PyPandas: A scalable data cleaning library

Pei-Lun Liao
New York University
pll273@nyu.edu

Chia-Hsien Lin
New York University
chl566@nyu.edu

Shang-Hung Tsai
New York University
st3127@nyu.edu

ABSTRACT

Data cleaning becomes a challenge in data mining and machine learning tasks. In this paper, we propose PyPandas, a scalable library running on Spark, to provide better solutions to data cleaning tasks. PyPandas provides basic cleaning features such as missing values handling, outlier detection and data scaling. Morevoer, advanced features like text cleaning are supported as well.

1 INTRODUCTION

In recent years, as storage devices become cheaper, storing big data in thousands of computers is easier than before. People start thinking about how to extract useful information from the huge datasets. The term, Big Data, was created to describe this phenomenon[**?**]. As the result, data mining[**? ?**] and machine learning[**?**] get popular nowadays. The quality of data is one of the key factors to successful data mining[**?**]. Hence, data cleaning becomes a challenge in the first step of data management and analysis[**? ? ?**].

To process huge dataset efficiently, distributed systems and parallel computing frameworks[**? ? ?**] were introduced. Spark[**?**] is one of the most popular open source projects for industry and academia. It is built on Hadoop[**?**] and provides a way to manage distributed memory. PySpark[**?**] is an extended library for Python programmers to work with Spark.

In the Python machine learning and data mining ecosystem, Pandas[**?**] is the most popular data management library. Pandas dataframe provides convenient ways to collect data and several data management methods for missing value filling, data indexing and data profiling.

However, Pandas can only be used in a single machine and could not be scaled to manage huge dataset. In this paper, we propose PyPandas, a library built on top of PySpark, that supports basic data management features, missing value handling or duplicated data removing. Moreover, we add common data processing features into our library to provide user friendly usage such as outlier detection and numerical data scaling. These features are important for the machine learning tasks and are supported in Scikit-Learn[**?**], a mainstream machine learning library, as well.

Furthermore, text data is another common data type in our storage. Text data is hard to process because of different languages, newly invented words, typos, abbreviations, urls and emoji etc. Although we have NLTK[**?**], a popular text cleaning library, there is no scalable text cleaning library. In this paper, we also provide tools for users to clean text easily.

The features of our library are summarized below.

- Scalable data management library runs on Spark
- Missing value handling, duplicated data removing, outlier detection and numerical data scaling
- Text processing such as url detection, punctuation removing, pattern searching and replacement.

2 RELATED WORK

There exists a number of open-source library for data cleaning and parallel data computing.

- Optimus[**?**] is a framework that can perform distributed data cleaning and preprocessing. This framework works well with Spark and its DataFrame can scale in big cluster. Optimus comes with helpful tools such as removing special characters and replacing null values.
- Dask[**?**] is another open-source library that supports parallel analytic computing. It provides scalable parallel data structures that extend interfaces like Pandas. At the same time, it offers low latency and high responsiveness.
- SparklingPandas[**?**] attempts to combine the power of Spark and Pandas to scale data analysis. It provides a Pandas-like API that is built using Spark's DataFrame class. Unfortunately, it only support spark v1.4 and Python 2.7, and its development has ended.

3 PROBLEM FORMULATION

4 METHODS, ARCHITECTURE AND DESIGN

In this project, we will to build our library on top of PySpark[**? ?**]. We will handle RDD data structure and implement useful data cleaning fucntions in our library. The ways to do that are collecting common cleaning functions into library and customizing map functions to perform advanced cleaning task such as pattern searching and replacement with regular expression. We will survey the Pandas[**?**] and Scikit-Learn[**?**] libraries to find out key features and create Pandas-like and Scikit-Learn-like API to provide easy usage and seamless adaptation.

4.1 Outlier Detection

4.1.1 Design. Outlier detection is a common problem in data cleaning. Outliers affects the experiment results and can lead to false conclusions. The data cleaning library we develop provide user-friendly solution to this problem.

4.1.2 Methods. The data cleaning library uses K-Means clustering method to detect outliers. Users can specify the number of clusters to use based their knowledge about the dataset. The system will choose random initial values, and perform 20 iterations of updates. Finally, a summary of the clusterings will be generated, which includes cluster centers, cluster sizes, distances, etc. User can then filter out outliers in each cluster by specifying a minimum distance from the cluster center, and all data points that are beyond that distance will be removed from the dataset.

4.1.3 Architecture. The implementation of the outlier detection functionality integrates the PySpark ml package. It is based on the KMeans clustering module, and it is very efficient and scalable. In addition, our data cleaning library implements User Defined

Functions to compute distance between data points and cluster centers, as well as other summary statistics. The architecture of this feature hides the complexity of K-Means clustering, while it exposes easily accessible APIs for users to remove outliers.

4.2 Scaling and Normalization

4.2.1 Design. Standardization of datasets is a common requirement for many machine learning algorithms, since objective functions will not work properly without scaling or normalization. Therefore, we provide several useful scaling functions and normalizing function. These functions can help standardize the specified columns and improve the performance of learning algorithms.

4.2.2 Methods. The standard_scale() function performs basic scaling on a particular column or a list of columns in a dataframe so that the values have unit standard deviation and/or zero mean. The min_max_scale() rescales columns to a user-defined range(e.g. $[0, 1]$) using min max scaling. Another scaling function max_abs_scale() transforms columns to range between -1 and 1 by dividing through the maximum absolute value in the columns. This operation does not shift/center the data, and thus does not destroy any sparsity. In addition, we provide normalize() function in our library. The normalize function can transform the given column(s) to have unit norm. The default p-norm value for normalization is 2.0, yet users can optionally specify the p-norm value.

4.2.3 Architecture. The implementation of the scaling and normalization functionality integrates the PySpark ml package as well. We include three the most commonly used scaling functions and one normalize function. The architecture and the API design is based on scikit-learn, the most popular machine learning library for the Python, so it is intuitive and easy to use.

5 EXPERIMENT

5.1 Dataset

Three datasets are chosen for the experiments. Each of them has different properties and can be used to test different aspects of the library. The summary of datasets could be found in table 1.

- 311 Service Requests[**?**] This dataset contains all 311 Service Requests since 2010. The majority of data has string type, including both categorical data (e.g. City, Status) and variable length data (e.g. Address, Description). This can be useful in testing the string-related features of the library.
- DOB Permit Issuance[**?**] The dataset consists of a list of permits for buildings in NYC issued by Department of Building. It has diverse data types, including categorical, string, numerical, and geographical data. Hence, it provides a good test case for the versatility of the library.
- DOB Job Application Filings[**?**] The dataset stores job applications filed through the Borough Offices in NYC. There are many columns containing numerical data, such as fee and number of stories. It can be used to test the data scaling and outlier detection features.

Table 1: Selected dataset

	(Row, Col)	Size	Missing Value
311 Service Requests	(9M, 53)	6.01 GB	28%
Permit Issuance	(3M, 60)	1.43 GB	20%
Job Application Filings	(5M, 89)	2.88 GB	40%

5.2 Evaluation

We will evaluate our project by comparing it against other existing open-source libraries, including Optimus[**?**], Dask[**?**], and SparkingPandas[**?**].

5.2.1 Features. We will compare our data cleaning features with previous works, particularly those frequently used functionality in data science such as null value handling, special character removal, duplicate detection, etc.

5.2.2 Performance. We will evaluate the performance of our library by measuring the time and space consumed and drawing comparison with existing libraries. This performance evaluation will show that this library is efficient and can be useful in production.

5.2.3 Usability. We might assess the usability of our library by doing a survey among a random group of users. We will compare the installation process and APIs provided and generate scores of user experience.

5.3 Result

6 CONCLUSIONS

Due to the lack of useful scalable data cleaning library on Spark, we propose PySpark, a scalable data cleaning library. We provide basic data cleaning features such as missing value handling, duplicated data removing and data scaling. Moreover, our library supports text cleaning feature with regular expression.