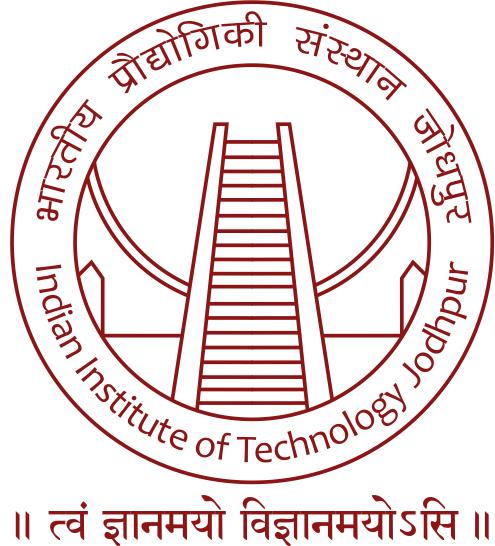
**SOFTWARE DEVELOPMENT AND ENGINEERING**

****

**A Project Report on**

**EXPLORATORY DATA ANALYSIS BENCHMARKING**

**DATAPREP.EDA Vs SPARK**

Submitted By,

Ashitabh Kumar - M20AIE221

Chitransh Sagar - MT19AIE230

**CONTENTS**

[**Introduction**](#_144vyzb2778c) **3**

[**General EDA Flow chart**](#_fmrewr1wk2hz) **5**

[**Purpose and Problem Statement**](#_8z2te13yyzdo) **5**

[**Datasets**](#_mh3te2c9byzi) **6**

[**Comparison**](#_mruhghe0x9xt) **7**

[**System Architecture**](#_kbpx72u6p1a8) **8**

[**Code Implementation and Snippet**](#_cy3sc9l0qu32) **9**

[Dataprep ( on Google Colab)](#_hb2x7sxi08l9) 9

[Great Expectations ( Spark) on Google Colab](#_t0t7cf5jlp2g) 11

[**Results**](#_2u00w28ucyg2) **12**

[**Cluster mode evaluation**](#_xgkbpf9qrgio) **14**

[Dataprep.EDA DASK](#_xfjyswkn7tjx) 15

[Dask scheduler on VM01](#_srwpqkc272p2) 15

[Dask Worker on other 3 VMs ( VM02,VM03,VM04)](#_vnrqgl9p5h2d) 15

[Dask dashboard](#_hzzc1oegptow) 16

[Great Expectations Spark](#_qxz28m6t1sh3) 16

[Master node](#_qxp3yg13945p) 16

[Worker node](#_1ibndwl84ar3) 16

[Spark Master UI](#_l6zv9wwodhzh) 17

[Jupyter notebooks for connecting to Dask](#_x519o6hlwa7r) 18

[Jupyter notebooks for connecting to Spark](#_g1e0o1yt2oih) 19

[**Features/Performance**](#_clsloaw64k81) **19**

[**Enhancement Opportunities**](#_h94loqa89lcx) **20**

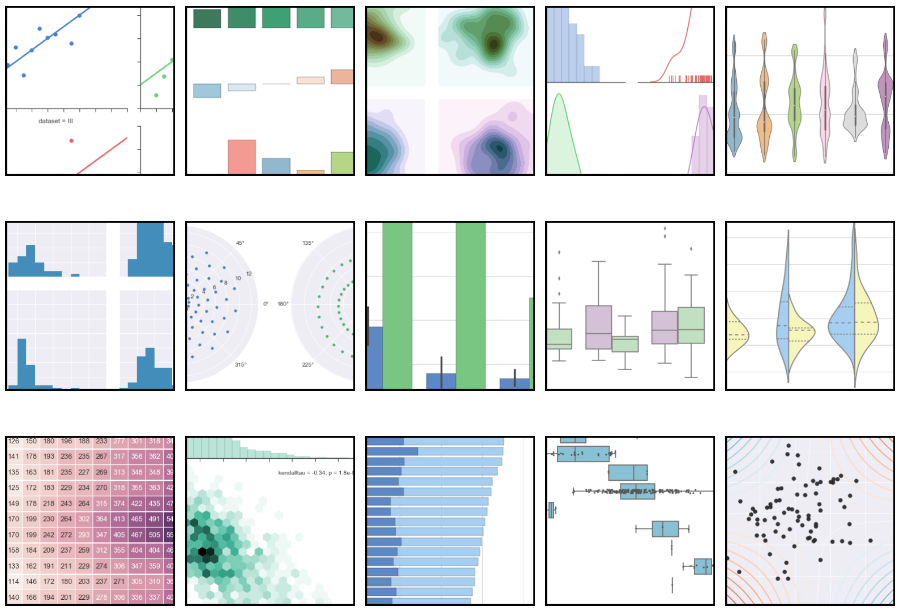
[**References**](#_gv4mpjul3thi) **20**

## 

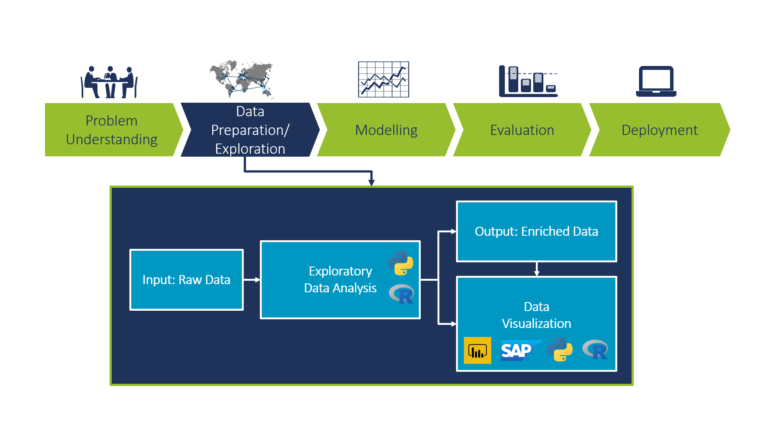
## Introduction

Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, mostly using statistical graphics and methods.It aims to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

There are various statistical KPI’s we generally use in EDA Task , some of which are shown below



## General EDA Flow chart



Our project is based on paper presented for Dataprep.EDA in ACM SIGMOD International Conference on Management of Data

<https://gateway.iitj.ac.in/proxy/48504b15/https/dl.acm.org/doi/10.1145/3448016.3457330>

This paper discusses a new EDA tool , developed and open source by the author Jinglin Peng. Source code for this tool , Dataprep.EDA, is available at <https://github.com/sfu-db/dataprep> .

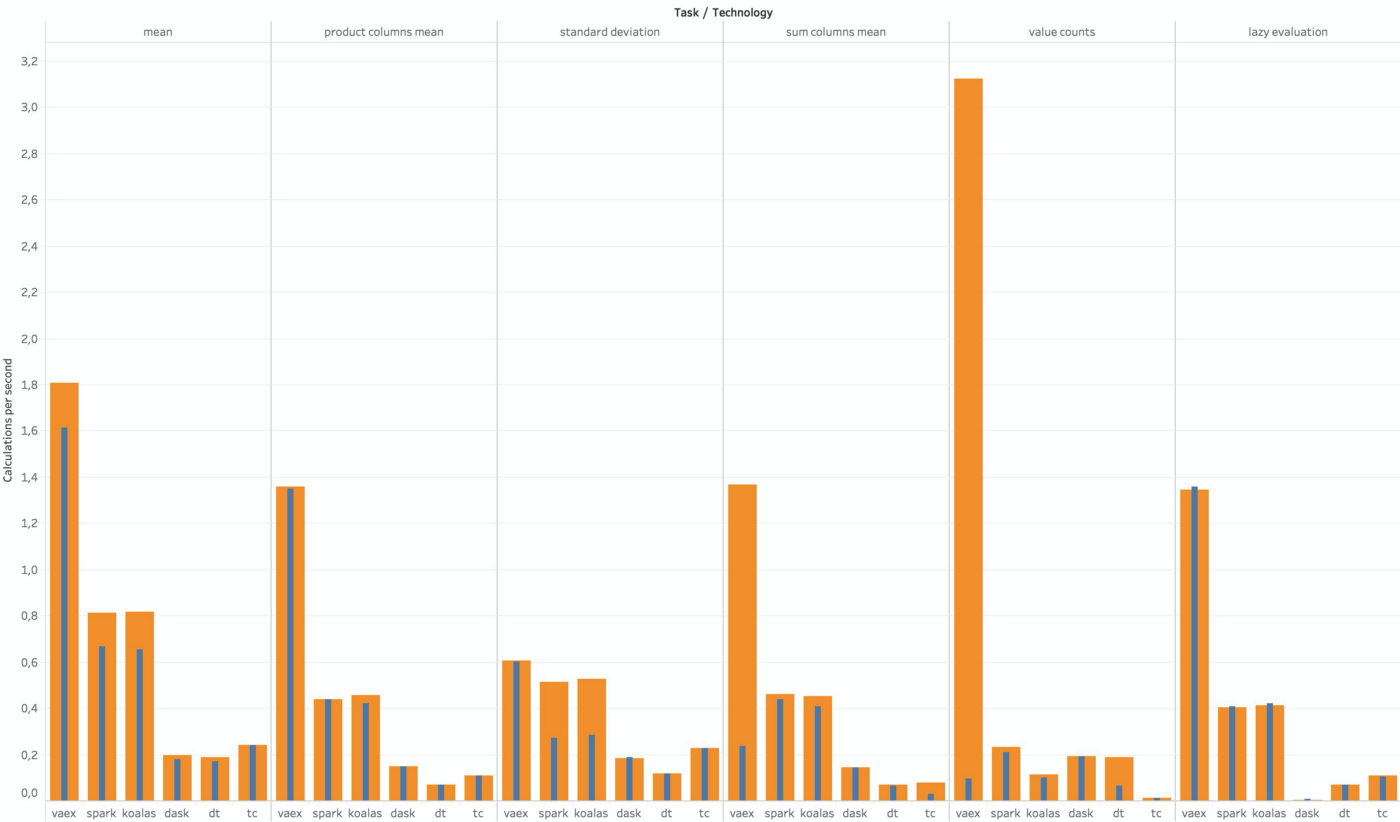
We will explore features provided by Dataprep.EDA and compare it with other similar options supported by Apache Spark **Great Expectations (**[**https://greatexpectations.io/**](https://greatexpectations.io/)**)**  . This analysis will help building unified platform for both Data exploration, Data Engineering and Machine Learning using Apache Spark

## Purpose and Problem Statement

For our project , we would like to benchmark Dataprep.EDA against Spark based EDA tools. Spark’s advantage comes from its capability to distribute load , therefore we will set up Dataprep.EDA on DASK , a distributed framework for Python.

We will set up the Dataprep.EDA library on a 3 node VM (Ubuntu) and will run it on a few large datasets from Kaggle/other open datasets. The performance statistics of core features will be recorded for benchmarking.

Spark is not the only available Big Data , however, we will keep only Spark as an option for this project as it is open source and supported by Databricks. Large community support and availability of enterprise support is necessary for wider adoption



## Datasets

In this project we are going to use an Open Source dataset “Smart Meters in London” to carry our EDA Tasks.It comprises of the following groups:-

1. Acorn\_Details
2. Daily Dataset.csv
3. Half hourly Dataset
4. HHBlock Dataset
5. Informations Households
6. US\_Bank Holidays
7. Weather Daily Darksky
8. Weather hourly Darkside

We will use Daily Dataset.csv (size : 335 MB) for our analysis

## Comparison

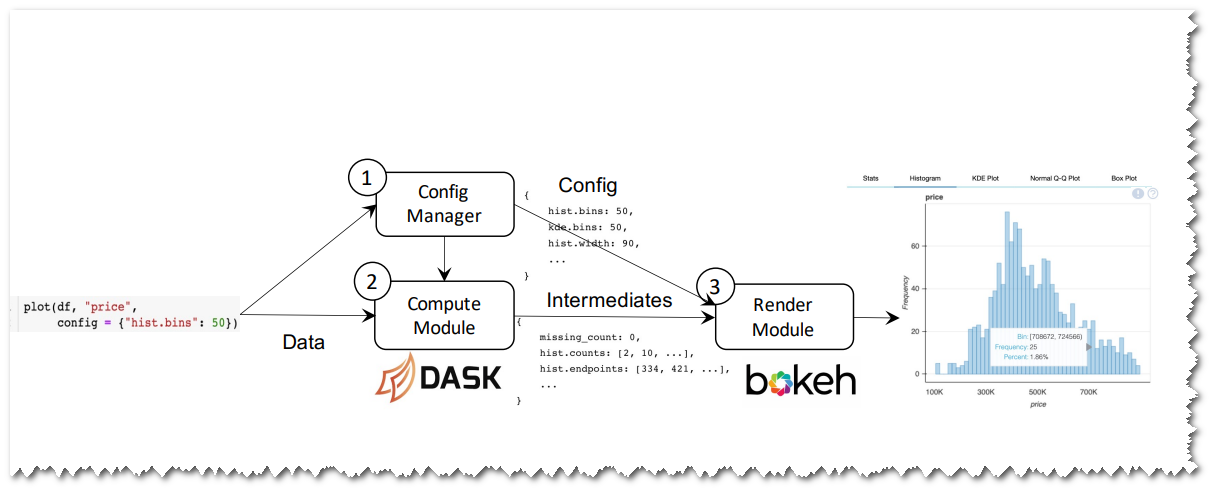
EDA is the process of exploring a dataset and getting an understanding of its main characteristics. The dataprep.eda package simplifies this process by allowing the user to explore important characteristics with simple APIs. Each API allows the user to analyze the dataset from a high level to a low level, and from different perspectives. Specifically, dataprep.eda provides the following functionality:

* Analyze column distributions with **plot()**. The function plot() explores the column distributions and statistics of the dataset. It will detect the column type, and then output various plots and statistics that are appropriate for the respective type. The user can optionally pass one or two columns of interest as parameters: If one column is passed, its distribution will be plotted in various ways, and column statistics will be computed. If two columns are passed, plots depicting the relationship between the two columns will be generated.
* Analyze correlations with plot\_correlation(). The function plot\_correlation() explores the correlation between columns in various ways and using multiple correlation metrics. By default, it plots correlation matrices with various metrics. The user can optionally pass one or two columns of interest as parameters: If one column is passed, the correlation between this column and all other columns will be computed and ranked. If two columns are passed, a scatter plot and regression line will be plotted.
* Analyze missing values with plot\_missing(). The function plot\_missing() enables thorough analysis of the missing values and their impact on the dataset. By default, it will generate various plots which display the amount of missing values for each column and any underlying patterns of the missing values in the dataset. To understand the impact of the missing values in one column on the other columns, the user can pass the column name as a parameter. Then, plot\_missing() will generate the distribution of each column with and without the missing values from the given column, enabling a thorough understanding of their impact.

## System Architecture

The **DataPrep.EDA** back-end is presented below, consisting of three components: 1 The Config Manager configures the system’s parameters, 2 the Compute module performs the computations on the data, and 3 the Render module creates the visualizations and

layouts. The Config Manager is used to organize the user-defined parameters and set default parameters in order to avoid setting and passing many parameters through the Compute and Render modules. The separation of the Compute module and the Render module has two benefits: First, the computations can be distributed to multiple visualizations. Second, the intermediate computations can be exposed to the user. This allows the user to create the visualizations with her desired plotting library

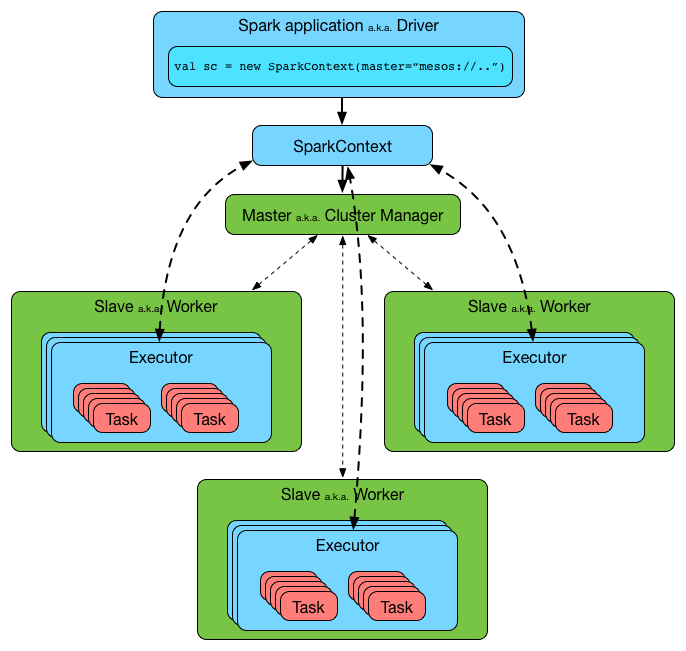


**Dask Background**. Dask is an open source library providing scalable analytics in Python. It offers similar APIs and data structures with other popular Python libraries, such as NumPy, Pandas, and Scikit-Learn. Internally, it partitions data into chunks, and runs computations over chunks in parallel.

The computations in Dask are lazy. Dask will first construct a computational graph that expresses the relationship between tasks. Then, it optimizes the graph to reduce computations such as removing unnecessary operators. Finally, it executes the graph when an eager operation like compute is called. Choice of Back-end Engine. We use Dask as the back-end engine of DataPrep.EDA for three reasons: (i) it is lightweight and fast in a single-node environment, (ii) it can scale to a distributed cluster, and (iii) it can optimize the computations required for multiple visualizations via lazy evaluation. We considered other engines like

Spark variants (PySpark and Koalas) and Modin , but found that they were less suitable for DataPrep.EDA than Dask. **Since Spark is designed for computations on very big data (TB to PB) in a large cluster, PySpark and Koalas are not lightweight like Dask and have a high scheduling overhead on a single node**. ( Here author mentioned that Spark is suitable for large datasets )

**Great Expectations** library just uses spark engine distributed computing capability to deliver result

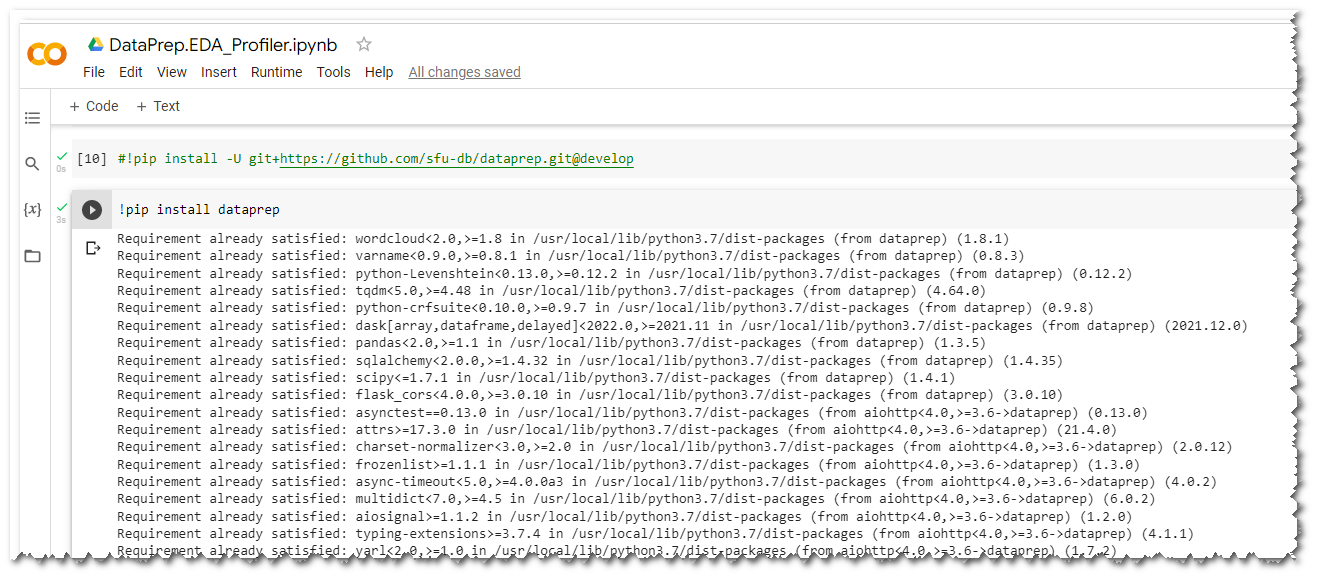


## Code Implementation and Snippet

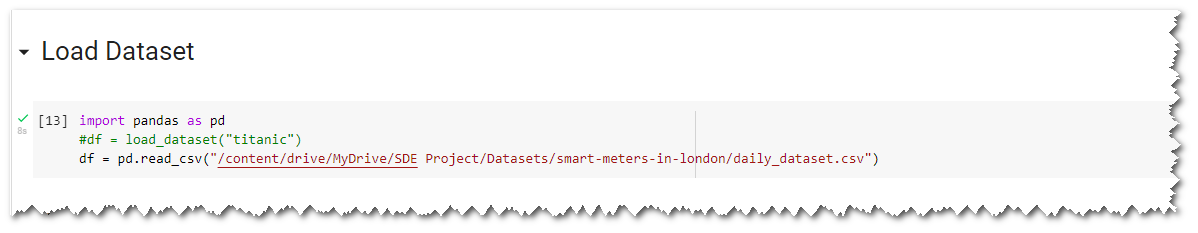
Initial code implementation is done on Google colab to confirm that both Dataprep.EDA and Spark Great Expectations can be executed on the same platform. We created 2 different notebooks for each and managed to make it work. Google colab doesn’t provide SPark, but we can manually setup latest Spark single node cluster

#### Dataprep ( on Google Colab)

Latest stable version or develop branch for experimental features



Reading project dataset via pandas



Plot the data profile (column distributions)

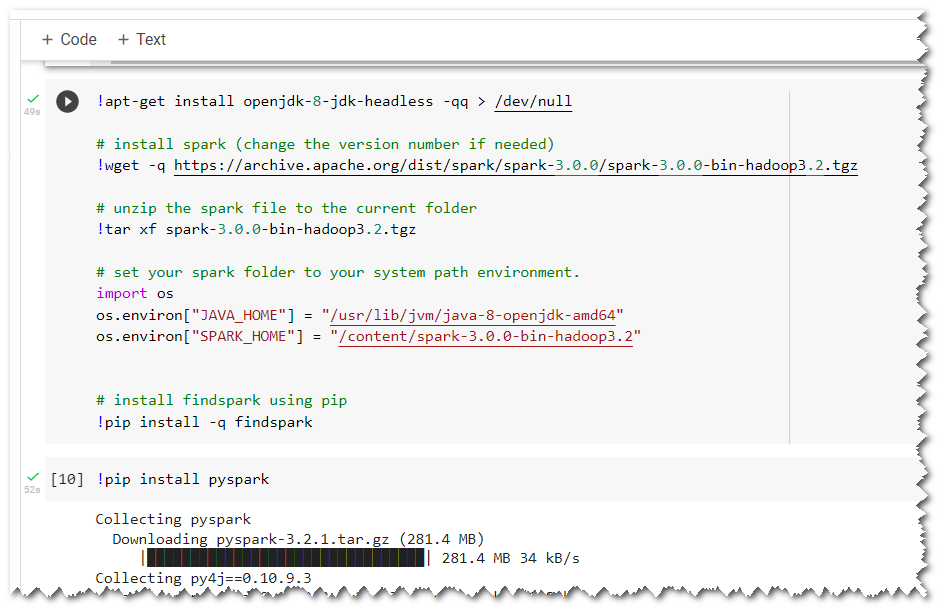
## 

#### Great Expectations ( Spark) on Google Colab

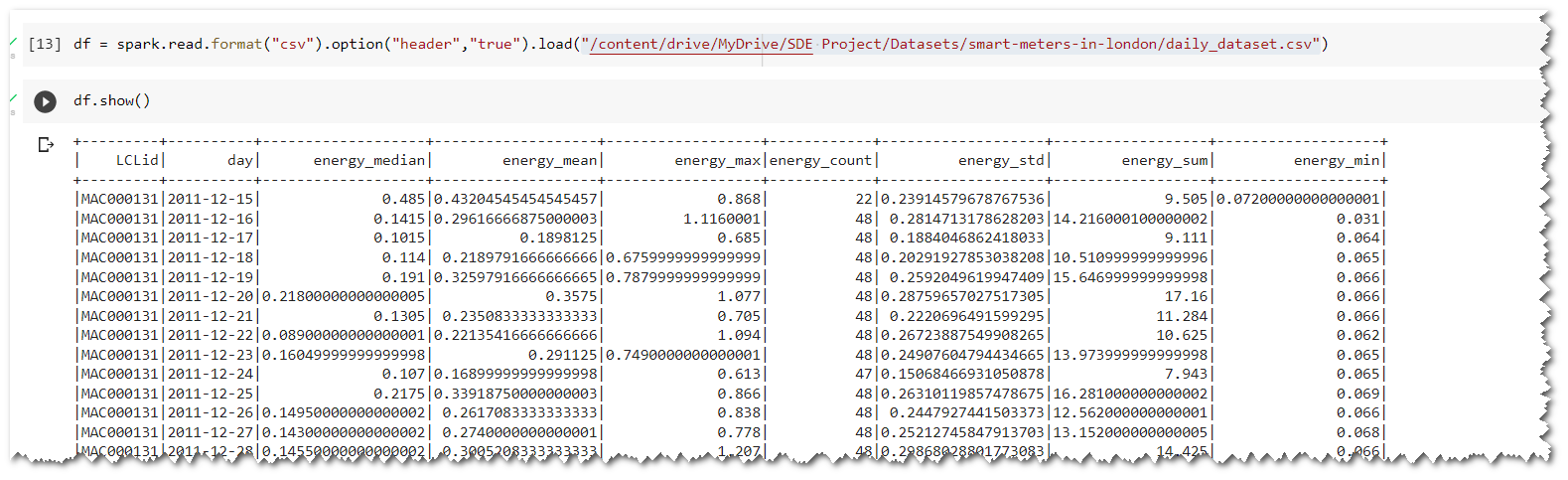
Latest version of Great Expectations

## 

Install Spark on Colab

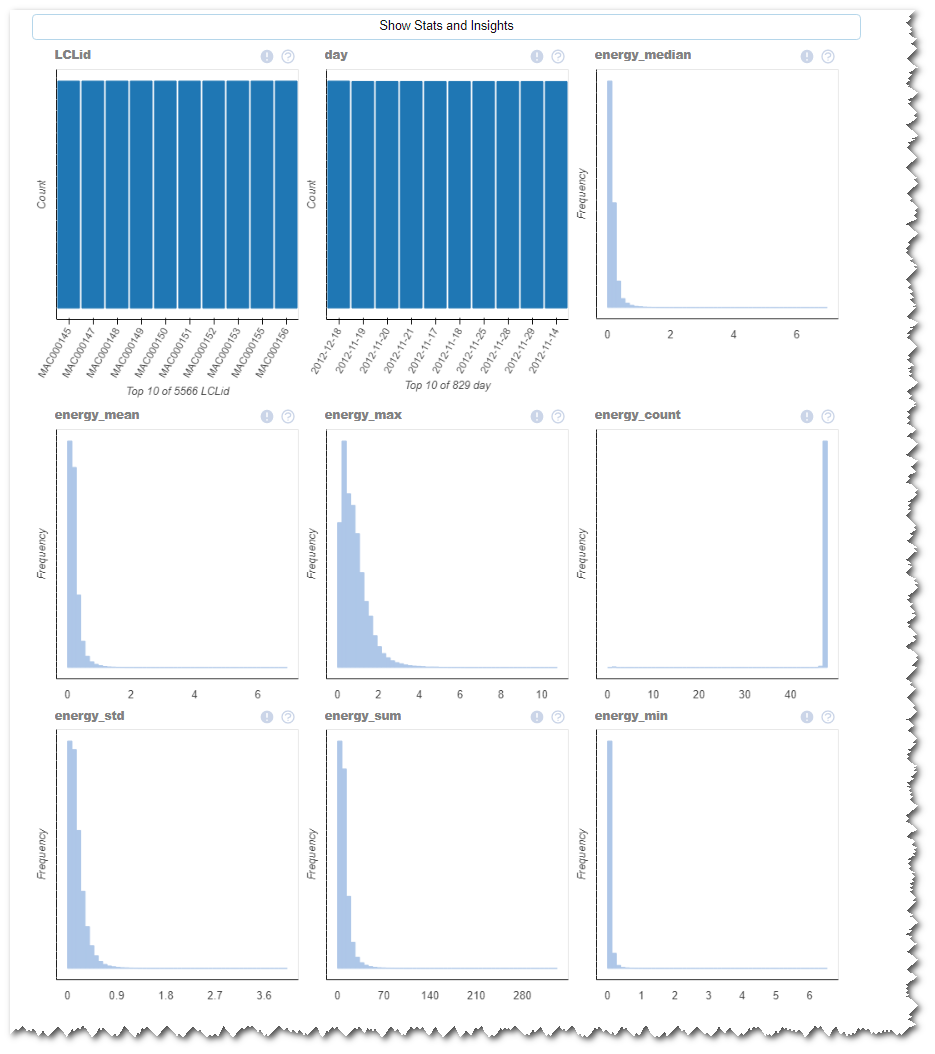


Load data using Spark dataframe

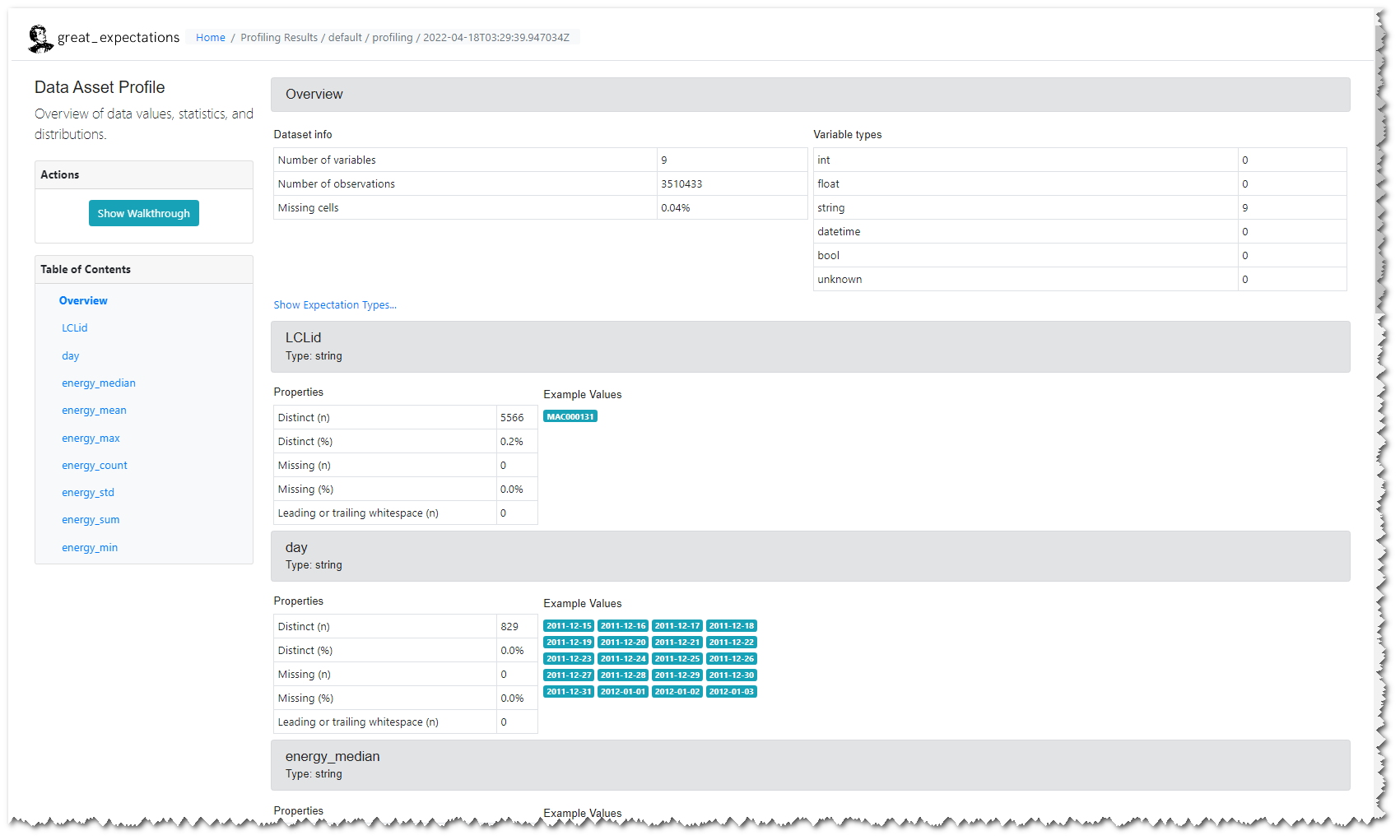


## Results

Dataprep.EDA screenshots



Great Expectations ( Spark) screenshots



## Cluster mode evaluation

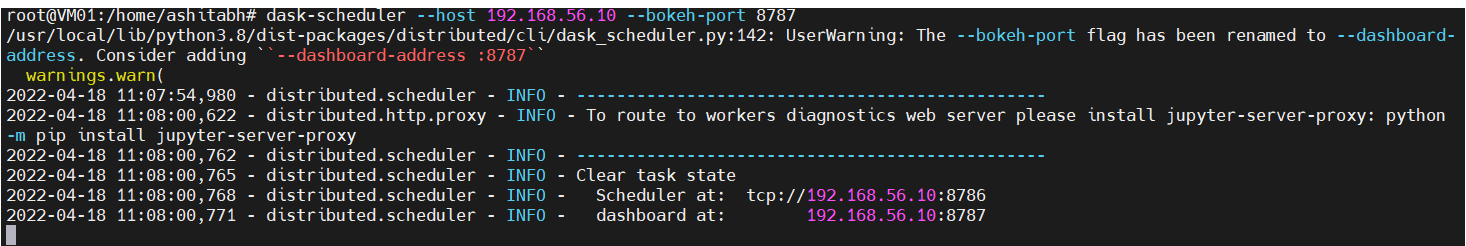
Since we were using free version of COlab, we can’t evaluate the real performance of distributed computation ( Dask vs Spark). So we created a 4 node VM ( Oracle VirtualBox)

## 

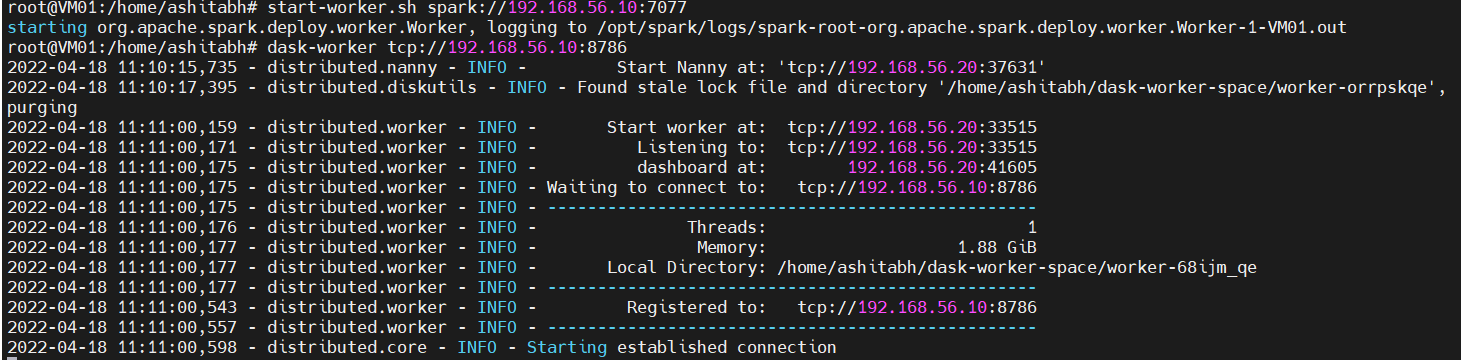
### Dataprep.EDA DASK

For Dataprep.EDA , we will be running Dask scheduler on VM01 and all 3 VM ( VM02,VM03,VM04) will be running worker node pointing to VM01

#### Dask scheduler on VM01



#### Dask Worker on other 3 VMs ( VM02,VM03,VM04)



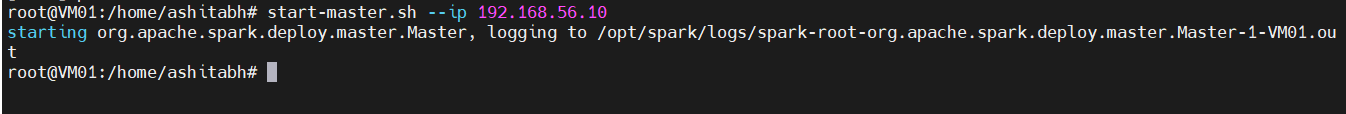
#### Dask dashboard

### 

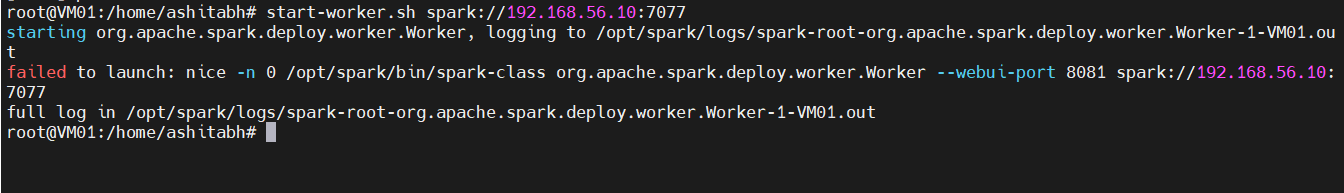
### Great Expectations Spark

For Spark , we will be running spark master on VM01 and workers on other 3 VMs( Vm02,VM03,VM04)

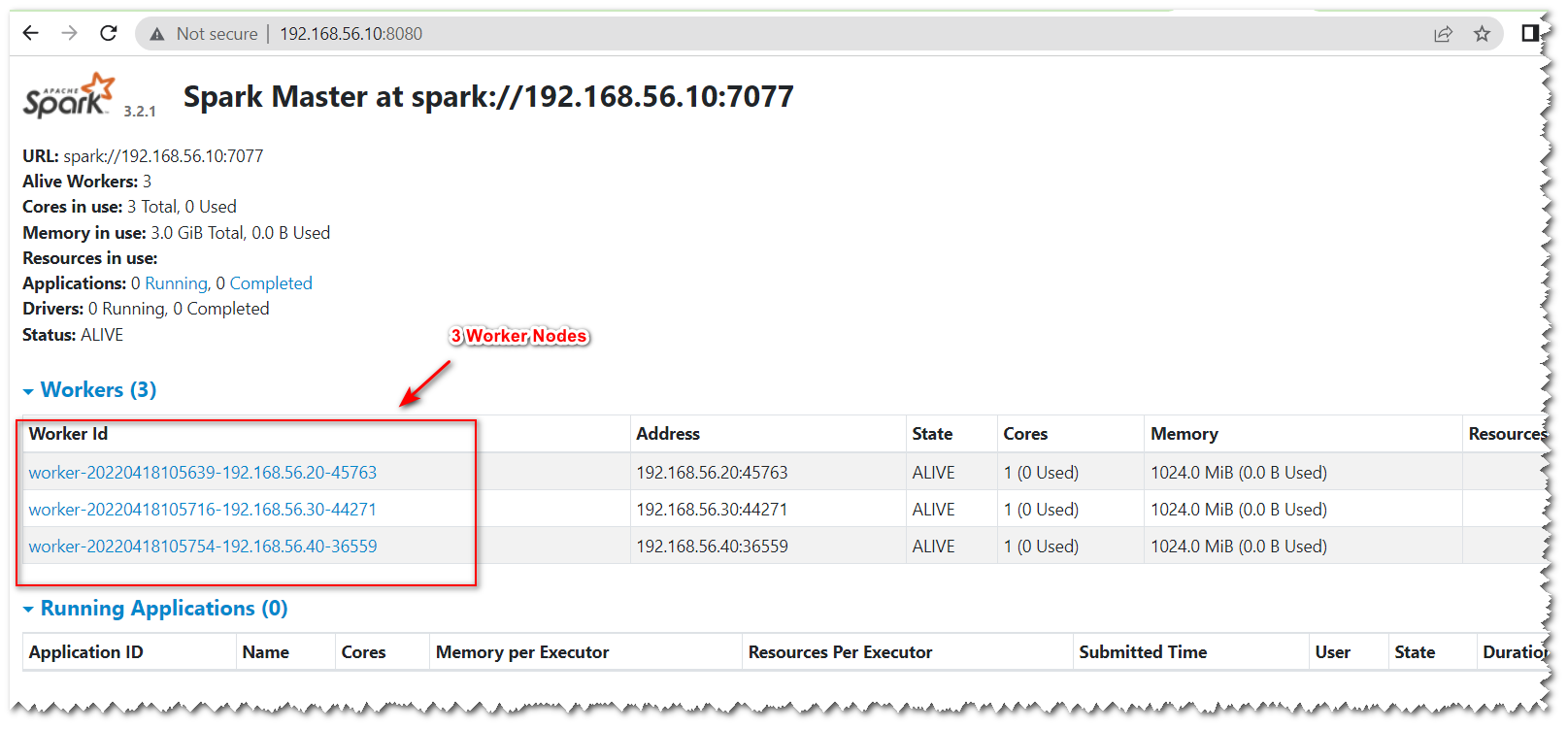
#### Master node



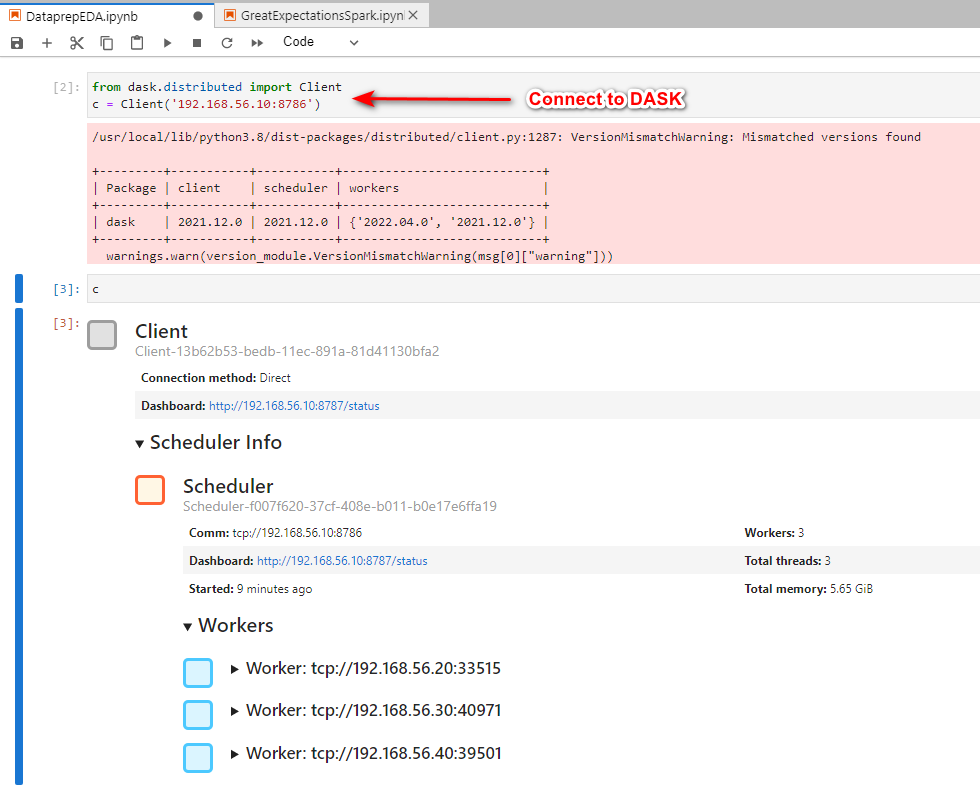
#### Worker node



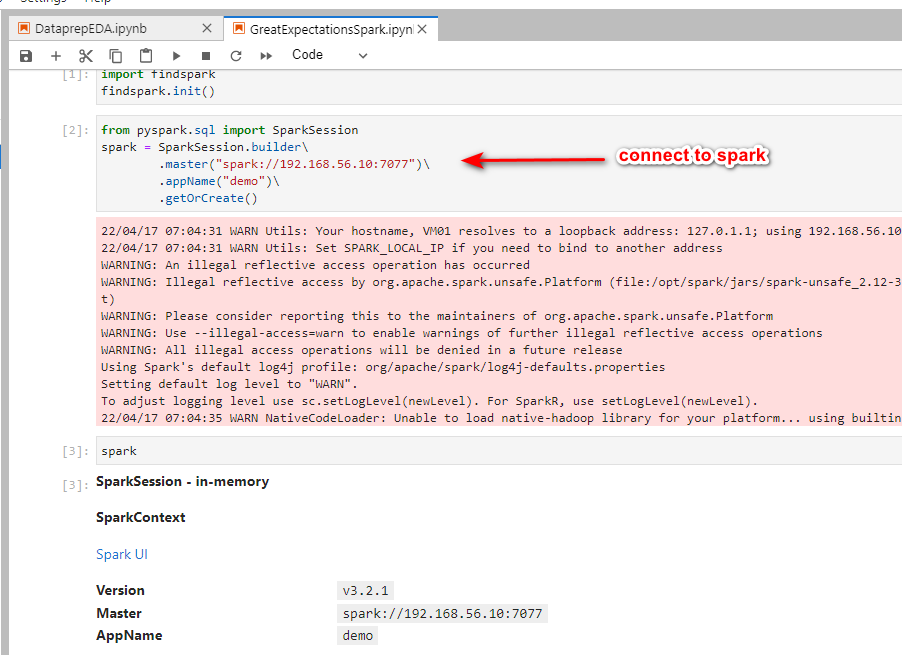
#### Spark Master UI



### Jupyter notebooks for connecting to Dask



### Jupyter notebooks for connecting to Spark



## Features/Performance

* Performance
  + Both DataPrep.eda and Great expectations support parallel execution.
* Handles Large Data
  + Both DataPrep.eda and Great expectations support large datasets as they can distribute data processing across clusters
* Managed service
  + All 3 major cloud providers ( Azure ,GCP and AWS) provide managed Databricks Spark cluster, so you don’t need to spend time and money for managing your cluster. Dask still needs manual deployment of clusters
* Smart Visualization
  + DataPrep.eda has better visualization for the data
* Integration with Data Engineering/Data Science Pipeline
  + Great Expectations provides API to automate data engineering/ data science pipelines to take automated actions based on rules setup for Data profile, missing data ,skewed data etc. DataPrep.EDA is targeted for better visualization , but APIs are not configurable to automate /integrate with existing DE/DS pipelines.It serves better as a standalone tool.

## Enhancement Opportunities

Spark provides a better opportunity to manage large datasets. However , Great expectation features were not targeted towards better visualization.

We can enhance visualization using the same Bokeh component and provide smooth transition to dask users.

In most of organizations , they have already adopted Spark for large data processing, so it will be easy for new users to run these jobs instead of creating another cluster for Dask workloads

## References

* <https://gateway.iitj.ac.in/proxy/48504b15/https/dl.acm.org/doi/10.1145/3448016.3457330>
* <https://vaex.io/blog/beyond-pandas-spark-dask-vaex-and-other-big-data-technologies-battling-head-to-head>
* https://towardsdatascience.com/exploratory-data-analysis-dataprep-eda-vs-pandas-profiling-7137683fe47f
* <https://docs.dask.org/en/stable/deploying-cli.html>
* <https://towardsdatascience.com/automating-eda-using-dataprep-2b541b6a3149>
* https://greatexpectations.io/blog/ml-ops-great-expectations/
* https://colab.research.google.com/drive/1kBCWoyagmfuwny-TPIaKgCubh2m9mDa-#scrollTo=6e7uG2\_N2zyx