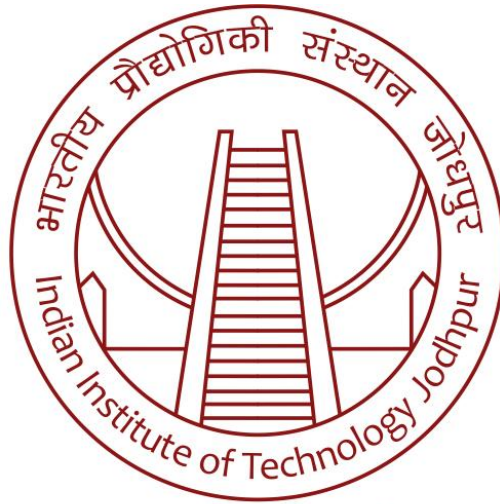


# SOFTWARE DEVELOPMENT AND ENGINEERING



॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

**A Project Report on**

## **EXPLORATORY DATA ANALYSIS BENCHMARKING** **DATAPREP.EDA Vs SPARK**

Submitted By,  
Ashitabh Kumar - M20AIE221  
Chitransh Sagar - MT19AIE230

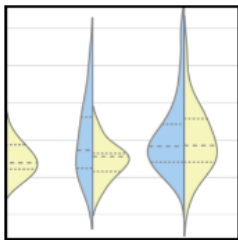
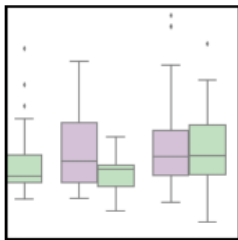
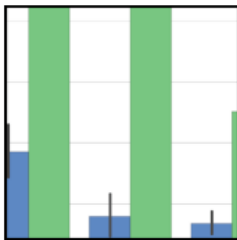
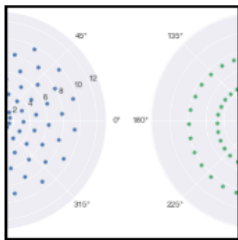
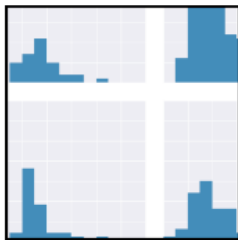
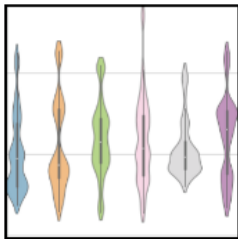
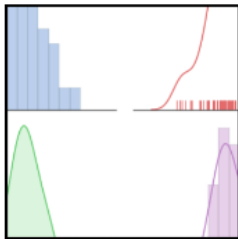
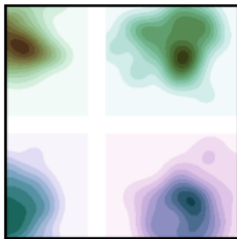
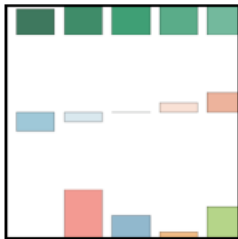
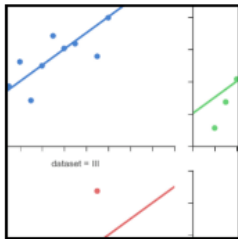
# CONTENTS

<b>Introduction</b>	<b>3</b>
<b>General EDA Flow chart</b>	<b>5</b>
<b>Purpose and Problem Statement</b>	<b>5</b>
<b>Datasets</b>	<b>6</b>
<b>Comparison</b>	<b>7</b>
<b>System Architecture</b>	<b>8</b>
<b>Code Implementation and Snippet</b>	<b>9</b>
Dataprep ( on Google Colab)	9
Great Expectations ( Spark) on Google Colab	11
<b>Results</b>	<b>12</b>
<b>Cluster mode evaluation</b>	<b>14</b>
Dataprep.EDA DASK	15
Dask scheduler on VM01	15
Dask Worker on other 3 VMs ( VM02,VM03,VM04)	15
Dask dashboard	16
Great Expectations Spark	16
Master node	16
Worker node	16
Spark Master UI	17
Jupyter notebooks for connecting to Dask	18
Jupyter notebooks for connecting to Spark	19
<b>Features/Performance</b>	<b>19</b>
<b>Enhancement Opportunities</b>	<b>20</b>
<b>References</b>	<b>20</b>

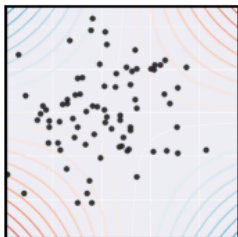
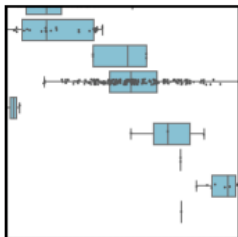
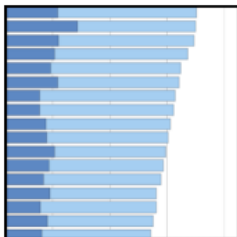
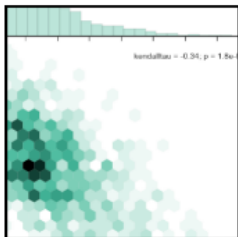
## Introduction

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.

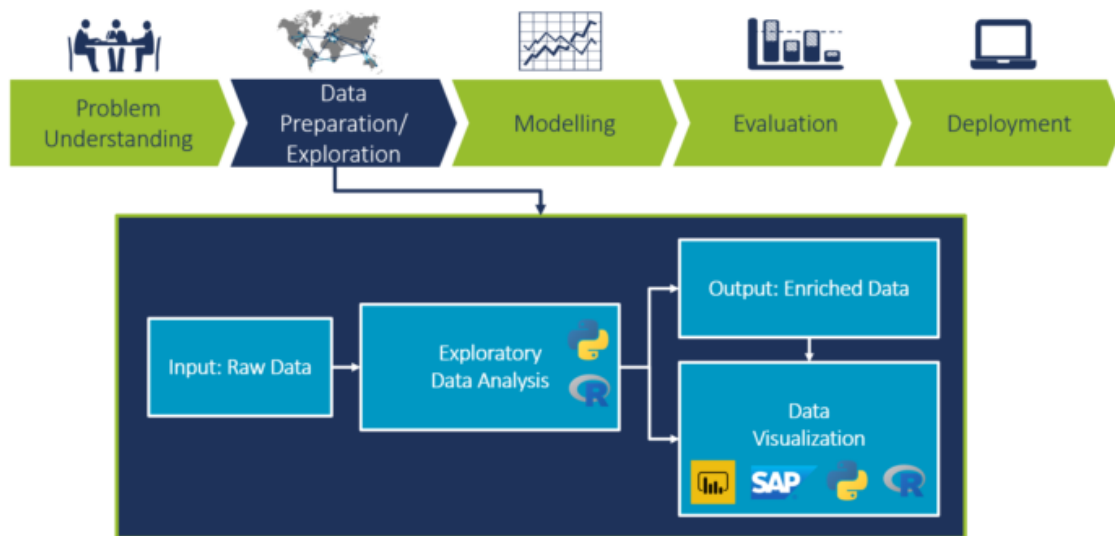
There are various statistical KPI's we generally use in EDA Task , some of which are shown below



126	150	180	195	185	253	277	301	318	31
141	178	193	236	235	267	317	356	362	45
135	163	191	235	227	269	315	348	345	35
125	172	183	229	234	270	318	365	363	43
149	178	218	243	264	315	374	422	435	47
170	199	230	264	302	364	413	465	491	54
170	199	242	272	293	347	405	467	505	56
158	184	209	237	269	312	355	404	434	46
133	162	191	211	229	274	306	347	369	40
114	146	172	190	203	237	271	305	319	36
140	166	194	201	229	278	305	336	337	45



## 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> .

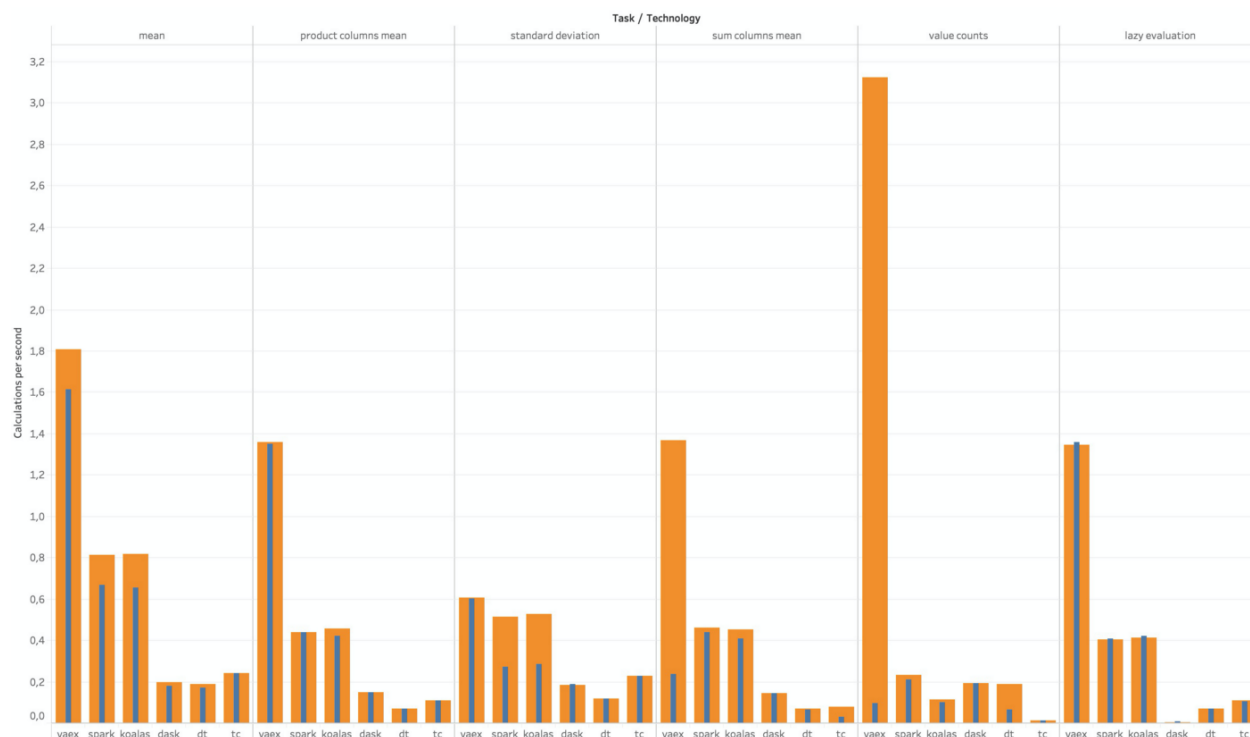
We will explore features provided by Dataprep.EDA and compare it with other similar options supported by Apache Spark **Great Expectations** (<https://greatexpectations.io/>) . This analysis will help building unified platform for both Data exploration, Data Engineering and Machine Learning using Apache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

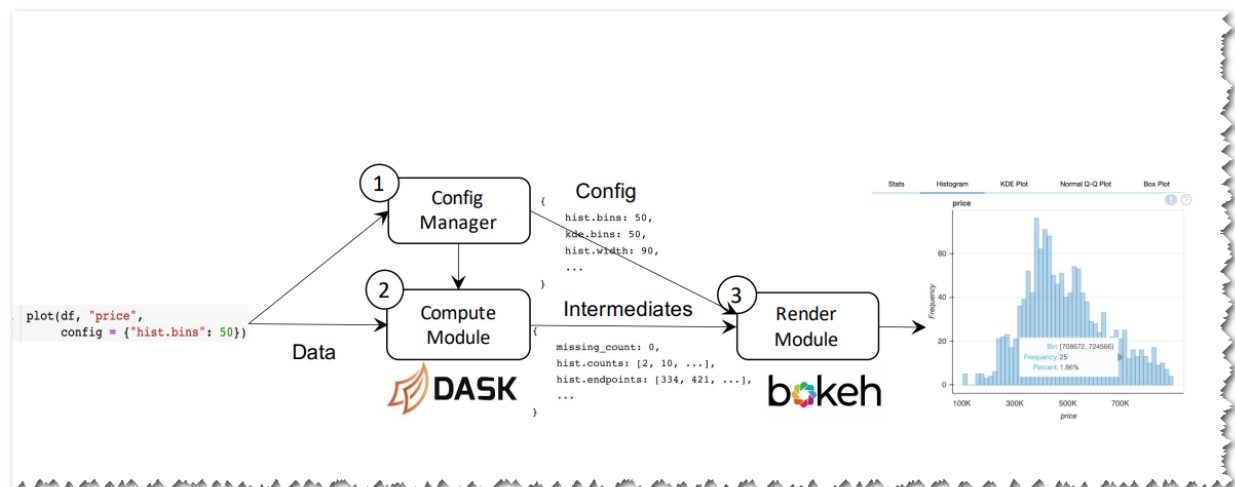
## Comparison

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

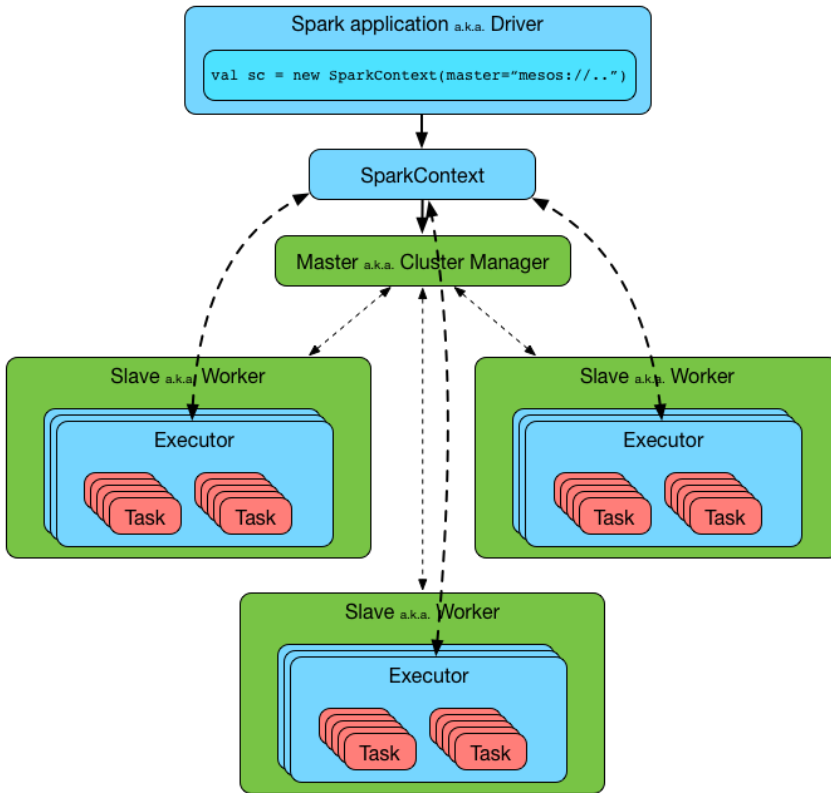


**Dask Background.** Dask is an open source library providing scalable analytics in Python. It offers similar APIs and data structures with other popular Python libraries, such as NumPy, Pandas, and Scikit-Learn. Internally, it partitions data into chunks, and runs computations over chunks in parallel.

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

```
co DataPrep.EDA_Profiler.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10] #!pip install -U git+https://github.com/sfu-db/dataprep.git@develop

{x} !pip install dataprep

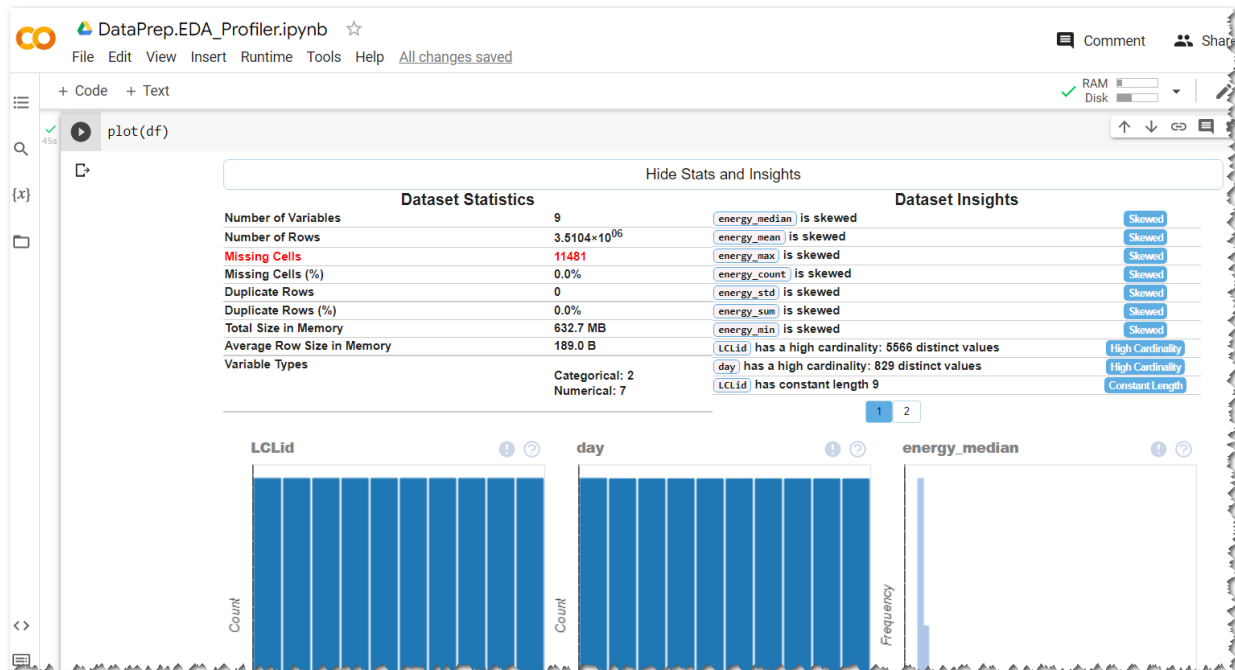
Requirement already satisfied: wordcloud<2.0,>=1.8 in /usr/local/lib/python3.7/dist-packages (from dataprep) (1.8.1)
Requirement already satisfied: varname<0.9.0,>=0.8.1 in /usr/local/lib/python3.7/dist-packages (from dataprep) (0.8.3)
Requirement already satisfied: python-Levenshtein<0.13.0,>=0.12.2 in /usr/local/lib/python3.7/dist-packages (from dataprep) (0.12.2)
Requirement already satisfied: tqdm<5.0,>=4.48 in /usr/local/lib/python3.7/dist-packages (from dataprep) (4.64.0)
Requirement already satisfied: python-crfsuite<0.10.0,>=0.9.7 in /usr/local/lib/python3.7/dist-packages (from dataprep) (0.9.8)
Requirement already satisfied: dask[array,dataframe,delayed]<2022.0,>=2021.11 in /usr/local/lib/python3.7/dist-packages (from dataprep) (2021.12.0)
Requirement already satisfied: pandas<2.0,>=1.1 in /usr/local/lib/python3.7/dist-packages (from dataprep) (1.3.5)
Requirement already satisfied: sqlalchemy<2.0.0,>=1.4.32 in /usr/local/lib/python3.7/dist-packages (from dataprep) (1.4.35)
Requirement already satisfied: scipy<1.7.1 in /usr/local/lib/python3.7/dist-packages (from dataprep) (1.4.1)
Requirement already satisfied: flask_cors<4.0.0,>=3.0.10 in /usr/local/lib/python3.7/dist-packages (from dataprep) (3.0.10)
Requirement already satisfied: async_test==0.13.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp<4.0,>=3.6->dataprep) (0.13.0)
Requirement already satisfied: attrs==17.3.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp<4.0,>=3.6->dataprep) (21.4.0)
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp<4.0,>=3.6->dataprep) (2.0.12)
Requirement already satisfied: frozenlist==1.1.1 in /usr/local/lib/python3.7/dist-packages (from aiohttp<4.0,>=3.6->dataprep) (1.3.0)
Requirement already satisfied: async_timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.7/dist-packages (from aiohttp<4.0,>=3.6->dataprep) (4.0.2)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.7/dist-packages (from aiohttp<4.0,>=3.6->dataprep) (6.0.2)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.7/dist-packages (from aiohttp<4.0,>=3.6->dataprep) (1.2.0)
Requirement already satisfied: typing_extensions>=3.7.4 in /usr/local/lib/python3.7/dist-packages (from aiohttp<4.0,>=3.6->dataprep) (4.1.1)
Requirement already satisfied: varl<2.0,>=1.0 in /usr/local/lib/python3.7/dist-packages (from aiohttp<4.0,>=3.6->dataprep) (1.7.2)
```

## Reading project dataset via pandas

```
Load Dataset

[13] import pandas as pd
      #df = load_dataset("titanic")
      df = pd.read_csv("/content/drive/MyDrive/SDE Project/Datasets/smart-meters-in-london/daily_dataset.csv")
```

## Plot the data profile (column distributions)



## Great Expectations ( Spark) on Google Colab

### Latest version of Great Expectations

```

Spark_Data_Profiler.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

'/content/drive/MyDrive/SDE Project/Datasets/smart-meters-in-london/daily_dataset.csv'

!pip install great_expectations

Requirement already satisfied: pickleshare in /usr/local/lib/python3.7/dist-packages (from Ipython>=7.16.3->great_expectations) (0.7.5)
Collecting prompt-toolkit<3.0.0,!=3.0.1,<3.1.0,>=2.0.0
  Downloading prompt_toolkit-3.0.29-py3-none-any.whl (381 kB)
    381 kB 67.0 MB/s
Requirement already satisfied: setuptools<=18.5 in /usr/local/lib/python3.7/dist-packages (from Ipython>=7.16.3->great_expectations) (57.4.0)
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.7/dist-packages (from Ipython>=7.16.3->great_expectations) (0.1.3)
Requirement already satisfied: backcall in /usr/local/lib/python3.7/dist-packages (from Ipython>=7.16.3->great_expectations) (0.2.0)
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-packages (from Ipython>=7.16.3->great_expectations) (2.6.1)
Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-packages (from Ipython>=7.16.3->great_expectations) (4.4.2)
Requirement already satisfied: jedi<=0.16 in /usr/local/lib/python3.7/dist-packages (from Ipython>=7.16.3->great_expectations) (0.18.1)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in /usr/local/lib/python3.7/dist-packages (from jedi<=0.16->Ipython>=7.16.3->great_expectations) (0.8.1)
Requirement already satisfied: MarkupSafe<=0.23 in /usr/local/lib/python3.7/dist-packages (from Jinja2<3.1.0,>=2.10->great_expectations) (2.0.1)
Collecting jsonpointer<=1.9
  Downloading jsonpointer-2.3-py2.py3-none-any.whl (7.8 kB)
Requirement already satisfied: attrs<=17.4.0 in /usr/local/lib/python3.7/dist-packages (from jsonschema>=2.5.1->great_expectations) (21.4.0)
Requirement already satisfied: importlib-resources<=1.4.0 in /usr/local/lib/python3.7/dist-packages (from jsonschema>=2.5.1->great_expectations) (5.12.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.7/dist-packages (from jsonschema>=2.5.1->great_expectations) (0.19.3)
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.7/dist-packages (from nbformat>=5.0->great_expectations) (4.9.2)
Requirement already satisfied: fastjsonschema in /usr/local/lib/python3.7/dist-packages (from nbformat>=5.0->great_expectations) (2.15.3)
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.7/dist-packages (from notebook>=6.4.10->great_expectations) (21.3.0)
Requirement already satisfied: jupyter-client<=5.3.4 in /usr/local/lib/python3.7/dist-packages (from notebook>=6.4.10->great_expectations) (5.3.3)
Requirement already satisfied: nbconvert<=5 in /usr/local/lib/python3.7/dist-packages (from notebook>=6.4.10->great_expectations) (5.6.1)
Requirement already satisfied: terminado<=0.8.3 in /usr/local/lib/python3.7/dist-packages (from notebook>=6.4.10->great_expectations) (0.13.3)
Requirement already satisfied: prometheus-client in /usr/local/lib/python3.7/dist-packages (from notebook>=6.4.10->great_expectations) (0.14.1)
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.7/dist-packages (from notebook>=6.4.10->great_expectations) (0.2.0)

```

### Install Spark on Colab

```
+ Code + Text

✓ 49s ▶ !apt-get install openjdk-8-jdk-headless -qq > /dev/null

# install spark (change the version number if needed)
!wget -q https://archive.apache.org/dist/spark/spark-3.0.0/spark-3.0.0-bin-hadoop3.2.tgz

# unzip the spark file to the current folder
!tar xf spark-3.0.0-bin-hadoop3.2.tgz

# set your spark folder to your system path environment.
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.0-bin-hadoop3.2"

# install findspark using pip
!pip install -q findspark

✓ [10] 52s !pip install pyspark

Collecting pyspark
  Downloading pyspark-3.2.1.tar.gz (281.4 MB)
    |████████████████████████████████████████| 281.4 MB 34 kB/s
Collecting py4j==0.10.9.3
```

## Load data using Spark dataframe

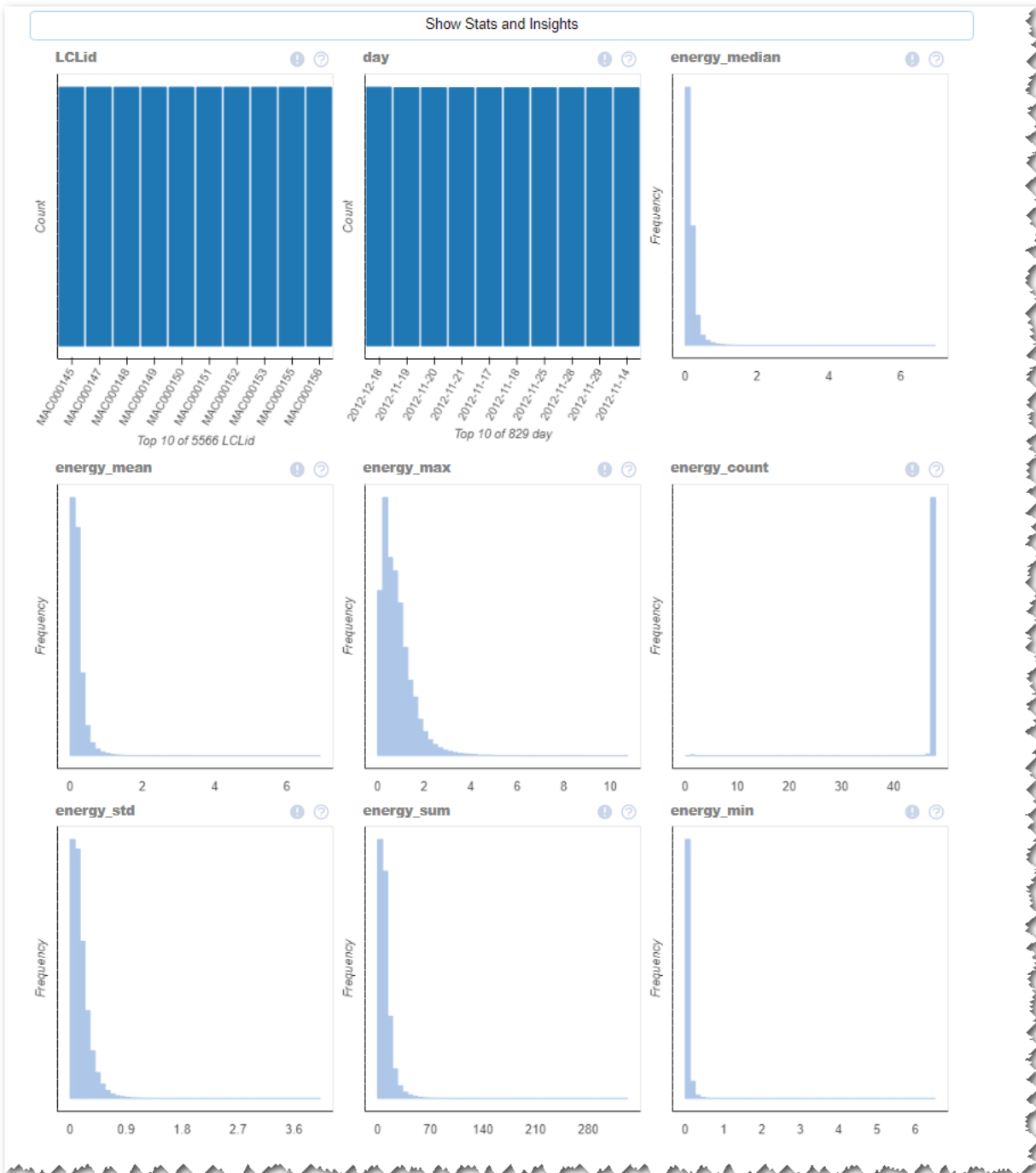
```
✓ [13] df = spark.read.format("csv").option("header", "true").load("/content/drive/MyDrive/SDE Project/Datasets/smart-meters-in-london/daily_dataset.csv")

✓ ▶ df.show()
```

LCLid	day	energy_median	energy_mean	energy_max	energy_count	energy_std	energy_sum	energy_min
MAC000131	2011-12-15	0.485	0.4320454545454545	0.868	22	0.23914579678767536	9.505	0.07200000000000001
MAC000131	2011-12-16	0.1415	0.29616666875000003	1.1160001	48	0.2814713178628203	14.216000100000002	0.031
MAC000131	2011-12-17	0.1015	0.1898125	0.685	48	0.1884046862418033	9.111	0.064
MAC000131	2011-12-18	0.114	0.21897916666666666	0.6759999999999999	48	0.20291927853038208	10.510999999999996	0.065
MAC000131	2011-12-19	0.191	0.32597916666666665	0.7879999999999999	48	0.2592049619947409	15.646999999999998	0.066
MAC000131	2011-12-20	0.21800000000000005	0.3575	1.077	48	0.28759657027517305	17.16	0.066
MAC000131	2011-12-21	0.1305	0.23508333333333333	0.705	48	0.2220696491599295	11.284	0.066
MAC000131	2011-12-22	0.08900000000000001	0.22135416666666666	1.094	48	0.26723887549908265	10.625	0.062
MAC000131	2011-12-23	0.16049999999999998	0.291125	0.7490000000000001	48	0.24907604794434665	13.973999999999998	0.065
MAC000131	2011-12-24	0.107	0.16899999999999998	0.613	47	0.15068466931050878	7.943	0.065
MAC000131	2011-12-25	0.2175	0.33918750000000003	0.866	48	0.26310119857478675	16.281000000000002	0.069
MAC000131	2011-12-26	0.14950000000000002	0.2617083333333333	0.838	48	0.2447927441503373	12.562000000000001	0.066
MAC000131	2011-12-27	0.14300000000000002	0.27400000000000001	0.778	48	0.25212745847913703	13.152000000000005	0.068
MAC000131	2011-12-28	0.14550000000000002	0.3005208333333333	1.207	48	0.29868928801773083	14.425	0.066

## Results

### Dataprep.EDA screenshots



Great Expectations ( Spark) screenshots

great\_expectations Home / Profiling Results / default / profiling / 2022-04-18T03:29:39.947034Z

### Data Asset Profile

Overview of data values, statistics, and distributions.

**Actions**

Show Walkthrough

**Table of Contents**

- Overview
- LCLid
- day
- energy\_median
- energy\_mean
- energy\_max
- energy\_count
- energy\_std
- energy\_sum
- energy\_min

#### Overview

**Dataset info**

Number of variables	9
Number of observations	3510433
Missing cells	0.04%

**Variable types**

int	0
float	0
string	9
datetime	0
bool	0
unknown	0

Show Expectation Types...

#### LCLid

Type: string

**Properties**

Distinct (n)	5566
Distinct (%)	0.2%
Missing (n)	0
Missing (%)	0.0%
Leading or trailing whitespace (n)	0

**Example Values**

MAC000111

#### day

Type: string

**Properties**

Distinct (n)	829
Distinct (%)	0.0%
Missing (n)	0
Missing (%)	0.0%
Leading or trailing whitespace (n)	0

**Example Values**

2011-12-10 2011-12-10 2011-12-10 2011-12-10  
2011-12-10 2011-12-20 2011-12-20 2011-12-20  
2011-12-20 2011-12-20 2011-12-20 2011-12-20  
2011-12-27 2011-12-28 2011-12-28 2011-12-30  
2011-12-31 2012-01-01 2012-01-02 2012-01-03

#### energy\_median

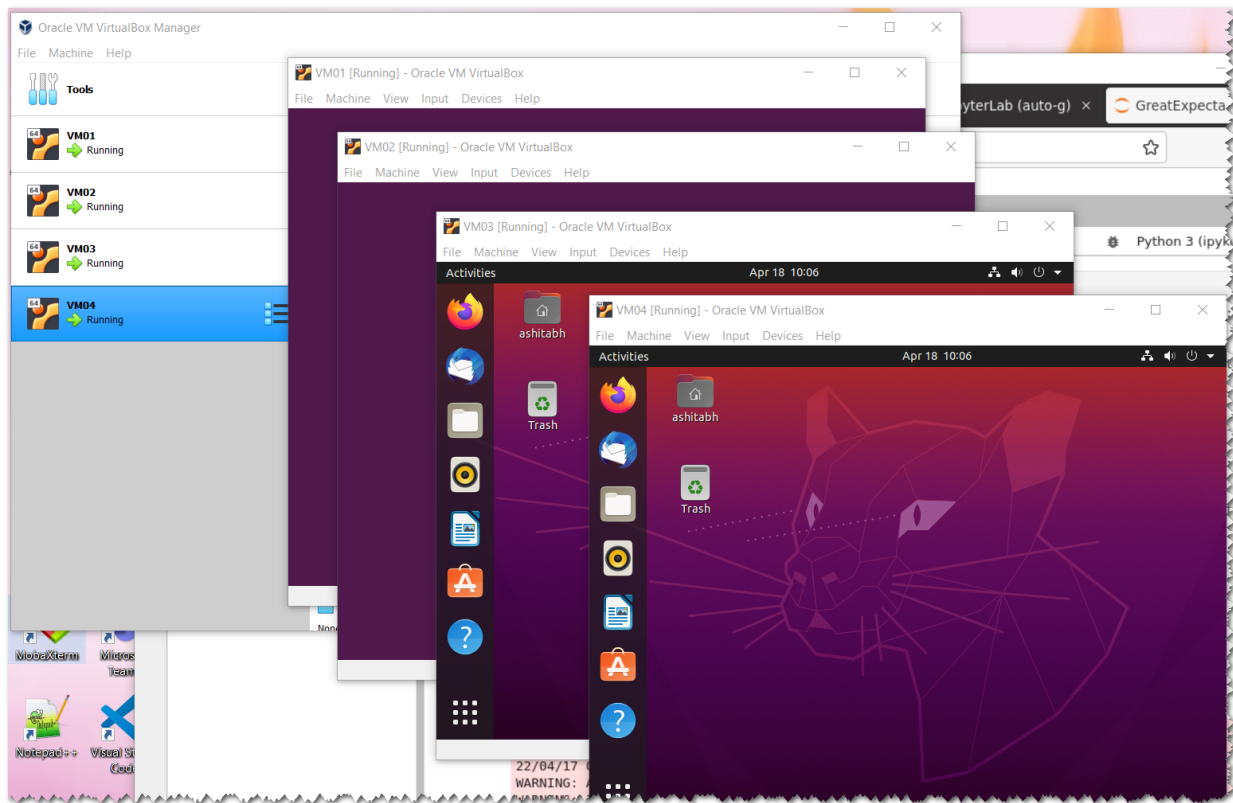
Type: string

**Properties**

**Example Values**

## 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

```
root@VM01:/home/ashitabh# dask-scheduler --host 192.168.56.10 --bokeh-port 8787
/usr/local/lib/python3.8/dist-packages/distributed/cli/dask_scheduler.py:142: UserWarning: The --bokeh-port flag has been renamed to --dashboard-address. Consider adding '--dashboard-address :8787'
  warnings.warn(
2022-04-18 11:07:54,980 - distributed.scheduler - INFO - -----
2022-04-18 11:08:00,622 - distributed.http.proxy - INFO - To route to workers diagnostics web server please install jupyter-server-proxy: python
-m pip install jupyter-server-proxy
2022-04-18 11:08:00,762 - distributed.scheduler - INFO - -----
2022-04-18 11:08:00,765 - distributed.scheduler - INFO - Clear task state
2022-04-18 11:08:00,768 - distributed.scheduler - INFO - Scheduler at: tcp://192.168.56.10:8786
2022-04-18 11:08:00,771 - distributed.scheduler - INFO - dashboard at: 192.168.56.10:8787
```

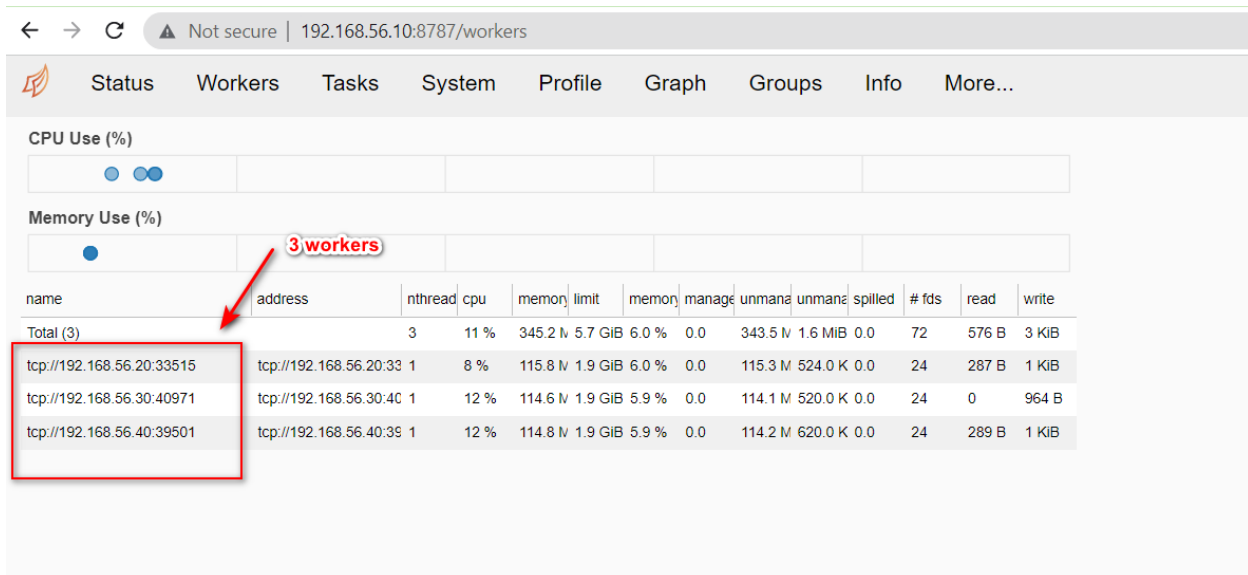
### Dask Worker on other 3 VMs ( VM02,VM03,VM04)

```

root@VM01:/home/ashitabh# start-worker.sh spark://192.168.56.10:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-VM01.out
root@VM01:/home/ashitabh# dask-worker tcp://192.168.56.10:8786
2022-04-18 11:10:15,735 - distributed.nanny - INFO - Start Nanny at: 'tcp://192.168.56.20:37631'
2022-04-18 11:10:17,395 - distributed.diskutils - INFO - Found stale lock file and directory '/home/ashitabh/dask-worker-space/worker-orrrpskqe',
purgin
2022-04-18 11:11:00,159 - distributed.worker - INFO - Start worker at: tcp://192.168.56.20:33515
2022-04-18 11:11:00,171 - distributed.worker - INFO - Listening to: tcp://192.168.56.20:33515
2022-04-18 11:11:00,175 - distributed.worker - INFO - dashboard at: 192.168.56.20:41605
2022-04-18 11:11:00,175 - distributed.worker - INFO - Waiting to connect to: tcp://192.168.56.10:8786
2022-04-18 11:11:00,175 - distributed.worker - INFO -
-----
2022-04-18 11:11:00,176 - distributed.worker - INFO - Threads: 1
2022-04-18 11:11:00,177 - distributed.worker - INFO - Memory: 1.88 GiB
2022-04-18 11:11:00,177 - distributed.worker - INFO - Local Directory: /home/ashitabh/dask-worker-space/worker-68ijm_je
2022-04-18 11:11:00,177 - distributed.worker - INFO -
-----
2022-04-18 11:11:00,543 - distributed.worker - INFO - Registered to: tcp://192.168.56.10:8786
2022-04-18 11:11:00,557 - distributed.worker - INFO -
-----
2022-04-18 11:11:00,598 - distributed.core - INFO - Starting established connection

```

## 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

```

root@VM01:/home/ashitabh# start-master.sh --ip 192.168.56.10
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark-root-org.apache.spark.deploy.master.Master-1-VM01.out
root@VM01:/home/ashitabh#

```

### Worker node

```


root@VM01:/home/ashitabh# start-worker.sh spark://192.168.56.10:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-VM01.out
failed to launch: nice -n 0 /opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker --webui-port 8081 spark://192.168.56.10:7077
full log in /opt/spark/logs/spark-root-org.apache.spark.deploy.worker.Worker-1-VM01.out
root@VM01:/home/ashitabh#

```



# Spark Master UI

← → ↺ ⚠ Not secure | 192.168.56.10:8080

 3.2.1

Spark Master at spark://192.168.56.10:7077

URL: spark://192.168.56.10:7077

Alive Workers: 3

Cores in use: 3 Total, 0 Used

Memory in use: 3.0 GiB Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

Status: ALIVE

3 Worker Nodes

▼ Workers (3)

Worker Id	Address	State	Cores	Memory	Resources
<a href="#">worker-20220418105639-192.168.56.20-45763</a>	192.168.56.20:45763	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
<a href="#">worker-20220418105716-192.168.56.30-44271</a>	192.168.56.30:44271	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	
<a href="#">worker-20220418105754-192.168.56.40-36559</a>	192.168.56.40:36559	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

▼ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

## Jupyter notebooks for connecting to Dask

DataprepEDA.ipynb

GreatExpectationsSpark.ipynb X

+

✕

📄

📄

▶

■

↺

▶▶

Code

▼

[2]:

```
from dask.distributed import Client
c = Client('192.168.56.10:8786')
```

← Connect to DASK

```
/usr/local/lib/python3.8/dist-packages/distributed/client.py:1287: VersionMismatchWarning: Mismatched versions found

+-----+-----+-----+-----+
| Package | client | scheduler | workers |
+-----+-----+-----+-----+
| dask    | 2021.12.0 | 2021.12.0 | {'2022.04.0', '2021.12.0'} |
+-----+-----+-----+-----+

warnings.warn(version_module.VersionMismatchWarning(msg[0]["warning"])))
```

[3]:

c

[3]:

Client

Client-13b62b53-bedb-11ec-891a-81d41130bfa2

Connection method: Direct

Dashboard: <http://192.168.56.10:8787/status>

▼ Scheduler Info

Scheduler

Scheduler-f007f620-37cf-408e-b011-b0e17e6ffa19

Comm: tcp://192.168.56.10:8786

Dashboard: <http://192.168.56.10:8787/status>

Started: 9 minutes ago

Workers: 3

Total threads: 3

Total memory: 5.65 GiB

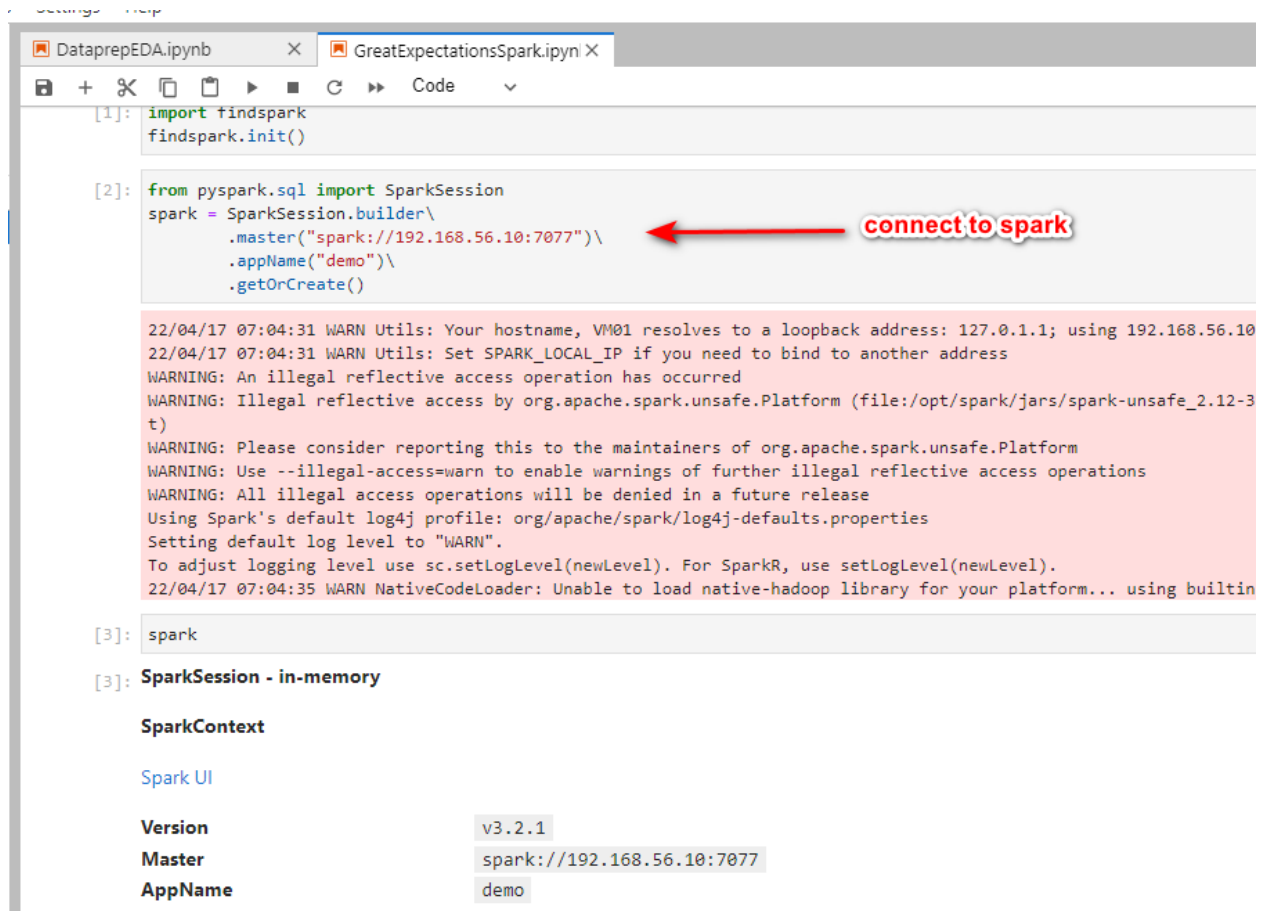
▼ Workers

▶ Worker: tcp://192.168.56.20:33515

▶ Worker: tcp://192.168.56.30:40971

▶ Worker: tcp://192.168.56.40:39501

## Jupyter notebooks for connecting to Spark



```
[1]: import findspark
findspark.init()

[2]: from pyspark.sql import SparkSession
spark = SparkSession.builder\
    .master("spark://192.168.56.10:7077")\
    .appName("demo")\
    .getOrCreate()

22/04/17 07:04:31 WARN Utils: Your hostname, VM01 resolves to a loopback address: 127.0.1.1; using 192.168.56.10
22/04/17 07:04:31 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/opt/spark/jars/spark-unsafe_2.12-3
t)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/04/17 07:04:35 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin

[3]: spark

[3]: SparkSession - in-memory

SparkContext

Spark UI

Version      v3.2.1
Master       spark://192.168.56.10:7077
AppName      demo
```

## 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-TPlKqCubh2m9mDa-#scrollTo=6e7uG2\\_N2zyx](https://colab.research.google.com/drive/1kBCWoyagmfuwny-TPlKqCubh2m9mDa-#scrollTo=6e7uG2_N2zyx)