**Pre-Processing of Job Dataset**

1. The first step of analyzing data set with basic exploration of the dataset. This done by simply checking the tops rows (using the head() command) and the shape of the dataset. From the analysis we can see that there are 1055500 rows and 6 columns.
2. After that to observe the statistical properties of the each column the describe method is used. Simple describe function return the statistical properties of the all of the numerical variables and the describe function with include=”o” argument returns the statistical properties of the all of the categorical variables. Here the start\_time and end\_time are the numerical variables and job\_name, inst\_id, user and status are the categorical variables
3. Then we check the meta data of the dataset by using the info method. This function returns the total number of non-null values present in each columns and the data type of each column.
4. Now I check the missing values of the dataset by using the isnull().sum() functions this returns the total number of null values present in each column.

**ANALYSIS:** Here a thing to note is that there are missing values in the start and end time, and the job with waiting, running status, will logically don't have starting and ending time. and removing the missing values of start time and end time will remove the class "waiting" from the status as it have missing values in both column. But according our analysis we are focused on reducing the execution time which can only be calculated when we have both of the values (i.e. start time and end time). So regardless of the impact of removing missing values from the start and end time on the dataset size, we will remove them. As the data without the start and end time is not important in our analysis.

This same will also be applied to all of the dataset which contain start time and end time

1. Dropna function is used to remove the missing values
2. Check the datatypes of the each column
3. Check the total number of unique values using the nunique function.
4. Then check the values of each column using the frequency table. This table is generated using the value\_counts function which return all of the uniques values with their total number of counts. Through this, we are more able to better understand the values of each column.
5. Feature Encoding: here we will encode the all of the categorical variable by replacing it with the numerical values. Here we can see that there are 3 values in the status column so we replace the terminated with 0, failed with 1 and running with 2. Here we did not encode the categorical ID, such as job\_name, inst\_id and user. As they will not be included in the analysis and are unique to their respective data.

**Pre-Processing of Task Dataset**

1. The first step of analyzing data set with basic exploration of the dataset. This done by simply checking the tops rows (using the head() command) and the shape of the dataset. From the analysis we can see that there are 1261049 rows and 10 columns.
2. Then we check the meta data of the dataset by using the info method. This function returns the total number of non-null values present in each columns and the data type of each column.
3. After that to observe the statistical properties of the each column the describe method is used. Simple describe function return the statistical properties of the all of the numerical variables and the describe function with include=”o” argument returns the statistical properties of the all of the categorical variables. Here the inst\_num, plan\_cpu, plan\_mem, plan\_gpu, start\_time and end\_time are the numerical variables and job\_name, task\_name, status and gpu\_type are the categorical variables
4. Now I check the missing values of the dataset by using the isnull().sum() functions this returns the total number of null values present in each column.

**ANALYSIS:** Here beside the start and end time, we do have missing values in other columns. we will simply remove the rows with missing values are the percentage of missing value are all below 30. so it will not greatly impact the dataset size.

This same will also be applied to all of the dataset which contain start time and end time

1. Dropna function is used to remove the missing values
2. Check the datatypes of the each column
3. Check the total number of unique values using the nunique function.
4. Then check the values of each column using the frequency table. This table is generated using the value\_counts function which return all of the uniques values with their total number of counts. Through this, we are more able to better understand the values of each column.

**ANALYSIS:** Here it can be seen that there are 3 unique values present in the data that are terminated, failed and running. The task\_name have 12 unique values and gpu\_type have 5 unique values

1. Feature Encoding: here we will encode the all of the categorical variable by replacing it with the numerical values. Here we can see that there are also 3 values in the status column so we replace the terminated with 0, failed with 1 and running with 2. Here we need to causation about the values that we assign. And try to stay consistent with these values. Like if the assign 0 to terminated then for all of the datasets the 0 will only be assigned to terminated in the status. In this way we will be able to merge the data without any error. In the same way the task\_name and gpu type are encoded. Here we did not encode the categorical ID, as they will not be included in the analysis and are unique to their respective data.

**Pre-Processing of Instance Dataset**

1. The first step of analyzing data set with basic exploration of the dataset. This done by simply checking the tops rows (using the head() command) and the shape of the dataset. From the analysis we can see that there are 7522001 rows and 9columns.
2. Then we check the meta data of the dataset by using the info method. This function returns the total number of non-null values present in each columns and the data type of each column.
3. After that to observe the statistical properties of the each column the describe method is used. Simple describe function return the statistical properties of the all of the numerical variables and the describe function with include=”o” argument returns the statistical properties of the all of the categorical variables. Here the start\_time and end\_time are the numerical variables and job\_name, task\_name, inst\_id, worker\_name, inst\_name, status and machine are the categorical variables
4. Now I check the missing values of the dataset by using the isnull().sum() functions this returns the total number of null values present in each column. And also calculate the percentage of missing values.

**ANALYSIS:** beside the start and end time we have missing values in other columns. and the percentage of missing values are below 5% so we will remove them.

1. Dropna function is used to remove the missing values
2. Check the datatypes of the each column
3. Check the total number of unique values using the nunique function.
4. Then check the values of each column using the frequency table. This table is generated using the value\_counts function which return all of the uniques values with their total number of counts. Through this, we are more able to better understand the values of each column.
5. Feature Encoding: here we will encode the all of the categorical variable by replacing it with the numerical values. Here we can see that there are 2 values in the status column so we replace the terminated with 0, and failed with 1. Here we need to causation about the values that we assign. And try to stay consistent with these values. Like if the assign 0 to terminated then for all of the datasets the 0 will only be assigned to terminated in the status. In this way we will be able to merge the data without any error. In the same way the task\_name is encoded. While maintaining the consistency between the values. Here we did not encode the categorical ID, as they will not be included in the analysis and are unique to their respective data.

**Pre-Processing of Sensor Dataset**

1. The first step of analyzing data set with basic exploration of the dataset. This done by simply checking the tops rows (using the head() command) and the shape of the dataset. From the analysis we can see that there are 3033231rows and 16 columns.
2. Then we check the meta data of the dataset by using the info method. This function returns the total number of non-null values present in each columns and the data type of each column.
3. After that to observe the statistical properties of the each column the describe method is used. Simple describe function return the statistical properties of the all of the numerical variables and the describe function with include=”o” argument returns the statistical properties of the all of the categorical variables. Here the cpu\_usage, gpu\_wrk\_util, avg\_mem, ma\_mem, avg\_gpu\_wrk\_mem, max\_gpu\_wrk\_mem, read, write, read\_count, and write\_count are the numerical variables and job\_name, task\_name, worker\_name, inst\_id, machine and gpu\_name are the categorical variables
4. Now I check the missing values of the dataset by using the isnull().sum() functions this returns the total number of null values present in each column. And also calculate the percentage of missing values.

**ANALYSIS:** the number of missing values are less then 1% so we will simply remove all of the missing values

1. Dropna function is used to remove the missing values
2. Check the datatypes of the each column
3. Check the total number of unique values using the nunique function.
4. Then check the values of each column using the frequency table. This table is generated using the value\_counts function which return all of the uniques values with their total number of counts. Through this, we are more able to better understand the values of each column.
5. Feature Encoding: here we will encode the all of the categorical variable by replacing it with the numerical values. As gpu\_name is the columns that does not appeared in the previous table so we will encode them directly. But the task name have appeared 2 time in the previous data that is task and instance dataset. We will consider the pervious values and assigned the values based on them accordingly. And 'TransformGraph','LeadingWorker' did not appeared before so we will assign the new values to them.

**Pre-Processing of Group Dataset**

1. The first step of analyzing data set with basic exploration of the dataset. This done by simply checking the tops rows (using the head() command) and the shape of the dataset. From the analysis we can see that there are 1055031 rows and 5 columns.
2. Then we check the Meta data of the dataset by using the info method. This function returns the total number of non-null values present in each columns and the data type of each column.
3. After that to observe the statistical properties of the each column the describe method is used. Simple describe function return the statistical properties of the all of the numerical variables and the describe function with include=”o” argument returns the statistical properties of the all of the categorical variables. Here all of the columns are the categorical variables
4. Now I check the missing values of the dataset by using the isnull().sum() functions this returns the total number of null values present in each column. And also calculate the percentage of missing values.

**ANALYSIS:** Here the missing values of gpu\_type\_spec is 98% and missing values of workload is 90%. these columns cannot add value to any analysis due to the lack of data. so it is better to remove these columns.

1. Dropna function is used to remove the columns with missing values
2. Check the datatypes of the each column
3. Check the total number of unique values using the nunique function.
4. Then check the values of each column using the frequency table. This table is generated using the value\_counts function which return all of the uniques values with their total number of counts. Through this, we are more able to better understand the values of each column.
5. Feature Encoding: here we will encode the all of the categorical variable by replacing it with the numerical values. Here we will encode the group column, as this column contain more than 1000 values we cannot manually replace the values, so we will use the label encoder class to replace the values with numerical values.

**Pre-Processing of Spec Dataset**

1. The first step of analyzing data set with basic exploration of the dataset. This done by simply checking the tops rows (using the head() command) and the shape of the dataset. From the analysis we can see that there are 1896 rows and 5 columns.
2. Then we check the meta data of the dataset by using the info method. This function returns the total number of non-null values present in each columns and the data type of each column.
3. After that to observe the statistical properties of the each column the describe method is used. Simple describe function return the statistical properties of the all of the numerical variables and the describe function with include=”o” argument returns the statistical properties of the all of the categorical variables. Here cap\_cpu, cap\_mem and cap\_gpu are the numerical variables and machine and gpu\_type are the categorical variables
4. Now I check the missing values of the dataset by using the isnull().sum() functions this returns the total number of null values present in each column. And also calculate the percentage of missing values.

**ANALYSIS:** there are zero missing values

1. Check the datatypes of the each column
2. Check the total number of unique values using the nunique function.
3. Then check the values of each column using the frequency table. This table is generated using the value\_counts function which return all of the uniques values with their total number of counts. Through this, we are more able to better understand the values of each column.
4. Feature Encoding: here we will encode the all of the categorical variable by replacing it with the numerical values. the gpu typeshave appeared in the previous dataset, so we will consider the pervious values and assigned the values based on them accordingly. And ‘CPU’ did not appeared before so we will assign the new value to it.

**Pre-Processing of Metric Dataset**

1. The first step of analyzing data set with basic exploration of the dataset. This done by simply checking the tops rows (using the head() command) and the shape of the dataset. From the analysis we can see that there are 2009422 rows and 12 columns.
2. Then we check the Meta data of the dataset by using the info method. This function returns the total number of non-null values present in each columns and the data type of each column.
3. After that to observe the statistical properties of the each column the describe method is used. Simple describe function return the statistical properties of the all of the numerical variables and the describe function with include=”o” argument returns the statistical properties of the all of the categorical variables. Here start\_time, end\_time, machine\_cpu\_iowait, machine\_cpu\_kernel, machine\_cpu\_user, machine\_gpu, machine\_load\_1, macine\_net\_recieve, machine\_num\_worker, and machine cpu are the numerical variables and worker\_anem and machine are the categorical variables
4. Now I check the missing values of the dataset by using the isnull().sum() functions this returns the total number of null values present in each column. And also calculate the percentage of missing values.

**ANALYSIS:** Here 2 columns that is machine\_cpu\_iowait and machine\_cpu have more then 50% missing values, so removing the missing rows of these columns will greatly impact the size of the data. more then half of the data will be removed. so its better to remove these 2 columns. and remove the missing rows of the other columns

1. Dropna function is used to remove the columns with missing values and missing values
2. Check the datatypes of the each column
3. Check the total number of unique values using the nunique function.
4. Then check the values of each column using the frequency table. This table is generated using the value\_counts function which return all of the uniques values with their total number of counts. Through this, we are more able to better understand the values of each column.
5. Feature Encoding: here since there are only Ids that are categorical by its nature, there is nothing to encode.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Machine specification** | **Machine metric** | **Task** | **Job** | **Instance** | **Sensor** | **Group** |
| **Number of records** | 1896 | 2009422 | 1261049 | 1055500 | 7522001 | 3033231 | 1055030 |
| **Number of missing data** | 0 | worker\_name 0  machine 0  start\_time 0  end\_time 0  machine\_cpu\_iowait 1090517  machine\_cpu\_kernel 280687  machine\_cpu\_usr 232094  machine\_gpu 241493  machine\_load\_1 268543  machine\_net\_receive 218147  machine\_num\_worker295257  machine\_cpu 1125894 | job\_name = 0 task\_name = 0 inst\_num = 130 status = 0 start\_time = 3714 end\_time = 349561 plan\_cpu = 18454 plan\_mem = 18454 plan\_gpu = 223965 gpu\_type = 217738 | job\_name = 0  inst\_id = 0  user = 0  status = 0  start\_time =3663  end\_time=296419 | job\_name 0  task\_name 0  inst\_name 0  worker\_name 331057  inst\_id 0  status 0  start\_time 1773468  end\_time 2267582  machine 331057 | job\_name 0  task\_name 0  worker\_name 0  inst\_id 0  machine 0  gpu\_name 0  cpu\_usage 5829  gpu\_wrk\_util 0  avg\_mem 1217  max\_mem 0  avg\_gpu\_wrk\_mem 0  max\_gpu\_wrk\_mem 0  read 3  write 3  read\_count 3  write\_count 3 | inst\_id 0  user 0  group 0 |
| **The attributes** | machine,gpu\_type,cap\_cpu,cap\_mem,cap\_gpu | worker\_name,machine,start\_time,end\_time,machine\_cpu\_iowait,machine\_cpu\_kernel,machine\_cpu\_usr,machine\_gpu,machine\_load\_1,machine\_net\_receive,machine\_num\_worker,machine\_cpu | job\_name,task\_name,inst\_num,status,start\_time,end\_time,plan\_cpu,plan\_mem,plan\_gpu,gpu\_type | job\_name,inst\_id, user, status, start\_time, end\_time | job\_name, task\_name, inst\_name, worker\_name, inst\_id, status, start\_time, end\_time, machine | job\_name, task\_name, worker\_name, inst\_id, machine, gpu\_name, cpu\_usage, gpu\_wrk\_util, avg\_mem, max\_mem, avg\_gpu\_wrk\_mem,  max\_gpu\_wrk\_mem, read, write, read\_count, write\_count | inst\_id, user, group |
| **Data analysis rules (missing, duplicate, no more one inst)** | Nothing | * 1. Drop any task is not finished =based on end time not NaN   2. Drop duplicate rows (worker name= not duplicate)   3. Drop any worker needs more than 50% of GPU = to avoid sensitive resources usage | * 1. Drop any task is not finished =based on end time not NaN, or status = not terminated   2. Drop duplicate rows (job name= not duplicate)   3. Drop any task has more than one inst = working on more than one machine at same time   4. Drop any task needs more than 50% of GPU = to avoid sensitive resources usage | 1. Drop any task is not finished =based on end time not NaN, or status = not terminated 2. Drop duplicate rows (job name= not duplicate) | 1. Drop any task is not finished =based on end time not NaN, or status = not terminated 2. Drop duplicate rows (job name= not duplicate) 3. Drop any task has more than one inst = working on more than one machine at same time | 1. Drop any task is not finished =based on end time not NaN, or status = not terminated 2. Drop duplicate rows (job name= not duplicate) 3. Drop any task has more than one inst = working on more than one machine at same time | Nothing |
| **The new number of records** | 1897 | 1406189 | 737136 | 759036 | 4758812 | 3027394 | 1055030 |
| **After assumptions (conditions) analysis - missing data** | 0 | worker\_name 0  machine 0  start\_time 0  end\_time 0  machine\_cpu\_iowait 0  machine\_cpu\_kernel 0  machine\_cpu\_usr 0  machine\_gpu 0  machine\_load\_1 0  machine\_net\_receive 0  machine\_num\_worker 0  machine\_cpu 0 | job\_name 0  task\_name 0  inst\_num 0  status 0  start\_time 0  end\_time 0  plan\_cpu 0  plan\_mem 0  plan\_gpu 0  gpu\_type  0  Aa solution: Could I plan put gpu=0  Gpu type= cpu | job\_name 0  inst\_id 0  user 0  status 0  start\_time 0  end\_time 0 | job\_name 0  task\_name 0  inst\_name 0  worker\_name 0  inst\_id 0  status 0  start\_time 0  end\_time 0  machine 0 | job\_name 0  task\_name 0  worker\_name 0  inst\_id 0  machine 0  gpu\_name 0  cpu\_usage 0  gpu\_wrk\_util 0  avg\_mem 0  max\_mem 0  avg\_gpu\_wrk\_mem 0  max\_gpu\_wrk\_mem 0  read 0  write 0  read\_count 0  write\_count 0 | 0 |
| **How many tasks in one node?** |  | 1168 worker per machine | 1 Instance and one task per each job | One Instance per each job | One instance and one task per each worker | One instance and one task per each worker | Average 737 instances per users |

**Merging Data Frames**

1. Merging start with metric and instance column these two columns are merged using all of the common columns that are, Common columns: end\_time, machine, start\_time, worker\_name. And the outer join is used as a method of merge. Outer join mean that merge all of the data with each other irrespective to whether it appear in other table or not. This types of method usually add the lot of null values in the dataset. So we will remove the missing values along with merging.

**ANALYSIS**: We see that machine\_cpu\_kernel, machine\_cpu\_usr, machine\_gpu, machine\_load\_1, machine\_net\_receive, and machine\_num\_worker have 70% values missing. so we will remove them and drop the duplicate values

1. Then we will merge the spec table with instance table using the common columns that is machine and outer join as a method of merge. And remove the missing values and duplicate values
2. Then we merge the group with the job columns using the same outer join method and remove the missing and duplicate values. The common columns are inst\_id, user.
3. Sensor is merge with the job table in the same way and the task is merge with the job using the same outer join method. The common columns are inst\_id, job\_name.
4. Task is merge with the job table using the common columns that are end\_time, start\_time, job\_name, status.
5. Now all of these tables are further merged with each other while making sure to remove those columns which have missing values more than 50% and for less missing values we remove those values and drop the duplicate values

Instance\_jobs

**Common columns: end\_time, inst\_id, inst\_name, job\_name, machine, start\_time, worker\_name, status, task\_name**

Metric\_instance

**Common columns: end\_time, machine, start\_time, worker\_name**

Metric

Instance

ig\_jobs

**Common columns: end\_time, inst\_id, job\_name, start\_time, status**

spec\_instance

**Common columns: machine**

Specification

Group\_jobs

**Common columns: inst\_id, user**

Group

igs\_jobs

**Common columns: end\_time, inst\_id, job\_name, start\_time, status, user**

Job

Sensor\_jobs

**Common columns: inst\_id, job\_name**

tigs\_jobs

**Common columns: end\_time, inst\_id, job\_name, start\_time, status, user, task\_name**

Sensor

task\_jobs

**Common columns: end\_time, start\_time, job\_name, status**

Task

|  |  |  |  |
| --- | --- | --- | --- |
| **Analysis on Merged Final Dataset (tigs\_jobs)** | | | |
| **Average number of instance per worker** | | | 1.0 |
| **Average number of instance per machine** | | | 278.4825 |
| **Average number of task working per machine** | | | 278.4825 |
| **Maximum number of task working on single machine** | | | 7532 |
| **How many task working on more than 0ne machine** | | | 11 |
| **How many jobs working on more than one machine** | | | 0 |
| **Average number of instance per task** | | | 43645.8181 |
| **Total number of instance per each task** | | **0 – tensorflow** | 324351 |
| **1 – worker** | 66577 |
| **2 – PyTorchWorker** | 57788 |
| **3 - xComputerWorker** | 18621 |
| **4 - evaluator** | 8248 |
| **5 - ps** | 2618 |
| **6 - TVMTuneMain** | 846 |
| **7 - OssToVoumeWorker** | 343 |
| **8 - OpenmpiWorker** | 464 |
| **9 - JupyterTask** | 224 |
| **10 – BladeMain** | 24 |
| **Average number task working per machine** | | | 278.482 |
| **Total number of task** | | | 11 |
| **Total number of task working on cpu** | | | 0 |
| **Total number of task working on gpu type** | | | 11 |
| **Total number of task working on gpu type** | **0 – MISC** | | 9 |
| **1 – T4** | | 11 |
| **2 – P100** | | 7 |
| **3 – V100** | | 9 |
| **4 – V100M32** | | 6 |
| **How many machines have same specifications** | | | 1724 |
| **How many task have same requirements** | | | 26 |
| **How many task start at same time on the same machine** | | | 48646 |
| **How many tasks end in the same time on the same machine** | | | 16715 |
| **Average execution time the task takes to run** | | | 5116.506 |
| **Average execution time when task working in gpu** | | | 8671.54 |
| **average execution time the task takes to run on each gpu type** | **0 – MISC** | | 5320.93 |
| **1 – T4** | | 2403.29 |
| **2 – P100** | | 8225.03 |
| **3 – V100** | | 12498.40 |
| **4 – V100M32** | | 14910.05 |
| **average Gpu memory utilize by the each task** | **0 – tensorflow** | | 5.70 |
| **1 – worker** | | 10.35 |
| **2 – PyTorchWorker** | | 16.78 |
| **3 - xComputerWorker** | | 34.01 |
| **4 - evaluator** | | 3.11 |
| **5 - ps** | | 0.04 |
| **6 - TVMTuneMain** | | 19.96 |
| **7 - OssToVoumeWorker** | | 0 |
| **8 - OpenmpiWorker** | | 6.15 |
| **9 - JupyterTask** | | 1.25 |
| **10 – BladeMain** | | 1.81 |
| **average CPU memory utilize by the each task** | **0 – tensorflow** | | 148.86 |
| **1 – worker** | | 111.053 |
| **2 – PyTorchWorker** | | 188.30 |
| **3 - xComputerWorker** | | 91.47 |
| **4 - evaluator** | | 126.87 |
| **5 - ps** | | 22.08 |
| **6 - TVMTuneMain** | | 270.99 |
| **7 - OssToVoumeWorker** | | 40.38 |
| **8 - OpenmpiWorker** | | 71.17 |
| **9 - JupyterTask** | | 23.91 |
| **10 - BladeMain** | | 128.28 |

**Remove Outlie**

Outliers occur in the numerical variable, so for that we will separate the numerical variable from other columns. outlier are detected through the box plots.

**ANALYSIS ON BOX PLOT**: From the above box plot we can see that we don't have outlier in the start time, end time and gpu name. and beside these columns all of the columns have outliers. so lets remove them through IQR method

To remove te outliers we uses the IQR method which replace the outlier values with their nearest IQP limits.

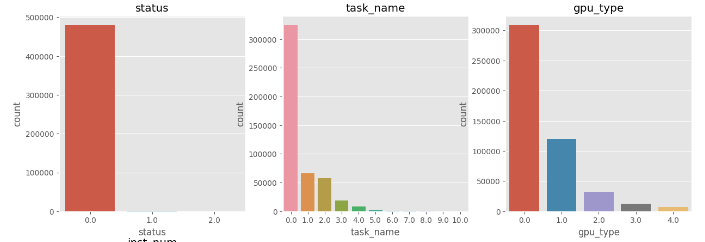
**Feature Engineering**

Here we have created 3 columns from the existing columns. that are the execution time, start\_date and end\_date. The execution time the duration of execution calculated by subtracting start time from the end time. And the start date and end date are the datetime format of the start time and end time respectively.

**Visualization**

**Bi-variate analysis**: Here we have observe the basic distribution of values in each column. For the categorical columns we uses the count plot and for the numerical columns we uses the distribution plot.

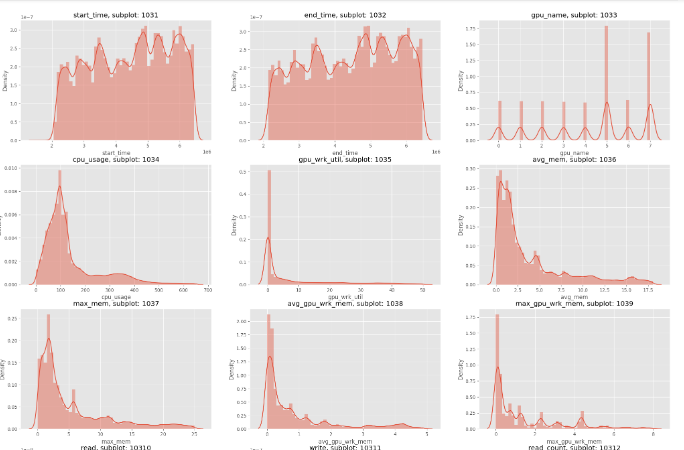
**ANALYSIS ON COUNT PLOT:**

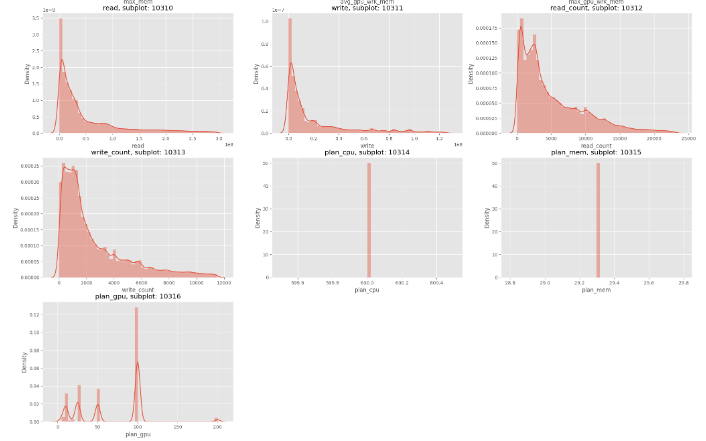


From the above plots we can see that:

* the status usually have 0 value that is the terminated
* In task\_name column 0 values have occured most
* in gpu\_type column the 0 value have occured most and 4 value occured least

**ANALYSIS ON DISTRIBUTION PLOTS:**



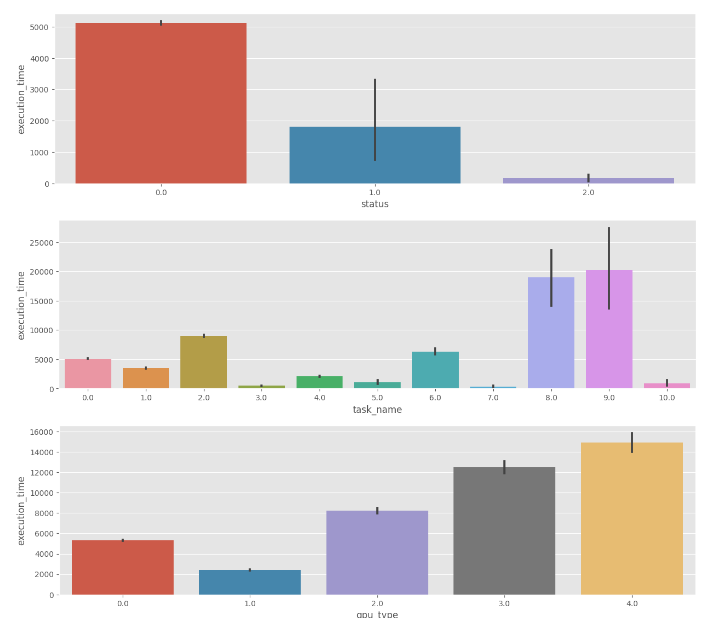


From the above plot we can observe how the values are distributed, here we can see that:

* start\_time and end\_ time have normal distribution
* write, write\_count, read, read\_count, max\_mem, avg\_gpu\_wrk\_mem, max\_gpu\_wrk\_mem, cpu\_usage, gpu\_wrk\_util and avg\_mem have postively skewed distribution and this mean that the median value is higher then the mean values.

**BI-variate analysis:** Here each categorical and numerical column is compared against the execution time to see how the values of each column varies with the execution time.

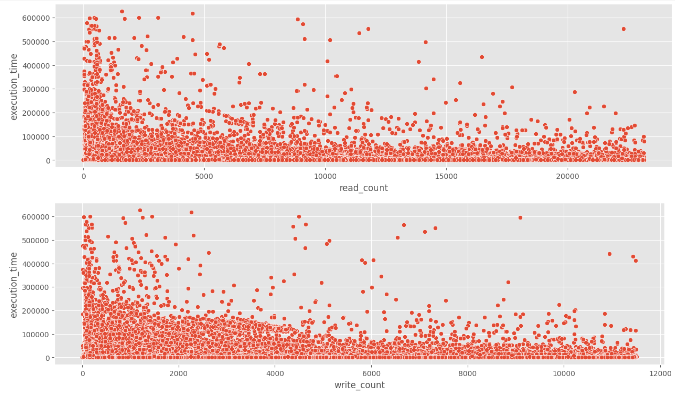
**ANALYSIS ON CATEGORICAL PLOTS:**

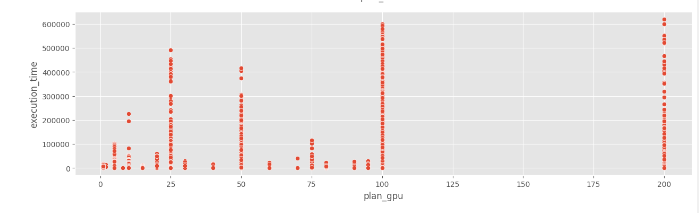


Here we can see that:

* the gpu\_type 4 have the highest execution time and 1 value have the lowest execution time.
* task name 8 and 9 have the highest execution time
* status zero have the highest execution time

**ANALYSIS ON NUMERICAL PLOTS:**





From the above plot we can see that:

* the execution time is high when the plan gpu is high
* as the value of write\_count, read\_count, write, read doesnot have very strong relationship with execution time but we can see that the as their increases the execution time slightly decreases

Correlation plots present the relationships between the values in terms of numbers.

**Calculating Episodes**

Episodes are the series of events that occur after one another. So to calculate the episode for the task scheduling task, we have taken the time in seconds as a measure of episode. So the each episode will last for 3 seconds. So to calculate the series of episodes which represent the data, we will create of time starting the in the minimum values of start time to the maximum value of the end time. By doing so we will be able to cover the entire time taken for execution. Since each episode will last for 3 seconds this mean that there will be an interval of 3 seconds between each episode. After splitting the data into a series of time we will assign a label/ number to each episode so that we can specifically see the total number of episodes.

After creating these 2 series of values (that is the split time in seconds and the respective episode number) we will structure the data according to it. in such a way that each row represent the each episode and its corresponding data represent the instance that is running during that episode.

After that we split the data into x and y, apply normalization and perform train test split.