National Level Online Hack-a-thon on Sustainable Energy by VIT University, Chennai

Contributed By:
M.Akshara(19BPS1061)
Sri Balaji J(19BEC1170)
R S Vimal(19BEE1045)

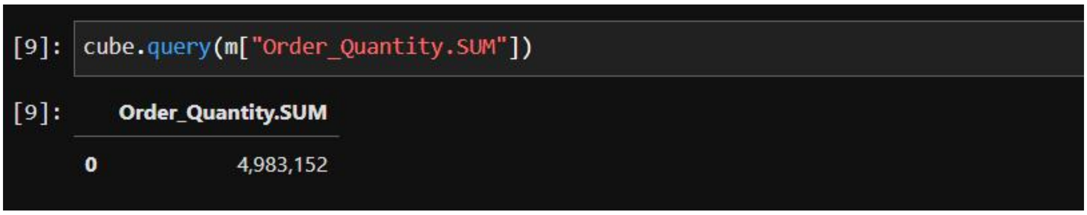
Problem statement: Jumpstart and create an application which can be used by energy management professionals using the below data set. https://github.com/jojo62000/Smarter_Decisions/blob/master/Chapter%203/Data/BO5341_IoTData.csv

Github link: https://github.com/Akshara2406/National-Level-Online-Hack-a-thon-on-Sustainable-Energy/tree/main

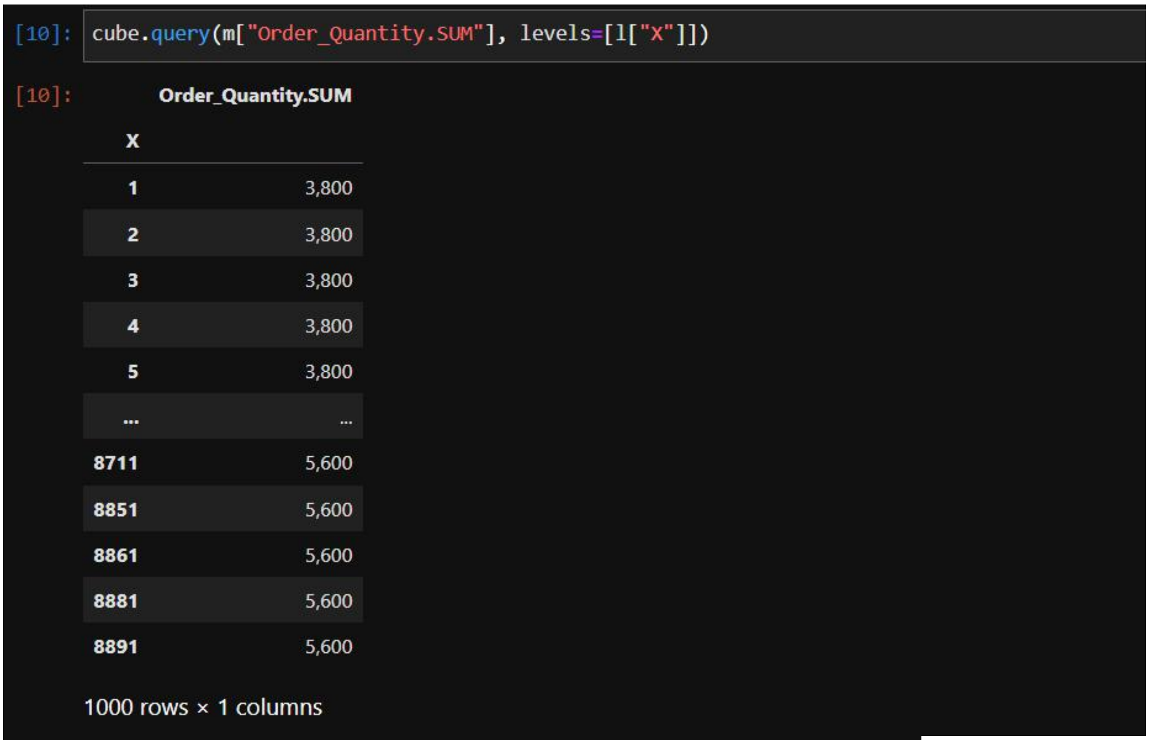
Kandi kit link :
https://kandi.openweaver.com/collections/utilities/hackathon-python-library(vit-ow-115

Outcome in the application :

1. OLAP operation of the data in front end (dice, slice, roll up/ down, filter)

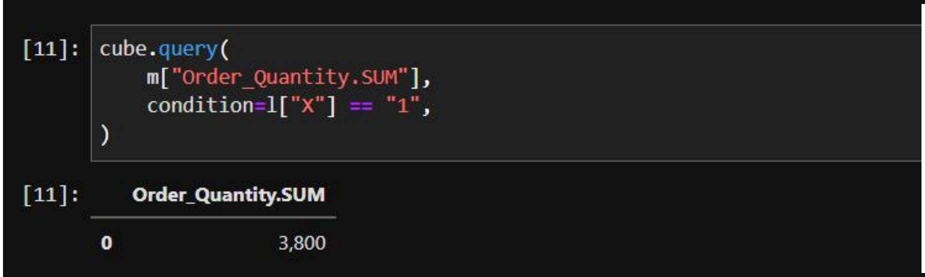


```
[9]: cube.query(m["Order_Quantity.SUM"])

[9]:    Order_Quantity.SUM
     0           4,983,152
```



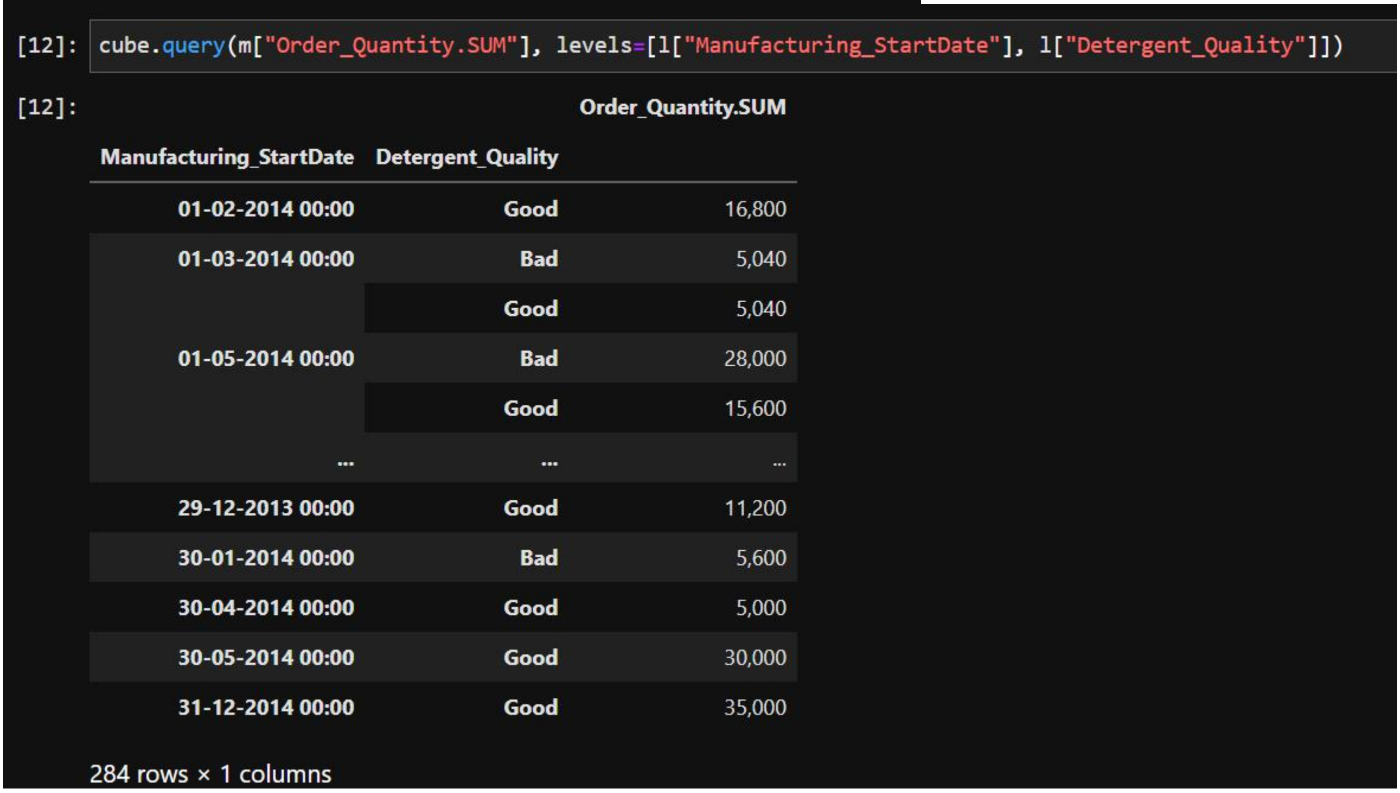
```
[10]: cube.query(m["Order_Quantity.SUM"], levels=[l["X"]])
```

```
[10]:      Order_Quantity.SUM
        X
        1               3,800
        2               3,800
        3               3,800
        4               3,800
        5               3,800
      ...                 ...
     8711               5,600
     8851               5,600
     8861               5,600
     8881               5,600
     8891               5,600

   1000 rows × 1 columns
```



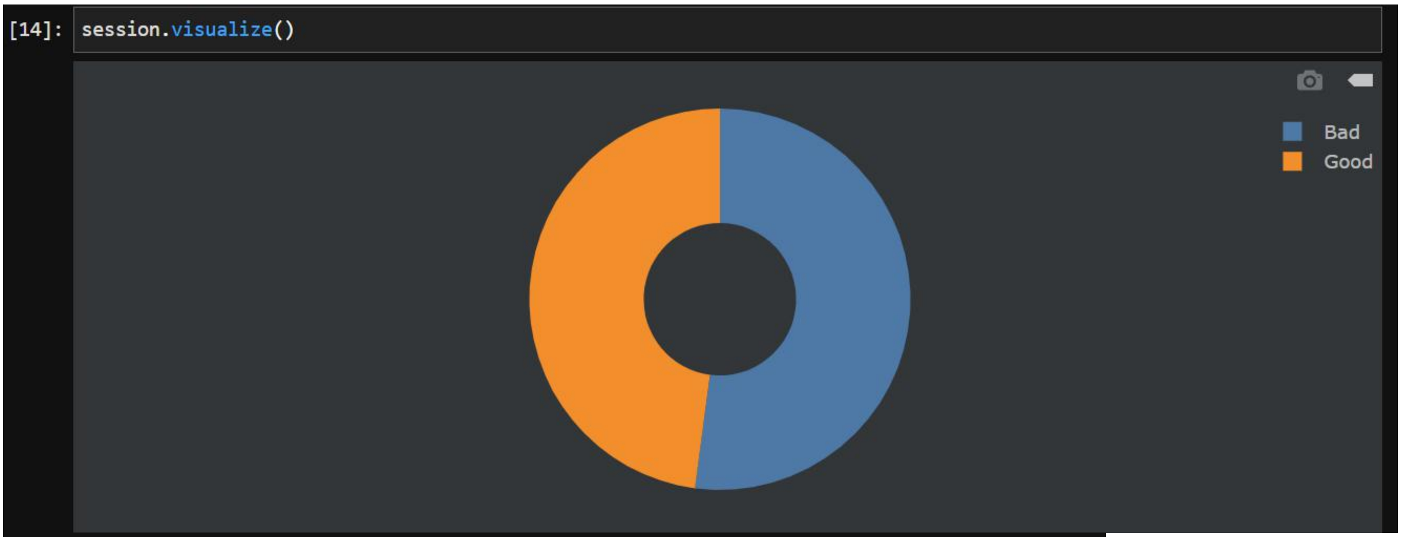
```
[11]: cube.query(
          m["Order_Quantity.SUM"],
          condition=l["X"] == "1",
      )
```

```
[11]:    Order_Quantity.SUM
       0              3,800
```

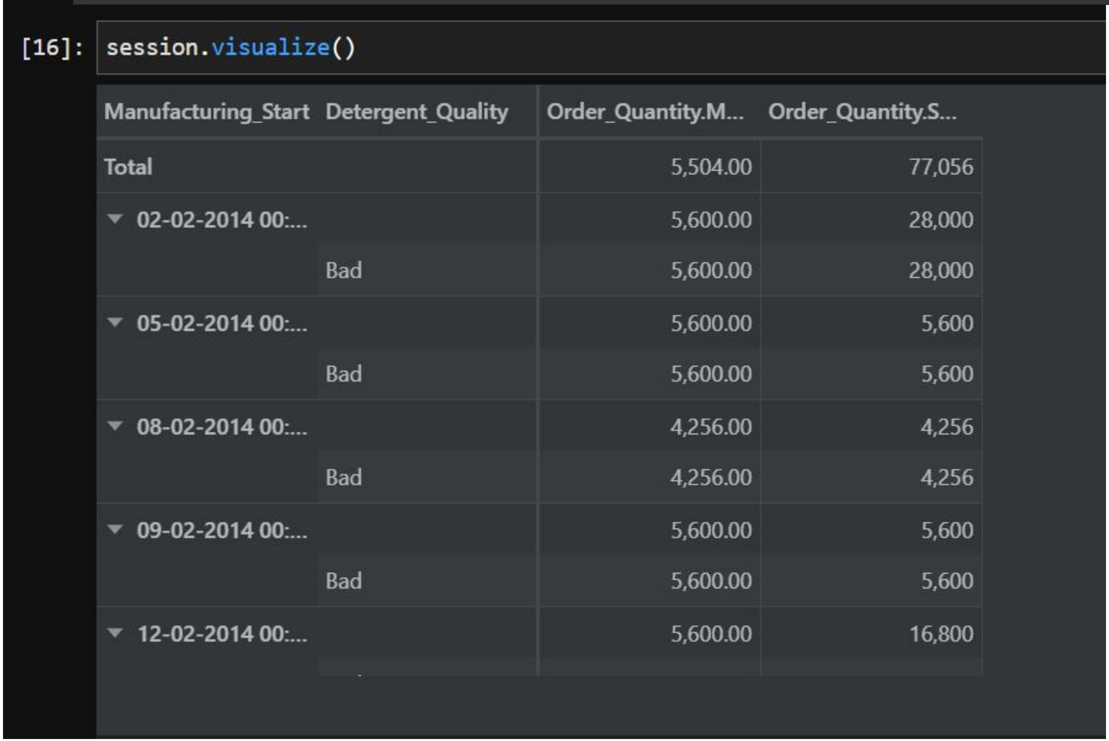


```
[12]: cube.query(m["Order_Quantity.SUM"], levels=[l["Manufacturing_StartDate"], l["Detergent_Quality"]])
```

```
[12]:                                              Order_Quantity.SUM
       Manufacturing_StartDate  Detergent_Quality
              01-02-2014 00:00               Good              16,800
              01-03-2014 00:00                Bad               5,040
                                             Good               5,040
              01-05-2014 00:00                Bad              28,000
                                             Good              15,600
                           ...                ...                 ...
              29-12-2013 00:00               Good              11,200
              30-01-2014 00:00                Bad               5,600
              30-04-2014 00:00               Good               5,000
              30-05-2014 00:00               Good              30,000
              31-12-2014 00:00               Good              35,000

      284 rows × 1 columns
```

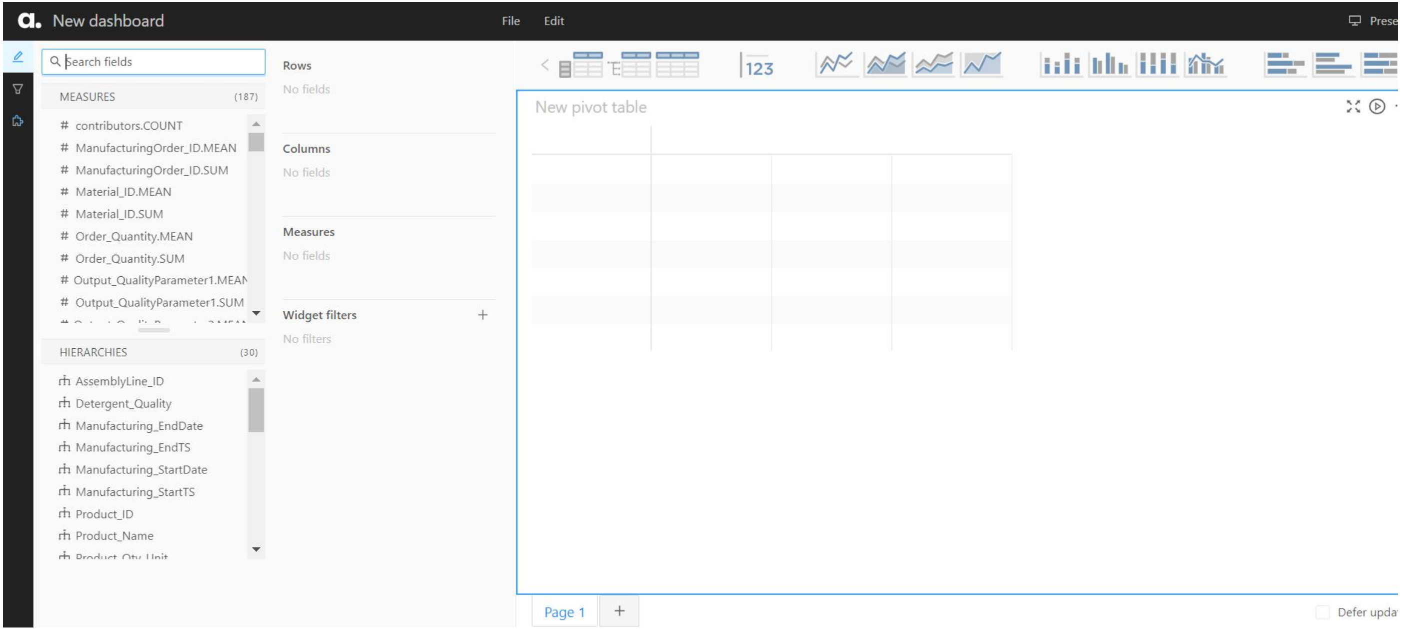


```
[14]: session.visualize()
```



```
[16]: session.visualize()
```

Manufacturing_Start	Detergent_Quality	Order_Quantity.M...	Order_Quantity.S...
Total		5,504.00	77,056
02-02-2014 00:...		5,600.00	28,000
	Bad	5,600.00	28,000
05-02-2014 00:...		5,600.00	5,600
	Bad	5,600.00	5,600
08-02-2014 00:...		4,256.00	4,256
	Bad	4,256.00	4,256
09-02-2014 00:...		5,600.00	5,600
	Bad	5,600.00	5,600
12-02-2014 00:...		5,600.00	16,800



MEASURES (187)

- contributors.COUNT
- ManufacturingOrder_ID.MEAN
- ManufacturingOrder_ID.SUM
- Material_ID.MEAN
- Material_ID.SUM
- Order_Quantity.MEAN
- Order_Quantity.SUM
- Output_QualityParameter1.MEAN
- Output_QualityParameter1.SUM

HIERARCHIES (30)

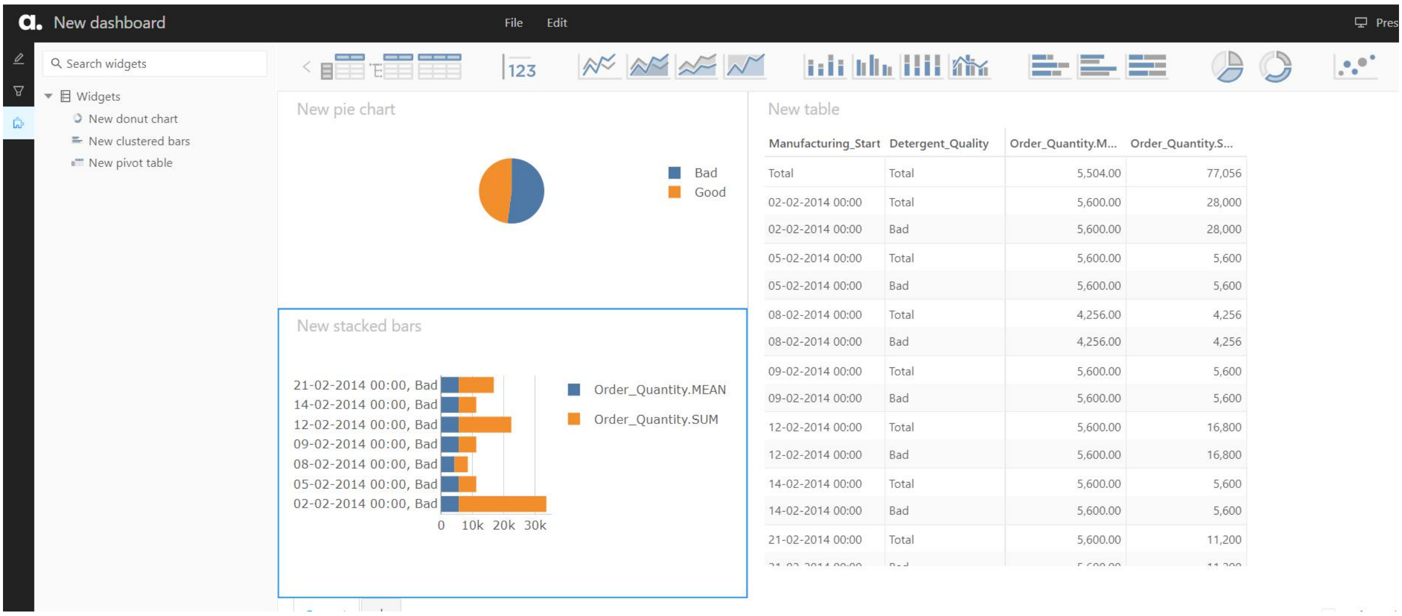
- AssemblyLine_ID
- Detergent_Quality
- Manufacturing_EndDate
- Manufacturing_EndTS
- Manufacturing_StartDate
- Manufacturing_StartTS
- Product_ID
- Product_Name
- Product_Qty_Unit

Rows
No fields

Columns
No fields

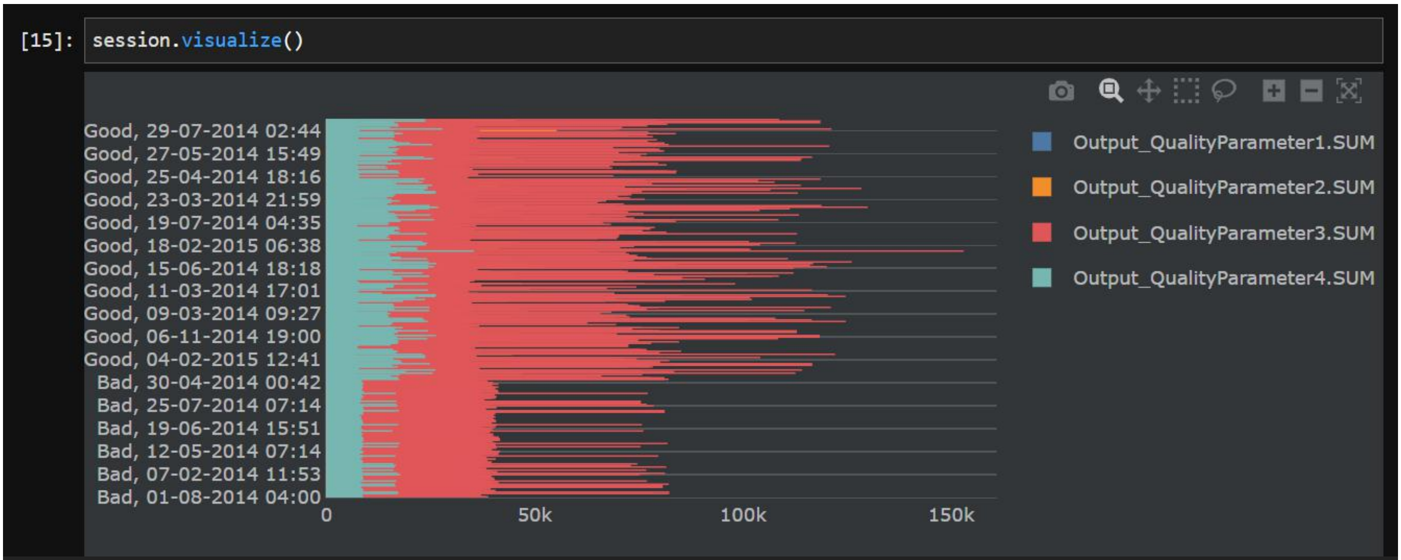
Measures
No fields

Widget filters +
No filters

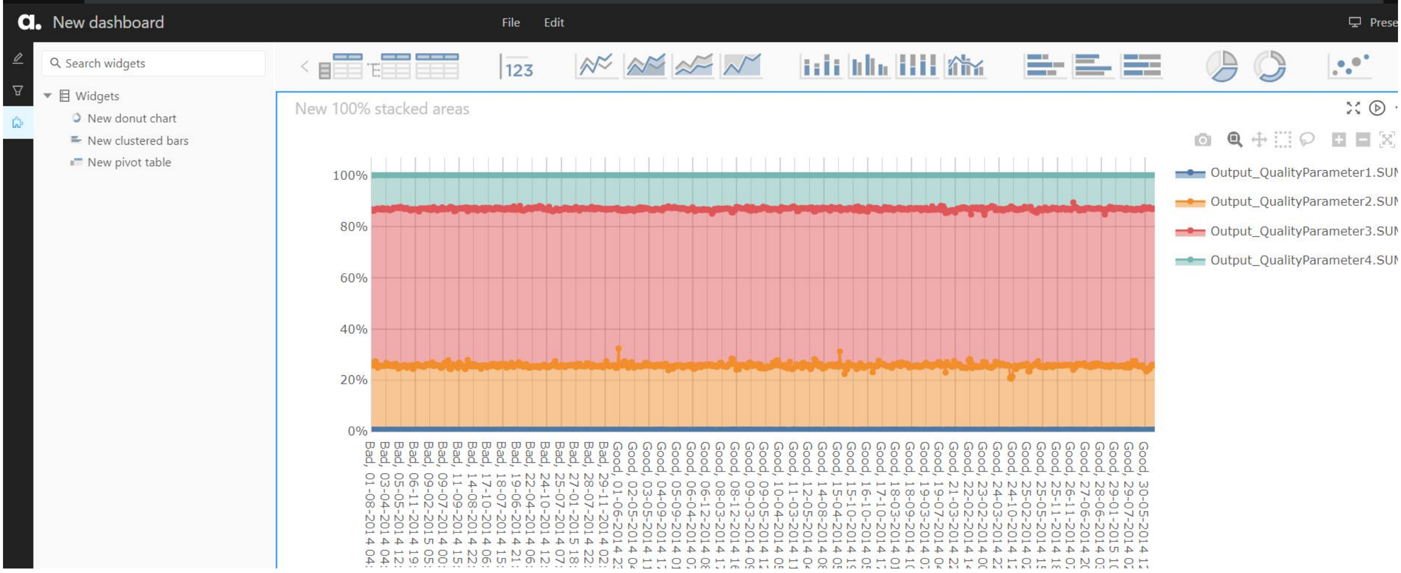


Manufacturing_Start	Detergent_Quality	Order_Quantity.M...	Order_Quantity.S...
Total	Total	5,504.00	77,056
02-02-2014 00:00	Total	5,600.00	28,000
02-02-2014 00:00	Bad	5,600.00	28,000
05-02-2014 00:00	Total	5,600.00	5,600
05-02-2014 00:00	Bad	5,600.00	5,600
08-02-2014 00:00	Total	4,256.00	4,256
08-02-2014 00:00	Bad	4,256.00	4,256
09-02-2014 00:00	Total	5,600.00	5,600
09-02-2014 00:00	Bad	5,600.00	5,600
12-02-2014 00:00	Total	5,600.00	16,800
12-02-2014 00:00	Bad	5,600.00	16,800
14-02-2014 00:00	Total	5,600.00	5,600
14-02-2014 00:00	Bad	5,600.00	5,600
21-02-2014 00:00	Total	5,600.00	11,200
21-02-2014 00:00	Bad	5,600.00	11,200

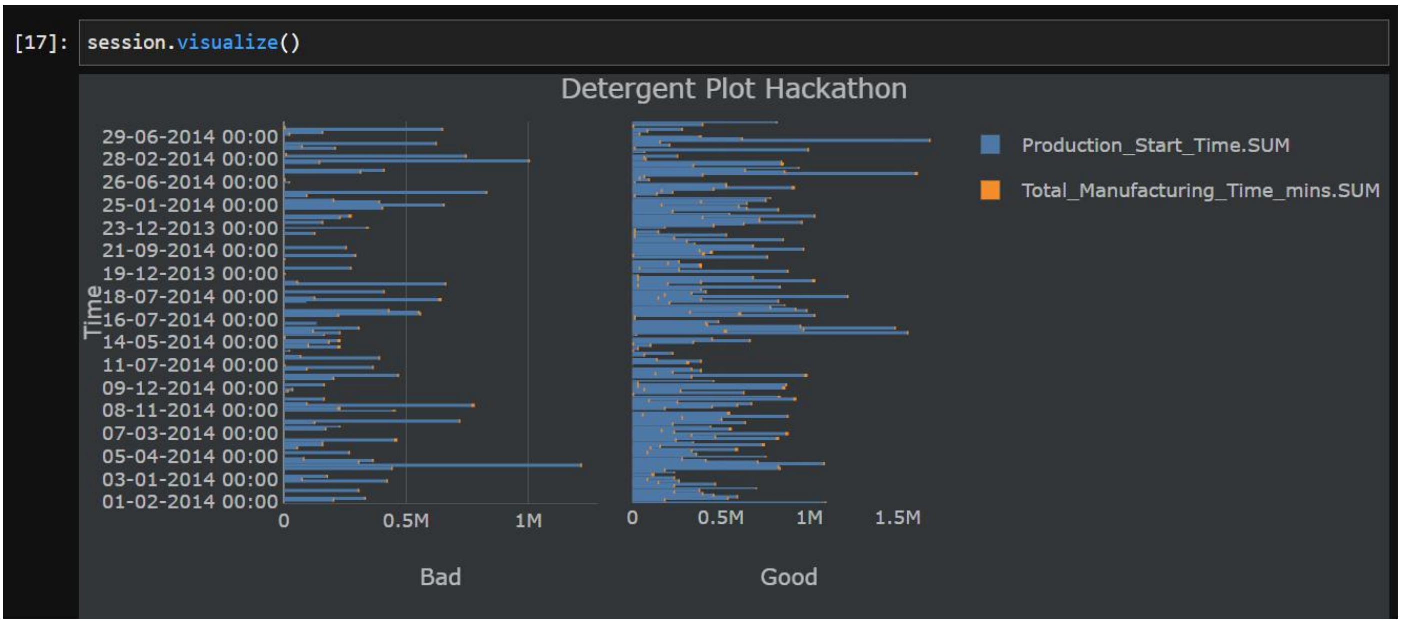
2. Ability to notify significant changes in the time series dataset imported in the tool



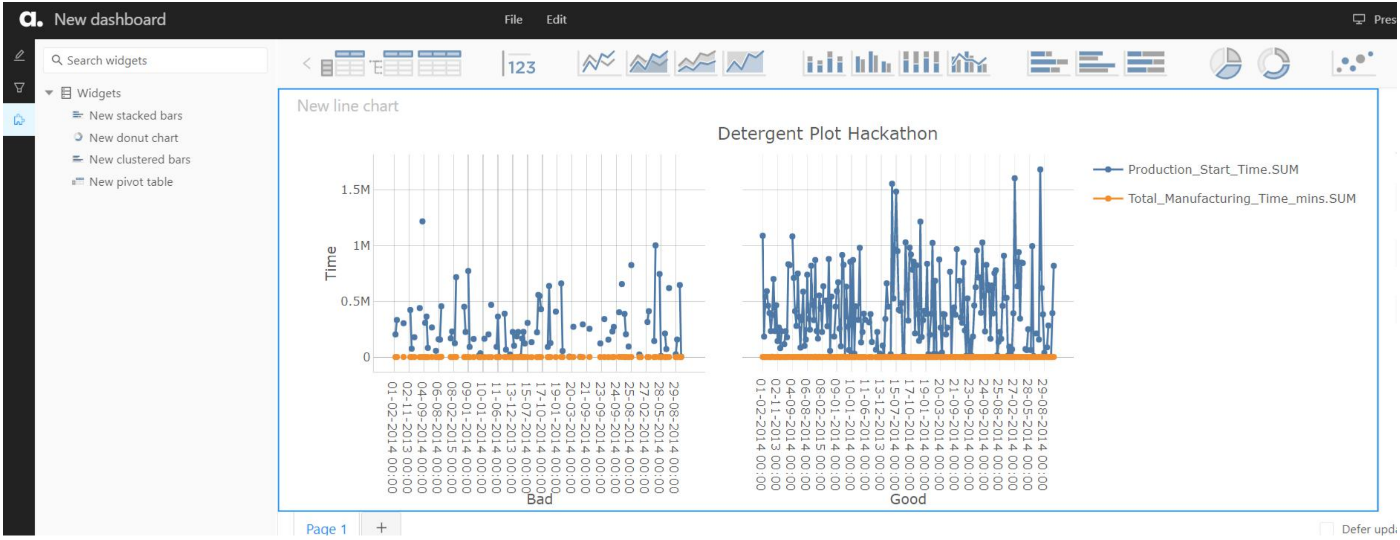
```
[15]: session.visualize()
```



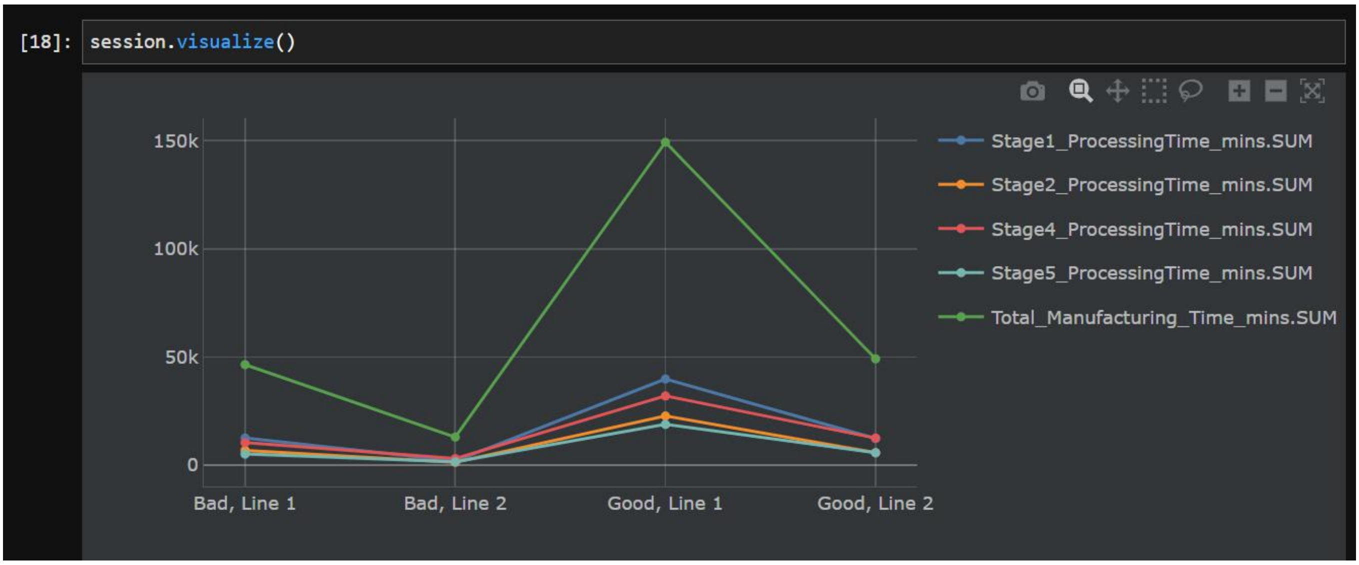
3. Ability to select from and to time stamp in the time series visualization and give a label or annotation



```
[17]: session.visualize()
```

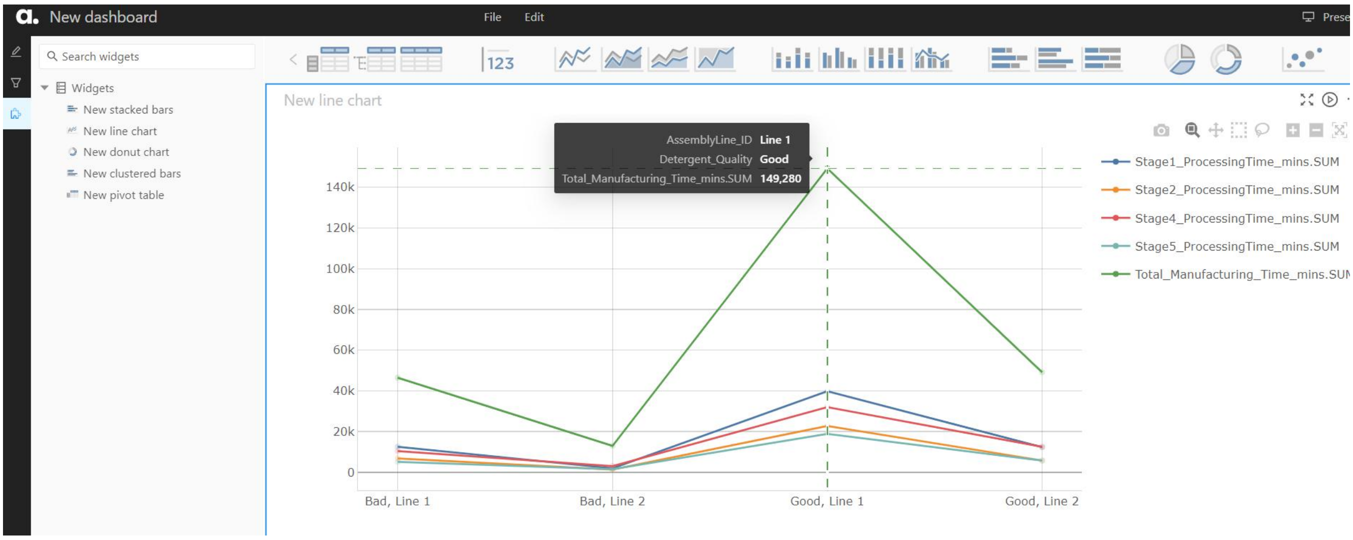


4. Annotation tool - Data labelling where the customer can import the data and the multiple columns render it in the chart where they can select from all and to frame and labelled the part and save in the database.

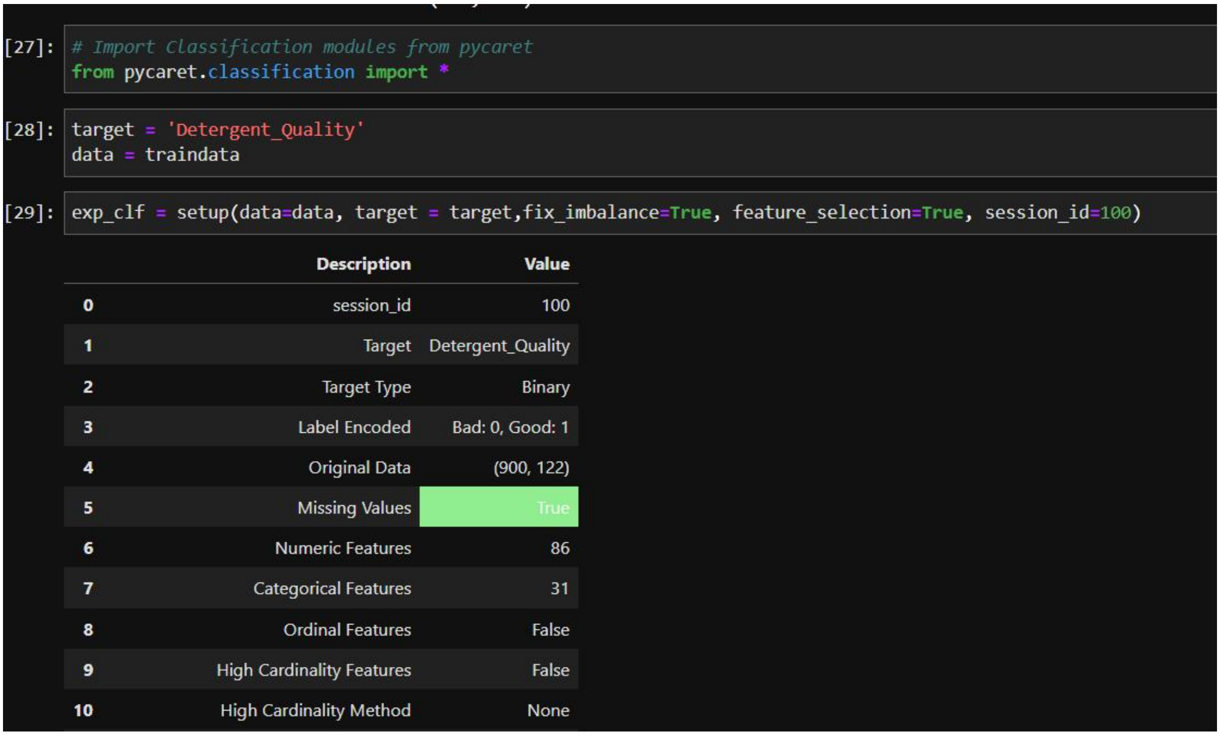


```
[18]: session.visualize()
```

150k
100k
50k
0
Bad, Line 1
Bad, Line 2
Good, Line 1
Good, Line 2
Stage1_ProcessingTime_mins.SUM
Stage2_ProcessingTime_mins.SUM
Stage4_ProcessingTime_mins.SUM
Stage5_ProcessingTime_mins.SUM
Total_Manufacturing_Time_mins.SUM



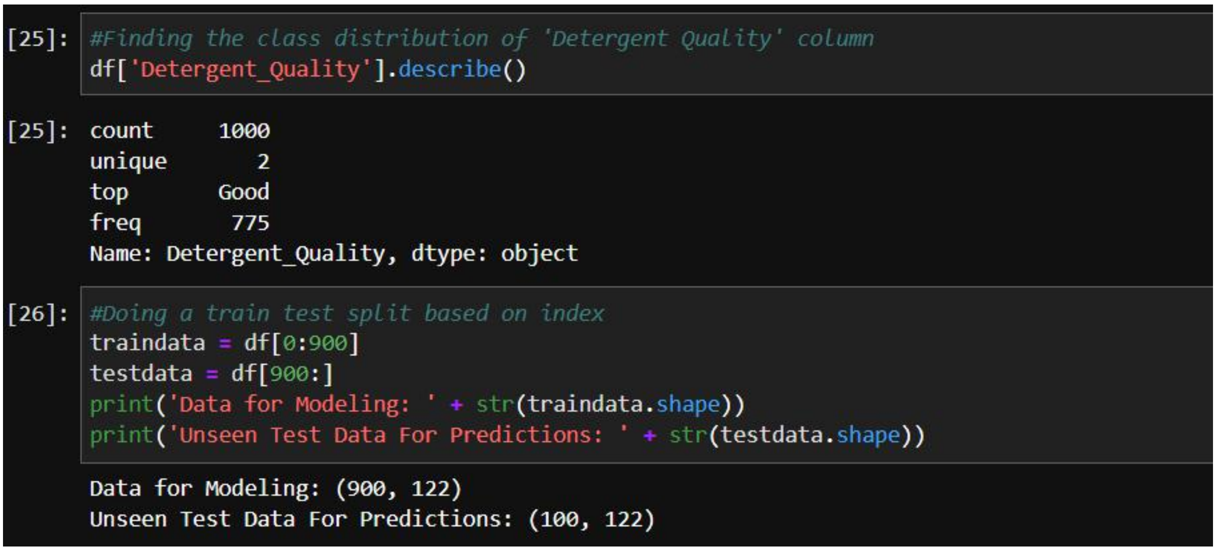
5. Exploratory data analysis- Where they can explore the data and find its relationship with the different parameters.



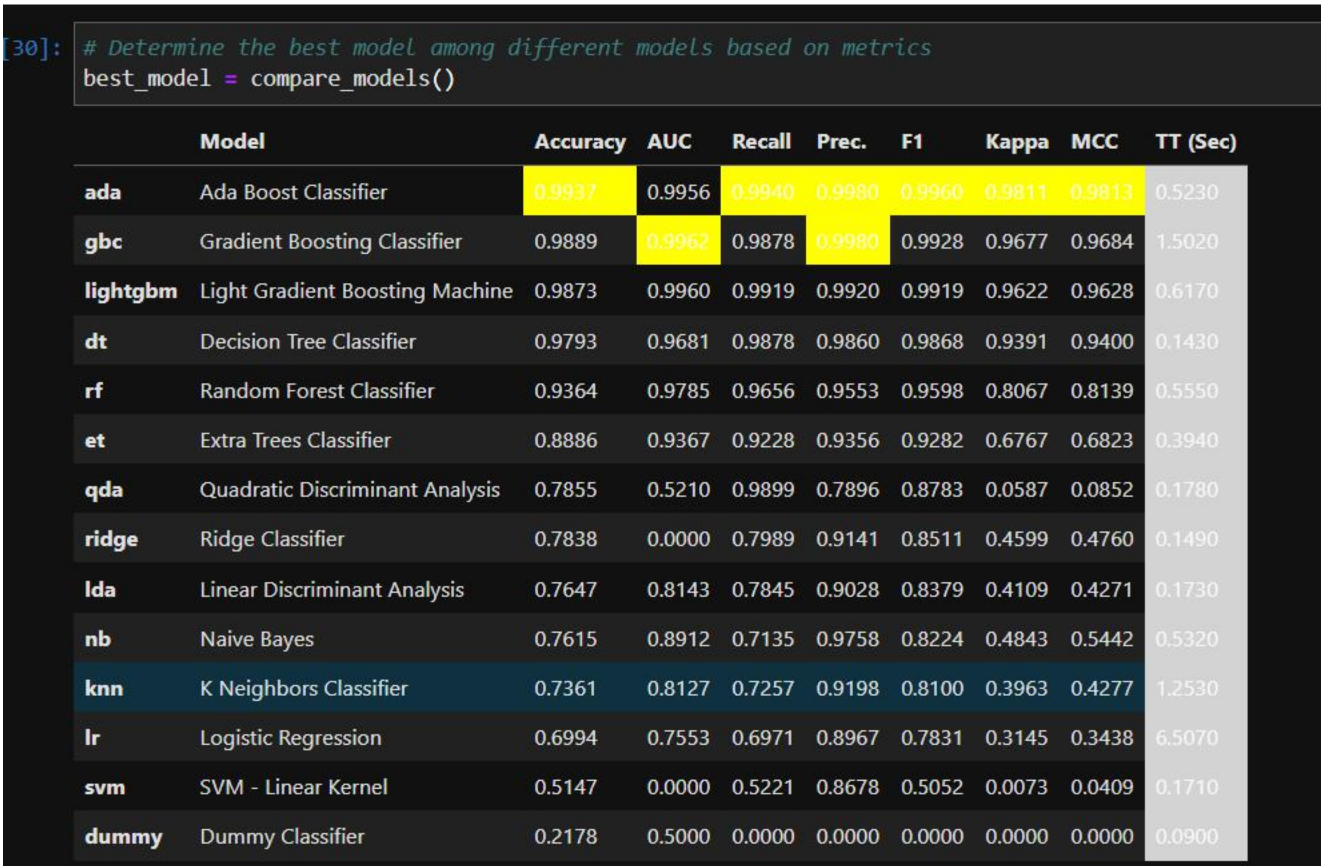
```
[27]: # Import Classification modules from pycaret
      from pycaret.classification import *

[28]: target = 'Detergent_Quality'
      data = traindata

[29]: exp_clf = setup(data=data, target = target,fix_imbalance=True, feature_selection=True, session_id=100)
```

	Description	Value
0	session_id	100
1	Target	Detergent_Quality
2	Target Type	Binary
3	Label Encoded	Bad: 0, Good: 1
4	Original Data	(900, 122)
5	Missing Values	True
6	Numeric Features	86
7	Categorical Features	31
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None

6. Prediction Analysis/ modelling - Where they can pass the data and application has to automatically select which model is best and show it’s all the model accuracy results.

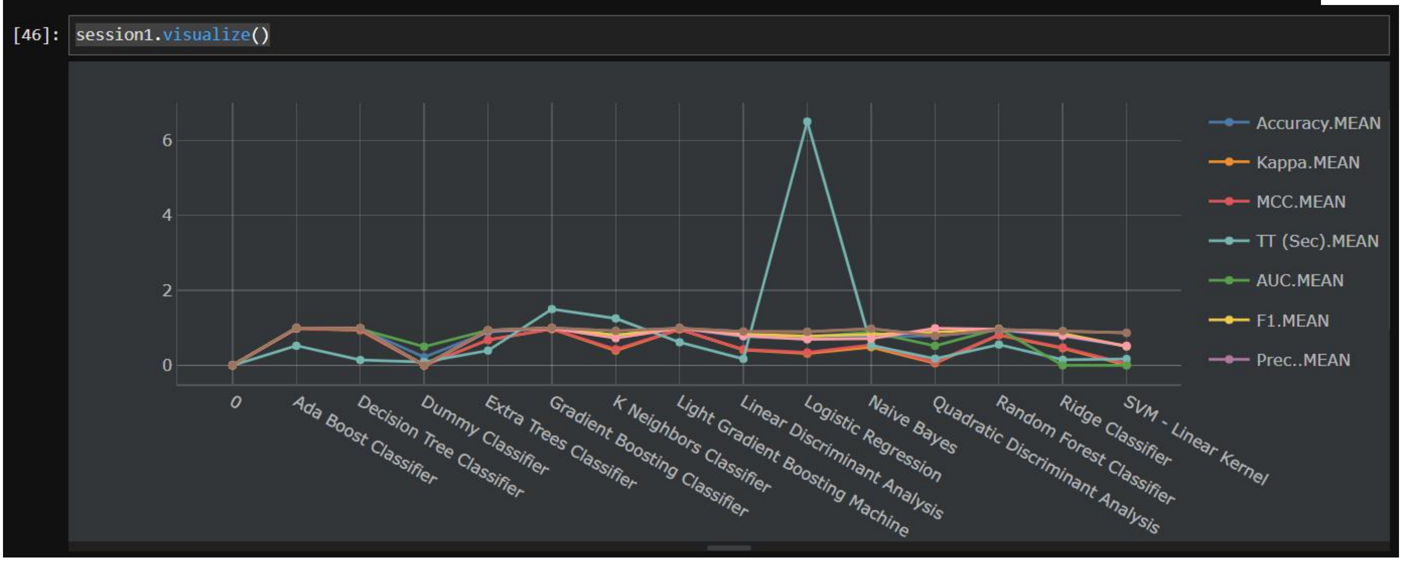


```
[25]: #Finding the class distribution of 'Detergent Quality' column
      df['Detergent_Quality'].describe()

[25]: count      1000
      unique        2
      top        Good
      freq        775
      Name: Detergent_Quality, dtype: object

[26]: #Doing a train test split based on index
      traindata = df[0:900]
      testdata = df[900:]
      print('Data for Modeling: ' + str(traindata.shape))
      print('Unseen Test Data For Predictions: ' + str(testdata.shape))

      Data for Modeling: (900, 122)
      Unseen Test Data For Predictions: (100, 122)
```

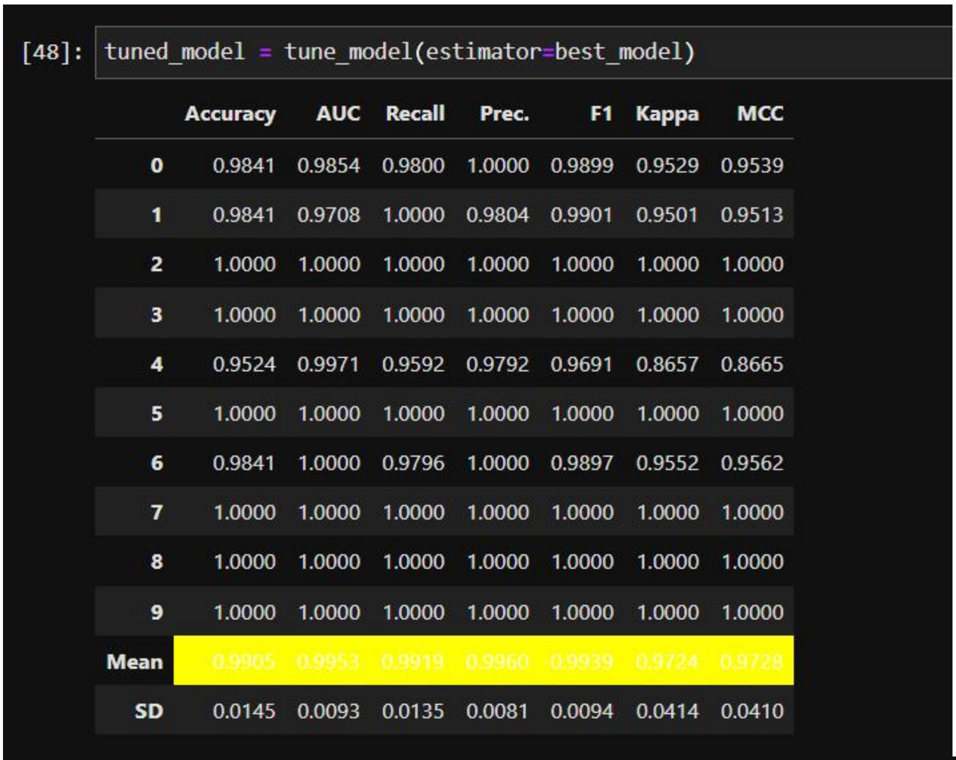


```
[30]: # Determine the best model among different models based on metrics
      best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
ada	Ada Boost Classifier	0.9937	0.9956	0.9940	0.9980	0.9960	0.9811	0.9813	0.5230
gbc	Gradient Boosting Classifier	0.9889	0.9962	0.9878	0.9980	0.9928	0.9677	0.9684	1.5020
lightgbm	Light Gradient Boosting Machine	0.9873	0.9960	0.9919	0.9920	0.9919	0.9622	0.9628	0.6170
dt	Decision Tree Classifier	0.9793	0.9681	0.9878	0.9860	0.9868	0.9391	0.9400	0.1430
rf	Random Forest Classifier	0.9364	0.9785	0.9656	0.9553	0.9598	0.8067	0.8139	0.5550
et	Extra Trees Classifier	0.8886	0.9367	0.9228	0.9356	0.9282	0.6767	0.6823	0.3940
qda	Quadratic Discriminant Analysis	0.7855	0.5210	0.9899	0.7896	0.8783	0.0587	0.0852	0.1780
ridge	Ridge Classifier	0.7838	0.0000	0.7989	0.9141	0.8511	0.4599	0.4760	0.1490
lda	Linear Discriminant Analysis	0.7647	0.8143	0.7845	0.9028	0.8379	0.4109	0.4271	0.1730
nb	Naive Bayes	0.7615	0.8912	0.7135	0.9758	0.8224	0.4843	0.5442	0.5320
knn	K Neighbors Classifier	0.7361	0.8127	0.7257	0.9198	0.8100	0.3963	0.4277	1.2530
lr	Logistic Regression	0.6994	0.7553	0.6971	0.8967	0.7831	0.3145	0.3438	6.5070
svm	SVM - Linear Kernel	0.5147	0.0000	0.5221	0.8678	0.5052	0.0073	0.0409	0.1710
dummy	Dummy Classifier	0.2178	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0900

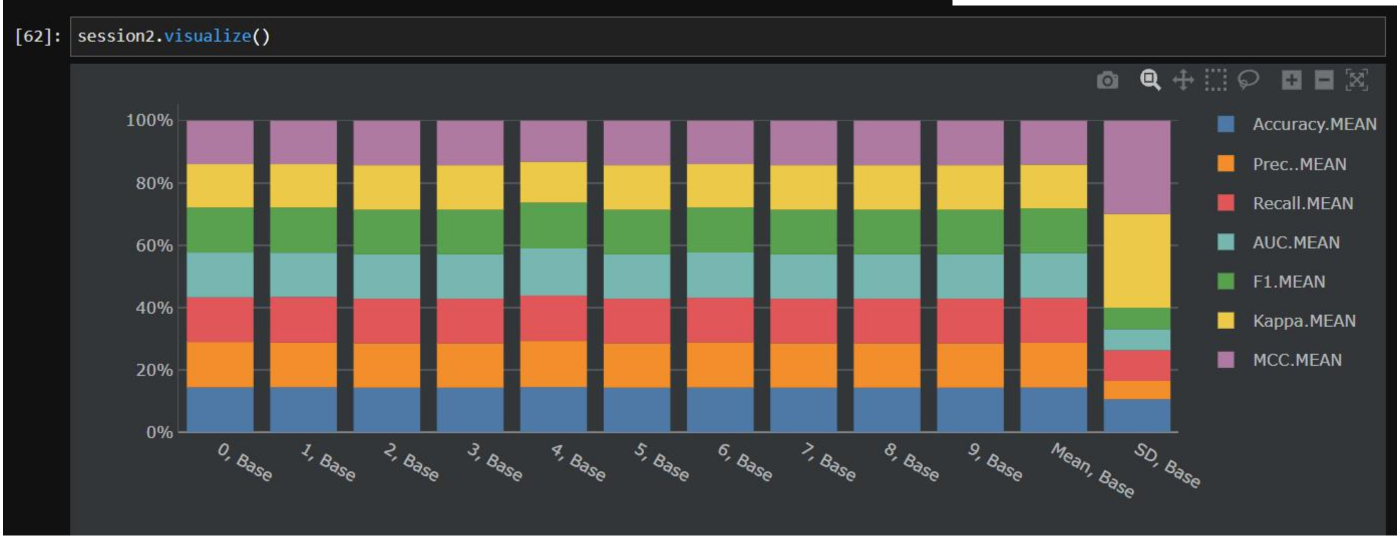


```
[46]: session1.visualize()
```



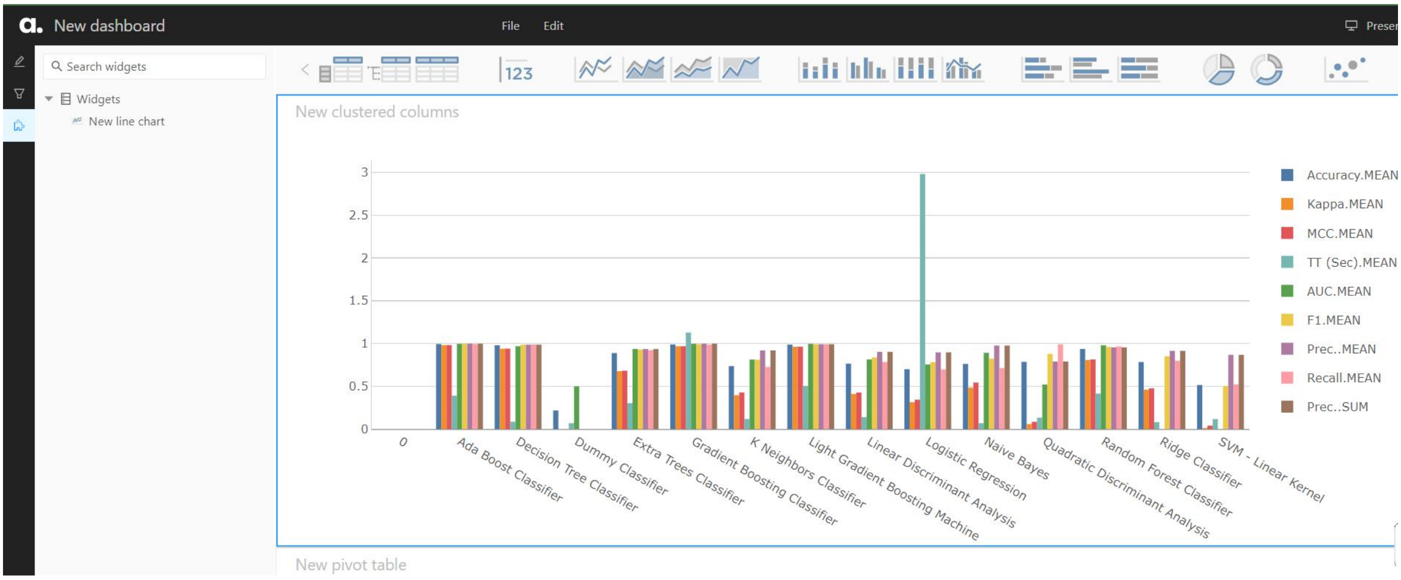
```
[48]: tuned_model = tune_model(estimator=best_model)
```

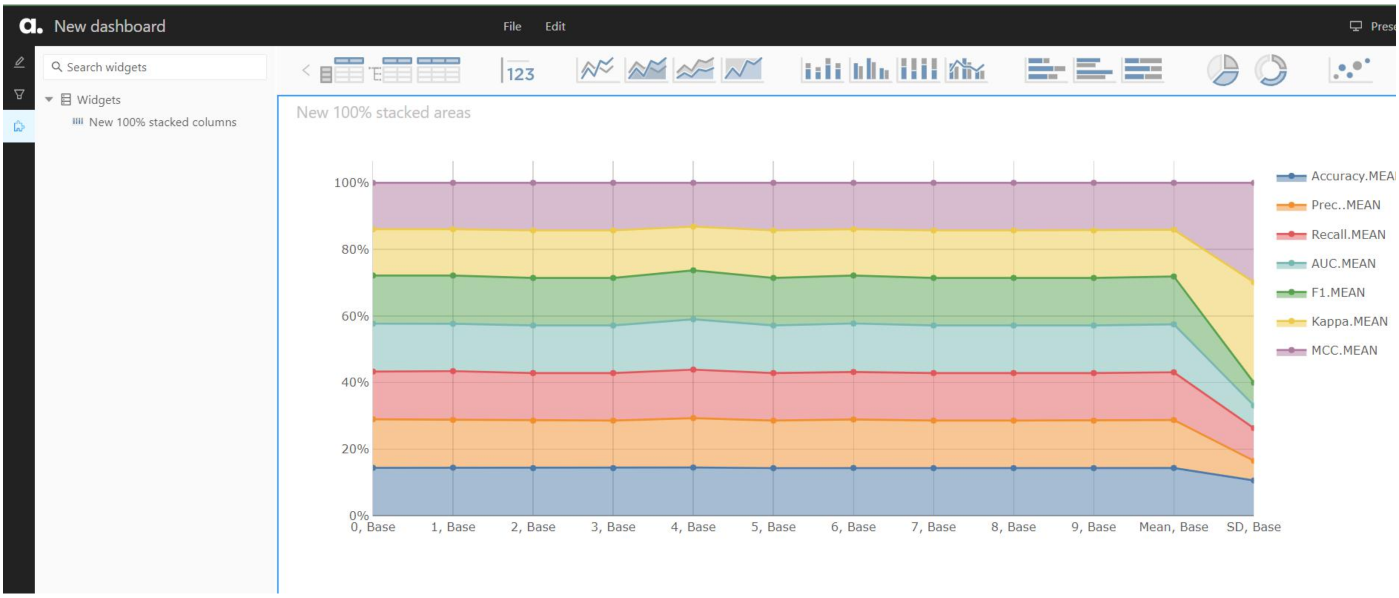
	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.9841	0.9854	0.9800	1.0000	0.9899	0.9529	0.9539
1	0.9841	0.9708	1.0000	0.9804	0.9901	0.9501	0.9513
2	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4	0.9524	0.9971	0.9592	0.9792	0.9691	0.8657	0.8665
5	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
6	0.9841	1.0000	0.9796	1.0000	0.9897	0.9552	0.9562
7	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
8	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
9	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Mean	0.9905	0.9953	0.9919	0.9960	0.9939	0.9724	0.9728
SD	0.0145	0.0093	0.0135	0.0081	0.0094	0.0414	0.0410



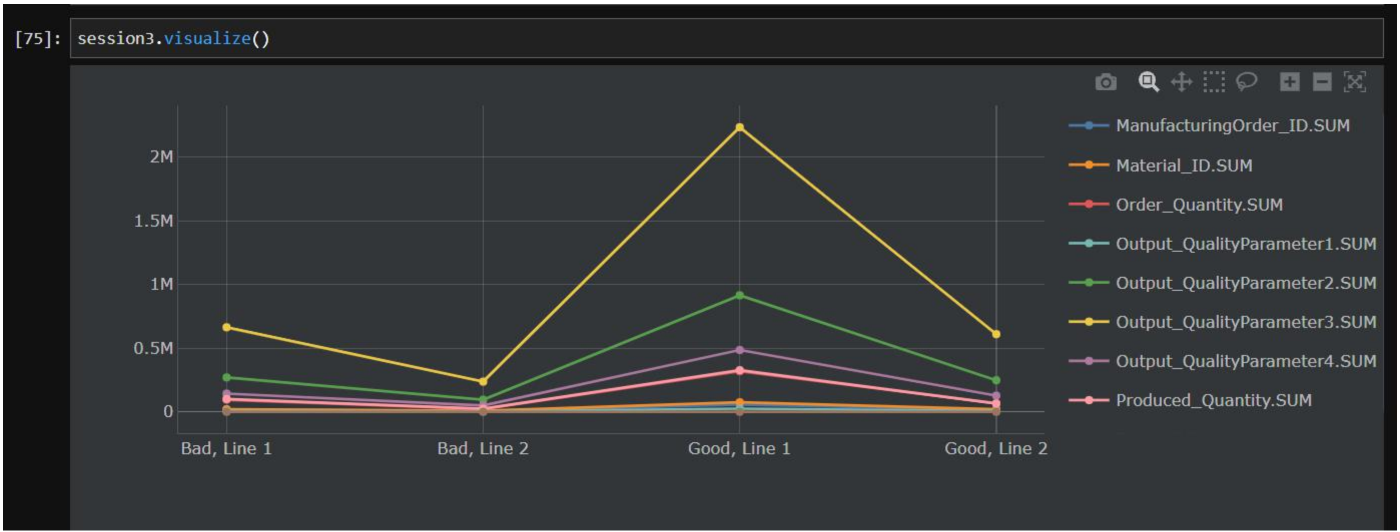
```
[62]: session2.visualize()
```

7. Have a dashboard to display aggregated values.



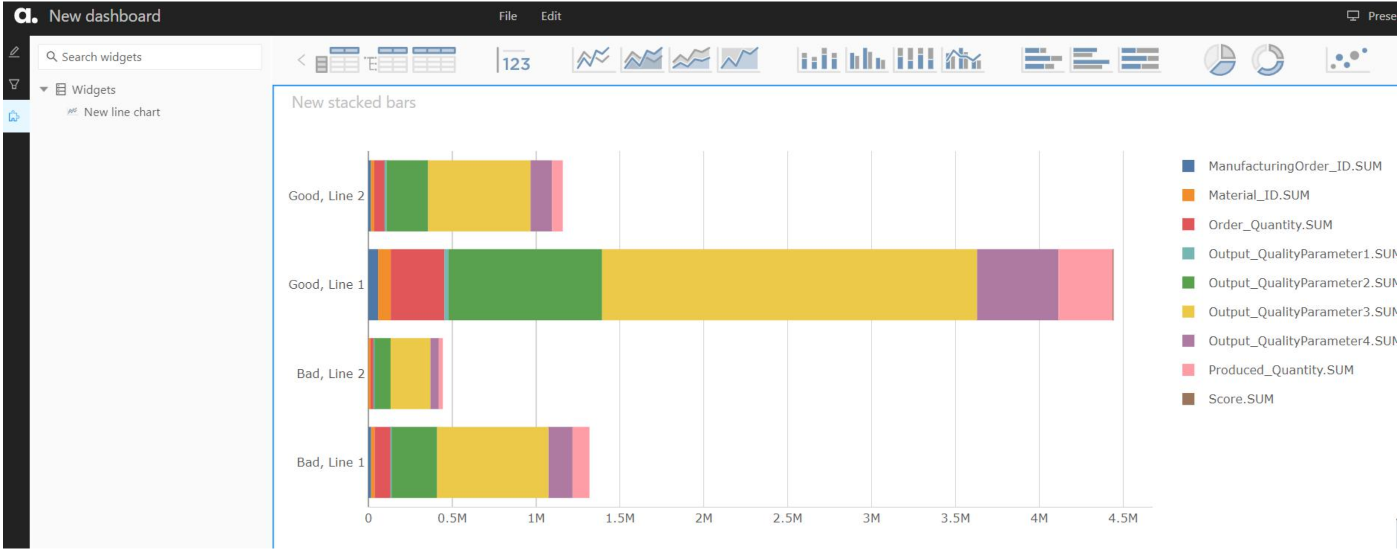


8. Results should be in pictorial representation.

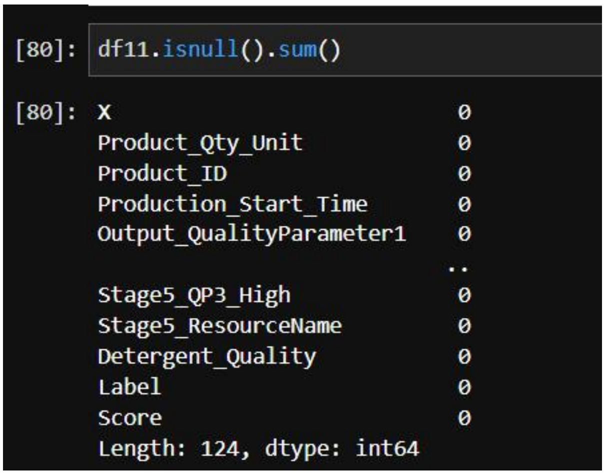


```
[75]: session3.visualize()
```

2M
1.5M
1M
0.5M
0
Bad, Line 1
Bad, Line 2
Good, Line 1
Good, Line 2
ManufacturingOrder_ID.SUM
Material_ID.SUM
Order_Quantity.SUM
Output_QualityParameter1.SUM
Output_QualityParameter2.SUM
Output_QualityParameter3.SUM
Output_QualityParameter4.SUM
Produced_Quantity.SUM



9. Data cleaning/Data sanitization must be done (Should not have null values).



```
[80]: df11.isnull().sum()

[80]: X                            0
      Product_Qty_Unit             0
      Product_ID                   0
      Production_Start_Time        0
      Output_QualityParameter1     0
                                  ..
      Stage5_QP3_High              0
      Stage5_ResourceName          0
      Detergent_Quality            0
      Label                        0
      Score                        0
      Length: 124, dtype: int64
```