BOM File Automation

For Deployment of Tyre

What is BOM File Automation ?

It’s all about Automating the Bill of Materials File for successful deployment of Tyre**.**

PROBLEM STATEMENT :

The RAW DATASET Excel workbook will be the input file, which contains sheets with nearly 10000+ RAW data's. Now new excel workbook with multiple sheets has to be created by automation (by program) which contains inherited data from raw data excel workbook.

SOLUTION :

This task executed by DATA CLEANING and DATA MINING process in response to requirements.

PROGRAMMING : python

DATA MINING TOOL : Pandas IN Jupyter Notebook.

Python IDE : ANACONDA

DESCRIPTION :

- **REQUIREMENTS:**
  1) Replacing CA* with GT* in Header Material Tuple.
  2) Delete all the yellow rows in raw dataset.
  3) Separate sheets should be created based on SW*,TRD*,RIM*,KYLE*,SF*,ADD SW* from BOM Component Tuple in dataset.

4) Another sheet should contain Tuples such as Header Material , BOM Component , Component quantity , Base UoM* for BOM , Created On , Created by , Last Changed on , Last Changed by.
5) Finally, the new excel work book is created containing all the above step 4 and step 5 sheets.

Algorithm :

Step 1 : Data Analysis is done manually

Step 2 : Data Cleaning is done by dropna() function in python

Step 3 : Data Mining Process is done by following method :

- METHOD : ***Agglomerative Data Clustering****

Agglomerative Data Clustering :

Hierarchical clustering algorithms actually fall into 2 categories: top-down or bottom-up. Bottom-up algorithms treat each data point as a single cluster at the outset and then successively merge (or agglomerate*) pairs of clusters until all clusters have been merged into a single cluster that contains all data points. Bottom-up hierarchical clustering is therefore called* hierarchical agglomerative clustering *or* HAC*. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.*

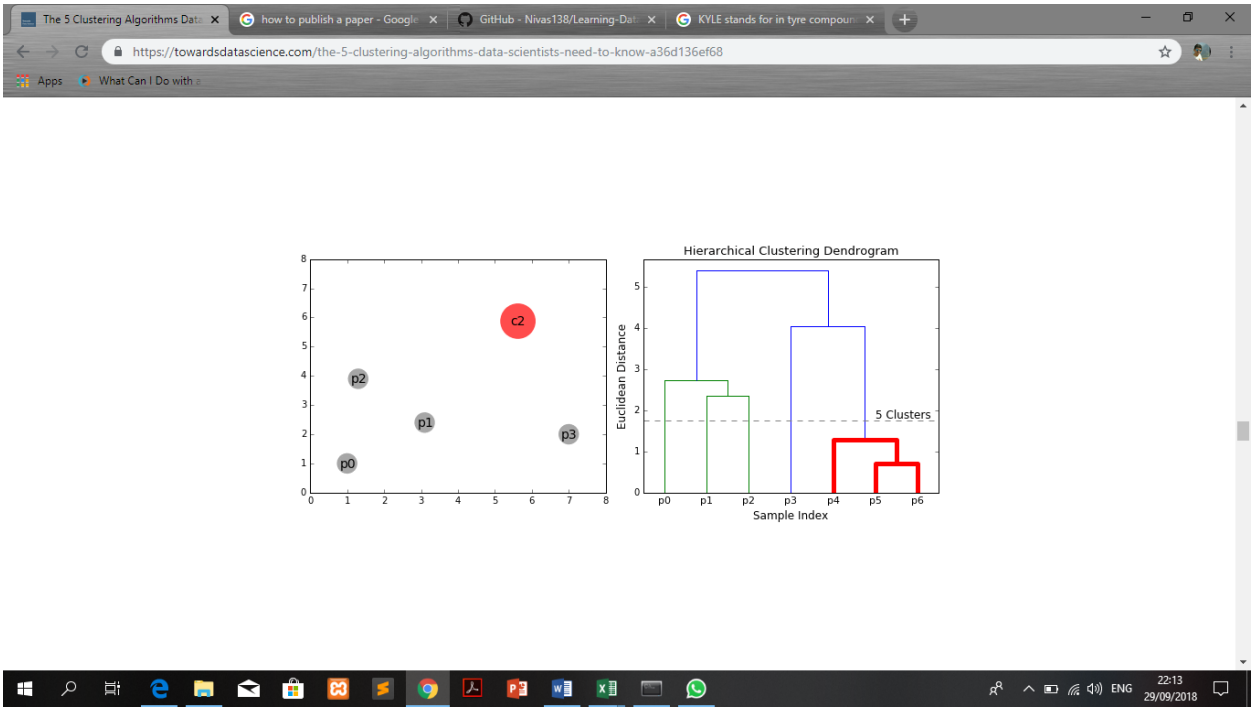
Steps :

1. *We begin by treating each data point as a single cluster i.e if there are X* data points in our dataset then we have X* clusters. We then select a distance metric that measures the distance between two clusters. As an example we will use average linkage which defines the distance between two clusters to be the average distance between data points in the first cluster and data points in the second cluster.*

2. *On each iteration we combine two clusters into one. The two clusters to be combined are selected as those with the smallest average linkage. I.e according to our selected distance metric, these two clusters have the smallest distance between each other and therefore are the most similar and should be combined.*

3. *Step 2 is repeated until we reach the root of the tree i.e we only have one cluster which contains all data points. In this way we can select how many clusters we want in the end, simply by choosing when to stop combining the clusters i.e when we stop building the tree!*

Hierarchical clustering does not require us to specify the number of clusters and we can even select which number of clusters looks best since we are building a tree. Additionally, the algorithm is not sensitive to the choice of distance metric; all of them tend to work equally well whereas with other clustering algorithms, the choice of distance metric is critical. A particularly good use case of hierarchical clustering methods is when the underlying data has a hierarchical structure and you want to recover the hierarchy; other clustering algorithms can't do this. These advantages of hierarchical clustering come at the cost of lower efficiency, as it has a time complexity of $O(n^3)$, unlike the linear complexity of K-Means and GMM.



- ✓ **Solution Program :**

```
print('BOM AUTOMATION FROM EXCEL')

#reading excel file

import pandas as pd
df=pd.read_excel('./ex.xlsx')

#required columns

req_coll = ['Header Material','BOM Component','Component quantity','Base UoM for BOM','Created On','Created By','Last Changed On','Last Changed By']
df=df[req_coll]



#null rows with coressponding columns

req_null = ['BOM Component','Base UoM for BOM','Created On','Created By','Last Changed On','Last Changed By']

#cleaning the yellow rows

df.dropna(axis=0,subset=req_null,inplace=True,how='all')
df.reset_index(drop=True,inplace=True)

#calculationg time

Count_Row=df.shape[0]
print("Total Data Running : " )
print(Count_Row)
import math
time=Count_Row/375
```

```
time1=math.ceil(time)
time2=str(time1)
print("Hello Dude !!")
print("I'm Running, Be Patient for maximum " + time2 + " minutes I will
notify you !!")
import time
import progressbar
with progressbar.ProgressBar() as bar:
```

#replace ca with gt

```
        count=0
        for i in bar(range(Count_Row)):
                time.sleep(0.1)
                df['Header Material'][count]='GT'+df['Header
Material'][count][2:]
                bar.update( i+1)
                count+=1
```

#sheet names

```
sheet_name=['sheet1','SW','TRD','RIM','KYLE','ADD','SF']
```

#saving all excel files in one book after list to dataframe , dataframe to excel using function

```
from pandas import ExcelWriter
def save_xls(df_all_list,excel_path,sheet_name_list):
   writer = ExcelWriter(excel_path)
   for n, df in enumerate(df_all_list):
      df.to_excel(writer,sheet_name_list[n],index=False)
   writer.save()
   print("Task Done!! Thank you for your patience !!")
   print('\nNow !! Go back to the same folder & get your required workbook
named as "new_required_book.xls" ')
bar.finish()
```

#rest all sheets initited list for fetching data

```
sw_list=[]
trd_list=[]
rim_list=[]
kyle_list=[]
add_list=[]
sf_list=[]
```

#checking bom component

```
count=0
for i in df['BOM Component']:
    if i[:2]=='SW':
        sw_list.append(count)
    elif i[:2]=='TE':
        trd_list.append(count)
    elif i[:2]=='RS':
        rim_list.append(count)
    elif i[:2]=='KS':
        kyle_list.append(count)
    elif i[:2]=='AS':
        add_list.append(count)
    elif i[:2]=='FS':
        sf_list.append(count)
    count+=1
```

#fetched the required list with required parameters

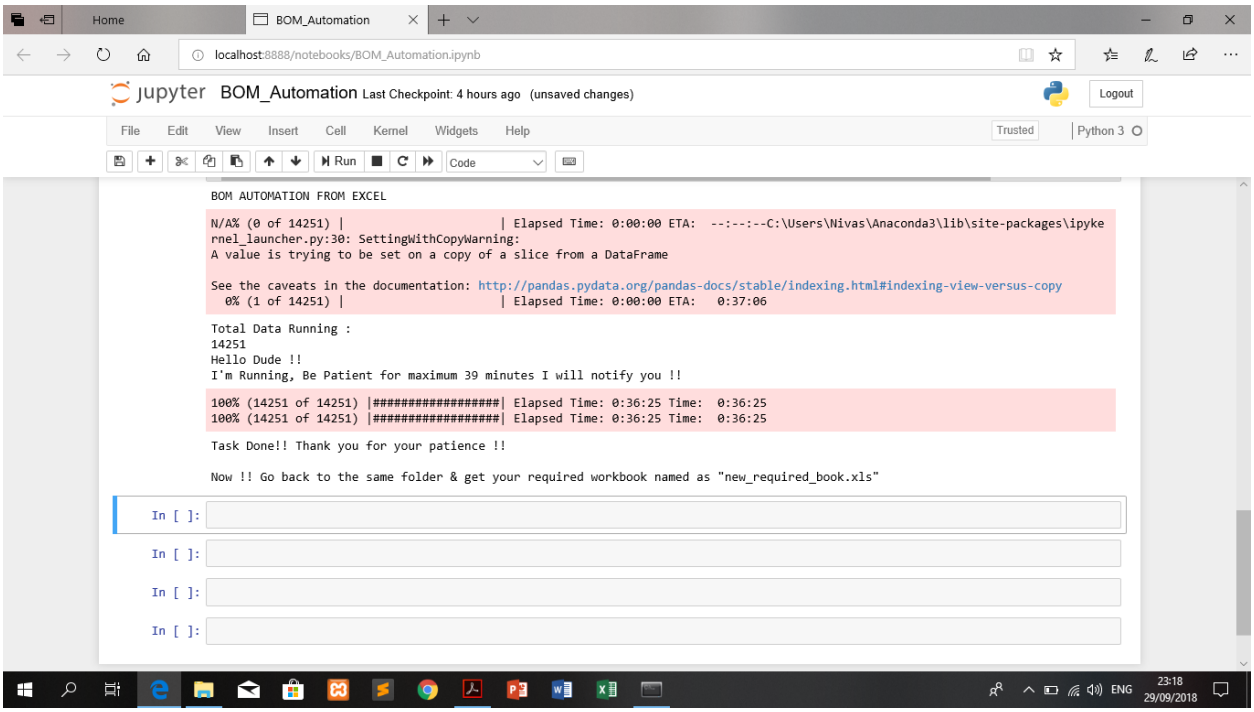
```
df_sw=df.iloc[sw_list,:4]
df_trd=df.iloc[trd_list,:4]
df_rim=df.iloc[rim_list,:4]
df_kyle=df.iloc[kyle_list,:4]
df_add=df.iloc[add_list,:4]
```

df_sf=df.iloc[sf_list,:4]
df_all=[df,df_sw,df_trd,df_rim,df_kyle,df_add,df_sf]

#list to dataframe and to excel function calling

save_xls(df_all,'new_required_book.xls',sheet_name)

Output:



✓ **DEPLOYMENT :**

The requirements are executed by above program from which new excel workbook is created for the supplies chain management of tyre deployment.