In [1]:

*# import json*

**import** **pandas** **as** **pd**

**from** **datetime** **import** datetime, timedelta

**from** **dateutil.parser** **import** parse *# to convert string date to pyton Date type so that we can compare dates with each other*

*# from pathlib import Path*

**import** **glob** *# to get the names and paths of the files in a directory/folder*

**1. Class which Convert records csv to into alarms**

In [2]:

*# format csvs*

**class** **CSV2Alarms**:

*"""*

*Summary:*

*Record mean 1 row of the csv file.*

*This class converts records into alarms.*

*"""*

**def** \_\_init\_\_(self,config): *#done*

self.config = config

print(f'>>Input file: **{**self.config["dir"]+self.config["in\_fname"]**}**')

self.df = **None**

*# reading raw csv fiile*

**try**:

self.df = pd.read\_csv(self.config["dir"]+self.config["in\_fname"], delimiter= ";" , usecols=self.config["cols"], encoding = "ISO-8859-1")

**except** **Exception** **as** e:

*# print(f"\n\n !!!!!!!!!!!!!!!!!! Excetpion 1 {e} !!!!!!!!!!!!!!!!!!!!!!!!!!")*

a = 1

**try**:

self.df = pd.read\_csv(self.config["dir"]+self.config["in\_fname"], delimiter= "," , usecols=self.config["cols"], encoding = "ISO-8859-1")

**except** **Exception** **as** e:

a =1

*# self.df = pd.read\_csv(self.config["dir"]+self.config["in\_fname"], encoding = "ISO-8859-1")*

*# print(f"\n\n !!!!!!!!!!!!!!!!!! Excetpion 2 {e} !!!!!!!!!!!!!!!!!!!!!!!!!!")*

print(">>File is read.")

*# removing the extra chars and returning the python Date*

**def** \_\_changeDate(self,d): *# done*

d = d.replace(".000000000","") *# removing extra zeros from the string date*

d = d.replace("/","-")

**return** parse(d) *# converting string to a python Date type e.g., float('5')*

*# Returning the alarm message type*

**def** \_\_getMessageType(self,message): *# done*

**if** message.find("Recover") != -1: *# finding the deactivation message*

**return** "Recover"

**elif** message.find("NR") != -1:

**return** "NR"

**else**:

**return** "Activation"

**def** \_\_getAlarmsFromDFs(self,df\_start, df\_end): *# done*

alarms = []

start\_records = [v **for** v **in** sorted(df\_start.to\_dict(orient="records"), key=**lambda** arg: arg["EventTime"], reverse=**False**)] *# sortting the dataframes rows in ascending order and storing them in a list*

end\_records = [v **for** v **in** sorted(df\_end.to\_dict(orient="records"), key=**lambda** arg: arg["EventTime"], reverse=**False**)]

i = 0

j = 0

*# print("End len",len(end\_records), "Start len", len(start\_records))*

**while** j < len(end\_records): *# ingore the row/entry if the end time is lesser to start time*

*# print(i,j)*

**if** len(start\_records)>0 **and** end\_records[j]["EventTime"] < start\_records[i]["EventTime"]:

j += 1

**else**:

**break**

**while** i < len(start\_records): *# find the one to one correspondence between startingtime and its coressponding most recent end time.*

**if** j <len(end\_records) **and** start\_records[i]["EventTime"] <= end\_records[j]["EventTime"]:

**if** i+1 < len(start\_records) **and** start\_records[i+1]["EventTime"] < end\_records[j]["EventTime"]: *# check for the next record*

i += 1

**continue**

alarm = {k: v **for** k, v **in** start\_records[i].items()} *# an alarm dictionary*

alarm["StartTime"] = alarm["EventTime"]

alarm["EndTime"] = end\_records[j]["EventTime"]

alarm["EndMessage"] = end\_records[j]["Message"]

**del** alarm["EventTime"]

alarms.append(alarm)

j += 1

**elif** j <len(end\_records) **and** start\_records[i]["EventTime"] > end\_records[j]["EventTime"]:

j +=1

**continue**

i += 1

**return** alarms

**def** \_\_convertRecordsToAlarmsV1(self,df\_source): *# done*

*""" Convert records from the same source to proper alarms with start and end time.*

*The record which contains "Recover" or "NR" in the Message column shows the deactivations.*

*Parameters*

*----------*

*records : list of dict*

*Each dict represent either activation of an alarm or deactivation of an alarm.*

*Returns*

*-------*

*alarms : list of dict*

*Each dict in the list is an alarm with the StartTime and EndTime of an alarm.*

*"""*

alarms = []

**for** condition **in** df\_source["Condition"].unique():

df\_condition = df\_source.loc[df\_source['Condition'].isin([condition])]

df\_start = df\_condition.loc[df\_condition['MessageType'].isin([

"Activation"])]

end\_types = [t **for** t **in** df\_condition["MessageType"].unique() **if** t !=

"Activation"] *# NR, Recover, NR+Reocver*

*# print(types)*

df\_end = df\_condition.loc[df\_condition['MessageType'].isin(end\_types)]

alarms += self.\_\_getAlarmsFromDFs(df\_start, df\_end)

**return** alarms

**def** formatCSV(self): *# done*

print(">>Column Types: ", end="")

**for** col **in** self.df.columns:

print(col, type(self.df[col][0]), end=", ")

**if** isinstance(self.df[col][0],str):

**try**:

self.df[col] = self.df[col].apply(**lambda** s: " ".join(s.split())) *# removing the extra spaces in a column*

**except** **Exception** **as** e:

print(f"**\n\n** !!!!!!!!!!!!!!!!!! Excetpion **{**e**}** !!!!!!!!!!!!!!!!!!!!!!!!!!")

print(type(self.df["EventTime"][0]))

self.df["EventTime"] = self.df["EventTime"].apply(self.\_\_changeDate)

self.df["MessageType"] = self.df["Message"].apply(self.\_\_getMessageType)

*# self.df["Month"] = self.df["EventTime"].apply(lambda arg: arg.month)*

print(f">>Before Filtering ACKS: **{**self.df.shape**}**")

self.df = self.df.loc[self.df['Message'].map(**lambda** arg: arg.find(self.config["ack-filter"])) == -1] *# removing the Acknowledgements*

print(f">>After Filtering ACKS: **{**self.df.shape**}**")

fpath = self.config["dir"] + self.config["formated\_fname"]

self.df.to\_csv(fpath, index=**False**) *# storing pandas dataframe into a csv file*

**return** self.df

**def** readFormattedCSV(self): *# done*

fpath = self.config["dir"] + self.config["formated\_fname"]

self.df = pd.read\_csv(fpath, low\_memory=**False**, parse\_dates=["EventTime"])

print(f">> Formating is complete. Outfile: **{**fpath**}**")

**return** self.df

**def** convertRecords2Alarms(self,df): *# done*

*# df = pd.read\_csv(p, low\_memory=False, usecols=cols,parse\_dates=["EventTime"])*

**assert** len(df["MachineName"].unique()) == 1 *# all the alarms should be related to the same unit*

alarms = []

sources\_ranks\_dict = df['SourceName'].value\_counts()

id = 0 *# for debugging*

**for** sname **in** sources\_ranks\_dict.keys(): *# for each source find the activation and deactivation times and covert them to alarms*

id += 1

*# TIC4544*

df\_sname = df.loc[df['SourceName'].isin([sname])] *# source DF*

*# types\_rank\_dict = df\_sname["MessageType"].value\_counts() # source ranks*

*# total = 0*

*# for key in types\_rank\_dict.keys():*

*# total += types\_rank\_dict[key]*

*# assert(total== sources\_ranks\_dict[sname]) # sum is equal to count*

print(f"Test:**{**sname**}**")

source\_alarms = self.\_\_convertRecordsToAlarmsV1(df\_sname) *# getting alarms related to one source (sname)*

print("end")

alarms += source\_alarms

print(f"[**{**id**}**]Source:**{**sname**}**, Conditions:**{**df\_sname['Condition'].unique()**}**, Total Alarms:**{**len(source\_alarms)**}**")

df\_out = pd.DataFrame(alarms)

df\_out["TimeDelta"] = df\_out[["StartTime", "EndTime"]].apply(**lambda** arg: timedelta.total\_seconds(arg[1]-arg[0]) , axis=1)

df\_out["Year-Month"] =df\_out["StartTime"].apply(**lambda** arg: (arg.year,arg.month))

file\_path = self.config["dir"]+self.config["alarm\_out\_fname"]

df\_out.to\_csv(file\_path, index = **False**)

print(f">>Conversion from records to alarms is complete. Outputfile : **{**file\_path**}**, Info : **{**df\_out.info()**}**")

**return** df\_out

In [3]:

*# def getPlantShutDownPeriods(filepath):*

*# df\_feed = pd.read\_excel(filepath)*

*# feed\_records = [v for v in sorted(df\_feed.to\_dict(orient="records"), key=lambda arg: arg["TimeStamp"], reverse=False)]*

*# plant\_shut\_down\_periods = []*

*# start\_time = None*

*# start\_flag = True*

*# for i in range(len(feed\_records)):*

*# if feed\_records[i]["47FIC011.PV"] <=50 and start\_flag ==True:*

*# start\_time = feed\_records[i]["TimeStamp"]*

*# start\_flag = False*

*# # print(f"{start\_time},{feed\_records[i]['47FIC011.PV']} ", end=" ")*

*# elif feed\_records[i]["47FIC011.PV"] >50 and i>0 and feed\_records[i-1]["47FIC011.PV"]<=50:*

*# plant\_shut\_down\_periods.append((start\_time,feed\_records[i]["TimeStamp"]))*

*# start\_flag =True*

*# # print(f'{feed\_records[i]["TimeStamp"]},{feed\_records[i-1]["47FIC011.PV"]},{feed\_records[i]["47FIC011.PV"]}')*

*# return plant\_shut\_down\_periods*

*# def monthlyAlarms2SingleFile(config):*

*# alarms\_dir, out\_file\_path ,cols = config["dir"], config["out\_file"], config['cols']*

*# fps = [f for f in glob.glob(alarms\_dir+ "\*.csv")]*

*# print(f">> Files to process {fps}")*

*# dfs\_list = []*

*# for f in fps:*

*# print(f">> === File: {f.split('/')[-1]}")*

*# df = pd.read\_csv(f, usecols = cols ,parse\_dates = ["StartTime","EndTime"])*

*# dfs\_list.append(df)*

*# df = pd.concat(dfs\_list, ignore\_index=True)*

*# df.to\_csv(out\_file\_path, index=False)*

*# return df*

**2. Configuration of input dirs and output dirs and file names**

In [10]:

ACK\_FILTER = "ACK"

*# PLANT = "plant\_33"*

in\_config = {

"dir": "../data/",

"csv\_names" : [f.split('/')[-1] **for** f **in** glob.glob(f"../../data/processed/alarms/"+ "\*.csv") **if** f.find("f-pre-1") !=-1], *# list comprehension*

"in\_fname": **None**, *# input file name*

"formated\_fname": **None**,

"alarm\_out\_fname": **None**,

"ack-filter":ACK\_FILTER,

'cols':["MachineName","SourceName","EventTime", "Message","Condition"]

}

print(in\_config)

out\_config = {

"dir": "../../data/processed/alarms/",

"out\_file": "../../data/processed/final/all-months-alarms-with-feed.csv",

"cols": ["SourceName", "Condition","StartTime", "EndTime","TimeDelta","Year-Month"]

}

alarm = **None**

temp\_df = **None**

obj = **None**

*# for f in in\_config["csv\_names"]:*

*# in\_config['in\_fname'] = "raw/"+f*

*# in\_config['formated\_fname'] = "raw/formatted\_"+f*

*# in\_config["alarm\_out\_fname"] = "alarms/alarms\_"+f*

*# obj = CSV2Alarms(in\_config)*

*# temp\_df = obj.formatCSV()*

**for** f **in** in\_config["csv\_names"]: *# reading csv files and first its formatting the csv files (i.e., removing extra zeros etc) and finally convertng the rows/records to alarms and storing back into a csv file in the processed folder*

in\_config['in\_fname'] = "raw/"+f

in\_config['formated\_fname'] = "raw/formatted\_"+f

in\_config["alarm\_out\_fname"] = "alarms/alarms\_"+f

obj = CSV2Alarms(in\_config)

*# temp\_df = alarm.formatCSV()*

temp\_df = obj.readFormattedCSV()

df\_alarms = obj.convertRecords2Alarms(temp\_df)

print(">> Complete")

In [11]:

*# plant\_shut\_down\_periods = getPlantShutDownPeriods("../../data/feed.xlsx")*

*# plant\_shut\_down\_periods*

*# # df\_f\_month = pd.read\_csv()*

In [12]:

df\_formatted = pd.read\_csv("../../data/processed/alarms/f-pre-1-2019\_5.csv")

df\_formatted

*# df = df\_formatted*

*# print(type(df["EventTime"][0]))*

*# # df =*

*# for t in plant\_shut\_down\_periods:*

*# start\_remove, end\_remove = t[0],t[1]*

*# df = df.loc[(df.EventTime < start\_remove) | (df.EventTime > end\_remove)]*