On this review of draft 5, we:

- implement the same BAs classification as on the Lead to Revenue Reports (for standardization and to reduce the number of BAs)
- will calculate the opportunity age
- will remove duplicated and pick the first, instead of the las

These changes were implemented to improve model precision for class "won".

H. Oliveira (Feb-13-2022)

In [264…
```
# Importing the required libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.compose import make_column_transformer
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error

#Reading in the data
som_df = pd.read_csv('20211030_Data_Consolidation.csv', low_memory=False)
som_df.shape
```

Out[264]:
```
(74748, 44)
```

In [265…
```
display(som_df.iloc[1,:])
```

Instance_ID	9171043922
Opportunity ID	91710
Report date	43921
Sold-to Party	[redacted]
Opp. Description	[redacted]
Last RG reached	Quotation approved
Exp. Sales Volume	100.000,00
Currency	CHF
Status	Lost
Quotation Type	Binding Quotation
Business Type	Customer Service
Employee Responsible	[redacted]
Business unit	MU
Chance of Realization	0.2
Chance for Bühler	0.4
Next action	To discuss the quote with the customer
PlanDate: Ready to Quote	NaN
PlanDate: Quot. approved	24.02.2017
Final Ship-to Country	GP
Exp Sales Vol in CHF	100000
Risk Check completed	NaN
Risk level	NaN
PlanDate: Order Rel.	43831
Attachment	NaN
Lead ID	NaN
Quotation No.	NaN
Start Date	42809
Act.Date: Order Rel.	1022
Act.Date: Handshake received	NaN
Act.Date: Quot. approved	NaN
Act.Date: BC Approved	NaN
Act.Date: Ready to Quote	NaN
Sold-to Party.1	NaN
Market segment	NaN
Final Ship-to ID	NaN
Sold-to Party Postal Code	97110
Sold-to Party City	Pointe a Pitre
Changed on	NaN
Reason	NaN
Final Ship-to Name	NaN
Final Ship-to Postal Code	NaN
Final Ship-to City	NaN
Winning Competitor	NaN
Main Contact	NaN

Name: 1, dtype: object

In [266…

```
# Dropping unused collumns

som_df = som_df.drop('Exp. Sales Volume', 1)
som_df = som_df.drop('Currency', 1)
som_df = som_df.drop('Next action', 1)
som_df = som_df.drop('PlanDate: Ready to Quote', 1)
som_df = som_df.drop('PlanDate: Quot. approved', 1)
#som_df = som_df.drop('Final Ship-to Country', 1)
#som_df = som_df.drop('Days to close', 1)
som_df = som_df.drop('Risk Check completed', 1)
som_df = som_df.drop('Risk level', 1)
som_df = som_df.drop('Attachment', 1)
som_df = som_df.drop('Lead ID', 1)
som_df = som_df.drop('Quotation No.', 1)
```

```
som_df = som_df.drop('Act.Date: Handshake received', 1)
som_df = som_df.drop('Act.Date: Quot. approved', 1)
som_df = som_df.drop('Act.Date: BC Approved', 1)
som_df = som_df.drop('Act.Date: Ready to Quote', 1)
som_df = som_df.drop('Sold-to Party.1', 1)
som_df = som_df.drop('Market segment', 1)
som_df = som_df.drop('Final Ship-to ID', 1)
som_df = som_df.drop('Changed on', 1)
som_df = som_df.drop('Reason', 1)
som_df = som_df.drop('Final Ship-to Name', 1)
som_df = som_df.drop('Final Ship-to Postal Code', 1)
som_df = som_df.drop('Final Ship-to City', 1)
som_df = som_df.drop('Winning Competitor', 1)
som_df = som_df.drop('Main Contact', 1)
```

In [ ]:

In [267…
```
display(som_df.iloc[1,:])
```

Instance_ID	9171043922
Opportunity ID	91710
Report date	43921
Sold-to Party	[redacted]
Opp. Description	[redacted]
Last RG reached	Quotation approved
Status	Lost
Quotation Type	Binding Quotation
Business Type	Customer Service
Employee Responsible	[redacted]
Business unit	MU
Chance of Realization	0.2
Chance for Bühler	0.4
Final Ship-to Country	GP
Exp Sales Vol in CHF	100000
PlanDate: Order Rel.	43831
Start Date	42809
Act.Date: Order Rel.	1022
Sold-to Party Postal Code	97110
Sold-to Party City	Pointe a Pitre

Name: 1, dtype: object

In [268…
```
mes_df = np.unique(som_df['Report date'])
print(mes_df)
```

```
[43891 43921 43951 43981 44011 44041 44071 44101 44131 44161 44191 44221
 44251 44281 44311 44341 44371 44401 44431]
```

In [269…
```
#removing inactive IDs from march/2020

#filtering March/2020 opportunities
march_df = som_df[som_df['Report date'] == mes_df[0]]

#filtering inactive opportunities in March/2020
inactive_march_df = march_df[march_df['Status'] != 'Active']

#selecting only the ids of March/20 that aren't 'active'
inactive_id = inactive_march_df['Opportunity ID']

#valid SOM = opportunities active by march/20; the inactive ones are removed
valid_som_df = som_df[~som_df['Opportunity ID'].isin(inactive_id.tolist())]
```

In [270…
```
cleaned_march_df = valid_som_df[valid_som_df['Report date'] == mes_df[0]]

for i in np.unique(valid_som_df.iloc[:,6]):
    print("No. of class", i,"instances", (sum(cleaned_march_df.iloc[:,6]==i)))
```

```
No. of class Active instances 2539
No. of class Cancelled by Buhler instances 0
No. of class Cancelled by Customer instances 0
No. of class Lost instances 0
No. of class User Error instances 0
No. of class User Error/Duplicate instances 0
No. of class Won instances 0
```

In [271…
```
for i in np.unique(valid_som_df.iloc[:,6]):
    print("No. of class", i,"instances", (sum(valid_som_df.iloc[:,6]==i)))
```

```
No. of class Active instances 36012
No. of class Cancelled by Buhler instances 5306
No. of class Cancelled by Customer instances 11144
No. of class Lost instances 2190
No. of class User Error instances 149
No. of class User Error/Duplicate instances 1077
No. of class Won instances 6539
```

Setting the 30 Days Status Column

In [272…
```
# creating supporting datasets

# id datframe
id_df = np.unique(valid_som_df['Opportunity ID'])

# status thirty dataframe
status_thirty = pd.DataFrame({'Instance_ID':[], 'status_thirty':[]})

#clustering opportunities with same ID
for q in id_df:
    id_cluster = valid_som_df[(valid_som_df['Opportunity ID'] == q)].sort_values(by=['
    lines = id_cluster.shape[0]
    date_range = id_cluster['Report date']

    #validated algoritm to get the opportunity status in 30 days
    for i in range(lines):

        if (id_cluster.iloc[i,2]+30) in date_range.values:
            c = id_cluster['Status'].where(id_cluster.iloc[:,2] == id_cluster.iloc[i,2
            instance = id_cluster.iloc[i,0]
            #print(c.values[0])
            #print()
            status_thirty = status_thirty.append({'Instance_ID': instance,'status_thir
        else:
            #c = 'NaN'
            status_thirty = status_thirty.append({'Instance_ID': instance,'status_thir

#display(status_thirty)
dataset30 = pd.merge(valid_som_df, status_thirty, on='Instance_ID')

print(dataset30.shape)
```

```
(62421, 21)
```

In [273…
```
display(dataset30.iloc[1,:])
```

```
Instance_ID                                  14435243922
Opportunity ID                                   144352
Report date                                       43921
Sold-to Party                                     [redacted]
Opp. Description                                  [redacted]
Last RG reached                    RG2: Ready to Quote
Status                          Cancelled by Customer
Quotation Type                                     NaN
Business Type                          Project/Plant
Employee Responsible                              [redacted]
Business unit                                      HN
Chance of Realization                             0.6
Chance for Bühler                                 0.8
Final Ship-to Country                              US
Exp Sales Vol in CHF                           250000
PlanDate: Order Rel.                              43861
Start Date                                        43787
Act.Date: Order Rel.                                 74
Sold-to Party Postal Code                          8512
Sold-to Party City                             Cranbury
status_thirty                   Cancelled by Customer
Name: 1, dtype: object
```

In [274…
```
dataset30.shape
```

Out[274]:
```
(62421, 21)
```

In [ ]:
```
```

Verifying the if the 30 days status is correct

In [275…
```
chaves = (np.unique(dataset30['status_thirty']))

instances_summary ={}
for a in chaves:
    instances_summary[a] = []

for i in chaves:
    soma_00=(sum(valid_som_df['Status']==i))
    soma_30=(sum(dataset30['status_thirty']==i))
    #soma_60=(sum(dataset60.iloc[:,20]==i))
    #soma_90=(sum(dataset90.iloc[:,20]==i))
    instances_summary[i].append(soma_00)
    instances_summary[i].append(soma_30)
    #instances_summary[i].append(soma_60)
    #instances_summary[i].append(soma_90)
    #print(soma)
    #print(instances_summary[i])
    #print(sum(dataset.iloc[:,22]==i))

instances_summary=pd.DataFrame.from_dict(instances_summary, orient='index', columns=['
print(instances_summary)
```

	Original	30_days
Active	36012	27864
Cancelled by Buhler	5306	5008
Cancelled by Customer	11144	10660
Lost	2190	2121
Nan	0	9518
User Error	149	130
User Error/Duplicate	1077	1001
Won	6539	6119

Cleaning dataset 30 & pluging in coordinates values

In [276…
```
dataset=dataset30

#filtering only active opportunities
dataset = dataset[dataset['Status'] == 'Active']
dataset.shape
```

Out[276]:
```
(36087, 21)
```

In [277…
```
#filtering only opportunities not declared as order released = on
dataset = dataset[dataset['Last RG reached'] != 'RG4: Order released']
dataset.shape
```

Out[277]:
```
(35947, 21)
```

In [278…
```
#filtering only opportunities with a defined status in 30 days
dataset = dataset[dataset['status_thirty'] != 'Nan']
dataset.shape
```

Out[278]:
```
(30958, 21)
```

In [279…
```
#getting the unique locations dataset for the coordinates plugin

active30 = dataset.drop_duplicates(subset='Opportunity ID', keep="first")

#display(active30.iloc[1,:])
valid_locations = active30[['Opportunity ID', 'Final Ship-to Country', 'Sold-to Party

display(valid_locations.iloc[0:10,:])
```

	Opportunity ID	Final Ship-to Country	Sold-to Party Postal Code	Sold-to Party City
8	140176	MX	72304	PUEBLA
12	131915	MX	50200	Toluca
15	149104	US	15144	Springdale
16	138993	MX	52000	Toluca
20	129208	MX	11320	Mexico DF
21	134576	MX	38010	Celaya
30	126795	MX	85236	NAVOJOA
31	146929	NI	NaN	Chinandega
32	146870	PA	2296	Panama
46	146865	US	23231	Richmond

In [280…
```
#Saving Locations (dataset with opps ids and location info)

valid_locations.to_csv('valid_locations.csv', index=False)
```

Adding accounts' geographical coordinates

In [281…
```
#Reading in the Opportunity ID ad ZIP data
loc_df = pd.read_csv('opp_coordinates.csv')

loc_df.head()
```

Out[281]:

	Opportunity ID	latitude	longitude
0	129431	20.521845	-100.813961
1	129402	20.521845	-100.813961
2	144504	20.521845	-100.813961
3	165817	20.521845	-100.813961
4	152583	20.521845	-100.813961

In [282…
```
print(loc_df.shape)
```

```
(6114, 3)
```

In [283…
```
dataset_loc=pd.merge(dataset, loc_df, on='Opportunity ID', how='left')

display(dataset_loc.iloc[0,:])
```

```
Instance_ID                          14017643922
Opportunity ID                        140176
Report date                            43921
Sold-to Party                          [redacted]
Opp. Description                       [redacted]
Last RG reached                RG2: Ready to Quote
Status                                  Active
Quotation Type                             NaN
Business Type                   Project/Plant
Employee Responsible                   [redacted]
Business unit                              BI
Chance of Realization                     0.4
Chance for Bühler                         0.4
Final Ship-to Country                      MX
Exp Sales Vol in CHF                  44992.6
PlanDate: Order Rel.                    43890
Start Date                              43718
Act.Date: Order Rel.                      172
Sold-to Party Postal Code               72304
Sold-to Party City                     PUEBLA
status_thirty                          Active
latitude                              18.8333
longitude                                -98
Name: 0, dtype: object
```

In [284…
```
dataset = dataset_loc
```

Adding accounts count of opp in the past 5 years

In [285…
```
#Reading in the Opportunities count data
opportunity_count = pd.read_csv('opportunity_count.csv')

opportunity_count.head()
```

Out[285]:

	Sold-to Party	Sum_opps_by_account
0	Indium Corporation / Clinton NY 13323	1.0
1	Harmac Pacific / Nanaimo BC V9X 1J2	1.0
2	Harpoon Brewery / Boston MA 02210	1.0
3	Harris Woolf / Fresno CA 93711	1.0
4	Saratoga Chips LLC / Fort Wayne IN 46804-5677	1.0

In [286…
```
dataset = pd.merge(dataset, opportunity_count, on='Sold-to Party', how='left' )
```

Adding employees' count of opp in the past 5 years

In [287…
```
#Reading in the Employees count data
employees_count = pd.read_csv('employees_count.csv')

employees_count.head()
```

Out[287]:

	Employee Responsible	Sum_opps_by_empl
0	sales ics / Minneapolis MN 55441-4509	1.0
1	Kimberly Balsamo / Cary NC 27511	1.0
2	Mahesh C K	1.0
3	Manuel Ammann / 9240 Uzwil	1.0
4	Marcelo Leal / Santaf&#65533; de Bogot&#65533;	1.0

In [288…
```
dataset = pd.merge(dataset, employees_count, on='Employee Responsible', how='left')
dataset.shape
```

Out[288]:
```
(30958, 25)
```

In [289…
```
display(dataset.iloc[1,:])
```

```
Instance_ID                                    13191543922
Opportunity ID                                      131915
Report date                                          43921
Sold-to Party                                        [redacted]
Opp. Description                                     [redacted]
Last RG reached                          Quotation approved
Status                                               Active
Quotation Type                                          NaN
Business Type                                Customer Service
Employee Responsible                                 [redacted]
Business unit                                            SC
Chance of Realization                                   0.4
Chance for Bühler                                       0.6
Final Ship-to Country                                    MX
Exp Sales Vol in CHF                                11593.2
PlanDate: Order Rel.                                  43917
Start Date                                            43577
Act.Date: Order Rel.                                    340
Sold-to Party Postal Code                             50200
Sold-to Party City                                   Toluca
status_thirty                          Cancelled by Customer
latitude                                            19.2925
longitude                                          -99.6569
Sum_opps_by_account                                      11
Sum_opps_by_empl                                        NaN
Name: 1, dtype: object
```

In [290…
```
display(dataset.count())
```

```
Instance_ID                 30958
Opportunity ID              30958
Report date                 30958
Sold-to Party               30958
Opp. Description            30958
Last RG reached             30950
Status                      30958
Quotation Type              19020
Business Type               30958
Employee Responsible        30958
Business unit               30958
Chance of Realization       30958
Chance for Bühler           30958
Final Ship-to Country       30945
Exp Sales Vol in CHF        30958
PlanDate: Order Rel.        30958
Start Date                  30958
Act.Date: Order Rel.        29421
Sold-to Party Postal Code   30541
Sold-to Party City          30958
status_thirty               30958
latitude                    30889
longitude                   30889
Sum_opps_by_account         22761
Sum_opps_by_empl            22841
dtype: int64
```

In [291…

```
#Replacing Nan with "0"
dataset = dataset.fillna(0)

display(dataset.count())
```

```
Instance_ID                 30958
Opportunity ID              30958
Report date                 30958
Sold-to Party               30958
Opp. Description            30958
Last RG reached             30958
Status                      30958
Quotation Type              30958
Business Type               30958
Employee Responsible        30958
Business unit               30958
Chance of Realization       30958
Chance for Bühler           30958
Final Ship-to Country       30958
Exp Sales Vol in CHF        30958
PlanDate: Order Rel.        30958
Start Date                  30958
Act.Date: Order Rel.        30958
Sold-to Party Postal Code   30958
Sold-to Party City          30958
status_thirty               30958
latitude                    30958
longitude                   30958
Sum_opps_by_account         30958
Sum_opps_by_empl            30958
dtype: int64
```

Theoretically, we shouldn't lose instances...

Organizing the number of BUs -> Business units

In [292…
```
adder = 0
for p in np.unique(dataset['Business unit']):
    print("Count of", p,"opportunities is", (sum(dataset.iloc[:,10]==p)))
    adder += 1
print("\n Number of BUs: ",adder)
```

```
Count of AG opportunities is 1
Count of AN opportunities is 2709
Count of BA opportunities is 1025
Count of BD opportunities is 1092
Count of BI opportunities is 1451
Count of BS opportunities is 78
Count of DR opportunities is 1860
Count of DS opportunities is 166
Count of ES opportunities is 2
Count of FR opportunities is 40
Count of FU opportunities is 2
Count of GD opportunities is 979
Count of GH opportunities is 344
Count of GS opportunities is 235
Count of HN opportunities is 2235
Count of IT opportunities is 43
Count of LA opportunities is 35
Count of LO opportunities is 2056
Count of MB opportunities is 690
Count of MO opportunities is 1340
Count of MU opportunities is 6265
Count of MX opportunities is 273
Count of RS opportunities is 845
Count of SC opportunities is 1971
Count of SM opportunities is 2060
Count of SO opportunities is 2130
Count of WF opportunities is 1031

 Number of BUs:  27
```

In [293…
```
# Creating BAs collumn

conditions = [(dataset['Business unit']=='AG') | (dataset['Business unit']=='DR'),
              (dataset['Business unit']=='FU') | (dataset['Business unit']=='AN') | (d
              (dataset['Business unit']=='HN') | (dataset['Business unit']=='ES'),

              (dataset['Business unit']=='BA') | (dataset['Business unit']=='MX'),
              (dataset['Business unit']=='MO') | (dataset['Business unit']=='SC'),

              (dataset['Business unit']=='BI') | (dataset['Business unit']=='WF') | (d

              (dataset['Business unit']=='BD') ,
              (dataset['Business unit']=='LO') | (dataset['Business unit']=='LA') |(da
              (dataset['Business unit']=='GD') | (dataset['Business unit']=='BS') ,

              (dataset['Business unit']=='GS') | (dataset['Business unit']=='GH') | (d

              (dataset['Business unit']=='DA') | (dataset['Business unit']=='DS') | (d
              (dataset['Business unit']=='MU') | (dataset['Business unit']=='SM')
```

```
            ]

choices = ['VN', 'VN', 'VN', 'CF', 'CF', 'HAAS', 'AM','AM','AM', 'GQ', 'DT', 'MS']

dataset['BA'] = np.select(conditions, choices)
```

In [294…
```
#display(dataset.iloc[1,:])
print("last column: ",dataset.shape[1]-1)
```

```
last column:  25
```

In [295…
```
display(dataset.iloc[0,:])
```

```
Instance_ID                                14017643922
Opportunity ID                                 140176
Report date                                     43921
Sold-to Party                                  [redacted]
Opp. Description                               [redacted]
Last RG reached                  RG2: Ready to Quote
Status                                         Active
Quotation Type                                      0
Business Type                          Project/Plant
Employee Responsible                           [redacted]
Business unit                                      BI
Chance of Realization                             0.4
Chance for Bühler                                 0.4
Final Ship-to Country                              MX
Exp Sales Vol in CHF                          44992.6
PlanDate: Order Rel.                            43890
Start Date                                      43718
Act.Date: Order Rel.                              172
Sold-to Party Postal Code                       72304
Sold-to Party City                            PUEBLA
status_thirty                                  Active
latitude                                      18.8333
longitude                                         -98
Sum_opps_by_account                                 0
Sum_opps_by_empl                                    0
BA                                               HAAS
Name: 0, dtype: object
```

In [296…
```
adder = 0
for p in np.unique(dataset.iloc[:,-1]):
    print("Count of", p,"opportunities is", (sum(dataset.iloc[:,-1]==p)))
    adder += 1
print('Consolidated BAs: ',adder)
```

```
Count of AM opportunities is 4240
Count of CF opportunities is 4609
Count of DT opportunities is 2296
Count of GQ opportunities is 2114
Count of HAAS opportunities is 2522
Count of MS opportunities is 8325
Count of VN opportunities is 6852
Consolidated BAs:  7
```

In [297…
```
#verifying if we have a NaN on Ba column

zero = dataset[dataset['BA']=='0']
```

```
display(zero)
```

Instance_ID	Opportunity ID	Report date	Sold-to Party	Opp. Description	Last RG reached	Status	Quotation Type	Business Type	Emp Respo

0 rows × 26 columns

In [298…

```
# dropping the columns replaced by numeric values

dataset = dataset.drop('Sold-to Party', 1)
dataset = dataset.drop('Opp. Description', 1)
dataset = dataset.drop('Status', 1) #all instances are alredy filtered as active opps
dataset = dataset.drop('Employee Responsible', 1)
dataset = dataset.drop('Business unit', 1)

dataset = dataset.drop('Final Ship-to Country', 1)

dataset = dataset.drop('Act.Date: Order Rel.', 1)
dataset = dataset.drop('Sold-to Party Postal Code', 1)
dataset = dataset.drop('Sold-to Party City', 1)
#dataset = dataset.drop('latitude', 1)
#dataset = dataset.drop('longitude', 1)

#dataset = dataset.drop('Sum_opps_by_account', 1) #all instances are active opps
#dataset = dataset.drop('Sum_opps_by_empl', 1) #all instances are active opps


## Summary for dropings selections


##Instance_ID                  30958
##Opportunity ID               30958
##Report date                  30958
## ->Sold-to Party               30958
## -> Opp. Description            30958
## Last RG reached              30950
## -> Status                      30958
## Quotation Type               19020
## Business Type                30958
## -> Employee Responsible         30958
## -> Business unit                30958
## Chance of Realization        30958
## Chance for Bühler            30958
## -> Final Ship-to Country        30945
## Exp Sales Vol in CHF         30958
## PlanDate: Order Rel.         30958
## Start Date                   30958
## -> Act.Date: Order Rel.         29421
## -> Sold-to Party Postal Code    30541
## -> Sold-to Party City           30958
## status_thirty                30958
```

```
## -> latitude                    30889
## ->longitude                   30889
## Latitude                     30958
## Longitude                    30958
## -> Sum_opps_by_account          22761
## Opps_last_5y                 30958
## -> Sum_opps_by_empl             22841
## Opps_empl_5y                 30958
```

In [ ]:

In [299…
```
display(dataset.iloc[0,:])
```

```
Instance_ID                       14017643922
Opportunity ID                         140176
Report date                             43921
Last RG reached          RG2: Ready to Quote
Quotation Type                             0
Business Type                 Project/Plant
Chance of Realization                    0.4
Chance for Bühler                        0.4
Exp Sales Vol in CHF                 44992.6
PlanDate: Order Rel.                    43890
Start Date                              43718
status_thirty                          Active
latitude                              18.8333
longitude                                 -98
Sum_opps_by_account                        0
Sum_opps_by_empl                           0
BA                                      HAAS
Name: 0, dtype: object
```

In [300…
```
dataset.shape
```

Out[300]:
```
(30958, 17)
```

Handling NaN and Zeros

In [301…
```
#Changing quotation type '0' to 'undefined' status

conditions = [(dataset['Quotation Type'] == 0),
              (dataset['Quotation Type'] != 0)
             ]

choices = ['Undefined', dataset['Quotation Type']]

dataset['Updated_Quotation_Type'] = np.select(conditions, choices)

# dropping the outdated status thirdy

dataset = dataset.drop('Quotation Type', 1)

print(dataset.shape)

#Checking on the Updated RG data quality
```

```
for p in np.unique(dataset.iloc[:,-1]):
    print("Count of", p,"is ", (sum(dataset.iloc[:,-1] == p)))
```

```
(30958, 17)
Count of Binding Quotation is  10736
Count of Budget Quotation is  6489
Count of Price Estimation is  1795
Count of Undefined is  11938
```

In [302…
```
display(dataset.iloc[0,:])
```

```
Instance_ID                        14017643922
Opportunity ID                          140176
Report date                              43921
Last RG reached           RG2: Ready to Quote
Business Type                   Project/Plant
Chance of Realization                     0.4
Chance for Bühler                         0.4
Exp Sales Vol in CHF                  44992.6
PlanDate: Order Rel.                    43890
Start Date                              43718
status_thirty                          Active
latitude                              18.8333
longitude                                 -98
Sum_opps_by_account                         0
Sum_opps_by_empl                            0
BA                                       HAAS
Updated_Quotation_Type              Undefined
Name: 0, dtype: object
```

In [303…
```
#Changing Last RG reached '0' to 'undefined' status

conditions = [(dataset['Last RG reached'] == 0),
              (dataset['Last RG reached'] != 0)
             ]

choices = ['Undefined', dataset['Last RG reached']]

dataset['Updated_RG'] = np.select(conditions, choices)

# dropping the outdated status thirdy

dataset = dataset.drop('Last RG reached', 1)

print(dataset.shape)

#Checking on the Updated RG data quality

for p in np.unique(dataset.iloc[:,-1]):
    print("Count of", p,"is ", (sum(dataset.iloc[:,-1] == p)))
```

```
(30958, 17)
Count of Basic Concept approved is  4407
Count of Quotation approved is  12730
Count of RG1: Feasibility checked is  5456
Count of RG2: Ready to Quote is  8054
Count of RG3: Handshake received is  303
Count of Undefined is  8
```

In [304…
```
dataset = dataset[dataset['Updated_RG']!="Undefined"]
```

```
dataset.shape
```

Out[304]: (30950, 17)

In [305…
```
#Checking on the Updated RG data quality

for p in np.unique(dataset['Updated_RG']):
    print("Count of", p,"is ", (sum(dataset['Updated_RG'] == p)))
```

```
Count of Basic Concept approved is  4407
Count of Quotation approved is  12730
Count of RG1: Feasibility checked is  5456
Count of RG2: Ready to Quote is  8054
Count of RG3: Handshake received is  303
```

In [306…
```
#Checking on the business type data quality (we don't want '0' or Nan)

for p in np.unique(dataset['Business Type']):
    print("Count of", p,"opp group", (sum(dataset['Business Type']==p)))
```

```
Count of Customer Service opp group 6273
Count of Project/Plant opp group 12767
Count of Single Machine opp group 11910
```

In [307…
```
display(dataset.iloc[1,:])
```

```
Instance_ID                          13191543922
Opportunity ID                           131915
Report date                               43921
Business Type                   Customer Service
Chance of Realization                        0.4
Chance for Bühler                            0.6
Exp Sales Vol in CHF                     11593.2
PlanDate: Order Rel.                      43917
Start Date                                43577
status_thirty             Cancelled by Customer
latitude                                19.2925
longitude                              -99.6569
Sum_opps_by_account                          11
Sum_opps_by_empl                              0
BA                                           CF
Updated_Quotation_Type                Undefined
Updated_RG                    Quotation approved
Name: 1, dtype: object
```

Calculating opportunity age

In [308…
```
dataset['opp_age'] = dataset['Report date'] - dataset ['Start Date']

display(dataset.iloc[0:10,[1,2,-1]])
```

	Opportunity ID	Report date	opp_age
0	140176	43921	203
1	131915	43921	344
2	149104	43921	46
3	138993	43921	224
4	129208	43921	385
5	134576	43921	298
6	126795	43921	425
7	146929	43921	78
8	146870	43921	81
9	146865	43921	81

Creating the final version of opportunity status in 30 days

In [309…
```
#Checking on the opportunity group data quality

for p in np.unique(dataset['status_thirty']):
    print("Count of", p,"is ", (sum(dataset['status_thirty']==p)))
```

```
Count of Active is  27817
Count of Cancelled by Buhler is  541
Count of Cancelled by Customer is  1331
Count of Lost is  312
Count of User Error is  23
Count of User Error/Duplicate is  89
Count of Won is  837
```

In [310…
```
# Final Status Thirty

conditions = [(dataset['status_thirty']=='User Error') | (dataset['status_thirty']=='U
              (dataset['status_thirty']=='Cancelled by Buhler') | (dataset['status_thi
              (dataset['status_thirty']=='Active'),
              (dataset['status_thirty']=='Won')
             ]

choices = ['Lost or Cancelled', "Active", 'Won']

dataset['Final_Status'] = np.select(conditions, choices)

# dropping the outdated status thirdy

dataset = dataset.drop('status_thirty', 1)

dataset.shape
```

Out[310]:
```
(30950, 18)
```

In [311…
```
#Checking on the status_30 data quality
```

```
for p in np.unique(dataset['Final_Status']):
    print("Count of", p,"is ", (sum(dataset['Final_Status']==p)))
```

```
Count of Active is  27817
Count of Lost or Cancelled is  2296
Count of Won is  837
```

In [312…
```
display(dataset.iloc[0,:])
```

```
Instance_ID                       14017643922
Opportunity ID                         140176
Report date                             43921
Business Type                   Project/Plant
Chance of Realization                     0.4
Chance for Bühler                         0.4
Exp Sales Vol in CHF                  44992.6
PlanDate: Order Rel.                    43890
Start Date                              43718
latitude                              18.8333
longitude                                 -98
Sum_opps_by_account                         0
Sum_opps_by_empl                            0
BA                                       HAAS
Updated_Quotation_Type             Undefined
Updated_RG                RG2: Ready to Quote
opp_age                                   203
Final_Status                           Active
Name: 0, dtype: object
```

In [313…
```
print(dataset.columns)
```

```
Index(['Instance_ID', 'Opportunity ID', 'Report date', 'Business Type',
       'Chance of Realization', 'Chance for Bühler', 'Exp Sales Vol in CHF',
       'PlanDate: Order Rel.', 'Start Date', 'latitude', 'longitude',
       'Sum_opps_by_account', 'Sum_opps_by_empl', 'BA',
       'Updated_Quotation_Type', 'Updated_RG', 'opp_age', 'Final_Status'],
      dtype='object')
```

In [314…
```
dataset.to_csv('dataset_som_training_30d_20220212_full.csv', index=False)
```

Dropping duplicates

this is a crucial step: should we pick the first or the last active opportunities? We'll pick the last

In [315…
```
#dropping duplicates

#Checking on the opportunity group data quality

dataset = pd.read_csv('dataset_som_training_30d_20220212_full.csv', low_memory=False)

for p in np.unique(dataset.iloc[:,-1]):
    print("Count of", p,"is ", (sum(dataset.iloc[:,-1]==p)))
```

```
Count of Active is  27817
Count of Lost or Cancelled is  2296
Count of Won is  837
```

In [316…
```
#dropping instances where opp ID and opp Final status are identical - this should redu
```

```
dataset = dataset.sort_values(by=['Opportunity ID','Report date'])

clean_dataset = dataset.drop_duplicates(
    subset = ['Opportunity ID', 'Final_Status'],
      keep = 'last'
            ).reset_index(drop = True)
```

In [317…
```
#Checking on the opportunity group data quality
last_column=(clean_dataset.shape[1]-1)
for p in np.unique(clean_dataset.iloc[:,last_column]):
    print("Count of", p,"is ", (sum(clean_dataset.iloc[:,last_column]==p)))
```

```
Count of Active is  5417
Count of Lost or Cancelled is  2292
Count of Won is  837
```

In [318…
```
dataset = clean_dataset
dataset.to_csv('dataset_som_training_30d_20220130_unique.csv', index=False)
dataset.shape
```

Out[318]:
```
(8546, 18)
```

In [319…
```
display(dataset.count())
```

```
Instance_ID               8546
Opportunity ID            8546
Report date               8546
Business Type             8546
Chance of Realization     8546
Chance for Bühler         8546
Exp Sales Vol in CHF      8546
PlanDate: Order Rel.      8546
Start Date                8546
latitude                  8546
longitude                 8546
Sum_opps_by_account       8546
Sum_opps_by_empl          8546
BA                        8546
Updated_Quotation_Type    8546
Updated_RG                8546
opp_age                   8546
Final_Status              8546
dtype: int64
```

In [320…
```
#removing again Nas
dataset_fill = dataset.fillna(0)
```

In [321…
```
#display(dataset_na.count())
display(dataset_fill.count())
```

```
Instance_ID               8546
Opportunity ID            8546
Report date               8546
Business Type             8546
Chance of Realization     8546
Chance for Bühler         8546
Exp Sales Vol in CHF      8546
PlanDate: Order Rel.      8546
Start Date                8546
latitude                  8546
longitude                 8546
Sum_opps_by_account       8546
Sum_opps_by_empl          8546
BA                        8546
Updated_Quotation_Type    8546
Updated_RG                8546
opp_age                   8546
Final_Status              8546
dtype: int64
```

In [322…
```
#dataset = dataset_fill
dataset.to_csv('20220214_dataset_som_clean.csv', index=False)
```

In [ ]:
```

```

Applying ML models

In [323…
```
dataset = pd.read_csv('20220214_dataset_som_clean.csv', low_memory=False)


# Assign X and y
X = dataset.iloc[:, 2:-1].values
y = dataset.iloc[:, -1].values
```

In [324…
```
display(dataset.iloc[0,:])
```

```
Instance_ID                         380744105
Opportunity ID                           3807
Report date                             44101
Business Type                  Project/Plant
Chance of Realization                       1
Chance for Bühler                         0.6
Exp Sales Vol in CHF                   617410
PlanDate: Order Rel.                    44165
Start Date                              39867
latitude                              44.4762
longitude                            -73.2129
Sum_opps_by_account                         0
Sum_opps_by_empl                          129
BA                                         CF
Updated_Quotation_Type       Budget Quotation
Updated_RG                RG2: Ready to Quote
opp_age                                  4234
Final_Status                           Active
Name: 0, dtype: object
```

In [325…
```
print(X[0:2,:])
```

```
[[44101 'Project/Plant' 1.0 0.6 617410.0 44165 39867 44.4761601
  -73.212906 0.0 129.0 'CF' 'Budget Quotation' 'RG2: Ready to Quote' 4234]
 [44191 'Project/Plant' 0.2 0.4 1647100.0 44499 40350 20.967075899999998
  -89.6237402 0.0 0.0 'CF' 'Budget Quotation' 'Basic Concept approved'
  3841]]
```

In [326…

```
#Dealing with categorical variables
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder


#Dealing with categorical variables Business Type
print(X[0,1])

col_trans = make_column_transformer((OneHotEncoder(), [1]), remainder='passthrough')
X = col_trans.fit_transform(X)
print(X[0,:])
print()
X = np.delete(X,0,1)
print(X[0,:])
print()

#Dealing with categorical variables Business Areas
print(X[0,-4])

col_trans = make_column_transformer((OneHotEncoder(), [-4]), remainder='passthrough')
X = col_trans.fit_transform(X)
print(X[0,:])
print()
X = np.delete(X,0,1)
print(X[0,:])
print()


#Dealing with categorical variables Updated_Quotation_Type
print(X[0,-3])

col_trans = make_column_transformer((OneHotEncoder(), [-3]), remainder='passthrough')
X = col_trans.fit_transform(X)
print(X[0,:])
print()
X = np.delete(X,0,1)
print(X[0,:])
print()

#Dealing with categorical variables Updated_RG
print(X[0,-2])
col_trans = make_column_transformer((OneHotEncoder(), [-2]), remainder='passthrough')
X = col_trans.fit_transform(X)
print(X[0,:])
print()
X = np.delete(X,0,1)
print(X[0,:])
print()
```

```
Project/Plant
[0.0 1.0 0.0 44101 1.0 0.6 617410.0 44165 39867 44.4761601 -73.212906 0.0
 129.0 'CF' 'Budget Quotation' 'RG2: Ready to Quote' 4234]

[1.0 0.0 44101 1.0 0.6 617410.0 44165 39867 44.4761601 -73.212906 0.0
 129.0 'CF' 'Budget Quotation' 'RG2: Ready to Quote' 4234]

CF
[0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 44101 1.0 0.6 617410.0 44165 39867
 44.4761601 -73.212906 0.0 129.0 'Budget Quotation' 'RG2: Ready to Quote'
 4234]

[1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 44101 1.0 0.6 617410.0 44165 39867
 44.4761601 -73.212906 0.0 129.0 'Budget Quotation' 'RG2: Ready to Quote'
 4234]

Budget Quotation
[0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 44101 1.0 0.6 617410.0
 44165 39867 44.4761601 -73.212906 0.0 129.0 'RG2: Ready to Quote' 4234]

[1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 44101 1.0 0.6 617410.0 44165
 39867 44.4761601 -73.212906 0.0 129.0 'RG2: Ready to Quote' 4234]

RG2: Ready to Quote
[0.0 0.0 0.0 1.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 44101 1.0
 0.6 617410.0 44165 39867 44.4761601 -73.212906 0.0 129.0 4234]

[0.0 0.0 1.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 44101 1.0 0.6
 617410.0 44165 39867 44.4761601 -73.212906 0.0 129.0 4234]
```

In [ ]:

In [327…
```
X.shape
```

Out[327]: (8546, 26)

In [328…
```
#Dependent variable ->y
from sklearn.preprocessing import LabelEncoder
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

print(np.unique(y))
```

```
[0 1 2]
```

In [329…
```
# save numpy array as csv file
from numpy import asarray
from numpy import savetxt
# define data
#data = asarray([[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]])
# save to csv file
#savetxt('data.csv', data, delimiter=',')

savetxt('20220212_X.csv',X,delimiter=',')

savetxt('20220212_y.csv',y,delimiter=',')
```

In [330…
```
# Splitting the data into Training Set and Test Set, stratified on 'quality'
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [331…
```
#X_train.dtype
```

In [332…
```
#Normalizing the features
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

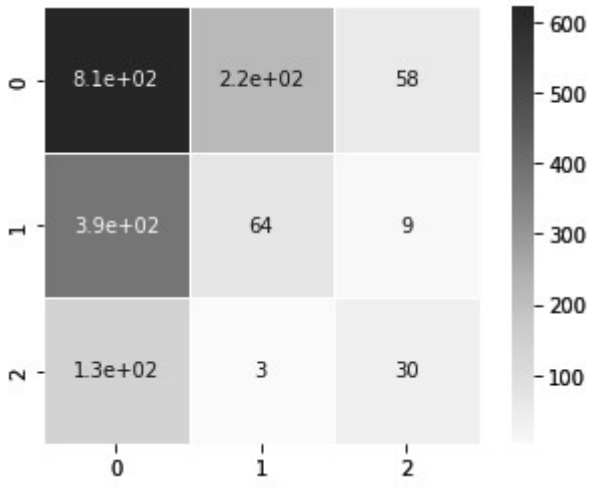
In [333…
```
#X_train
```

In [334…
```
# Random Forest Classifier Grid Search
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

rfObj = RandomForestClassifier()

grid_param = {
    'n_estimators': [10, 50, 100, 150, 200],    # Previously found to be around 200
    'criterion': ['gini', 'entropy'],
    'bootstrap': [True, False],
    'class_weight': ['balanced', 'balanced_subsample'],
    'random_state' : [0]
}

gd_sr = GridSearchCV(estimator=rfObj, param_grid=grid_param, cv=5, n_jobs=-1,
                     scoring='balanced_accuracy')
gd_sr.fit(X_train, y_train)

print("Best parameters:", gd_sr.best_params_)
```

```
Best parameters: {'bootstrap': True, 'class_weight': 'balanced', 'criterion': 'entrop
y', 'n_estimators': 10, 'random_state': 0}
```

In [335…
```
# First stage of y prediction: 5, 6, or other(0)

y_pred = gd_sr.predict(X_test)
print("Classification Report")
print(classification_report(y_test, y_pred))


import seaborn as sns

CFM = confusion_matrix(y_test , y_pred)
matrix=sns.heatmap(CFM, linewidths=1,vmax=622,
                   square=True, cmap="Blues",annot=True)
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.61      0.74      0.67      1084
           1       0.22      0.14      0.17       459
           2       0.31      0.18      0.23       167

    accuracy                           0.53      1710
   macro avg       0.38      0.35      0.36      1710
weighted avg       0.48      0.53      0.49      1710
```



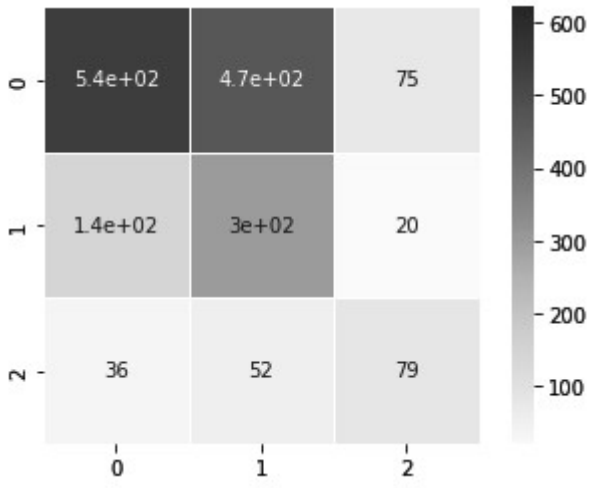
In [336…
```
# Save Model Using Pickle
import pickle
filename = '20220213_forest_classifier.sav'
pickle.dump(gd_sr, open(filename, 'wb'))
```

In [ ]:
```
```

In [337…
```
#Naive Bayes
#Fitting Classifier to Training Set. Create a classifier object here and call it class
from sklearn.naive_bayes import GaussianNB
classifierObj = GaussianNB()
classifierObj.fit(X_train , y_train)

#Making predictions on the Test Set
y_pred = classifierObj.predict(X_test)
```

In [338…
```
print("Classification Report")
print(classification_report(y_test, y_pred))

CFM = confusion_matrix(y_test , y_pred)
matrix=sns.heatmap(CFM, linewidths=1,vmax=622,
                   square=True, cmap="Blues",annot=True)
```

```
Classification Report
              precision    recall  f1-score   support

           0       0.75      0.50      0.60      1084
           1       0.36      0.65      0.47       459
           2       0.45      0.47      0.46       167

    accuracy                           0.54      1710
   macro avg       0.52      0.54      0.51      1710
weighted avg       0.62      0.54      0.55      1710
```



In [ ]: