#Business Case:

ABC Tech is an mid-size organisation operation in IT-enabled business segment over a decade. On an average ABC Tech receives 22-25k IT incidents/tickets , which were handled to best practice ITIL framework with incident management , problem management, change management and configuration management processes. These ITIL practices attained matured process level and a recent audit confirmed that further improvement initiatives may not yield return of investment.

ABC Tech management is looking for ways to improve the incident management process as recent customer survey results shows that incident management is rated as poor. Machine Learning as way to improve ITSM processes ABC Tech management recently attended Machine Learning conference on ML for ITSM. Machine learning looks prospective to improve ITSM processes through prediction and automation. They came up with 4 key areas, where ML can help ITSM process in ABC Tech.

1. Predicting High Priority Tickets: To predict priority 1 & 2 tickets, so that they can take preventive measures or fix the problem before it surfaces.
2. Forecast the incident volume in different fields , quarterly and annual. So that they can be better prepared with resources and technology planning.
3. Auto tag the tickets with right priorities and right departments so that reassigning and related delay can be reduced.
4. Predict RFC (Request for change) and possible failure / misconfiguration of ITSM assets.

#importing neccessory libraries

```
#basic modules
import mysql.connector
from mysql.connector import Error

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import datetime
import pickle

import warnings
warnings.filterwarnings('ignore')

#sklearn modules
##data preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
```

```python
##model creation
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import
RandomForestClassifier,BaggingClassifier,GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from xgboost import XGBClassifier

from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.arima.model import ARIMA


#model evaluation
from sklearn.metrics import
confusion_matrix,classification_report,ConfusionMatrixDisplay,f1_score
,recall_score,accuracy_score

encoder=LabelEncoder()
```

#basic checks

```python
host = "18.136.157.135"
username='dm_team'
database = 'project_itsm'
password='DM!$Team@&27920!'

try:
    connection = mysql.connector.connect(host=host,
                                        database = database,
                                         user=username,
                                         password=password)
    if connection.is_connected():
        print(f"Connected to MySql:{host}")

        sql_query = "Select * from dataset_list"

        df = pd.read_sql_query(sql_query,connection)

        display(df)

except Error as err:
    print(f"Error:{err}")
finally:
    if connection.is_connected():
        connection.close()
        print("Connection is closed")

df.to_csv("Ticket.csv", index=False)
```

```
Connected to MySql:18.136.157.135

          CI_Name          CI_Cat                     CI_Subcat        WBS  \
0      SUB000508  subapplication     Web Based Application  WBS000162   
1      WBA000124     application     Web Based Application  WBS000088   
2      DTA000024     application       Desktop Application  WBS000092   
3      WBA000124     application     Web Based Application  WBS000088   
4      WBA000124     application     Web Based Application  WBS000088   
...          ...             ...                       ...        ...   
46601  SBA000464     application  Server Based Application  WBS000073   
46602  SBA000461     application  Server Based Application  WBS000073   
46603  LAP000019        computer                    Laptop  WBS000091   
46604  WBA000058     application     Web Based Application  WBS000073   
46605  DCE000077        hardware       DataCenterEquipment  WBS000267   

      Incident_ID  Status Impact Urgency Priority   number_cnt  ...  \
0       IM0000004  Closed      4       4        4  0.601292279  ...   
1       IM0000005  Closed      3       3        3  0.415049969  ...   
2       IM0000006  Closed     NS       3       NA  0.517551335  ...   
3       IM0000011  Closed      4       4        4  0.642927218  ...   
4       IM0000012  Closed      4       4        4  0.345258343  ...   
...           ...     ...    ...     ...      ...          ...  ...   
46601   IM0047053  Closed      4       4        4   0.23189604  ...   
46602   IM0047054  Closed      4       4        4  0.805153085  ...   
46603   IM0047055  Closed      5       5        5  0.917466294  ...   
46604   IM0047056  Closed      4       4        4  0.701278158  ...   
46605   IM0047057  Closed      3       3        3  0.902319509  ...   

            Reopen_Time     Resolved_Time        Close_Time  
Handle_Time_hrs  \
0                        04-11-2013 13:50  04-11-2013 13:51   
3,87,16,91,111   
1      02-12-2013 12:31  02-12-2013 12:36  02-12-2013 12:36   
4,35,47,86,389   
2                        13-01-2014 15:12  13-01-2014 15:13   
4,84,31,19,444   
```

```
3                         14-11-2013 09:31  14-11-2013 09:31
4,32,18,33,333
4                         08-11-2013 13:55  08-11-2013 13:55
3,38,39,03,333
...                 ...               ...               ...
...
46601                     31-03-2014 16:29  31-03-2014 16:29
0,095
46602                     31-03-2014 15:29  31-03-2014 15:29
0,428333333
46603                     31-03-2014 15:32  31-03-2014 15:32
0,071666667
46604                     31-03-2014 15:42  31-03-2014 15:42
0,116944444
46605                     31-03-2014 22:47  31-03-2014 22:47
0,586388889

                       Closure_Code No_of_Related_Interactions  \
0                             Other                         1
1                          Software                         1
2      No error - works as designed                         1
3                    Operator error                         1
4                             Other                         1
...                             ...                       ...
46601                         Other                         1
46602                    User error                         1
46603                      Hardware                         1
46604                      Software                         1
46605                      Hardware                         1

      Related_Interaction No_of_Related_Incidents
No_of_Related_Changes  \
0                SD0000007                       2

1                SD0000011                       1

2                SD0000017

3                SD0000025

4                SD0000029

...                    ...                     ...                     ..
.
46601            SD0147021

46602            SD0146967

46603            SD0146982
```

```
46604            SD0146986

46605            SD0147088


      Related_Change
0
1
2
3
4
...                 ...
46601
46602
46603
46604
46605

[46606 rows x 25 columns]

Connection is closed

df = pd.read_csv('Ticket.csv')

df = df.replace('', pd.NA)

pd.set_option('display.max_columns',None)

df.head()

     CI_Name          CI_Cat              CI_Subcat        WBS
Incident_ID  \
0  SUB000508  subapplication  Web Based Application  WBS000162
IM0000004
1  WBA000124     application  Web Based Application  WBS000088
IM0000005
2  DTA000024     application    Desktop Application  WBS000092
IM0000006
3  WBA000124     application  Web Based Application  WBS000088
IM0000011
4  WBA000124     application  Web Based Application  WBS000088
IM0000012

   Status Impact Urgency  Priority  number_cnt
Category  \
0  Closed      4       4       4.0    0.601292
incident
1  Closed      3       3       3.0    0.415050
incident
2  Closed     NS       3       NaN    0.517551  request for
information
```

```
3  Closed      4       4       4.0    0.642927
incident
4  Closed      4       4       4.0    0.345258
incident

   KB_number Alert_Status  No_of_Reassignments         Open_Time  \
0  KM0000553       closed                 26.0  05-02-2012 13:32
1  KM0000611       closed                 33.0  12-03-2012 15:44
2  KM0000339       closed                  3.0  29-03-2012 12:36
3  KM0000611       closed                 13.0  17-07-2012 11:49
4  KM0000611       closed                  2.0  10-08-2012 11:01

        Reopen_Time     Resolved_Time        Close_Time
Handle_Time_hrs  \
0               NaN  04-11-2013 13:50  04-11-2013 13:51
3,87,16,91,111
1  02-12-2013 12:31  02-12-2013 12:36  02-12-2013 12:36
4,35,47,86,389
2               NaN  13-01-2014 15:12  13-01-2014 15:13
4,84,31,19,444
3               NaN  14-11-2013 09:31  14-11-2013 09:31
4,32,18,33,333
4               NaN  08-11-2013 13:55  08-11-2013 13:55
3,38,39,03,333

                   Closure_Code  No_of_Related_Interactions  \
0                         Other                         1.0
1                      Software                         1.0
2  No error - works as designed                         1.0
3                Operator error                         1.0
4                         Other                         1.0

  Related_Interaction  No_of_Related_Incidents  No_of_Related_Changes
\
0           SD0000007                      2.0                    NaN

1           SD0000011                      1.0                    NaN

2           SD0000017                      NaN                    NaN

3           SD0000025                      NaN                    NaN

4           SD0000029                      NaN                    NaN


  Related_Change
0            NaN
1            NaN
2            NaN
```

```
3           NaN
4           NaN
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46606 entries, 0 to 46605
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   CI_Name                     46606 non-null  object
 1   CI_Cat                      46495 non-null  object
 2   CI_Subcat                   46495 non-null  object
 3   WBS                         46606 non-null  object
 4   Incident_ID                 46606 non-null  object
 5   Status                      46606 non-null  object
 6   Impact                      46606 non-null  object
 7   Urgency                     46606 non-null  object
 8   Priority                    45226 non-null  float64
 9   number_cnt                  46606 non-null  float64
 10  Category                    46606 non-null  object
 11  KB_number                   46606 non-null  object
 12  Alert_Status                46606 non-null  object
 13  No_of_Reassignments         46605 non-null  float64
 14  Open_Time                   46606 non-null  object
 15  Reopen_Time                 2284 non-null   object
 16  Resolved_Time               44826 non-null  object
 17  Close_Time                  46606 non-null  object
 18  Handle_Time_hrs             46605 non-null  object
 19  Closure_Code                46146 non-null  object
 20  No_of_Related_Interactions  46492 non-null  float64
 21  Related_Interaction         46606 non-null  object
 22  No_of_Related_Incidents     1222 non-null   float64
 23  No_of_Related_Changes       560 non-null    float64
 24  Related_Change              560 non-null    object
dtypes: float64(6), object(19)
memory usage: 8.9+ MB
```

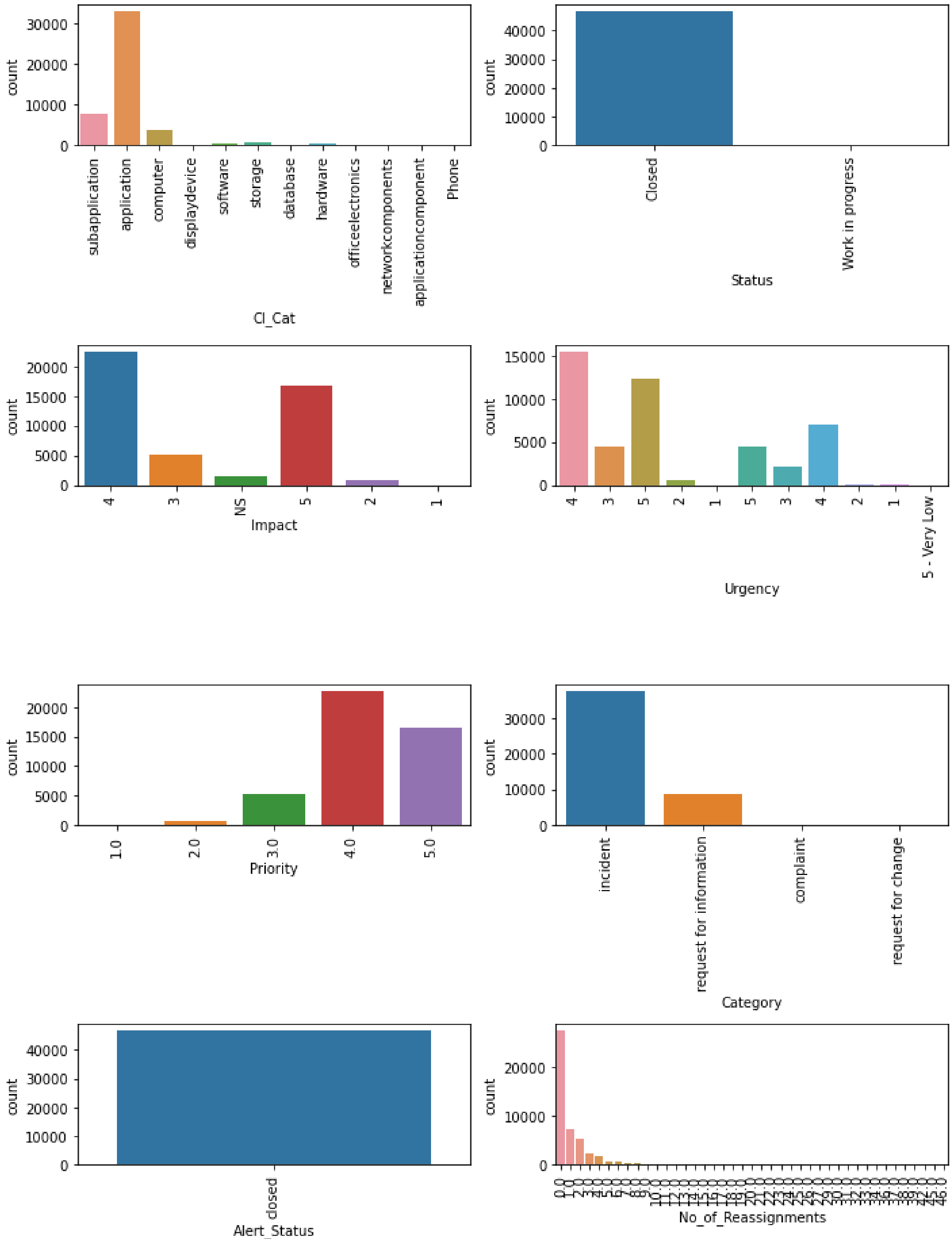
```
exclude_columns
=['CI_Name','CI_Subcat','WBS','Incident_ID','number_cnt','KB_number','
Open_Time','Reopen_Time','Resolved_Time',

'Close_Time','Handle_Time_hrs','Related_Interaction','No_of_Related_In
cidents','No_of_Related_Changes','Related_Change']
print(len([column for column in df.columns if column not in
exclude_columns]))
print('\n')
print([column for column in df.columns if column not in
exclude_columns])
```

```
10
```

```
['CI_Cat', 'Status', 'Impact', 'Urgency', 'Priority', 'Category',
'Alert_Status', 'No_of_Reassignments', 'Closure_Code',
'No_of_Related_Interactions']
```

#EDA

```
pl_no=1
plt.figure(figsize=(10,18))
for i in [column for column in df.columns if column not in
exclude_columns]:

  plt.subplot(5,2,pl_no)
  sns.countplot(x=i,data=df)
  plt.xlabel(i)
  plt.xticks(rotation=90)
  pl_no+=1
plt.tight_layout()
```



count
CI_Cat
subapplication
application
computer
displaydevice
software
storage
database
hardware
officeelectronics
networkcomponents
applicationcomponent
Phone
Status
Closed
Work in progress
Impact
NS
Urgency
5 - Very Low
Priority
Category
incident
request for information
complaint
request for change
Alert_Status
closed
No_of_Reassignments

##Insight-1

- in `CI_cat`, that is in the department section of the dataset, it is found that the `application` is having more count compared to others
- Th `Status` of almost all of tickets is in closed state
- In the `impact`,`urgency` and `priorty` columns most of the tickets are having imapct and urgency of either 4 or 5
- and the most of the tickets are belonging to the `incident` category
- `No_of_Reassignments` column indicating that most of the tickets solved at `first assignment` and also `some entries are there having reassigned many times`
- `others` and `software` were indicated as the major closure code after the tikcet resloving

---

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46606 entries, 0 to 46605
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   CI_Name                     46606 non-null  object 
 1   CI_Cat                      46495 non-null  object 
 2   CI_Subcat                   46495 non-null  object 
 3   WBS                         46606 non-null  object 
 4   Incident_ID                 46606 non-null  object 
 5   Status                      46606 non-null  object 
 6   Impact                      46606 non-null  object 
 7   Urgency                     46606 non-null  object 
 8   Priority                    45226 non-null  float64
 9   number_cnt                  46606 non-null  float64
 10  Category                    46606 non-null  object 
 11  KB_number                   46606 non-null  object 
 12  Alert_Status                46606 non-null  object 
 13  No_of_Reassignments         46605 non-null  float64
 14  Open_Time                   46606 non-null  object 
 15  Reopen_Time                 2284 non-null   object 
 16  Resolved_Time               44826 non-null  object 
 17  Close_Time                  46606 non-null  object 
 18  Handle_Time_hrs             46605 non-null  object 
 19  Closure_Code                46146 non-null  object 
 20  No_of_Related_Interactions  46492 non-null  float64
 21  Related_Interaction         46606 non-null  object 
 22  No_of_Related_Incidents     1222 non-null   float64
 23  No_of_Related_Changes       560 non-null    float64
 24  Related_Change              560 non-null    object 
dtypes: float64(6), object(19)
memory usage: 8.9+ MB
```

- **converting the `null count to the percentage` of the null values present for better handling purpuse**

```
null_df=pd.DataFrame((df.isnull().sum()/len(df))*100,columns=['per'])
null_df['count']=df.isnull().sum()
null_df
```

	per	count
CI_Name	0.000000	0
CI_Cat	0.238167	111
CI_Subcat	0.238167	111
WBS	0.000000	0
Incident_ID	0.000000	0
Status	0.000000	0
Impact	0.000000	0
Urgency	0.000000	0
Priority	2.960992	1380
number_cnt	0.000000	0
Category	0.000000	0
KB_number	0.000000	0
Alert_Status	0.000000	0
No_of_Reassignments	0.002146	1
Open_Time	0.000000	0
Reopen_Time	95.099343	44322
Resolved_Time	3.819251	1780
Close_Time	0.000000	0
Handle_Time_hrs	0.002146	1
Closure_Code	0.986997	460
No_of_Related_Interactions	0.244604	114
Related_Interaction	0.000000	0
No_of_Related_Incidents	97.378020	45384
No_of_Related_Changes	98.798438	46046
Related_Change	98.798438	46046

droppping the `columns which are above 50%`null values

```
null_df[null_df['per']>50]
```

	per	count
Reopen_Time	95.099343	44322
No_of_Related_Incidents	97.378020	45384
No_of_Related_Changes	98.798438	46046
Related_Change	98.798438	46046

```
df.drop(null_df[null_df['per']>50].index,axis=1,inplace=True)

for i in df.columns:
    if df[i].dtype=='object':

        print(f'{i} {len(df[i].unique())}')
        print('----'*5)
```

```
CI_Name 3019
--------------------
CI_Cat 13
--------------------
CI_Subcat 65
--------------------
WBS 274
--------------------
Incident_ID 46606
--------------------
Status 2
--------------------
Impact 6
--------------------
Urgency 11
--------------------
Category 4
--------------------
KB_number 1825
--------------------
Alert_Status 1
--------------------
Open_Time 34636
--------------------
Resolved_Time 33628
--------------------
Close_Time 34528
--------------------
Handle_Time_hrs 30639
--------------------
Closure_Code 15
--------------------
Related_Interaction 43060
--------------------
```

#check for unique values

---

checking all the unique values of categorical columns

fixing the number 66 after knowing the maximum length of discrete columns

```python
for i in df.columns:
    if df[i].dtype=='object':
        if len(df[i].unique())<66:
            print(f'{i} -----> {df[i].unique()}')
            print('----'*10)
```

```
CI_Cat -----> ['subapplication' 'application' 'computer' nan
'displaydevice' 'software'
 'storage' 'database' 'hardware' 'officeelectronics'
'networkcomponents'
 'applicationcomponent' 'Phone']
----------------------------------------
CI_Subcat -----> ['Web Based Application' 'Desktop Application'
'Server Based Application'
 'SAP' 'Client Based Application' 'Citrix' 'Standard Application'
 'Windows Server' 'Laptop' 'Linux Server' nan 'Monitor'
 'Automation Software' 'SAN' 'Banking Device' 'Desktop' 'Database'
 'Oracle Server' 'Keyboard' 'Printer' 'Exchange' 'System Software'
'VDI'
 'Encryption' 'Omgeving' 'MigratieDummy' 'Scanner' 'Controller'
 'DataCenterEquipment' 'KVM Switches' 'Switch' 'Database Software'
 'Network Component' 'Unix Server' 'Lines' 'ESX Cluster' 'zOS Server'
 'SharePoint Farm' 'NonStop Server' 'Application Server'
 'Security Software' 'Thin Client' 'zOS Cluster' 'Router' 'VMWare'
 'Net Device' 'Neoview Server' 'MQ Queue Manager' 'UPS' 'Number'
 'Iptelephony' 'Windows Server in extern beheer' 'Modem' 'X86 Server'
 'ESX Server' 'Virtual Tape Server' 'IPtelephony' 'NonStop Harddisk'
 'Firewall' 'RAC Service' 'zOS Systeem' 'Instance' 'NonStop Storage'
 'Protocol' 'Tape Library']
----------------------------------------
Status -----> ['Closed' 'Work in progress']
----------------------------------------
Impact -----> ['4' '3' 'NS' '5' '2' '1']
----------------------------------------
Urgency -----> [4 3 5 2 1 '5' '3' '4' '2' '1' '5 - Very Low']
----------------------------------------
Category -----> ['incident' 'request for information' 'complaint'
'request for change']
----------------------------------------
Alert_Status -----> ['closed']
----------------------------------------
Closure_Code -----> ['Other' 'Software' 'No error - works as designed'
'Operator error'
 'Unknown' 'Data' 'Referred' 'Hardware' 'Questions' 'User error'
'Inquiry'
 'User manual not used' 'Kwaliteit van de output' nan 'Overig']
----------------------------------------
```

#data preprocessing column by column

- ML algorithms will work well if first undertood the data well, so im doing this way to understand the data well to preprocess the well
- preprocessing the data column by column for better cleaning of data
- in the preprocessing the stages follwing these steps
    a. null value imputation
    b. label encoding
    c. force typecating
    d. dropping the unnessesory columns

  and other required preprocessing steps

---

df['CI_Name']

- as name doesnot impact on ticket priority dropping the name column

---

```
df.drop('CI_Name',axis=1,inplace=True)

df.head()

           CI_Cat               CI_Subcat       WBS Incident_ID
Status  \
0  subapplication  Web Based Application  WBS000162   IM0000004
Closed
1     application  Web Based Application  WBS000088   IM0000005
Closed
2     application    Desktop Application  WBS000092   IM0000006
Closed
3     application  Web Based Application  WBS000088   IM0000011
Closed
4     application  Web Based Application  WBS000088   IM0000012
Closed

  Impact Urgency  Priority  number_cnt                 Category
KB_number  \
0      4       4       4.0    0.601292                 incident
KM0000553
```

```
1      3       3       3.0    0.415050                 incident
KM0000611
2     NS       3       NaN    0.517551  request for information
KM0000339
3      4       4       4.0    0.642927                 incident
KM0000611
4      4       4       4.0    0.345258                 incident
KM0000611

  Alert_Status  No_of_Reassignments         Open_Time
Resolved_Time  \
0       closed                 26.0  05-02-2012 13:32  04-11-2013
13:50
1       closed                 33.0  12-03-2012 15:44  02-12-2013
12:36
2       closed                  3.0  29-03-2012 12:36  13-01-2014
15:12
3       closed                 13.0  17-07-2012 11:49  14-11-2013
09:31
4       closed                  2.0  10-08-2012 11:01  08-11-2013
13:55

         Close_Time Handle_Time_hrs                 Closure_Code  \
0  04-11-2013 13:51  3,87,16,91,111                        Other
1  02-12-2013 12:36  4,35,47,86,389                     Software
2  13-01-2014 15:13  4,84,31,19,444  No error - works as designed
3  14-11-2013 09:31  4,32,18,33,333               Operator error
4  08-11-2013 13:55  3,38,39,03,333                        Other

   No_of_Related_Interactions Related_Interaction
0                         1.0           SD0000007
1                         1.0           SD0000011
2                         1.0           SD0000017
3                         1.0           SD0000025
4                         1.0           SD0000029
```

df['CI_Cat']

```
df['CI_Cat'].value_counts()
```

```
application            32900
subapplication          7782
computer                3643
storage                  703
hardware                 442
software                 333
database                 214
displaydevice            212
officeelectronics        152
networkcomponents        107
```

```
applicationcomponent          5
Phone                         2
Name: CI_Cat, dtype: int64
```

```
df['CI_Cat'].isnull().sum()
```

```
111
```

```
df.loc[df['CI_Cat'].isnull(),'CI_Cat']='application'
```

```
df['CI_Cat'].value_counts()
```

```
application               33011
subapplication             7782
computer                   3643
storage                     703
hardware                    442
software                    333
database                    214
displaydevice               212
officeelectronics           152
networkcomponents           107
applicationcomponent          5
Phone                         2
Name: CI_Cat, dtype: int64
```

- transforming the columns from categorical columns to numerical columns using `label_encoder`

```
df['CI_Cat']=encoder.fit_transform(df['CI_Cat'])
```

```
data.head()
```

```
   CI_Cat              CI_Subcat        WBS Incident_ID  Status Impact  \
0      11  Web Based Application  WBS000162   IM0000004  Closed      4

1       1  Web Based Application  WBS000088   IM0000005  Closed      3

2       1    Desktop Application  WBS000092   IM0000006  Closed     NS

3       1  Web Based Application  WBS000088   IM0000011  Closed      4

4       1  Web Based Application  WBS000088   IM0000012  Closed      4


  Urgency  Priority  number_cnt                 Category  KB_number  \
0       4       4.0    0.601292                 incident  KM0000553
1       3       3.0    0.415050                 incident  KM0000611
2       3       NaN    0.517551  request for information  KM0000339
3       4       4.0    0.642927                 incident  KM0000611
4       4       4.0    0.345258                 incident  KM0000611
```

```
  Alert_Status  No_of_Reassignments        Open_Time
Resolved_Time  \
0       closed                 26.0  05-02-2012 13:32  04-11-2013
13:50
1       closed                 33.0  12-03-2012 15:44  02-12-2013
12:36
2       closed                  3.0  29-03-2012 12:36  13-01-2014
15:12
3       closed                 13.0  17-07-2012 11:49  14-11-2013
09:31
4       closed                  2.0  10-08-2012 11:01  08-11-2013
13:55

         Close_Time Handle_Time_hrs                  Closure_Code  \
0  04-11-2013 13:51  3,87,16,91,111                         Other
1  02-12-2013 12:36  4,35,47,86,389                      Software
2  13-01-2014 15:13  4,84,31,19,444  No error - works as designed
3  14-11-2013 09:31  4,32,18,33,333                Operator error
4  08-11-2013 13:55  3,38,39,03,333                         Other

   No_of_Related_Interactions Related_Interaction
0                         1.0           SD0000007
1                         1.0           SD0000011
2                         1.0           SD0000017
3                         1.0           SD0000025
4                         1.0           SD0000029
```

df['CI_Subcat']

```
df['CI_Subcat'].unique()
```

```
array(['Web Based Application', 'Desktop Application',
       'Server Based Application', 'SAP', 'Client Based Application',
       'Citrix', 'Standard Application', 'Windows Server', 'Laptop',
       'Linux Server', nan, 'Monitor', 'Automation Software', 'SAN',
       'Banking Device', 'Desktop', 'Database', 'Oracle Server',
       'Keyboard', 'Printer', 'Exchange', 'System Software', 'VDI',
       'Encryption', 'Omgeving', 'MigratieDummy', 'Scanner',
'Controller',
       'DataCenterEquipment', 'KVM Switches', 'Switch',
       'Database Software', 'Network Component', 'Unix Server',
'Lines',
       'ESX Cluster', 'zOS Server', 'SharePoint Farm', 'NonStop
Server',
       'Application Server', 'Security Software', 'Thin Client',
       'zOS Cluster', 'Router', 'VMWare', 'Net Device', 'Neoview
Server',
       'MQ Queue Manager', 'UPS', 'Number', 'Iptelephony',
       'Windows Server in extern beheer', 'Modem', 'X86 Server',
```

```
       'ESX Server', 'Virtual Tape Server', 'IPtelephony',
       'NonStop Harddisk', 'Firewall', 'RAC Service', 'zOS Systeem',
       'Instance', 'NonStop Storage', 'Protocol', 'Tape Library'],
      dtype=object)

df['CI_Subcat'].isnull().sum()

111

df['CI_Subcat'].mode()

0    Server Based Application
dtype: object
```

- there were 111 null values present this columns replacing those with mode i.e. `server based application`

```
df.loc[df['CI_Subcat'].isnull(),'CI_Subcat']=df['CI_Subcat'].mode()[0]

df['CI_Subcat'].isnull().sum()

0
```

- using `lable_encoder` to transform categorical to numerical columns

```
df['CI_Subcat']=encoder.fit_transform(df['CI_Subcat'])

df.head()

   CI_Cat  CI_Subcat        WBS Incident_ID  Status Impact Urgency
Priority  \
0      11         57  WBS000162   IM0000004  Closed      4       4
4.0
1       1         57  WBS000088   IM0000005  Closed      3       3
3.0
2       1         10  WBS000092   IM0000006  Closed     NS       3
NaN
3       1         57  WBS000088   IM0000011  Closed      4       4
4.0
4       1         57  WBS000088   IM0000012  Closed      4       4
4.0

   number_cnt                 Category  KB_number Alert_Status  \
0    0.601292                 incident  KM0000553       closed
1    0.415050                 incident  KM0000611       closed
2    0.517551  request for information  KM0000339       closed
3    0.642927                 incident  KM0000611       closed
4    0.345258                 incident  KM0000611       closed

   No_of_Reassignments         Open_Time     Resolved_Time
Close_Time  \
0                 26.0  05-02-2012 13:32  04-11-2013 13:50  04-11-2013
```

```
13:51
1                  33.0  12-03-2012 15:44  02-12-2013 12:36  02-12-2013
12:36
2                   3.0  29-03-2012 12:36  13-01-2014 15:12  13-01-2014
15:13
3                  13.0  17-07-2012 11:49  14-11-2013 09:31  14-11-2013
09:31
4                   2.0  10-08-2012 11:01  08-11-2013 13:55  08-11-2013
13:55

  Handle_Time_hrs                  Closure_Code
No_of_Related_Interactions  \
0  3,87,16,91,111                         Other
1.0
1  4,35,47,86,389                      Software
1.0
2  4,84,31,19,444  No error - works as designed
1.0
3  4,32,18,33,333                Operator error
1.0
4  3,38,39,03,333                         Other
1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

df['WBS']

```
len(df['WBS'].unique())

274
```

- extracting the unique numbers of the WBS system

```
df['WBS']=df['WBS'].apply(lambda x: x[-3:])

df['WBS']

0        162
1        088
2        092
3        088
4        088
        ...
46601    073
46602    073
```

```
46603    091
46604    073
46605    267
Name: WBS, Length: 46606, dtype: object
```

```
df['WBS']=df['WBS'].astype(int)

df.head()
```

```
   CI_Cat  CI_Subcat  WBS Incident_ID  Status Impact Urgency  Priority
\
0      11         57  162   IM0000004  Closed      4       4       4.0

1       1         57   88   IM0000005  Closed      3       3       3.0

2       1         10   92   IM0000006  Closed     NS       3       NaN

3       1         57   88   IM0000011  Closed      4       4       4.0

4       1         57   88   IM0000012  Closed      4       4       4.0


   number_cnt                 Category  KB_number Alert_Status  \
0    0.601292                 incident  KM0000553       closed
1    0.415050                 incident  KM0000611       closed
2    0.517551  request for information  KM0000339       closed
3    0.642927                 incident  KM0000611       closed
4    0.345258                 incident  KM0000611       closed

   No_of_Reassignments         Open_Time     Resolved_Time
Close_Time  \
0                 26.0  05-02-2012 13:32  04-11-2013 13:50  04-11-2013
13:51
1                 33.0  12-03-2012 15:44  02-12-2013 12:36  02-12-2013
12:36
2                  3.0  29-03-2012 12:36  13-01-2014 15:12  13-01-2014
15:13
3                 13.0  17-07-2012 11:49  14-11-2013 09:31  14-11-2013
09:31
4                  2.0  10-08-2012 11:01  08-11-2013 13:55  08-11-2013
13:55

  Handle_Time_hrs                  Closure_Code
No_of_Related_Interactions  \
0  3,87,16,91,111                         Other
1.0
1  4,35,47,86,389                      Software
1.0
2  4,84,31,19,444  No error - works as designed
1.0
3  4,32,18,33,333                Operator error
```

```
1.0
4  3,38,39,03,333                                Other
1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

df['Incident_ID']

```
len(df['Incident_ID'].unique())

46606
```

- there are `46606` unique values in this column and it carries no weight to data so dropping the column

```
df.drop('Incident_ID',axis=1,inplace=True)

df.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority  number_cnt  \
0      11         57  162  Closed       4        4       4.0    0.601292

1       1         57   88  Closed       3        3       3.0    0.415050

2       1         10   92  Closed      NS        3       NaN    0.517551

3       1         57   88  Closed       4        4       4.0    0.642927

4       1         57   88  Closed       4        4       4.0    0.345258


                  Category  KB_number Alert_Status
No_of_Reassignments  \
0                 incident  KM0000553       closed
26.0
1                 incident  KM0000611       closed
33.0
2  request for information  KM0000339       closed
3.0
3                 incident  KM0000611       closed
13.0
4                 incident  KM0000611       closed
2.0

          Open_Time      Resolved_Time        Close_Time
```

```
Handle_Time_hrs  \
0  05-02-2012 13:32  04-11-2013 13:50  04-11-2013 13:51
3,87,16,91,111
1  12-03-2012 15:44  02-12-2013 12:36  02-12-2013 12:36
4,35,47,86,389
2  29-03-2012 12:36  13-01-2014 15:12  13-01-2014 15:13
4,84,31,19,444
3  17-07-2012 11:49  14-11-2013 09:31  14-11-2013 09:31
4,32,18,33,333
4  10-08-2012 11:01  08-11-2013 13:55  08-11-2013 13:55
3,38,39,03,333

                  Closure_Code  No_of_Related_Interactions  \
0                        Other                         1.0
1                     Software                         1.0
2  No error - works as designed                         1.0
3               Operator error                         1.0
4                        Other                         1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

df['Status']

```
df['Status'].unique()

array(['Closed', 'Work in progress'], dtype=object)
```

- using label encoder to convert the categorical columns to numerical columns

```
df['Status'].isnull().sum()

0

df['Status']=encoder.fit_transform(df['Status'])

df.head()

   CI_Cat  CI_Subcat  WBS  Status Impact Urgency  Priority  number_cnt
\
0      11         57  162       0      4       4       4.0    0.601292

1       1         57   88       0      3       3       3.0    0.415050

2       1         10   92       0     NS       3       NaN    0.517551

3       1         57   88       0      4       4       4.0    0.642927
```

```
4       1         57   88        0      4       4       4.0    0.345258


                  Category  KB_number Alert_Status
No_of_Reassignments  \
0                 incident  KM0000553       closed
26.0
1                 incident  KM0000611       closed
33.0
2  request for information  KM0000339       closed
3.0
3                 incident  KM0000611       closed
13.0
4                 incident  KM0000611       closed
2.0

          Open_Time     Resolved_Time        Close_Time
Handle_Time_hrs  \
0  05-02-2012 13:32  04-11-2013 13:50  04-11-2013 13:51
3,87,16,91,111
1  12-03-2012 15:44  02-12-2013 12:36  02-12-2013 12:36
4,35,47,86,389
2  29-03-2012 12:36  13-01-2014 15:12  13-01-2014 15:13
4,84,31,19,444
3  17-07-2012 11:49  14-11-2013 09:31  14-11-2013 09:31
4,32,18,33,333
4  10-08-2012 11:01  08-11-2013 13:55  08-11-2013 13:55
3,38,39,03,333

                   Closure_Code  No_of_Related_Interactions  \
0                         Other                         1.0
1                      Software                         1.0
2  No error - works as designed                         1.0
3                Operator error                         1.0
4                         Other                         1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

df['Impact']

```
df['Impact'].value_counts()

4     22556
5     16741
```

```
3      5234
NS     1380
2       692
1         3
Name: Impact, dtype: int64
```

```
df['Impact'].mode()[0]
```

```
'4'
```

- replacing the null values with mode of the column i.e 4

```
df.loc[df['Impact']=='NS','Impact']=df['Impact'].mode()[0]
```

```
df.loc[df['Impact']=='NS']
```

```
Empty DataFrame
Columns: [CI_Cat, CI_Subcat, WBS, Status, Impact, Urgency, Priority,
number_cnt, Category, KB_number, Alert_Status, No_of_Reassignments,
Open_Time, Resolved_Time, Close_Time, Handle_Time_hrs, Closure_Code,
No_of_Related_Interactions, Related_Interaction]
Index: []
```

```
df['Impact'].dtype
```

```
dtype('O')
```

```
df['Impact']=df['Impact'].astype(int)
```

```
df['Impact'].unique()
```

```
array([4, 3, 5, 2, 1])
```

df['Urgency']

```
df['Urgency'].value_counts()
```

```
4                15526
5                12284
4                 7062
5                 4495
3                 4419
3                 2117
2                  538
2                  158
1                    5
5 - Very Low         1
1                    1
Name: Urgency, dtype: int64
```

- as only 1 entry there in data dropping the column

```
df.drop(df.loc[df['Urgency']=='5 - Very
Low'].index,axis=0,inplace=True)

df['Urgency'].value_counts()
```

```
4    15526
5    12284
4     7062
5     4495
3     4419
3     2117
2      538
2      158
1        5
1        1
Name: Urgency, dtype: int64
```

```
df.drop_duplicates()
```

```
       CI_Cat  CI_Subcat  WBS  Status  Impact Urgency  Priority
number_cnt  \
0          11         57  162       0       4       4       4.0
0.601292
1           1         57   88       0       3       3       3.0
0.415050
2           1         10   92       0       4       3       NaN
0.517551
3           1         57   88       0       4       4       4.0
0.642927
4           1         57   88       0       4       4       4.0
0.345258
...       ...        ...  ...     ...     ...     ...       ...
...
46601       1         45   73       0       4       4       4.0
0.231896
46602       1         45   73       0       4       4       4.0
0.805153
46603       3         21   91       0       5       5       5.0
0.917466
46604       1         57   73       0       4       4       4.0
0.701278
46605       6          6  267       0       3       3       3.0
0.902320

                      Category  KB_number Alert_Status
No_of_Reassignments  \
0                     incident  KM0000553       closed
26.0
1                     incident  KM0000611       closed
33.0
```

```
2      request for information  KM0000339       closed
3.0
3                     incident  KM0000611       closed
13.0
4                     incident  KM0000611       closed
2.0
...                        ...        ...          ...
...
46601                 incident  KM0001314       closed
0.0
46602                 incident  KM0002360       closed
0.0
46603                 incident  KM0000315       closed
0.0
46604                 incident  KM0001287       closed
0.0
46605                 incident  KM0000182       closed
0.0

              Open_Time     Resolved_Time        Close_Time
Handle_Time_hrs  \
0      05-02-2012 13:32  04-11-2013 13:50  04-11-2013 13:51
3,87,16,91,111
1      12-03-2012 15:44  02-12-2013 12:36  02-12-2013 12:36
4,35,47,86,389
2      29-03-2012 12:36  13-01-2014 15:12  13-01-2014 15:13
4,84,31,19,444
3      17-07-2012 11:49  14-11-2013 09:31  14-11-2013 09:31
4,32,18,33,333
4      10-08-2012 11:01  08-11-2013 13:55  08-11-2013 13:55
3,38,39,03,333
...                 ...               ...               ...
...
46601  31-03-2014 16:23  31-03-2014 16:29  31-03-2014 16:29
0,095
46602  31-03-2014 15:03  31-03-2014 15:29  31-03-2014 15:29
0,428333333
46603  31-03-2014 15:28  31-03-2014 15:32  31-03-2014 15:32
0,071666667
46604  31-03-2014 15:35  31-03-2014 15:42  31-03-2014 15:42
0,116944444
46605  31-03-2014 17:24  31-03-2014 22:47  31-03-2014 22:47
0,586388889

                      Closure_Code  No_of_Related_Interactions  \
0                            Other                         1.0
1                         Software                         1.0
2      No error - works as designed                         1.0
3                   Operator error                         1.0
```

```
4                                 Other                          1.0
...                                 ...                          ...
46601                             Other                          1.0
46602                        User error                          1.0
46603                          Hardware                          1.0
46604                          Software                          1.0
46605                          Hardware                          1.0

      Related_Interaction
0               SD0000007
1               SD0000011
2               SD0000017
3               SD0000025
4               SD0000029
...                   ...
46601           SD0147021
46602           SD0146967
46603           SD0146982
46604           SD0146986
46605           SD0147088

[46605 rows x 19 columns]
```

```
df['Urgency']=df['Urgency'].astype(int)
```

```
df['Urgency'].unique()
```

```
array([4, 3, 5, 2, 1])
```

```
df.shape
```

```
(46605, 19)
```

```
df.head()
```

```
   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4       4.0
0.601292
1       1         57   88       0       3        3       3.0
0.415050
2       1         10   92       0       4        3       NaN
0.517551
3       1         57   88       0       4        4       4.0
0.642927
4       1         57   88       0       4        4       4.0
0.345258

                  Category  KB_number Alert_Status
No_of_Reassignments  \
0                 incident  KM0000553       closed
```

```
26.0
1                  incident  KM0000611       closed
33.0
2  request for information  KM0000339       closed
3.0
3                  incident  KM0000611       closed
13.0
4                  incident  KM0000611       closed
2.0

          Open_Time     Resolved_Time        Close_Time
Handle_Time_hrs  \
0  05-02-2012 13:32  04-11-2013 13:50  04-11-2013 13:51
3,87,16,91,111
1  12-03-2012 15:44  02-12-2013 12:36  02-12-2013 12:36
4,35,47,86,389
2  29-03-2012 12:36  13-01-2014 15:12  13-01-2014 15:13
4,84,31,19,444
3  17-07-2012 11:49  14-11-2013 09:31  14-11-2013 09:31
4,32,18,33,333
4  10-08-2012 11:01  08-11-2013 13:55  08-11-2013 13:55
3,38,39,03,333

                   Closure_Code  No_of_Related_Interactions  \
0                         Other                         1.0
1                      Software                         1.0
2  No error - works as designed                         1.0
3                Operator error                         1.0
4                         Other                         1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

df['Priority']

```
df['Priority'].unique()
```

```
array([ 4.,  3., nan,  5.,  2.,  1.])
```

```
df['Priority'].value_counts()
```

```
4.0    22717
5.0    16485
3.0     5323
2.0      697
```

```
1.0         3
Name: Priority, dtype: int64
```

- replacing the null values with mode

```
df['Priority'].mode()

0    4.0
dtype: float64

df.loc[df['Priority'].isna(),'Priority']=df['Priority'].mode()[0]

df['Priority']=df['Priority'].astype(int)

df.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
3       1         57   88       0       4        4         4
0.642927
4       1         57   88       0       4        4         4
0.345258

                  Category  KB_number Alert_Status
No_of_Reassignments  \
0                 incident  KM0000553       closed
26.0
1                 incident  KM0000611       closed
33.0
2  request for information  KM0000339       closed
3.0
3                 incident  KM0000611       closed
13.0
4                 incident  KM0000611       closed
2.0

          Open_Time     Resolved_Time        Close_Time
Handle_Time_hrs  \
0  05-02-2012 13:32  04-11-2013 13:50  04-11-2013 13:51
3,87,16,91,111
1  12-03-2012 15:44  02-12-2013 12:36  02-12-2013 12:36
4,35,47,86,389
2  29-03-2012 12:36  13-01-2014 15:12  13-01-2014 15:13
4,84,31,19,444
3  17-07-2012 11:49  14-11-2013 09:31  14-11-2013 09:31
```

```
4,32,18,33,333
4  10-08-2012 11:01  08-11-2013 13:55  08-11-2013 13:55
3,38,39,03,333

                  Closure_Code  No_of_Related_Interactions  \
0                        Other                         1.0
1                     Software                         1.0
2  No error - works as designed                         1.0
3               Operator error                         1.0
4                        Other                         1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

##df['number_cnt']

```
df['number_cnt']=df['number_cnt'].astype(float)
```

df['Category']

```
df['Category'].unique()

array(['incident', 'request for information', 'complaint',
       'request for change'], dtype=object)
```

- transforming the catergorical columns to numerical columns

```
df['Category']=encoder.fit_transform(df['Category'])

df.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
3       1         57   88       0       4        4         4
0.642927
4       1         57   88       0       4        4         4
0.345258

   Category  KB_number Alert_Status  No_of_Reassignments
Open_Time  \
```

```
0          1  KM0000553       closed                   26.0  05-02-2012
13:32
1          1  KM0000611       closed                   33.0  12-03-2012
15:44
2          3  KM0000339       closed                    3.0  29-03-2012
12:36
3          1  KM0000611       closed                   13.0  17-07-2012
11:49
4          1  KM0000611       closed                    2.0  10-08-2012
11:01

      Resolved_Time         Close_Time Handle_Time_hrs  \
0  04-11-2013 13:50  04-11-2013 13:51  3,87,16,91,111
1  02-12-2013 12:36  02-12-2013 12:36  4,35,47,86,389
2  13-01-2014 15:12  13-01-2014 15:13  4,84,31,19,444
3  14-11-2013 09:31  14-11-2013 09:31  4,32,18,33,333
4  08-11-2013 13:55  08-11-2013 13:55  3,38,39,03,333

                   Closure_Code  No_of_Related_Interactions  \
0                         Other                         1.0
1                      Software                         1.0
2  No error - works as designed                         1.0
3                Operator error                         1.0
4                         Other                         1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

df['Alert_Status']

```
df['Alert_Status'].unique()

array(['closed'], dtype=object)

df['Alert_Status'].value_counts()

closed    46605
Name: Alert_Status, dtype: int64
```

- as almost all the columns are in `closed` state dropping the columns

```
df.drop('Alert_Status',axis=1,inplace=True)

df.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
```

```
0       11          57  162        0       4        4         4
0.601292
1        1          57   88        0       3        3         3
0.415050
2        1          10   92        0       4        3         4
0.517551
3        1          57   88        0       4        4         4
0.642927
4        1          57   88        0       4        4         4
0.345258

   Category  KB_number  No_of_Reassignments         Open_Time  \
0         1  KM0000553                 26.0  05-02-2012 13:32
1         1  KM0000611                 33.0  12-03-2012 15:44
2         3  KM0000339                  3.0  29-03-2012 12:36
3         1  KM0000611                 13.0  17-07-2012 11:49
4         1  KM0000611                  2.0  10-08-2012 11:01

      Resolved_Time        Close_Time Handle_Time_hrs  \
0  04-11-2013 13:50  04-11-2013 13:51  3,87,16,91,111
1  02-12-2013 12:36  02-12-2013 12:36  4,35,47,86,389
2  13-01-2014 15:12  13-01-2014 15:13  4,84,31,19,444
3  14-11-2013 09:31  14-11-2013 09:31  4,32,18,33,333
4  08-11-2013 13:55  08-11-2013 13:55  3,38,39,03,333

                  Closure_Code  No_of_Related_Interactions  \
0                        Other                         1.0
1                     Software                         1.0
2  No error - works as designed                         1.0
3               Operator error                         1.0
4                        Other                         1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

df['KB_number']

```
len(df['KB_number'].unique())

1824
```

- extracting the last 4 numbers of column

```
df['KB_number']=df['KB_number'].apply(lambda x: x[-4:])

df['KB_number']=df['KB_number'].astype(int)
```

```
df.head()
```

```
   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
3       1         57   88       0       4        4         4
0.642927
4       1         57   88       0       4        4         4
0.345258

   Category  KB_number  No_of_Reassignments         Open_Time  \
0         1        553                 26.0  05-02-2012 13:32
1         1        611                 33.0  12-03-2012 15:44
2         3        339                  3.0  29-03-2012 12:36
3         1        611                 13.0  17-07-2012 11:49
4         1        611                  2.0  10-08-2012 11:01

      Resolved_Time        Close_Time Handle_Time_hrs  \
0  04-11-2013 13:50  04-11-2013 13:51  3,87,16,91,111
1  02-12-2013 12:36  02-12-2013 12:36  4,35,47,86,389
2  13-01-2014 15:12  13-01-2014 15:13  4,84,31,19,444
3  14-11-2013 09:31  14-11-2013 09:31  4,32,18,33,333
4  08-11-2013 13:55  08-11-2013 13:55  3,38,39,03,333

                 Closure_Code  No_of_Related_Interactions  \
0                       Other                         1.0
1                    Software                         1.0
2  No error - works as designed                       1.0
3              Operator error                         1.0
4                       Other                         1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

```
len(df['KB_number'].unique())
```

```
1824
```

df['No_of_Reassignments']

```
len(df['No_of_Reassignments'].unique())
```

```
42
```

```python
df['No_of_Reassignments'].mode()
```

```
0    0.0
dtype: float64
```

```python
df['No_of_Reassignments'].isnull().sum()
```

```
1
```

- replacing the null values with mode i.e 0

```python
df.loc[df['No_of_Reassignments'].isnull(),'No_of_Reassignments']=df['No_of_Reassignments'].mode()[0]
```

```python
df['No_of_Reassignments'].isnull().sum()
```

```
0
```

```python
df['No_of_Reassignments']=df['No_of_Reassignments'].astype(int)
```

```python
df.head()
```

```
   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority  
number_cnt  \
0      11         57  162       0       4        4         4  
0.601292
1       1         57   88       0       3        3         3  
0.415050
2       1         10   92       0       4        3         4  
0.517551
3       1         57   88       0       4        4         4  
0.642927
4       1         57   88       0       4        4         4  
0.345258

   Category  KB_number  No_of_Reassignments         Open_Time  \
0         1        553                   26  05-02-2012 13:32   
1         1        611                   33  12-03-2012 15:44   
2         3        339                    3  29-03-2012 12:36   
3         1        611                   13  17-07-2012 11:49   
4         1        611                    2  10-08-2012 11:01   

      Resolved_Time        Close_Time Handle_Time_hrs  \
0  04-11-2013 13:50  04-11-2013 13:51   3,87,16,91,111   
1  02-12-2013 12:36  02-12-2013 12:36   4,35,47,86,389   
2  13-01-2014 15:12  13-01-2014 15:13   4,84,31,19,444   
3  14-11-2013 09:31  14-11-2013 09:31   4,32,18,33,333   
4  08-11-2013 13:55  08-11-2013 13:55   3,38,39,03,333   
```

```
                    Closure_Code  No_of_Related_Interactions  \
0                          Other                         1.0
1                       Software                         1.0
2  No error - works as designed                          1.0
3                 Operator error                         1.0
4                          Other                         1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

df['Open_Time']

```
df['Open_Time'].isnull().sum()

0
```

- converting the open time column to datetime format

```
df['Open_Time']=pd.to_datetime(df['Open_Time'])

df.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
3       1         57   88       0       4        4         4
0.642927
4       1         57   88       0       4        4         4
0.345258

   Category  KB_number  No_of_Reassignments           Open_Time  \
0         1        553                   26 2012-05-02 13:32:00
1         1        611                   33 2012-12-03 15:44:00
2         3        339                    3 2012-03-29 12:36:00
3         1        611                   13 2012-07-17 11:49:00
4         1        611                    2 2012-10-08 11:01:00

      Resolved_Time        Close_Time Handle_Time_hrs  \
0  04-11-2013 13:50  04-11-2013 13:51  3,87,16,91,111
1  02-12-2013 12:36  02-12-2013 12:36  4,35,47,86,389
2  13-01-2014 15:12  13-01-2014 15:13  4,84,31,19,444
```

```
3  14-11-2013 09:31  14-11-2013 09:31  4,32,18,33,333
4  08-11-2013 13:55  08-11-2013 13:55  3,38,39,03,333

                   Closure_Code  No_of_Related_Interactions  \
0                         Other                         1.0
1                      Software                         1.0
2  No error - works as designed                         1.0
3                Operator error                         1.0
4                         Other                         1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

df['Resolved_Time']

```
df['Resolved_Time']=pd.to_datetime(df['Resolved_Time'])

df.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
3       1         57   88       0       4        4         4
0.642927
4       1         57   88       0       4        4         4
0.345258

   Category  KB_number  No_of_Reassignments           Open_Time  \
0         1        553                   26 2012-05-02 13:32:00
1         1        611                   33 2012-12-03 15:44:00
2         3        339                    3 2012-03-29 12:36:00
3         1        611                   13 2012-07-17 11:49:00
4         1        611                    2 2012-10-08 11:01:00

        Resolved_Time        Close_Time Handle_Time_hrs  \
0 2013-04-11 13:50:00  04-11-2013 13:51  3,87,16,91,111
1 2013-02-12 12:36:00  02-12-2013 12:36  4,35,47,86,389
2 2014-01-13 15:12:00  13-01-2014 15:13  4,84,31,19,444
3 2013-11-14 09:31:00  14-11-2013 09:31  4,32,18,33,333
4 2013-08-11 13:55:00  08-11-2013 13:55  3,38,39,03,333
```

```
                 Closure_Code  No_of_Related_Interactions  \
0                       Other                         1.0
1                    Software                         1.0
2  No error - works as designed                        1.0
3              Operator error                         1.0
4                       Other                         1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

```
df['Resolved_Time'].isnull().sum()
```

```
1780
```

```
df['Resolved_Time'].mode()[0]
```

```
Timestamp('2013-10-10 12:53:00')
```

```
df.loc[df['Resolved_Time'].isnull(),'Resolved_Time']=df['Resolved_Time
'].mode()[0]
```

```
df['Resolved_Time'].isnull().sum()
```

```
0
```

```
df['Resolved_Time']=pd.to_datetime(df['Resolved_Time'])
```

```
# data.drop('Resolved_Time',axis=1,inplace=True)
```

```
df.head()
```

```
   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
3       1         57   88       0       4        4         4
0.642927
4       1         57   88       0       4        4         4
0.345258

   Category  KB_number  No_of_Reassignments           Open_Time  \
0         1        553                   26 2012-05-02 13:32:00
1         1        611                   33 2012-12-03 15:44:00
2         3        339                    3 2012-03-29 12:36:00
```

```
3         1        611                    13 2012-07-17 11:49:00
4         1        611                     2 2012-10-08 11:01:00

        Resolved_Time        Close_Time Handle_Time_hrs  \
0 2013-04-11 13:50:00  04-11-2013 13:51  3,87,16,91,111
1 2013-02-12 12:36:00  02-12-2013 12:36  4,35,47,86,389
2 2014-01-13 15:12:00  13-01-2014 15:13  4,84,31,19,444
3 2013-11-14 09:31:00  14-11-2013 09:31  4,32,18,33,333
4 2013-08-11 13:55:00  08-11-2013 13:55  3,38,39,03,333

                   Closure_Code  No_of_Related_Interactions  \
0                         Other                         1.0
1                      Software                         1.0
2  No error - works as designed                         1.0
3                Operator error                         1.0
4                         Other                         1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

df['Close_Time']

```
df['Close_Time'].isnull().sum()

0

df['Close_Time']=pd.to_datetime(df['Close_Time'])

df.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
3       1         57   88       0       4        4         4
0.642927
4       1         57   88       0       4        4         4
0.345258

   Category  KB_number  No_of_Reassignments           Open_Time  \
0         1        553                   26 2012-05-02 13:32:00
1         1        611                   33 2012-12-03 15:44:00
```

```
2         3        339                       3 2012-03-29 12:36:00
3         1        611                      13 2012-07-17 11:49:00
4         1        611                       2 2012-10-08 11:01:00

        Resolved_Time          Close_Time Handle_Time_hrs  \
0 2013-04-11 13:50:00 2013-04-11 13:51:00  3,87,16,91,111
1 2013-02-12 12:36:00 2013-02-12 12:36:00  4,35,47,86,389
2 2014-01-13 15:12:00 2014-01-13 15:13:00  4,84,31,19,444
3 2013-11-14 09:31:00 2013-11-14 09:31:00  4,32,18,33,333
4 2013-08-11 13:55:00 2013-08-11 13:55:00  3,38,39,03,333

                   Closure_Code  No_of_Related_Interactions  \
0                         Other                         1.0
1                      Software                         1.0
2  No error - works as designed                         1.0
3                Operator error                         1.0
4                         Other                         1.0

  Related_Interaction
0           SD0000007
1           SD0000011
2           SD0000017
3           SD0000025
4           SD0000029
```

df['Handle_Time_hrs']

- manually creating the `handle_time_hrs` as the given `handle_time_hrs` is not carrying any meaningfull information
- converting the difference days to hours taken

```
df.drop('Handle_Time_hrs',axis=1,inplace=True)

df['Handle_Time_hrs_conv']=abs(df['Close_Time']-df['Open_Time'])

a=[]
for i in df['Handle_Time_hrs_conv'].index:
    a.append((df['Handle_Time_hrs_conv'][i].total_seconds())/3600)

df['Handle_Time_hrs_conv']=a

df.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
```

```
3       1          57   88        0        4         4          4
0.642927
4       1          57   88        0        4         4          4
0.345258

   Category  KB_number  No_of_Reassignments           Open_Time  \
0         1        553                   26 2012-05-02 13:32:00
1         1        611                   33 2012-12-03 15:44:00
2         3        339                    3 2012-03-29 12:36:00
3         1        611                   13 2012-07-17 11:49:00
4         1        611                    2 2012-10-08 11:01:00

        Resolved_Time          Close_Time
Closure_Code  \
0 2013-04-11 13:50:00 2013-04-11 13:51:00
Other
1 2013-02-12 12:36:00 2013-02-12 12:36:00
Software
2 2014-01-13 15:12:00 2014-01-13 15:13:00  No error - works as
designed
3 2013-11-14 09:31:00 2013-11-14 09:31:00                Operator
error
4 2013-08-11 13:55:00 2013-08-11 13:55:00
Other

   No_of_Related_Interactions Related_Interaction
Handle_Time_hrs_conv
0                         1.0           SD0000007
8256.316667
1                         1.0           SD0000011
1700.866667
2                         1.0           SD0000017
15722.616667
3                         1.0           SD0000025
11637.700000
4                         1.0           SD0000029
7370.900000
```

df['Closure_Code']

- as the closure code will not determine the ticket priority and importance as its done at the posterior stage of ticket resolving

```
df.drop('Closure_Code',axis=1,inplace=True)

df.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
```

```
0.601292
1       1         57   88        0       3        3         3
0.415050
2       1         10   92        0       4        3         4
0.517551
3       1         57   88        0       4        4         4
0.642927
4       1         57   88        0       4        4         4
0.345258

   Category  KB_number  No_of_Reassignments           Open_Time  \
0         1        553                   26 2012-05-02 13:32:00
1         1        611                   33 2012-12-03 15:44:00
2         3        339                    3 2012-03-29 12:36:00
3         1        611                   13 2012-07-17 11:49:00
4         1        611                    2 2012-10-08 11:01:00

        Resolved_Time          Close_Time  No_of_Related_Interactions
\
0 2013-04-11 13:50:00 2013-04-11 13:51:00                         1.0

1 2013-02-12 12:36:00 2013-02-12 12:36:00                         1.0

2 2014-01-13 15:12:00 2014-01-13 15:13:00                         1.0

3 2013-11-14 09:31:00 2013-11-14 09:31:00                         1.0

4 2013-08-11 13:55:00 2013-08-11 13:55:00                         1.0


  Related_Interaction  Handle_Time_hrs_conv
0           SD0000007           8256.316667
1           SD0000011           1700.866667
2           SD0000017          15722.616667
3           SD0000025          11637.700000
4           SD0000029           7370.900000
```

df['No_of_Related_Interactions']

```
df['No_of_Related_Interactions'].isnull().sum()
```

```
114
```

```
len(df['No_of_Related_Interactions'].unique())
```

```
50
```

```
df['No_of_Related_Interactions'].mode()
```

```
0    1.0
dtype: float64
```

- replacing the null values with mode

```
df.loc[df['No_of_Related_Interactions'].isnull(),'No_of_Related_Intera
ctions']=df['No_of_Related_Interactions'].mode()[0]

df['No_of_Related_Interactions'].isnull().sum()

0

df['No_of_Related_Interactions']=df['No_of_Related_Interactions'].asty
pe(int)

df.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
3       1         57   88       0       4        4         4
0.642927
4       1         57   88       0       4        4         4
0.345258

   Category  KB_number  No_of_Reassignments           Open_Time  \
0         1        553                   26 2012-05-02 13:32:00
1         1        611                   33 2012-12-03 15:44:00
2         3        339                    3 2012-03-29 12:36:00
3         1        611                   13 2012-07-17 11:49:00
4         1        611                    2 2012-10-08 11:01:00

        Resolved_Time          Close_Time  No_of_Related_Interactions
\
0 2013-04-11 13:50:00 2013-04-11 13:51:00                           1

1 2013-02-12 12:36:00 2013-02-12 12:36:00                           1

2 2014-01-13 15:12:00 2014-01-13 15:13:00                           1

3 2013-11-14 09:31:00 2013-11-14 09:31:00                           1

4 2013-08-11 13:55:00 2013-08-11 13:55:00                           1


  Related_Interaction  Handle_Time_hrs_conv
0           SD0000007           8256.316667
1           SD0000011           1700.866667
2           SD0000017          15722.616667
```

```
3          SD0000025          11637.700000
4          SD0000029           7370.900000
```

df['Related_Interaction']

```
len(df['Related_Interaction'].unique())
```

```
43059
```

```
df.drop('Related_Interaction',axis=1,inplace=True)
```

Preprocessed dataset for machine learning

```
df.head()
```

```
   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority  
number_cnt  \
0      11         57  162       0       4        4         4   
0.601292   
1       1         57   88       0       3        3         3   
0.415050   
2       1         10   92       0       4        3         4   
0.517551   
3       1         57   88       0       4        4         4   
0.642927   
4       1         57   88       0       4        4         4   
0.345258   

   Category  KB_number  No_of_Reassignments           Open_Time  \
0         1        553                   26 2012-05-02 13:32:00   
1         1        611                   33 2012-12-03 15:44:00   
2         3        339                    3 2012-03-29 12:36:00   
3         1        611                   13 2012-07-17 11:49:00   
4         1        611                    2 2012-10-08 11:01:00   

        Resolved_Time          Close_Time  No_of_Related_Interactions 
\
0 2013-04-11 13:50:00 2013-04-11 13:51:00                           1   

1 2013-02-12 12:36:00 2013-02-12 12:36:00                           1   

2 2014-01-13 15:12:00 2014-01-13 15:13:00                           1   

3 2013-11-14 09:31:00 2013-11-14 09:31:00                           1   

4 2013-08-11 13:55:00 2013-08-11 13:55:00                           1   


   Handle_Time_hrs_conv  
0           8256.316667  
```

```
1           1700.866667
2          15722.616667
3          11637.700000
4           7370.900000
```

```
df.shape
```

```
(46605, 16)
```

Task 1

1. Predicting High Priority Tickets: To predict priority 1 & 2 tickets, so that they can take preventive measures or fix the problem before it surfaces.

```
# sns.pairplot(data=data)
```

- as we already used these columns and converted to `handle_time_hrs` dropping these columns

```
df.isnull().sum()
```

```
CI_Cat                         0
CI_Subcat                      0
WBS                            0
Status                         0
Impact                         0
Urgency                        0
Priority                       0
number_cnt                     0
Category                       0
KB_number                      0
No_of_Reassignments            0
Open_Time                      0
Resolved_Time                  0
Close_Time                     0
No_of_Related_Interactions     0
Handle_Time_hrs_conv           0
dtype: int64
```

```
data=df.drop(['Open_Time','Resolved_Time','Close_Time'],axis=1)
```

```
data.head()
```

```
   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
```

```
3       1         57   88       0       4        4         4
0.642927
4       1         57   88       0       4        4         4
0.345258

   Category  KB_number  No_of_Reassignments
No_of_Related_Interactions  \
0         1        553                   26
1
1         1        611                   33
1
2         3        339                    3
1
3         1        611                   13
1
4         1        611                    2
1

   Handle_Time_hrs_conv
0           8256.316667
1           1700.866667
2          15722.616667
3          11637.700000
4           7370.900000

data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 46605 entries, 0 to 46605
Data columns (total 13 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   CI_Cat                      46605 non-null  int32
 1   CI_Subcat                   46605 non-null  int32
 2   WBS                         46605 non-null  int32
 3   Status                      46605 non-null  int32
 4   Impact                      46605 non-null  int32
 5   Urgency                     46605 non-null  int32
 6   Priority                    46605 non-null  int32
 7   number_cnt                  46605 non-null  float64
 8   Category                    46605 non-null  int32
 9   KB_number                   46605 non-null  int32
 10  No_of_Reassignments         46605 non-null  int32
 11  No_of_Related_Interactions  46605 non-null  int32
 12  Handle_Time_hrs_conv        46605 non-null  float64
dtypes: float64(2), int32(11)
memory usage: 4.3 MB

scaler=MinMaxScaler()
```

```
X=data.drop(['Priority','Urgency'],axis=1)

X.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  number_cnt  Category
KB_number  \
0      11         57  162       0       4    0.601292         1
553
1       1         57   88       0       3    0.415050         1
611
2       1         10   92       0       4    0.517551         3
339
3       1         57   88       0       4    0.642927         1
611
4       1         57   88       0       4    0.345258         1
611

   No_of_Reassignments  No_of_Related_Interactions
Handle_Time_hrs_conv
0                   26                           1
8256.316667
1                   33                           1
1700.866667
2                    3                           1
15722.616667
3                   13                           1
11637.700000
4                    2                           1
7370.900000

y=data['Priority'].map({1:1,2:1,3:0,4:0,5:0})

y.value_counts()

0    45905
1      700
Name: Priority, dtype: int64
```

train test split

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42,stratify=y)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

(32623, 11)
(13982, 11)
```

```
(32623,)
(13982,)
```

scaling

```
X_train_scaled=scaler.fit_transform(X_train)
X_test_scaled=scaler.transform(X_test)

X_train_scaled=pd.DataFrame(X_train_scaled,columns=X_train.columns)
X_train_scaled.head()
```

```
     CI_Cat  CI_Subcat       WBS  Status  Impact  number_cnt  Category  \
0  0.090909   0.904762  0.210682     0.0    1.00    0.080536  0.333333

1  0.272727   0.142857  0.264095     0.0    1.00    0.467506  0.333333

2  0.090909   0.904762  0.937685     0.0    0.50    0.734377  0.333333

3  0.090909   0.714286  0.008902     0.0    0.75    0.695219  0.333333

4  0.272727   0.031746  0.427300     0.0    0.25    0.867096  0.333333


   KB_number  No_of_Reassignments  No_of_Related_Interactions  \
0   0.330935             0.043478                         0.0
1   0.609818             0.021739                         0.0
2   0.356327             0.000000                         0.0
3   0.010157             0.021739                         0.0
4   0.115108             0.000000                         0.0

   Handle_Time_hrs_conv
0              0.319154
1              0.298705
2              0.000113
3              0.002981
4              0.004057
```

```
X_test_scaled=pd.DataFrame(X_test_scaled,columns=X_test.columns)
X_test_scaled.head()
```

```
     CI_Cat  CI_Subcat       WBS  Status  Impact  number_cnt  Category  \
0  0.909091   0.650794  0.373887     0.0    1.00    0.352607  0.333333

1  0.090909   0.714286  0.774481     0.0    1.00    0.104232  1.000000

2  0.454545   0.428571  0.264095     0.0    0.75    0.284961  0.333333

3  0.272727   0.333333  0.264095     0.0    0.50    0.307601  0.333333
```

```
4   0.090909   0.904762  0.210682     0.0     0.75    0.273778  0.333333


   KB_number  No_of_Reassignments  No_of_Related_Interactions  \
0   0.289886             0.043478                     0.00000
1   0.771477             0.043478                     0.00000
2   0.132882             0.021739                     0.00271
3   0.580195             0.043478                     0.00000
4   0.863733             0.000000                     0.00000

   Handle_Time_hrs_conv
0              0.000242
1              0.044762
2              0.004598
3              0.004462
4              0.000005
```

function for model selection task1

Logic behind the function

1. first creating a dictionary with the name model_summary and initiating with null values with proper keys
2. function called `model_selection` will take model as parameter 3.initially the model will be `initiated` within the function and will be `stored in the variable called model`
3. model will be `fitted on x_train and y_train` 5.model will `first predict on test data` 6.after prediction all the evaluation `metric values will be appended to dictionary` with corresponding key values. 7.then it will `print the confusion matrix and classification report` of that model 8.the same `steps will also the performed` on train data ---

```
model_summary={'model_name_train':[],'f1_score_train':
[],'recall_score_train':[],'accuracy_score_train':[],
               'model_name_test':[],'f1_score_test':
[],'recall_score_test':[],'accuracy_score_test':[]}

def model_selction_1(model):

    #model initialization ,fitting and predicting
    print(model)
    model=model()
    model.fit(X_train,y_train)
    model_pred=model.predict(X_test)
```

```
    #appending the metrics to the dictionary created
    model_summary['model_name_test'].append(model.__class__.__name__)

model_summary['f1_score_test'].append(f1_score(y_test,model_pred,avera
ge='macro'))

model_summary['recall_score_test'].append(recall_score(y_test,model_pr
ed,average='macro'))

model_summary['accuracy_score_test'].append(accuracy_score(y_test,mode
l_pred))

    #printing the confusion metrics and classification report
    print('metrics on test data')
    print(confusion_matrix(y_test,model_pred))
    print('\n')
    print(classification_report(y_test,model_pred))

    #predictions on train data
    model_pred1=model.predict(X_train)

    #appending the metrics to the dictionary created
    model_summary['model_name_train'].append(model.__class__.__name__)

model_summary['f1_score_train'].append(f1_score(y_train,model_pred1,av
erage='macro'))

model_summary['recall_score_train'].append(recall_score(y_train,model_
pred1,average='macro'))

model_summary['accuracy_score_train'].append(accuracy_score(y_train,mo
del_pred1))

    #printing the confusion metrics and classification report
    print('metrics on train data')
    print(confusion_matrix(y_train,model_pred1))
    print('\n')
    print(classification_report(y_train,model_pred1))
    print('==='*10)


models=[LogisticRegression,DecisionTreeClassifier,RandomForestClassifi
er,

BaggingClassifier,KNeighborsClassifier,GaussianNB,SVC,GradientBoosting
Classifier]

for i in models:
    model_selction_1(i)
```

```
<class 'sklearn.linear_model._logistic.LogisticRegression'>
metrics on test data
[[13772     0]
 [  200    10]]


              precision    recall  f1-score   support

           0       0.99      1.00      0.99     13772
           1       1.00      0.05      0.09       210

    accuracy                           0.99     13982
   macro avg       0.99      0.52      0.54     13982
weighted avg       0.99      0.99      0.98     13982

metrics on train data
[[32132     1]
 [  466    24]]


              precision    recall  f1-score   support

           0       0.99      1.00      0.99     32133
           1       0.96      0.05      0.09       490

    accuracy                           0.99     32623
   macro avg       0.97      0.52      0.54     32623
weighted avg       0.99      0.99      0.98     32623

==============================
<class 'sklearn.tree._classes.DecisionTreeClassifier'>
metrics on test data
[[13771     1]
 [    2   208]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13772
           1       1.00      0.99      0.99       210

    accuracy                           1.00     13982
   macro avg       1.00      1.00      1.00     13982
weighted avg       1.00      1.00      1.00     13982

metrics on train data
[[32133     0]
 [    0   490]]


              precision    recall  f1-score   support
```

```
           0       1.00      1.00      1.00     32133
           1       1.00      1.00      1.00       490

    accuracy                           1.00     32623
   macro avg       1.00      1.00      1.00     32623
weighted avg       1.00      1.00      1.00     32623

==============================
<class 'sklearn.ensemble._forest.RandomForestClassifier'>
metrics on test data
[[13772     0]
 [    2   208]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13772
           1       1.00      0.99      1.00       210

    accuracy                           1.00     13982
   macro avg       1.00      1.00      1.00     13982
weighted avg       1.00      1.00      1.00     13982

metrics on train data
[[32133     0]
 [    0   490]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     32133
           1       1.00      1.00      1.00       490

    accuracy                           1.00     32623
   macro avg       1.00      1.00      1.00     32623
weighted avg       1.00      1.00      1.00     32623

==============================
<class 'sklearn.ensemble._bagging.BaggingClassifier'>
metrics on test data
[[13772     0]
 [    2   208]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13772
           1       1.00      0.99      1.00       210

    accuracy                           1.00     13982
```

```
   macro avg       1.00      1.00      1.00     13982
weighted avg       1.00      1.00      1.00     13982

metrics on train data
[[32133     0]
 [    0   490]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     32133
           1       1.00      1.00      1.00       490

    accuracy                           1.00     32623
   macro avg       1.00      1.00      1.00     32623
weighted avg       1.00      1.00      1.00     32623

==============================
<class 'sklearn.neighbors._classification.KNeighborsClassifier'>
metrics on test data
[[13734    38]
 [   91   119]]


              precision    recall  f1-score   support

           0       0.99      1.00      1.00     13772
           1       0.76      0.57      0.65       210

    accuracy                           0.99     13982
   macro avg       0.88      0.78      0.82     13982
weighted avg       0.99      0.99      0.99     13982

metrics on train data
[[32084    49]
 [  160   330]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     32133
           1       0.87      0.67      0.76       490

    accuracy                           0.99     32623
   macro avg       0.93      0.84      0.88     32623
weighted avg       0.99      0.99      0.99     32623

==============================
<class 'sklearn.naive_bayes.GaussianNB'>
metrics on test data
[[13770     2]
```

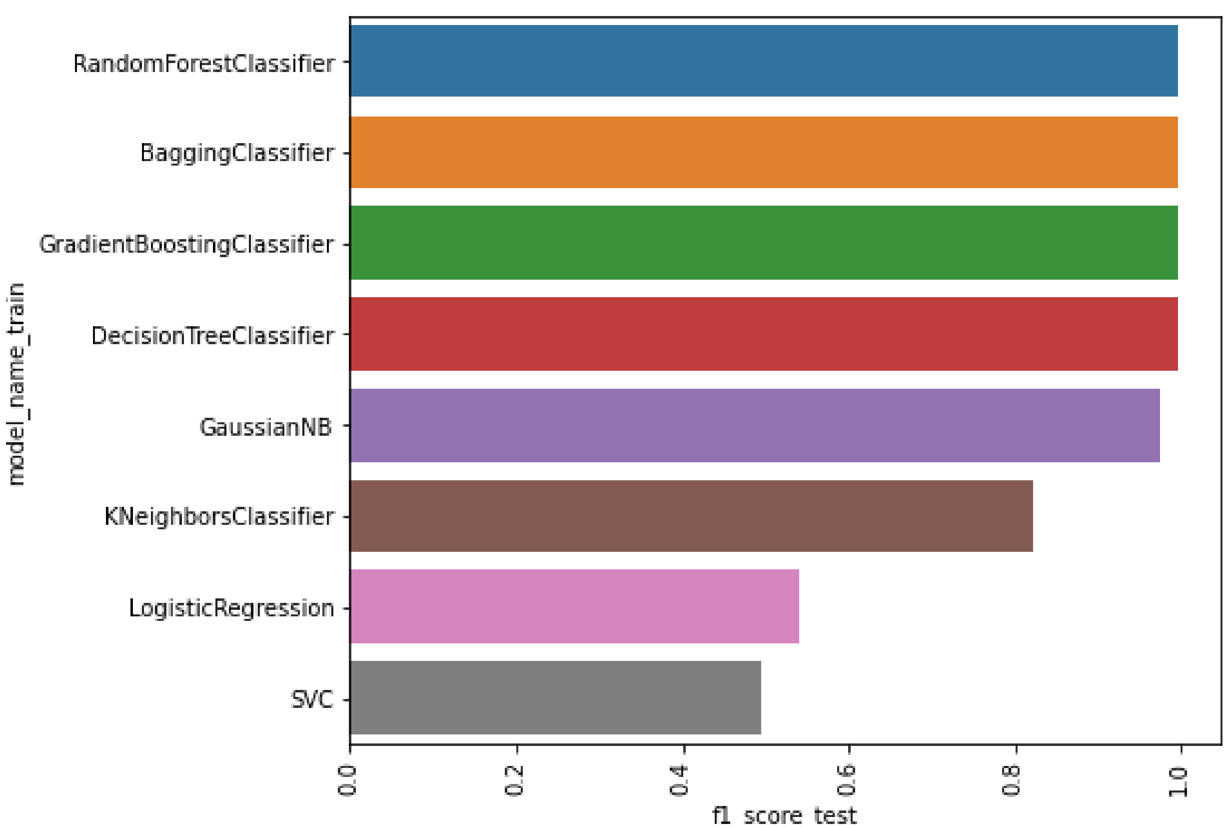
```
 [   18   192]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13772
           1       0.99      0.91      0.95       210

    accuracy                           1.00     13982
   macro avg       0.99      0.96      0.97     13982
weighted avg       1.00      1.00      1.00     13982

metrics on train data
[[32126     7]
 [   29   461]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     32133
           1       0.99      0.94      0.96       490

    accuracy                           1.00     32623
   macro avg       0.99      0.97      0.98     32623
weighted avg       1.00      1.00      1.00     32623

==============================
<class 'sklearn.svm._classes.SVC'>
metrics on test data
[[13772     0]
 [  210     0]]


              precision    recall  f1-score   support

           0       0.98      1.00      0.99     13772
           1       0.00      0.00      0.00       210

    accuracy                           0.98     13982
   macro avg       0.49      0.50      0.50     13982
weighted avg       0.97      0.98      0.98     13982

metrics on train data
[[32133     0]
 [  490     0]]


              precision    recall  f1-score   support

           0       0.98      1.00      0.99     32133
           1       0.00      0.00      0.00       490
```

```
    accuracy                           0.98     32623
   macro avg       0.49      0.50      0.50     32623
weighted avg       0.97      0.98      0.98     32623

==============================
<class 'sklearn.ensemble._gb.GradientBoostingClassifier'>
metrics on test data
[[13772     0]
 [    2   208]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13772
           1       1.00      0.99      1.00       210

    accuracy                           1.00     13982
   macro avg       1.00      1.00      1.00     13982
weighted avg       1.00      1.00      1.00     13982

metrics on train data
[[32133     0]
 [    0   490]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     32133
           1       1.00      1.00      1.00       490

    accuracy                           1.00     32623
   macro avg       1.00      1.00      1.00     32623
weighted avg       1.00      1.00      1.00     32623

==============================
```

```python
summary=pd.DataFrame(model_summary).sort_values('f1_score_test',ascend
ing=False).drop('model_name_test',axis=1)

summary
```

```
             model_name_train  f1_score_train  recall_score_train  \
2      RandomForestClassifier        1.000000            1.000000
3           BaggingClassifier        1.000000            1.000000
7  GradientBoostingClassifier        1.000000            1.000000
1      DecisionTreeClassifier        1.000000            1.000000
5                  GaussianNB        0.980931            0.970299
4        KNeighborsClassifier        0.878124            0.835972
0          LogisticRegression        0.542995            0.524474
6                         SVC        0.496217            0.500000
```

```
   accuracy_score_train  f1_score_test  recall_score_test
accuracy_score_test
2              1.000000       0.997571           0.995238
0.999857
3              1.000000       0.997571           0.995238
0.999857
7              1.000000       0.997571           0.995238
0.999857
1              1.000000       0.996366           0.995202
0.999785
5              0.998896       0.974885           0.957070
0.998570
4              0.993593       0.821913           0.781954
0.990774
0              0.985685       0.541850           0.523810
0.985696
6              0.984980       0.496217           0.500000
0.984981

plt.figure(figsize=(7,6))
sns.barplot(y=summary['model_name_train'],x=summary['f1_score_test'])
plt.xticks(rotation=90)
plt.show()
```



Model selection for task 1

- from the above graph it is found that the `RandomForestClassifier`,`bagging_classifier`,`gradiant boosting` performing well compared to other algorithms
- and it is performing well above 95 percentage so not using `optimization techniques` separatly
- im considering the `RandomForestClassifier`, `gradiant boosting` model over `bagging_classifier` as it performing better in more number of times compared to baggining classifer
- will create the `RandomForestClassifier` model for further use

```
#model creation
#model initialization
high_priority_model=RandomForestClassifier()

#fitting the model
high_priority_model.fit(X_train,y_train)

#predicting using the model
high_priority_pred=high_priority_model.predict(X_test)

#printing the confusion metrics and classification report
print('metrics on test data')
print('confusion matrix')
print(confusion_matrix(y_test,high_priority_pred))
print('\n')
print('classification report')
print(classification_report(y_test,high_priority_pred))
print('==='*10)
```

```
metrics on test data
confusion matrix
[[13772     0]
 [    2   208]]


classification report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13772
           1       1.00      0.99      1.00       210

    accuracy                           1.00     13982
   macro avg       1.00      1.00      1.00     13982
weighted avg       1.00      1.00      1.00     13982

==============================
```

TASK-2 | FORECASTING

2. Forecast the incident volume in different fields , quarterly and annual. So that they can be better prepared with resources and technology planning.

```
data_1=df.copy()

data_1.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
3       1         57   88       0       4        4         4
0.642927
4       1         57   88       0       4        4         4
0.345258

   Category  KB_number  No_of_Reassignments           Open_Time  \
0         1        553                   26 2012-05-02 13:32:00
1         1        611                   33 2012-12-03 15:44:00
2         3        339                    3 2012-03-29 12:36:00
3         1        611                   13 2012-07-17 11:49:00
4         1        611                    2 2012-10-08 11:01:00

        Resolved_Time          Close_Time  No_of_Related_Interactions
\
0 2013-04-11 13:50:00 2013-04-11 13:51:00                           1

1 2013-02-12 12:36:00 2013-02-12 12:36:00                           1

2 2014-01-13 15:12:00 2014-01-13 15:13:00                           1

3 2013-11-14 09:31:00 2013-11-14 09:31:00                           1

4 2013-08-11 13:55:00 2013-08-11 13:55:00                           1


   Handle_Time_hrs_conv
0           8256.316667
1           1700.866667
2          15722.616667
3          11637.700000
4           7370.900000

data_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 46605 entries, 0 to 46605
Data columns (total 16 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   CI_Cat                        46605 non-null  int32
 1   CI_Subcat                     46605 non-null  int32
 2   WBS                           46605 non-null  int32
 3   Status                        46605 non-null  int32
 4   Impact                        46605 non-null  int32
 5   Urgency                       46605 non-null  int32
 6   Priority                      46605 non-null  int32
 7   number_cnt                    46605 non-null  float64
 8   Category                      46605 non-null  int32
 9   KB_number                     46605 non-null  int32
 10  No_of_Reassignments           46605 non-null  int32
 11  Open_Time                     46605 non-null  datetime64[ns]
 12  Resolved_Time                 46605 non-null  datetime64[ns]
 13  Close_Time                    46605 non-null  datetime64[ns]
 14  No_of_Related_Interactions    46605 non-null  int32
 15  Handle_Time_hrs_conv          46605 non-null  float64
dtypes: datetime64[ns](3), float64(2), int32(11)
memory usage: 5.3 MB
```

- **sorting the data based on the ticket opening time**

```
timeseries_data=data_1.sort_values('Open_Time')

timeseries_data.head()

    CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
11       1         57   88       0       4        4         4
0.291928
12       1         57   55       0       4        4         4
0.776486
9        1         57   55       0       4        4         4
0.306670
2        1         10   92       0       4        3         4
0.517551
0       11         57  162       0       4        4         4
0.601292

    Category  KB_number  No_of_Reassignments           Open_Time  \
```

```
11          1        611                       8 2012-01-10 10:49:00
12          1        401                       5 2012-02-10 12:12:00
9           1        401                       2 2012-03-09 16:04:00
2           3        339                       3 2012-03-29 12:36:00
0           1        553                      26 2012-05-02 13:32:00

            Resolved_Time          Close_Time  No_of_Related_Interactions \
11 2013-08-11 14:18:00 2013-08-11 14:22:00                           1

12 2014-04-02 09:38:00 2014-04-02 09:38:00                           2

9  2013-08-11 14:33:00 2013-08-11 14:35:00                           1

2  2014-01-13 15:12:00 2014-01-13 15:13:00                           1

0  2013-04-11 13:50:00 2013-04-11 13:51:00                           1


    Handle_Time_hrs_conv
11          13899.550000
12          18765.433333
9           12478.516667
2           15722.616667
0            8256.316667
```

- as each time a single ticket raised from each department
- taking only CI_Cat column along with open_time
- will also consider only date neglecting the time in the timestamp

```
forecast_data=timeseries_data[['CI_Cat','Open_Time']]

forecast_data['Open_Time']=forecast_data['Open_Time'].dt.date

forecast_data.head()
```

```
    CI_Cat   Open_Time
11       1  2012-01-10
12       1  2012-02-10
9        1  2012-03-09
2        1  2012-03-29
0       11  2012-05-02
```

- **grouping is doing through the concept of pivot_table**

```
pivot_table = forecast_data.pivot_table(index='Open_Time',
columns='CI_Cat', aggfunc='size')

pd.set_option('display.max_rows',None)

pivot_table

CI_Cat        0      1    2     3    4     5     6    7     8     9
10  \
Open_Time

2012-01-10  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-02-10  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-03-09  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-03-29  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-05-02  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-05-12  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-07-12  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-07-17  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-08-15  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-08-22  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-08-29  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-09-21  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-10-08  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-10-12  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-10-15  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-10-18  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-10-23  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-11-21  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2012-12-03  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
```

```
NaN
2012-12-24  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-01-03  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-01-05  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-01-07  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-01-08  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-01-10  NaN  286.0  NaN  55.0  3.0   1.0  10.0  NaN   NaN   1.0
4.0
2013-01-11  NaN  233.0  NaN  18.0  NaN   3.0   1.0  NaN   NaN   5.0
3.0
2013-01-12  NaN    4.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   2.0
NaN
2013-01-15  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-01-22  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-01-23  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-01-30  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-01-31  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-02-07  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-02-09  NaN    7.0  NaN   2.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-02-10  NaN  294.0  NaN  34.0  NaN   NaN   2.0  1.0   1.0   4.0
5.0
2013-02-11  NaN    3.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   2.0
NaN
2013-02-12  NaN  279.0  NaN  24.0  1.0   1.0   1.0  1.0   1.0   4.0
5.0
2013-02-18  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-02-19  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-02-20  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-02-25  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-02-26  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-02-28  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
```

```
2013-03-04  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-03-05  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-03-06  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-03-07  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-03-09  NaN    8.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-03-10  NaN  333.0  NaN  29.0  NaN   2.0   4.0  NaN   1.0   3.0
5.0
2013-03-11  NaN    3.0  NaN   NaN  NaN   NaN   1.0  NaN   NaN   NaN
2.0
2013-03-12  NaN  282.0  NaN  43.0  NaN   1.0   2.0  4.0   2.0   NaN
7.0
2013-03-15  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-03-21  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-03-26  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-03-27  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-04-02  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-04-03  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-04-04  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-04-06  NaN    3.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-04-07  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-04-09  NaN    8.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-04-10  NaN  260.0  NaN  30.0  NaN   NaN   2.0  NaN   1.0   NaN
1.0
2013-04-11  NaN  321.0  NaN  33.0  NaN   2.0   2.0  2.0   NaN   2.0
16.0
2013-04-12  NaN  267.0  NaN  25.0  1.0   3.0   1.0  NaN   NaN   3.0
6.0
2013-04-16  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-04-17  NaN    3.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-04-19  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-04-22  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
```

```
NaN
2013-04-24  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-04-25  NaN    3.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-04-26  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-03  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-04  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-06  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-07  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-08  NaN    3.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-09  NaN    5.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-10  NaN    4.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-11  NaN  322.0  NaN  35.0  3.0   2.0   2.0  2.0   2.0   3.0
10.0
2013-05-12  NaN  229.0  1.0  29.0  NaN  10.0   1.0  NaN   6.0   2.0
6.0
2013-05-13  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-15  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-22  NaN    5.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-23  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-24  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-27  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-29  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-30  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-05-31  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-02  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-05  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-06  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
```

```
2013-06-08  NaN    2.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-09  NaN    7.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-10  NaN    3.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-11  NaN  328.0  1.0  38.0  NaN   2.0   4.0  NaN   4.0   7.0
7.0
2013-06-12  NaN  226.0  NaN  20.0  NaN   1.0   NaN  1.0   1.0   NaN
5.0
2013-06-13  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-14  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-17  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-18  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-19  NaN    2.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-20  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-24  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-26  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-27  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-06-28  NaN    3.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-05  NaN    5.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-08  NaN    7.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-10  NaN  347.0  NaN  29.0  1.0   7.0   2.0  NaN   2.0   1.0
4.0
2013-07-11  NaN  270.0  NaN  42.0  NaN   1.0   2.0  NaN   1.0   4.0
7.0
2013-07-12  NaN    4.0  NaN   1.0  NaN   NaN   1.0  NaN   NaN   NaN
NaN
2013-07-15  NaN    6.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-16  NaN    7.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-17  NaN    3.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-19  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-22  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
```

```
NaN
2013-07-23  NaN    4.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-24  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-25  NaN    3.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-26  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-29  NaN    6.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-30  NaN    4.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-07-31  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-02  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-07  NaN    4.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-08  NaN    5.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-10  NaN  285.0  NaN  27.0  2.0   3.0   2.0  5.0   1.0   1.0
4.0
2013-08-11  NaN  195.0  NaN  26.0  NaN   1.0   4.0  NaN   2.0   4.0
5.0
2013-08-12  NaN    1.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-13  NaN    6.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-14  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-15  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-16  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-19  NaN    5.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-20  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-21  NaN    5.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-22  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-23  NaN    6.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-26  NaN    4.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-27  NaN    8.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
```

```
2013-08-28  NaN    3.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-29  NaN    4.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-08-30  NaN    8.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-04  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-07  NaN    3.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-08  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-09  NaN   10.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-10  NaN  247.0  NaN  32.0  NaN   2.0   1.0  1.0   1.0   1.0
2.0
2013-09-11  NaN    4.0  NaN   1.0  NaN   NaN   NaN  1.0   NaN   NaN
NaN
2013-09-12  NaN  306.0  NaN  35.0  NaN   1.0   1.0  NaN   NaN   NaN
4.0
2013-09-13  NaN    7.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-16  NaN   13.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-17  NaN   27.0  NaN   2.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-18  NaN   21.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-19  NaN   16.0  NaN   3.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-20  NaN   13.0  NaN   2.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-23  NaN   38.0  NaN   5.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-24  NaN   68.0  NaN   6.0  1.0   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-25  NaN   59.0  NaN   2.0  NaN   NaN   NaN  NaN   NaN   1.0
NaN
2013-09-26  NaN   70.0  NaN  17.0  NaN   NaN   2.0  NaN   1.0   1.0
1.0
2013-09-27  NaN   68.0  NaN  11.0  NaN   2.0   NaN  NaN   NaN   NaN
NaN
2013-09-28  NaN    NaN  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-09-30  NaN  172.0  NaN  15.0  3.0   NaN   1.0  NaN   1.0   NaN
NaN
2013-10-04  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-10-05  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
```

```
NaN
2013-10-06  NaN    4.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-10-07  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-10-09  NaN   15.0  NaN   NaN  NaN   1.0   NaN  NaN   NaN   NaN
NaN
2013-10-10  NaN  276.0  NaN  42.0  NaN   1.0   2.0  2.0   NaN   1.0
1.0
2013-10-11  NaN    5.0  NaN   NaN  NaN   NaN   NaN  1.0   NaN   NaN
NaN
2013-10-12  NaN  283.0  NaN  40.0  3.0   3.0   2.0  2.0   1.0   1.0
6.0
2013-10-13  NaN    2.0  NaN   NaN  NaN   NaN   NaN  1.0   NaN   NaN
NaN
2013-10-14  NaN  349.0  NaN  36.0  1.0   3.0   2.0  1.0   2.0   2.0
2.0
2013-10-15  NaN  319.0  NaN  38.0  2.0   5.0   4.0  NaN   2.0   6.0
2.0
2013-10-16  NaN  192.0  NaN  25.0  NaN   2.0   3.0  NaN   2.0   2.0
5.0
2013-10-17  NaN  266.0  NaN  33.0  1.0   4.0   5.0  NaN   1.0   1.0
4.0
2013-10-18  NaN  201.0  NaN  28.0  NaN   NaN   NaN  NaN   3.0   4.0
1.0
2013-10-19  NaN    2.0  NaN   2.0  NaN   NaN   1.0  NaN   NaN   NaN
NaN
2013-10-20  NaN    4.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-10-21  NaN  278.0  1.0  34.0  NaN   1.0   2.0  NaN   1.0   5.0
2.0
2013-10-22  NaN  260.0  NaN  28.0  NaN   2.0   NaN  NaN   1.0   4.0
7.0
2013-10-23  NaN  167.0  NaN  29.0  1.0   NaN   2.0  2.0   1.0   4.0
1.0
2013-10-24  NaN  253.0  NaN  29.0  2.0   3.0   3.0  NaN   1.0   2.0
4.0
2013-10-25  NaN  192.0  NaN  14.0  NaN   1.0   1.0  1.0   NaN   1.0
8.0
2013-10-27  NaN    4.0  NaN   NaN  NaN   NaN   1.0  NaN   NaN   NaN
NaN
2013-10-28  NaN  281.0  NaN  43.0  NaN   2.0   5.0  1.0   1.0   4.0
8.0
2013-10-29  NaN  264.0  NaN  34.0  NaN   1.0   3.0  1.0   2.0   3.0
3.0
2013-10-30  NaN  284.0  NaN  35.0  2.0   2.0   1.0  1.0   4.0   1.0
3.0
2013-10-31  NaN  219.0  NaN  33.0  1.0   NaN   4.0  1.0   1.0   7.0
4.0
```

```
2013-11-03  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-11-06  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-11-07  NaN    5.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-11-09  NaN   24.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   1.0
1.0
2013-11-10  NaN  227.0  NaN  22.0  NaN   NaN   3.0  3.0   1.0   1.0
5.0
2013-11-11  NaN  281.0  NaN  47.0  4.0   3.0   2.0  NaN   2.0   2.0
13.0
2013-11-12  NaN  243.0  NaN  14.0  NaN   2.0   1.0  1.0   1.0   1.0
10.0
2013-11-13  NaN  250.0  NaN  25.0  NaN   NaN   NaN  NaN   1.0   3.0
3.0
2013-11-14  NaN  247.0  NaN  41.0  NaN   NaN   NaN  NaN   1.0   3.0
6.0
2013-11-15  NaN  186.0  NaN  26.0  NaN   1.0   1.0  NaN   3.0   1.0
5.0
2013-11-16  NaN   26.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-11-17  NaN    3.0  1.0   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-11-18  NaN  425.0  NaN  45.0  NaN   6.0   1.0  2.0   NaN   3.0
5.0
2013-11-19  NaN  321.0  NaN  39.0  NaN   2.0   2.0  1.0   NaN   2.0
5.0
2013-11-20  NaN  231.0  NaN  27.0  4.0   2.0   1.0  3.0   2.0   1.0
5.0
2013-11-21  NaN  268.0  NaN  43.0  NaN   3.0   NaN  NaN   2.0   2.0
5.0
2013-11-22  NaN  267.0  NaN  30.0  NaN   NaN   2.0  NaN   1.0   4.0
4.0
2013-11-23  NaN    6.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-11-24  NaN    5.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-11-25  NaN  339.0  1.0  33.0  1.0   NaN   2.0  1.0  10.0   1.0
5.0
2013-11-26  NaN  272.0  NaN  28.0  1.0   2.0   5.0  1.0   2.0   2.0
3.0
2013-11-27  NaN  264.0  NaN  22.0  2.0   2.0   3.0  NaN   1.0   1.0
3.0
2013-11-28  1.0  286.0  NaN  27.0  2.0   1.0   2.0  NaN   1.0   4.0
7.0
2013-11-29  NaN  196.0  NaN  15.0  NaN   2.0   2.0  1.0   2.0   NaN
6.0
2013-11-30  NaN    6.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
```

```
NaN
2013-12-03  NaN    NaN  NaN    NaN  NaN    NaN    NaN  NaN    NaN    NaN
NaN
2013-12-06  NaN    2.0  NaN    NaN  NaN    NaN    NaN  NaN    NaN    NaN
NaN
2013-12-07  NaN    3.0  NaN    NaN  NaN    NaN    NaN  NaN    NaN    NaN
NaN
2013-12-08  NaN    4.0  NaN    NaN  NaN    NaN    NaN  NaN    NaN    NaN
NaN
2013-12-09  NaN   13.0  NaN    2.0  NaN    NaN    NaN  NaN    NaN    NaN
NaN
2013-12-10  NaN    3.0  NaN    NaN  NaN    NaN    NaN  1.0    NaN    NaN
NaN
2013-12-11  NaN  277.0  NaN   44.0  1.0    NaN    3.0  2.0    NaN    3.0
3.0
2013-12-12  NaN  249.0  NaN   38.0  NaN    6.0    4.0  NaN    2.0    2.0
11.0
2013-12-13  NaN  186.0  NaN   29.0  1.0    NaN    NaN  2.0    1.0    1.0
5.0
2013-12-14  NaN    2.0  NaN    1.0  NaN    NaN    NaN  1.0    NaN    NaN
NaN
2013-12-15  NaN    1.0  NaN    NaN  NaN    NaN    NaN  NaN    NaN    1.0
NaN
2013-12-16  NaN  296.0  NaN   47.0  2.0    1.0    NaN  NaN    NaN    2.0
7.0
2013-12-17  NaN  275.0  NaN   32.0  NaN    NaN    3.0  NaN    1.0    1.0
10.0
2013-12-18  NaN  240.0  NaN   30.0  NaN    2.0    2.0  1.0    1.0    1.0
5.0
2013-12-19  NaN  249.0  NaN   32.0  3.0    2.0    4.0  1.0    NaN    1.0
7.0
2013-12-20  NaN  180.0  NaN   23.0  1.0    1.0    1.0  NaN    NaN    NaN
1.0
2013-12-21  NaN    3.0  NaN    4.0  NaN    NaN    NaN  1.0    NaN    NaN
NaN
2013-12-22  NaN    3.0  NaN    NaN  NaN    NaN    NaN  NaN    NaN    1.0
2.0
2013-12-23  NaN  223.0  NaN   35.0  NaN    1.0    NaN  1.0    1.0    4.0
2.0
2013-12-24  NaN  256.0  NaN   13.0  1.0    1.0    NaN  3.0    2.0    2.0
2.0
2013-12-25  NaN    1.0  NaN    NaN  NaN    NaN    NaN  NaN    NaN    NaN
NaN
2013-12-26  NaN    2.0  NaN    NaN  NaN    NaN    NaN  NaN    NaN    NaN
NaN
2013-12-27  NaN  120.0  NaN   20.0  NaN    1.0    2.0  NaN    1.0    1.0
6.0
2013-12-28  NaN    2.0  NaN    NaN  NaN    NaN    NaN  NaN    NaN    NaN
NaN
```

```
2013-12-29  NaN    2.0  NaN   2.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2013-12-30  NaN  200.0  NaN  21.0  2.0   1.0   3.0  NaN   NaN   NaN
6.0
2013-12-31  NaN  198.0  NaN  11.0  NaN   2.0   NaN  NaN   1.0   2.0
1.0
2014-01-02  NaN    6.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2014-01-03  NaN    6.0  NaN   NaN  NaN   NaN   1.0  NaN   NaN   1.0
NaN
2014-01-13  NaN  243.0  NaN  32.0  NaN   1.0   4.0  1.0   1.0   2.0
2.0
2014-01-14  NaN  247.0  NaN  37.0  1.0   2.0   4.0  1.0   NaN   6.0
3.0
2014-01-15  NaN  225.0  NaN  22.0  2.0   NaN   3.0  NaN   1.0   2.0
6.0
2014-01-16  NaN  228.0  NaN  29.0  1.0   2.0   4.0  NaN   1.0   1.0
6.0
2014-01-17  NaN  190.0  NaN  15.0  3.0   1.0   2.0  1.0   2.0   2.0
5.0
2014-01-18  NaN    9.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
1.0
2014-01-19  NaN    5.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2014-01-20  NaN  298.0  NaN  30.0  3.0   NaN   2.0  1.0   NaN   3.0
9.0
2014-01-21  NaN  330.0  NaN  48.0  3.0   NaN   6.0  NaN   2.0   3.0
9.0
2014-01-22  NaN  259.0  NaN  23.0  2.0   NaN   5.0  1.0   NaN   3.0
4.0
2014-01-23  NaN  278.0  NaN  25.0  4.0   3.0   6.0  NaN   2.0   1.0
4.0
2014-01-24  NaN  219.0  NaN  17.0  2.0   4.0   1.0  NaN   NaN   2.0
4.0
2014-01-25  NaN    7.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2014-01-26  NaN    2.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2014-01-27  NaN  402.0  NaN  18.0  6.0   NaN   1.0  1.0   NaN   3.0
9.0
2014-01-28  NaN  268.0  NaN  29.0  1.0   NaN   5.0  1.0   2.0   3.0
4.0
2014-01-29  NaN  288.0  NaN  31.0  NaN   1.0   5.0  1.0   1.0   4.0
7.0
2014-01-30  NaN  313.0  NaN  28.0  1.0   1.0   5.0  NaN   1.0  15.0
8.0
2014-01-31  NaN  245.0  NaN  18.0  1.0   3.0   2.0  1.0   NaN   4.0
2.0
2014-02-01  NaN  233.0  NaN  30.0  4.0   1.0  12.0  NaN   2.0   3.0
4.0
```

2014-02-02	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2014-02-03	NaN	1.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	2.0
2014-02-13	NaN	262.0	NaN	27.0	1.0	3.0	4.0	1.0	NaN	3.0	4.0
2014-02-14	NaN	172.0	NaN	24.0	3.0	2.0	3.0	1.0	NaN	NaN	5.0
2014-02-15	NaN	13.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2014-02-16	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	1.0
2014-02-17	1.0	337.0	NaN	34.0	1.0	6.0	3.0	NaN	NaN	2.0	7.0
2014-02-18	NaN	274.0	NaN	24.0	3.0	NaN	1.0	2.0	1.0	2.0	2.0
2014-02-19	NaN	252.0	NaN	21.0	2.0	7.0	8.0	NaN	1.0	2.0	4.0
2014-02-20	NaN	240.0	NaN	20.0	1.0	1.0	2.0	2.0	1.0	1.0	8.0
2014-02-21	NaN	207.0	NaN	34.0	NaN	2.0	2.0	1.0	NaN	1.0	7.0
2014-02-22	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2014-02-23	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2014-02-24	NaN	253.0	NaN	25.0	8.0	NaN	1.0	NaN	2.0	2.0	4.0
2014-02-25	NaN	257.0	NaN	27.0	7.0	1.0	1.0	1.0	NaN	1.0	5.0
2014-02-26	NaN	226.0	NaN	18.0	4.0	NaN	4.0	1.0	NaN	NaN	3.0
2014-02-27	NaN	251.0	NaN	26.0	6.0	2.0	5.0	NaN	NaN	2.0	6.0
2014-02-28	NaN	181.0	NaN	15.0	1.0	NaN	1.0	1.0	3.0	2.0	12.0
2014-03-01	NaN	182.0	NaN	20.0	NaN	NaN	6.0	NaN	3.0	1.0	9.0
2014-03-02	NaN	345.0	NaN	29.0	6.0	NaN	4.0	NaN	1.0	4.0	7.0
2014-03-03	NaN	185.0	NaN	24.0	4.0	NaN	1.0	NaN	NaN	4.0	6.0
2014-03-13	NaN	200.0	NaN	30.0	4.0	NaN	3.0	1.0	NaN	4.0	3.0
2014-03-14	NaN	197.0	NaN	33.0	1.0	NaN	4.0	NaN	NaN	5.0	7.0
2014-03-15	NaN	3.0	NaN	1.0	NaN	NaN	1.0	NaN	NaN	NaN	NaN
2014-03-16	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

```
NaN
2014-03-17  NaN  228.0  NaN  24.0  3.0   4.0   4.0  NaN   1.0   4.0
6.0
2014-03-18  NaN  233.0  NaN  26.0  NaN   1.0   4.0  NaN   1.0   1.0
6.0
2014-03-19  NaN  232.0  NaN  19.0  1.0   3.0   5.0  1.0   NaN   5.0
6.0
2014-03-20  NaN  168.0  NaN  19.0  NaN   3.0   6.0  2.0   NaN   2.0
5.0
2014-03-21  NaN  160.0  NaN  12.0  4.0   2.0   3.0  NaN   NaN   1.0
8.0
2014-03-22  NaN    1.0  NaN   NaN  NaN   NaN   1.0  1.0   NaN   NaN
1.0
2014-03-23  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   1.0
NaN
2014-03-24  NaN  245.0  NaN  26.0  3.0   2.0   5.0  NaN   1.0   2.0
11.0
2014-03-25  NaN  218.0  NaN  23.0  2.0   1.0   1.0  NaN   NaN   3.0
7.0
2014-03-26  NaN  214.0  NaN   8.0  2.0   NaN   1.0  2.0   1.0   3.0
5.0
2014-03-27  NaN  188.0  NaN  14.0  NaN   2.0   2.0  NaN  10.0   2.0
2.0
2014-03-28  NaN  136.0  NaN   9.0  NaN   NaN   5.0  NaN   1.0   2.0
7.0
2014-03-29  NaN    2.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2014-03-30  NaN    2.0  NaN   1.0  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2014-03-31  NaN  160.0  NaN   3.0  3.0   NaN   3.0  1.0   NaN   1.0
2.0
2014-04-01  NaN    NaN  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2014-04-02  NaN  298.0  NaN  37.0  4.0   1.0   3.0  1.0   2.0   4.0
4.0
2014-04-03  NaN  220.0  NaN  27.0  5.0   1.0   3.0  NaN   NaN   1.0
5.0
2014-05-01  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2014-05-02  NaN  316.0  NaN  23.0  4.0   NaN   2.0  2.0   NaN   2.0
9.0
2014-05-03  NaN  201.0  NaN  26.0  2.0   2.0   8.0  NaN   NaN   NaN
6.0
2014-06-01  NaN  280.0  NaN  29.0  7.0   NaN   5.0  1.0   3.0   3.0
9.0
2014-06-02  NaN  281.0  NaN  29.0  NaN   1.0   3.0  2.0   3.0   3.0
10.0
2014-06-03  NaN  231.0  NaN  31.0  2.0   NaN   2.0  2.0   3.0   4.0
7.0
```

```
2014-07-01  NaN  297.0  NaN  25.0  3.0   1.0   1.0  1.0   3.0   9.0
8.0
2014-07-02  NaN  247.0  NaN  21.0  6.0   2.0   4.0  1.0   NaN   NaN
9.0
2014-07-03  NaN  199.0  NaN  17.0  1.0   1.0   2.0  2.0   NaN   6.0
1.0
2014-08-01  NaN  249.0  NaN  33.0  2.0   3.0   3.0  1.0   1.0   4.0
9.0
2014-08-02  NaN    3.0  NaN   1.0  NaN   NaN   1.0  NaN   NaN   NaN
NaN
2014-08-03  NaN    6.0  NaN   1.0  NaN   NaN   NaN  1.0   NaN   NaN
NaN
2014-09-01  NaN  253.0  NaN  34.0  1.0   2.0   5.0  1.0   1.0   3.0
5.0
2014-09-02  NaN    NaN  NaN   NaN  NaN   NaN   1.0  NaN   NaN   NaN
NaN
2014-09-03  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2014-10-01  NaN  224.0  NaN  16.0  4.0   1.0  10.0  NaN   NaN   1.0
4.0
2014-10-02  NaN  296.0  NaN  29.0  3.0   2.0   5.0  1.0   1.0   4.0
5.0
2014-10-03  NaN  232.0  NaN  31.0  4.0   6.0  10.0  NaN   NaN   2.0
10.0
2014-11-01  NaN    3.0  NaN   NaN  NaN   NaN  50.0  NaN   NaN   NaN
NaN
2014-11-02  NaN  271.0  NaN  21.0  2.0   NaN   2.0  1.0   NaN   2.0
5.0
2014-11-03  NaN  241.0  NaN  33.0  3.0   4.0   2.0  1.0   1.0   5.0
5.0
2014-12-01  NaN    1.0  NaN   NaN  NaN   NaN   NaN  NaN   NaN   NaN
NaN
2014-12-02  NaN  228.0  NaN  23.0  3.0   4.0   2.0  4.0   1.0   1.0
4.0
2014-12-03  NaN  221.0  NaN  24.0  1.0   NaN  17.0  NaN   NaN   4.0
4.0

CI_Cat        11
Open_Time
2012-01-10   NaN
2012-02-10   NaN
2012-03-09   NaN
2012-03-29   NaN
2012-05-02   1.0
2012-05-12   1.0
2012-07-12   NaN
2012-07-17   NaN
2012-08-15   NaN
2012-08-22   NaN
```

```
2012-08-29   NaN
2012-09-21   NaN
2012-10-08   NaN
2012-10-12   NaN
2012-10-15   NaN
2012-10-18   NaN
2012-10-23   NaN
2012-11-21   1.0
2012-12-03   NaN
2012-12-24   NaN
2013-01-03   1.0
2013-01-05   NaN
2013-01-07   NaN
2013-01-08   NaN
2013-01-10  79.0
2013-01-11  45.0
2013-01-12   NaN
2013-01-15   NaN
2013-01-22   NaN
2013-01-23   NaN
2013-01-30   1.0
2013-01-31   1.0
2013-02-07   1.0
2013-02-09   3.0
2013-02-10  71.0
2013-02-11   NaN
2013-02-12  73.0
2013-02-18   NaN
2013-02-19   NaN
2013-02-20   NaN
2013-02-25   1.0
2013-02-26   NaN
2013-02-28   NaN
2013-03-04   NaN
2013-03-05   NaN
2013-03-06   1.0
2013-03-07   NaN
2013-03-09   1.0
2013-03-10  78.0
2013-03-11   NaN
2013-03-12  50.0
2013-03-15   NaN
2013-03-21   NaN
2013-03-26   1.0
2013-03-27   1.0
2013-04-02   NaN
2013-04-03   1.0
2013-04-04   NaN
2013-04-06   NaN
```

2013-04-07	1.0
2013-04-09	1.0
2013-04-10	51.0
2013-04-11	76.0
2013-04-12	61.0
2013-04-16	NaN
2013-04-17	NaN
2013-04-19	1.0
2013-04-22	NaN
2013-04-24	NaN
2013-04-25	NaN
2013-04-26	NaN
2013-05-03	NaN
2013-05-04	NaN
2013-05-06	1.0
2013-05-07	NaN
2013-05-08	NaN
2013-05-09	4.0
2013-05-10	1.0
2013-05-11	73.0
2013-05-12	46.0
2013-05-13	NaN
2013-05-15	NaN
2013-05-22	NaN
2013-05-23	NaN
2013-05-24	NaN
2013-05-27	1.0
2013-05-29	NaN
2013-05-30	NaN
2013-05-31	NaN
2013-06-02	NaN
2013-06-05	1.0
2013-06-06	NaN
2013-06-08	NaN
2013-06-09	2.0
2013-06-10	1.0
2013-06-11	67.0
2013-06-12	41.0
2013-06-13	1.0
2013-06-14	NaN
2013-06-17	NaN
2013-06-18	NaN
2013-06-19	NaN
2013-06-20	NaN
2013-06-24	NaN
2013-06-26	NaN
2013-06-27	1.0
2013-06-28	1.0
2013-07-05	NaN

```
2013-07-08   NaN
2013-07-10  63.0
2013-07-11  52.0
2013-07-12   NaN
2013-07-15   1.0
2013-07-16   NaN
2013-07-17   1.0
2013-07-19   NaN
2013-07-22   NaN
2013-07-23   1.0
2013-07-24   NaN
2013-07-25   NaN
2013-07-26   NaN
2013-07-29   NaN
2013-07-30   NaN
2013-07-31   NaN
2013-08-02   NaN
2013-08-07   NaN
2013-08-08   1.0
2013-08-10  77.0
2013-08-11  38.0
2013-08-12   NaN
2013-08-13   NaN
2013-08-14   1.0
2013-08-15   NaN
2013-08-16   NaN
2013-08-19   2.0
2013-08-20   1.0
2013-08-21   NaN
2013-08-22   2.0
2013-08-23   NaN
2013-08-26   1.0
2013-08-27   NaN
2013-08-28   1.0
2013-08-29   NaN
2013-08-30   NaN
2013-09-04   NaN
2013-09-07   NaN
2013-09-08   NaN
2013-09-09   NaN
2013-09-10  71.0
2013-09-11   NaN
2013-09-12  59.0
2013-09-13   1.0
2013-09-16   1.0
2013-09-17   1.0
2013-09-18   3.0
2013-09-19   2.0
2013-09-20   2.0
```

```
2013-09-23   6.0
2013-09-24   5.0
2013-09-25   9.0
2013-09-26  10.0
2013-09-27  10.0
2013-09-28   NaN
2013-09-30  27.0
2013-10-04   NaN
2013-10-05   NaN
2013-10-06   NaN
2013-10-07   NaN
2013-10-09   3.0
2013-10-10  66.0
2013-10-11   1.0
2013-10-12  37.0
2013-10-13   NaN
2013-10-14  63.0
2013-10-15  67.0
2013-10-16  57.0
2013-10-17  58.0
2013-10-18  55.0
2013-10-19   NaN
2013-10-20   NaN
2013-10-21  57.0
2013-10-22  51.0
2013-10-23  57.0
2013-10-24  77.0
2013-10-25  62.0
2013-10-27   NaN
2013-10-28  59.0
2013-10-29  54.0
2013-10-30  53.0
2013-10-31  46.0
2013-11-03   NaN
2013-11-06   NaN
2013-11-07   2.0
2013-11-09   NaN
2013-11-10  67.0
2013-11-11  61.0
2013-11-12  42.0
2013-11-13  42.0
2013-11-14  50.0
2013-11-15  62.0
2013-11-16   5.0
2013-11-17   1.0
2013-11-18  82.0
2013-11-19  66.0
2013-11-20  63.0
2013-11-21  62.0
```

```
2013-11-22  49.0
2013-11-23   2.0
2013-11-24   NaN
2013-11-25  58.0
2013-11-26  69.0
2013-11-27  70.0
2013-11-28  63.0
2013-11-29  44.0
2013-11-30   1.0
2013-12-03   1.0
2013-12-06   NaN
2013-12-07   1.0
2013-12-08   1.0
2013-12-09   6.0
2013-12-10   NaN
2013-12-11  60.0
2013-12-12  76.0
2013-12-13  48.0
2013-12-14   NaN
2013-12-15   NaN
2013-12-16  56.0
2013-12-17  54.0
2013-12-18  49.0
2013-12-19  69.0
2013-12-20  45.0
2013-12-21   NaN
2013-12-22   NaN
2013-12-23  52.0
2013-12-24  76.0
2013-12-25   NaN
2013-12-26   NaN
2013-12-27  46.0
2013-12-28   NaN
2013-12-29   1.0
2013-12-30  54.0
2013-12-31  54.0
2014-01-02   NaN
2014-01-03   1.0
2014-01-13  89.0
2014-01-14  58.0
2014-01-15  73.0
2014-01-16  49.0
2014-01-17  68.0
2014-01-18   3.0
2014-01-19   NaN
2014-01-20  66.0
2014-01-21  75.0
2014-01-22  78.0
2014-01-23  59.0
```

```
2014-01-24  55.0
2014-01-25   1.0
2014-01-26   NaN
2014-01-27  60.0
2014-01-28  80.0
2014-01-29  70.0
2014-01-30  83.0
2014-01-31  59.0
2014-02-01  55.0
2014-02-02   NaN
2014-02-03   NaN
2014-02-13  43.0
2014-02-14  39.0
2014-02-15   3.0
2014-02-16   NaN
2014-02-17  70.0
2014-02-18  56.0
2014-02-19  67.0
2014-02-20  70.0
2014-02-21  69.0
2014-02-22   NaN
2014-02-23   NaN
2014-02-24  43.0
2014-02-25  53.0
2014-02-26  61.0
2014-02-27  63.0
2014-02-28  58.0
2014-03-01  62.0
2014-03-02  70.0
2014-03-03  58.0
2014-03-13  58.0
2014-03-14  57.0
2014-03-15   NaN
2014-03-16   1.0
2014-03-17  42.0
2014-03-18  48.0
2014-03-19  59.0
2014-03-20  89.0
2014-03-21  57.0
2014-03-22   NaN
2014-03-23   NaN
2014-03-24  46.0
2014-03-25  51.0
2014-03-26  56.0
2014-03-27  49.0
2014-03-28  45.0
2014-03-29   2.0
2014-03-30   NaN
2014-03-31  44.0
```

```
2014-04-01   1.0
2014-04-02  91.0
2014-04-03  56.0
2014-05-01   NaN
2014-05-02  67.0
2014-05-03  53.0
2014-06-01  53.0
2014-06-02  82.0
2014-06-03  69.0
2014-07-01  57.0
2014-07-02  65.0
2014-07-03  51.0
2014-08-01  55.0
2014-08-02   NaN
2014-08-03   2.0
2014-09-01  61.0
2014-09-02   NaN
2014-09-03   NaN
2014-10-01  57.0
2014-10-02  75.0
2014-10-03  57.0
2014-11-01   1.0
2014-11-02  69.0
2014-11-03  51.0
2014-12-01   NaN
2014-12-02  68.0
2014-12-03  44.0
```

- converting the pivot table to dataframe

```
final_df=pd.DataFrame(pivot_table)
```

- converting the index format from object type to datetime format

```
final_df.index=pd.to_datetime(final_df.index)
```

- filling the null values with 0

```
final_df.fillna(0,inplace=True)

len(final_df)
```

```
331
```

- resampling the data on day

- # converting the daily data to quaterly year data

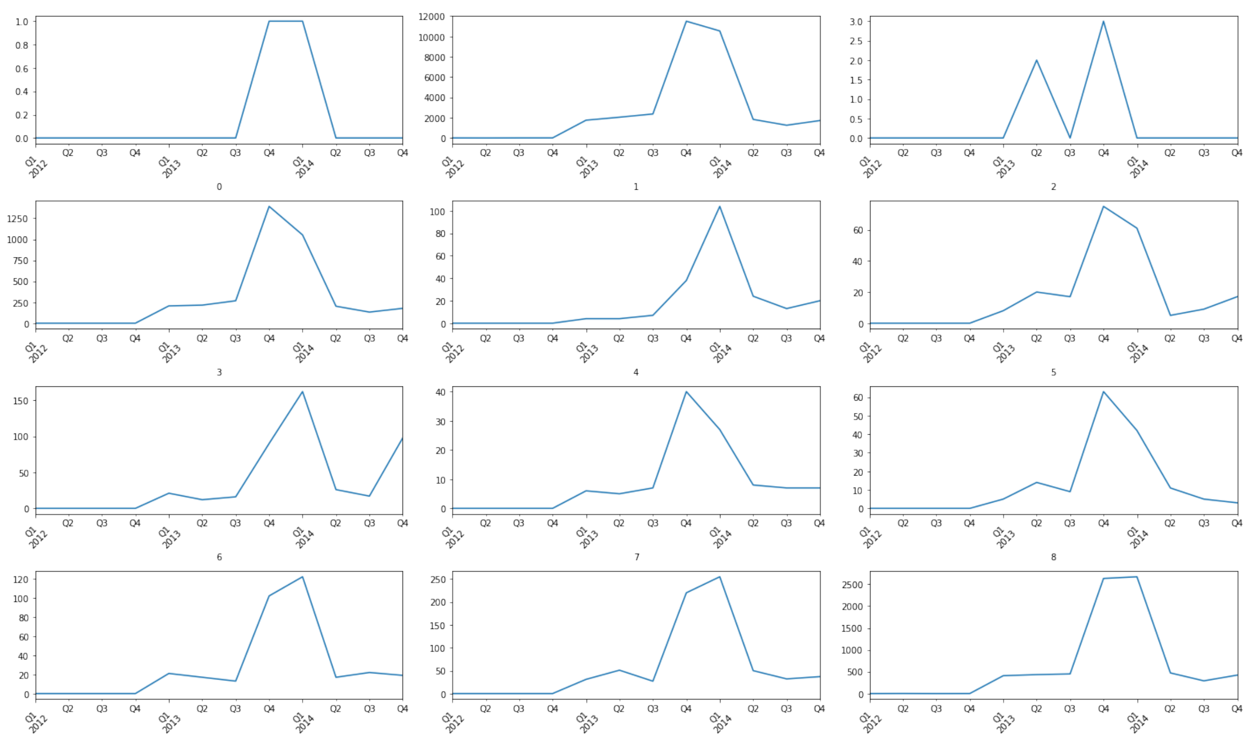
```
daily_data = final_df.resample('D', closed='right',
label='right').asfreq()

quaterly_data = daily_data.resample('Q').sum()

quaterly_data

CI_Cat        0        1    2       3      4     5      6     7     8
9   \
Open_Time

2012-03-31  0.0      4.0  0.0     0.0    0.0   0.0    0.0   0.0   0.0
0.0
2012-06-30  0.0      0.0  0.0     0.0    0.0   0.0    0.0   0.0   0.0
0.0
2012-09-30  0.0      6.0  0.0     0.0    0.0   0.0    0.0   0.0   0.0
0.0
2012-12-31  0.0      8.0  0.0     0.0    0.0   0.0    0.0   0.0   0.0
0.0
2013-03-31  0.0   1751.0  0.0   206.0    4.0   8.0   21.0   6.0   5.0
21.0
2013-06-30  0.0   2044.0  2.0   215.0    4.0  20.0   12.0   5.0  14.0
17.0
2013-09-30  0.0   2363.0  0.0   267.0    7.0  17.0   16.0   7.0   9.0
13.0
2013-12-31  1.0  11496.0  3.0  1392.0   38.0  75.0   90.0  40.0  63.0
102.0
2014-03-31  1.0  10538.0  0.0  1052.0  104.0  61.0  162.0  27.0  42.0
122.0
2014-06-30  0.0   1828.0  0.0   202.0   24.0   5.0   26.0   8.0  11.0
17.0
2014-09-30  0.0   1255.0  0.0   132.0   13.0   9.0   17.0   7.0   5.0
22.0
2014-12-31  0.0   1717.0  0.0   177.0   20.0  17.0   98.0   7.0   3.0
19.0

CI_Cat        10       11
Open_Time
2012-03-31   0.0      0.0
2012-06-30   0.0      2.0
2012-09-30   0.0      0.0
2012-12-31   0.0      1.0
```

```
2013-03-31   31.0   408.0
2013-06-30   51.0   433.0
2013-09-30   27.0   449.0
2013-12-31  220.0  2633.0
2014-03-31  255.0  2671.0
2014-06-30   50.0   472.0
2014-09-30   32.0   291.0
2014-12-31   37.0   422.0
```

```
quaterly_data.shape
```

```
(12, 12)
```

```
plt.figure(figsize=(20,12))
pl_no=1
for i in quaterly_data.columns:
  plt.subplot(4,3,pl_no)
  quaterly_data[i].plot()
  plt.xlabel(i)
  plt.xticks(rotation=45)
  pl_no+=1
plt.tight_layout()
```



```
quaterly_data
```

```
CI_Cat         0         1    2       3      4     5       6     7     8
9    \
```

```
Open_Time

2012-03-31  0.0      4.0  0.0     0.0    0.0   0.0    0.0   0.0   0.0
0.0
2012-06-30  0.0      0.0  0.0     0.0    0.0   0.0    0.0   0.0   0.0
0.0
2012-09-30  0.0      6.0  0.0     0.0    0.0   0.0    0.0   0.0   0.0
0.0
2012-12-31  0.0      8.0  0.0     0.0    0.0   0.0    0.0   0.0   0.0
0.0
2013-03-31  0.0   1751.0  0.0   206.0    4.0   8.0   21.0   6.0   5.0
21.0
2013-06-30  0.0   2044.0  2.0   215.0    4.0  20.0   12.0   5.0  14.0
17.0
2013-09-30  0.0   2363.0  0.0   267.0    7.0  17.0   16.0   7.0   9.0
13.0
2013-12-31  1.0  11496.0  3.0  1392.0   38.0  75.0   90.0  40.0  63.0
102.0
2014-03-31  1.0  10538.0  0.0  1052.0  104.0  61.0  162.0  27.0  42.0
122.0
2014-06-30  0.0   1828.0  0.0   202.0   24.0   5.0   26.0   8.0  11.0
17.0
2014-09-30  0.0   1255.0  0.0   132.0   13.0   9.0   17.0   7.0   5.0
22.0
2014-12-31  0.0   1717.0  0.0   177.0   20.0  17.0   98.0   7.0   3.0
19.0

CI_Cat         10      11
Open_Time
2012-03-31    0.0     0.0
2012-06-30    0.0     2.0
2012-09-30    0.0     0.0
2012-12-31    0.0     1.0
2013-03-31   31.0   408.0
2013-06-30   51.0   433.0
2013-09-30   27.0   449.0
2013-12-31  220.0  2633.0
2014-03-31  255.0  2671.0
2014-06-30   50.0   472.0
2014-09-30   32.0   291.0
2014-12-31   37.0   422.0
```

Stationarity check

- performing `adfuller_statistic test` on the data to check the stationarity of data

- after performin gthe adfuler test , `differencing` is performed on the data to make the data stationary

```
from statsmodels.tsa.stattools import adfuller

def perform_adf_test(data):

  stationary_cols=[]
  non_stationary_cols=[]

  for column in data.columns:

    result = adfuller(data[column])

    if result[1]<=0.05:
      print(f"{column} is stationary ")
    else:
      print(f"{column} is  not stationary ")

data_diff_1=quaterly_data.diff()

data_diff_1.dropna()

CI_Cat         0       1    2       3     4     5      6     7     8
9   \
Open_Time

2012-06-30  0.0    -4.0  0.0     0.0   0.0   0.0    0.0   0.0   0.0
0.0
2012-09-30  0.0     6.0  0.0     0.0   0.0   0.0    0.0   0.0   0.0
0.0
2012-12-31  0.0     2.0  0.0     0.0   0.0   0.0    0.0   0.0   0.0
0.0
2013-03-31  0.0  1743.0  0.0   206.0   4.0   8.0   21.0   6.0   5.0
21.0
2013-06-30  0.0   293.0  2.0     9.0   0.0  12.0   -9.0  -1.0   9.0
-4.0
2013-09-30  0.0   319.0 -2.0    52.0   3.0  -3.0    4.0   2.0  -5.0
-4.0
2013-12-31  1.0  9133.0  3.0  1125.0  31.0  58.0   74.0  33.0  54.0
89.0
2014-03-31  0.0  -958.0 -3.0  -340.0  66.0 -14.0   72.0 -13.0 -21.0
20.0
2014-06-30 -1.0 -8710.0  0.0  -850.0 -80.0 -56.0 -136.0 -19.0 -31.0 -
105.0
2014-09-30  0.0  -573.0  0.0   -70.0 -11.0   4.0   -9.0  -1.0  -6.0
5.0
2014-12-31  0.0   462.0  0.0    45.0   7.0   8.0   81.0   0.0  -2.0
-3.0
```

```
CI_Cat          10      11
Open_Time
2012-06-30    0.0     2.0
2012-09-30    0.0    -2.0
2012-12-31    0.0     1.0
2013-03-31   31.0   407.0
2013-06-30   20.0    25.0
2013-09-30  -24.0    16.0
2013-12-31  193.0  2184.0
2014-03-31   35.0    38.0
2014-06-30 -205.0 -2199.0
2014-09-30  -18.0  -181.0
2014-12-31    5.0   131.0
```

```
perform_adf_test(data_diff_1.dropna())
```

```
0 is  not stationary
1 is  not stationary
2 is stationary
3 is  not stationary
4 is  not stationary
5 is stationary
6 is stationary
7 is stationary
8 is stationary
9 is  not stationary
10 is  not stationary
11 is  not stationary
```

```
data_diff_2=quaterly_data.diff().diff()
```

```
data_diff_3=quaterly_data.diff().diff().diff()
```

```
perform_adf_test(data_diff_2.dropna())
```

```
0 is  not stationary
1 is stationary
2 is stationary
3 is  not stationary
4 is  not stationary
5 is stationary
6 is  not stationary
7 is stationary
8 is stationary
9 is  not stationary
10 is  not stationary
11 is stationary
```

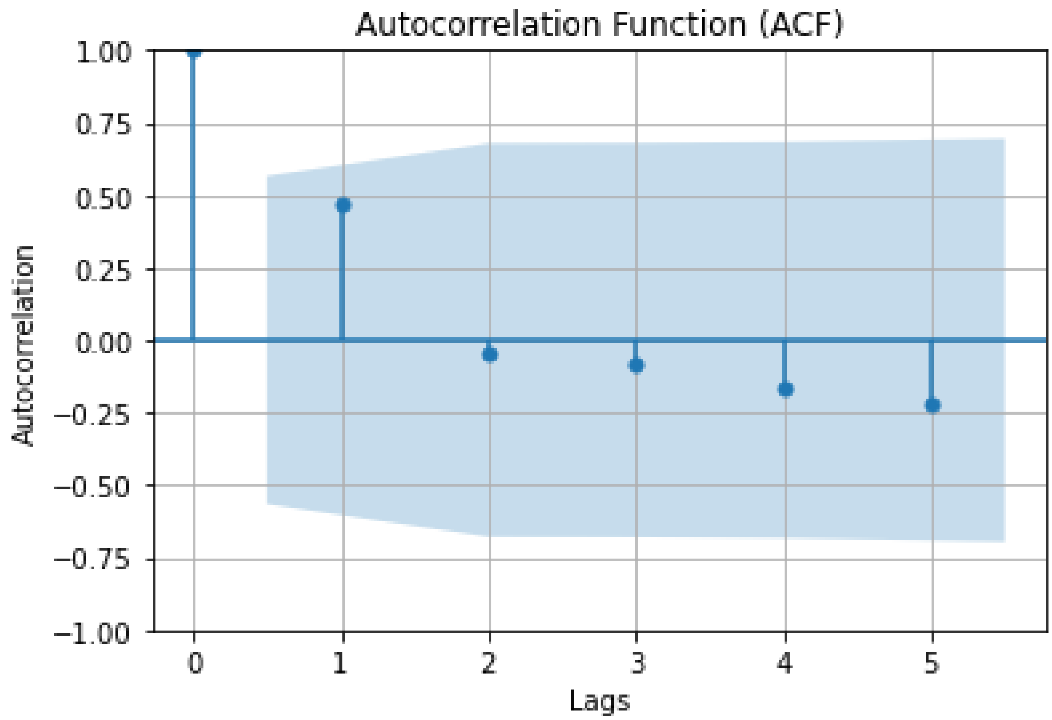
d value

as we can see at d=2 most os the columns are having coming under stationary data type so selecting the d=2 for further use

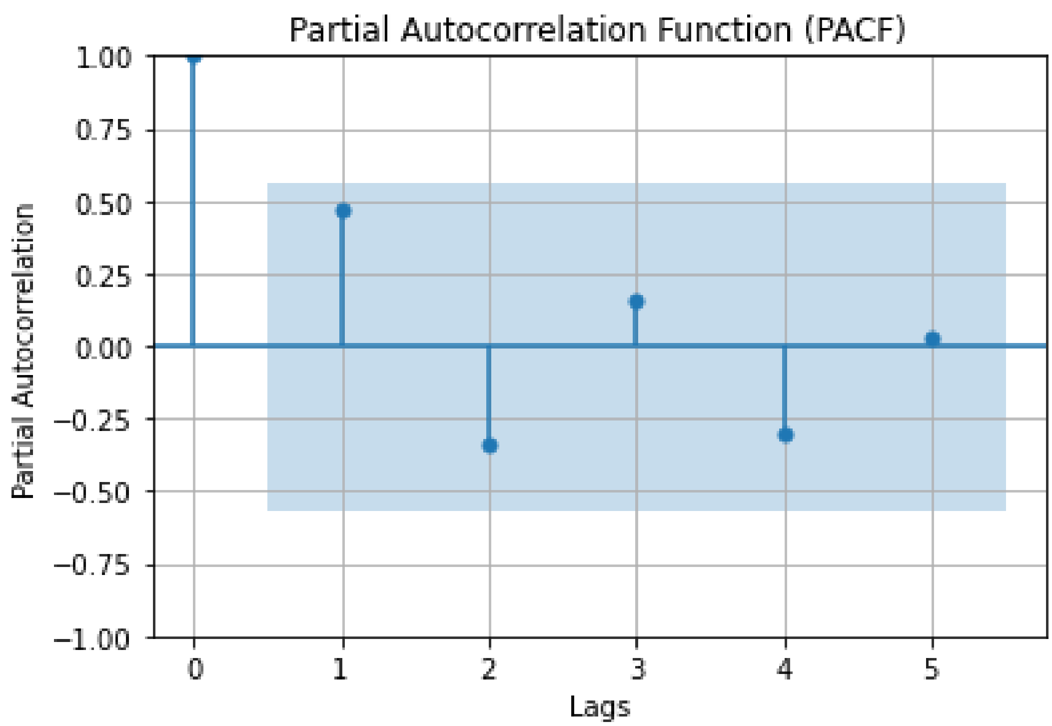
---

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Plot ACF
plt.figure(figsize=(12, 6))
plot_acf(quaterly_data[10], lags=5, alpha=0.05)
plt.xlabel('Lags')
plt.ylabel('Autocorrelation')
plt.title('Autocorrelation Function (ACF)')
plt.grid(True)
plt.show()

# Plot PACF
plt.figure(figsize=(12, 6))
plot_pacf(quaterly_data[10], lags=5, alpha=0.05)
plt.xlabel('Lags')
plt.ylabel('Partial Autocorrelation')
plt.title('Partial Autocorrelation Function (PACF)')
plt.grid(True)
plt.show()
```

```
<Figure size 864x432 with 0 Axes>
```




```
<Figure size 864x432 with 0 Axes>
```



p and q value

- from the auto_corelation selcted the q as 1

- and partial autocorelation plot selected the p value as 1

  1.`p=1` -----> pacf_plot

  2. `q=1` ----> acf_plot

Arima model

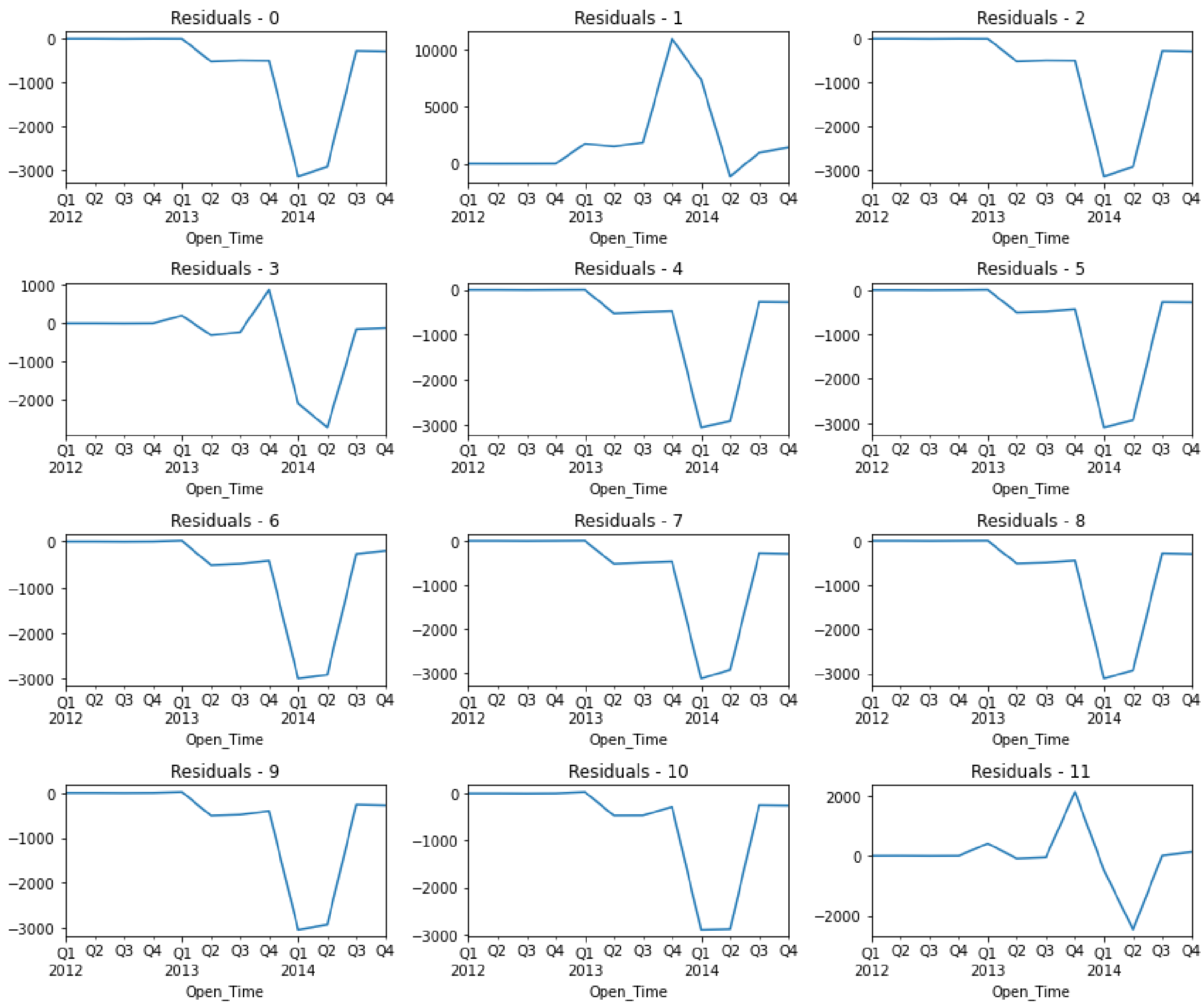
Arima model forecasts and forecast plots

```
#  Perform the forecasting for each column
arima_forecast = {}
steps = 12
for column in quaterly_data.columns:
    model = ARIMA(quaterly_data[column], order=(1, 2, 1))  # ARIMA(1,
0, 0) model
    model_fit = model.fit()
    forecast = model_fit.forecast(steps=steps)
    arima_forecast[column] = forecast
```

Residual anlysys for arima model evaluation

After fitting the ARIMA model to the training data, you can analyze the residuals (differences between the actual values and the model's predictions). Check whether the residuals have constant variance, are normally distributed, and show no significant autocorrelation.

```
plt.figure(figsize=(12, 10))
pl_no = 1

for column, results in arima_forecast.items():
    residuals = quaterly_data[column] - model_fit.fittedvalues
    plt.subplot(4, 3, pl_no)
    residuals.plot()
    plt.title(f'Residuals - {column}')
    pl_no += 1

plt.tight_layout()
plt.show()
```




```
arima_forecast_data=pd.DataFrame(arima_forecast)

arima_forecast_data=arima_forecast_data.astype(int)
arima_forecast_data
```

	0	1	2	3	4	5	6	7	8	9	10	11
2015-03-31	0	1876	0	195	21	18	104	7	3	20	40	466
2015-06-30	0	2013	0	211	23	20	113	8	3	22	43	503
2015-09-30	0	2147	0	226	24	21	122	8	3	24	47	538
2015-12-31	0	2282	0	242	26	23	130	9	4	25	50	574
2016-03-31	0	2416	0	257	28	25	139	10	4	27	53	609
2016-06-30	0	2550	0	273	30	26	148	10	4	29	57	645
2016-09-30	0	2685	0	289	31	28	156	11	5	30	60	680
2016-12-31	0	2819	0	304	33	29	165	12	5	32	63	716
2017-03-31	0	2954	0	320	35	31	174	12	5	34	67	752
2017-06-30	0	3088	0	336	37	32	183	13	5	36	70	787
2017-09-30	0	3223	0	351	39	34	191	13	6	37	73	823
2017-12-31	0	3357	0	367	40	36	200	14	6	39	77	858

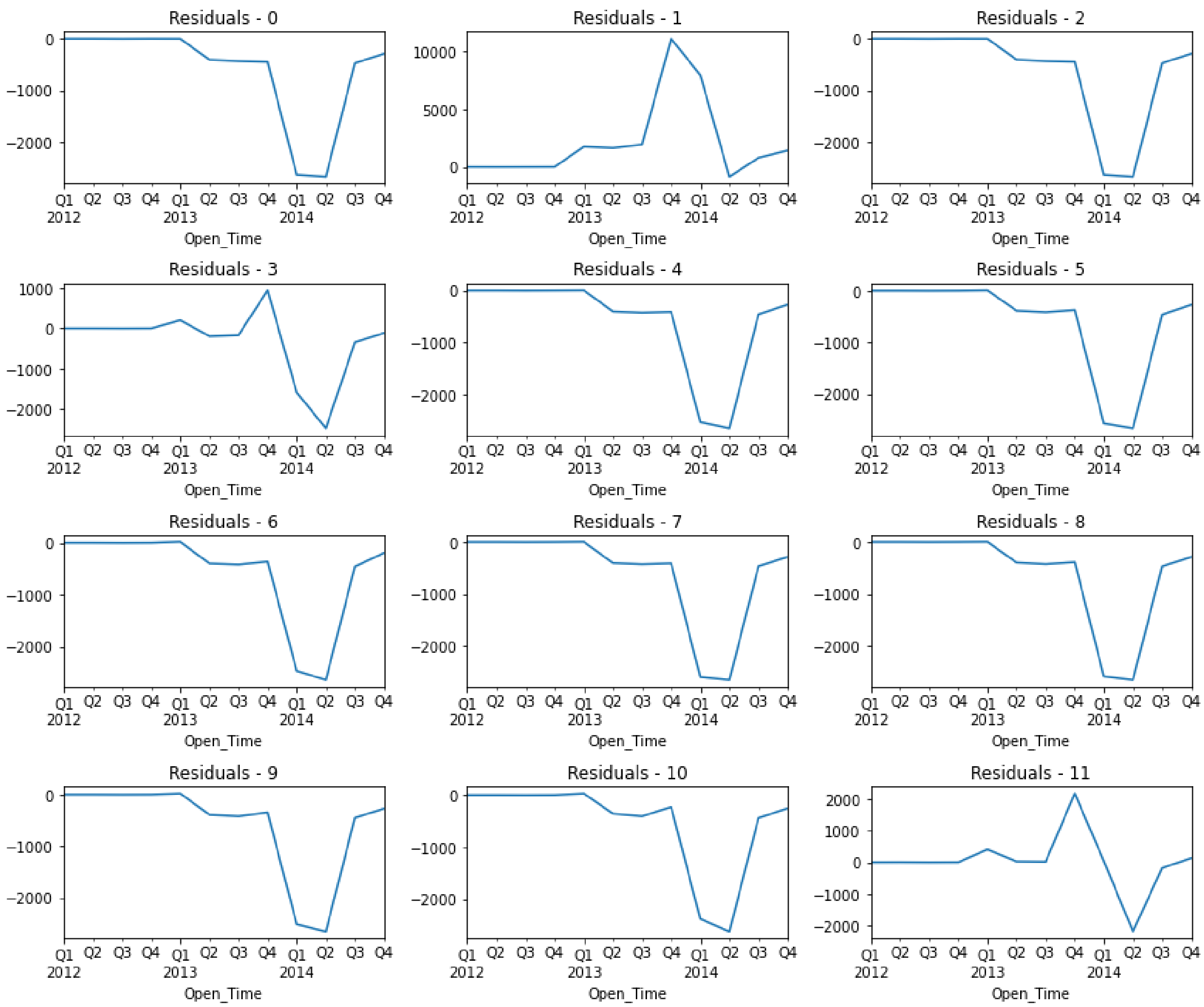
Sarimax model

```
columns_to_forecast = quaterly_data.columns

# Perform the forecasting for each column
sarima_forecast = {}
for column in columns_to_forecast:
    model = SARIMAX(quaterly_data[column], order=(1, 1, 1),
seasonal_order=(1, 1, 1, 12))  # SARIMAX(1, 0, 0)(1, 0, 0, 12) model
    model_fit = model.fit()
    forecast = model_fit.forecast(steps=12)  # Forecast for the next
12 months
    sarima_forecast[column] = forecast
```

Residual anlysys for sarima model evaluation

After fitting the SARIMA model to the training data, you can analyze the residuals (differences between the actual values and the model's predictions). Check whether the residuals have constant variance, are normally distributed, and show no significant autocorrelation.

```
plt.figure(figsize=(12, 10))
pl_no = 1

for column, results in sarima_forecast.items():
    residuals = quaterly_data[column] - model_fit.fittedvalues
    plt.subplot(4, 3, pl_no)
    residuals.plot()
    plt.title(f'Residuals - {column}')
    pl_no += 1

plt.tight_layout()
plt.show()
```




```
sarima_forecast_data=pd.DataFrame(sarima_forecast)

sarima_forecast_data=sarima_forecast_data.astype(int)
sarima_forecast_data
```

	0	1	2	3	4	5	6	7	8	9	10	11
2015-03-31	0	1719	0	177	20	17	98	7	3	19	37	422
2015-06-30	0	1715	0	177	20	17	98	7	3	19	37	424
2015-09-30	0	1721	0	177	20	17	98	7	3	19	37	422
2015-12-31	0	1723	0	177	20	17	98	7	3	19	37	423
2016-03-31	0	3466	0	383	24	25	119	13	8	40	68	830
2016-06-30	0	3759	2	392	24	37	110	12	17	36	88	855
2016-09-30	0	4078	0	444	27	34	114	14	12	32	64	871
2016-12-31	1	13211	3	1569	58	92	188	47	66	121	257	3055
2017-03-31	1	12253	0	1229	124	78	260	34	45	141	292	3093
2017-06-30	0	3543	0	379	44	21	124	15	13	35	87	894
2017-09-30	0	2970	0	309	33	26	115	14	8	41	69	713
2017-12-31	0	3432	0	354	40	34	196	14	6	38	74	844

About forecast models

- created 2 forecasting models for predicting the volumns quaterly and annualy
- out of `arima_model` and `sarima_model` ,sarima model performing very well in forecasting and i plotted the results above.
- also evaluated the models based on their residuals, comparing to the other p,d,q combinations (1,2,1) is giving good result

TASK 3

1. Auto tag the tickets with right priorities and right departments so that reassigning and related delay can be reduced

```
data_3=df.copy()

data_3.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
3       1         57   88       0       4        4         4
0.642927
4       1         57   88       0       4        4         4
0.345258

   Category  KB_number  No_of_Reassignments           Open_Time  \
0         1        553                   26 2012-05-02 13:32:00
1         1        611                   33 2012-12-03 15:44:00
2         3        339                    3 2012-03-29 12:36:00
3         1        611                   13 2012-07-17 11:49:00
4         1        611                    2 2012-10-08 11:01:00

        Resolved_Time          Close_Time  No_of_Related_Interactions
\
0 2013-04-11 13:50:00 2013-04-11 13:51:00                           1

1 2013-02-12 12:36:00 2013-02-12 12:36:00                           1

2 2014-01-13 15:12:00 2014-01-13 15:13:00                           1

3 2013-11-14 09:31:00 2013-11-14 09:31:00                           1
```

```
4 2013-08-11 13:55:00 2013-08-11 13:55:00                           1


   Handle_Time_hrs_conv
0           8256.316667
1           1700.866667
2          15722.616667
3          11637.700000
4           7370.900000
```

```
data_3=data_3.drop(['Open_Time','Resolved_Time','Close_Time'],axis=1)
```

```
X1=data_3.drop(['Priority','CI_Cat','Urgency'],axis=1)
```

```
X1.head()
```

```
   CI_Subcat  WBS  Status  Impact  number_cnt  Category  KB_number  \
0         57  162       0       4    0.601292         1        553
1         57   88       0       3    0.415050         1        611
2         10   92       0       4    0.517551         3        339
3         57   88       0       4    0.642927         1        611
4         57   88       0       4    0.345258         1        611

   No_of_Reassignments  No_of_Related_Interactions
Handle_Time_hrs_conv
0                   26                           1
8256.316667
1                   33                           1
1700.866667
2                    3                           1
15722.616667
3                   13                           1
11637.700000
4                    2                           1
7370.900000
```

```
y1=data_3['Priority']
```

```
y1.head()
```

```
0    4
1    3
2    4
3    4
4    4
Name: Priority, dtype: int32
```

```
y2=data_3['CI_Cat']
```

Function for model selection Task 3

Logic behind the function

1. first creating a dictionary with the name model_summary and initiating with null values with proper keys
2. function called `model_selection` will take model as parameter 3.initially the model will be `initiated` within the function and will be `stored in the variable called model`
3. model will be `fitted on x_train and y_train` 5.model will `first predict on test data` 6.after prediction all the evaluation `metric values will be appended to dictionary` with corresponding key values. 7.then it will `print the confusion matrix and classification report` of that model 8.the same `steps will also the performed` on train data ---

```
model_summary_1={'model_name_train':[],'f1_score_train':
[],'recall_score_train':[],'accuracy_score_train':[],
               'model_name_test':[],'f1_score_test':
[],'recall_score_test':[],'accuracy_score_test':[]}


def model_selction_2(model):

    #model initialization ,fitting and predicting
    print(model)
    model=model()
    model.fit(X_train,y_train)
    model_pred=model.predict(X_test)

    #appending the metrics to the dictionary created

model_summary_1['model_name_test'].append(model.__class__.__name__)

model_summary_1['f1_score_test'].append(f1_score(y_test,model_pred,ave
rage='macro'))

model_summary_1['recall_score_test'].append(recall_score(y_test,model_
pred,average='macro'))

model_summary_1['accuracy_score_test'].append(accuracy_score(y_test,mo
del_pred))

    #printing the confusion metrics and classification report
    print('metrics on test data')
```

```
    print(confusion_matrix(y_test,model_pred))
    print('\n')
    print(classification_report(y_test,model_pred))

    #predictions on train data
    model_pred1=model.predict(X_train)

    #appending the metrics to the dictionary created

model_summary_1['model_name_train'].append(model.__class__.__name__)

model_summary_1['f1_score_train'].append(f1_score(y_train,model_pred1,
average='macro'))

model_summary_1['recall_score_train'].append(recall_score(y_train,mode
l_pred1,average='macro'))

model_summary_1['accuracy_score_train'].append(accuracy_score(y_train,
model_pred1))

    #printing the confusion metrics and classification report
    print('metrics on train data')
    print(confusion_matrix(y_train,model_pred1))
    print('\n')
    print(classification_report(y_train,model_pred1))
    print('==='*10)

X_train, X_test, y_train, y_test = train_test_split(X1, y1,
test_size=0.3, random_state=42,stratify=y1)

for i in models:
    model_selction_2(i)
```

```
<class 'sklearn.linear_model._logistic.LogisticRegression'>
metrics on test data
[[   0    0    0    1    0]
 [   0    0    0   61  148]
 [   0    0    0 1242  355]
 [   0    0    7 6006 1216]
 [   0    0    0 4133  813]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00       209
           3       0.00      0.00      0.00      1597
           4       0.52      0.83      0.64      7229
           5       0.32      0.16      0.22      4946

    accuracy                           0.49     13982
```

```
   macro avg       0.17      0.20      0.17     13982
weighted avg       0.38      0.49      0.41     13982

metrics on train data
[[    0     0     0     1     1]
 [    0     0     0   140   348]
 [    0     0     0  2910   816]
 [    0     0    27 14046  2795]
 [    0     0     0  9737  1802]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         2
           2       0.00      0.00      0.00       488
           3       0.00      0.00      0.00      3726
           4       0.52      0.83      0.64     16868
           5       0.31      0.16      0.21     11539

    accuracy                           0.49     32623
   macro avg       0.17      0.20      0.17     32623
weighted avg       0.38      0.49      0.41     32623

==============================
<class 'sklearn.tree._classes.DecisionTreeClassifier'>
metrics on test data
[[   1    0    0    0    0]
 [   0  207    2    0    0]
 [   0    4 1566   25    2]
 [   0    0   18 7193   18]
 [   0    0    0   18 4928]]


              precision    recall  f1-score   support

           1       1.00      1.00      1.00         1
           2       0.98      0.99      0.99       209
           3       0.99      0.98      0.98      1597
           4       0.99      1.00      0.99      7229
           5       1.00      1.00      1.00      4946

    accuracy                           0.99     13982
   macro avg       0.99      0.99      0.99     13982
weighted avg       0.99      0.99      0.99     13982

metrics on train data
[[    2     0     0     0     0]
 [    0   488     0     0     0]
 [    0     0  3726     0     0]
 [    0     0     0 16868     0]
```

```
 [    0     0     0     0 11539]]


              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2
           2       1.00      1.00      1.00       488
           3       1.00      1.00      1.00      3726
           4       1.00      1.00      1.00     16868
           5       1.00      1.00      1.00     11539

    accuracy                           1.00     32623
   macro avg       1.00      1.00      1.00     32623
weighted avg       1.00      1.00      1.00     32623

==============================
<class 'sklearn.ensemble._forest.RandomForestClassifier'>
metrics on test data
[[   0    1    0    0    0]
 [   0  207    2    0    0]
 [   0    0 1565   30    2]
 [   0    0    4 7201   24]
 [   0    0    0    0 4946]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       1.00      0.99      0.99       209
           3       1.00      0.98      0.99      1597
           4       1.00      1.00      1.00      7229
           5       0.99      1.00      1.00      4946

    accuracy                           1.00     13982
   macro avg       0.80      0.79      0.79     13982
weighted avg       1.00      1.00      1.00     13982

metrics on train data
[[    2     0     0     0     0]
 [    0   488     0     0     0]
 [    0     0  3726     0     0]
 [    0     0     0 16867     1]
 [    0     0     0     0 11539]]


              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2
           2       1.00      1.00      1.00       488
           3       1.00      1.00      1.00      3726
```

```
           4       1.00      1.00      1.00     16868
           5       1.00      1.00      1.00     11539

    accuracy                           1.00     32623
   macro avg       1.00      1.00      1.00     32623
weighted avg       1.00      1.00      1.00     32623

==============================
<class 'sklearn.ensemble._bagging.BaggingClassifier'>
metrics on test data
[[   1    0    0    0    0]
 [   0  207    2    0    0]
 [   0    0 1573   22    2]
 [   0    0   12 7199   18]
 [   0    0    0    3 4943]]


              precision    recall  f1-score   support

           1       1.00      1.00      1.00         1
           2       1.00      0.99      1.00       209
           3       0.99      0.98      0.99      1597
           4       1.00      1.00      1.00      7229
           5       1.00      1.00      1.00      4946

    accuracy                           1.00     13982
   macro avg       1.00      0.99      1.00     13982
weighted avg       1.00      1.00      1.00     13982

metrics on train data
[[    2     0     0     0     0]
 [    0   486     2     0     0]
 [    0     0  3724     2     0]
 [    0     0     8 16855     5]
 [    0     0     0     1 11538]]


              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2
           2       1.00      1.00      1.00       488
           3       1.00      1.00      1.00      3726
           4       1.00      1.00      1.00     16868
           5       1.00      1.00      1.00     11539

    accuracy                           1.00     32623
   macro avg       1.00      1.00      1.00     32623
weighted avg       1.00      1.00      1.00     32623

==============================
```

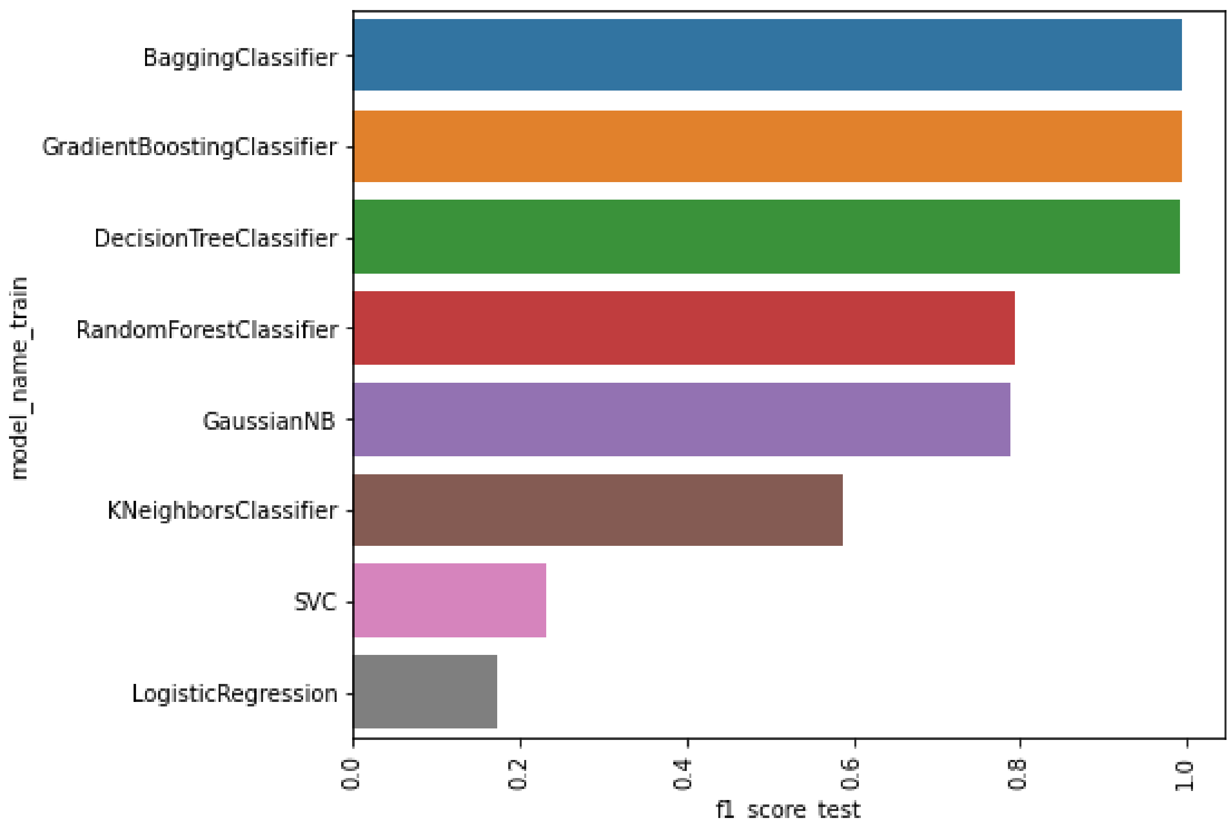
```
<class 'sklearn.neighbors._classification.KNeighborsClassifier'>
metrics on test data
[[   0    0    0    1    0]
 [   0  133   31   41    4]
 [   0   31 1010  417  139]
 [   0   38  290 6152  749]
 [   0    5  101  937 3903]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.64      0.64      0.64       209
           3       0.71      0.63      0.67      1597
           4       0.82      0.85      0.83      7229
           5       0.81      0.79      0.80      4946

    accuracy                           0.80     13982
   macro avg       0.60      0.58      0.59     13982
weighted avg       0.80      0.80      0.80     13982

metrics on train data
[[    0     0     0     1     1]
 [    0   354    63    55    16]
 [    0    59  2751   700   216]
 [    0    44   459 15273  1092]
 [    0     8   173  1474  9884]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         2
           2       0.76      0.73      0.74       488
           3       0.80      0.74      0.77      3726
           4       0.87      0.91      0.89     16868
           5       0.88      0.86      0.87     11539

    accuracy                           0.87     32623
   macro avg       0.66      0.65      0.65     32623
weighted avg       0.87      0.87      0.87     32623

==============================
<class 'sklearn.naive_bayes.GaussianNB'>
metrics on test data
[[   0    1    0    0    0]
 [   0  203    5    1    0]
 [   0    5 1558   32    2]
 [   0    0    3 7152   74]
 [   0    0    0    3 4943]]
```

```
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.97      0.97      0.97       209
           3       0.99      0.98      0.99      1597
           4       0.99      0.99      0.99      7229
           5       0.98      1.00      0.99      4946

    accuracy                           0.99     13982
   macro avg       0.79      0.79      0.79     13982
weighted avg       0.99      0.99      0.99     13982

metrics on train data
[[    2     0     0     0     0]
 [    0   480     8     0     0]
 [    0     6  3652    68     0]
 [    0     3     5 16681   179]
 [    0     0     0     6 11533]]

              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2
           2       0.98      0.98      0.98       488
           3       1.00      0.98      0.99      3726
           4       1.00      0.99      0.99     16868
           5       0.98      1.00      0.99     11539

    accuracy                           0.99     32623
   macro avg       0.99      0.99      0.99     32623
weighted avg       0.99      0.99      0.99     32623

==============================
<class 'sklearn.svm._classes.SVC'>

metrics on test data
[[   0    0    0    1    0]
 [   0    0    0   93  116]
 [   0    0    0 1228  369]
 [   0    0    0 6213 1016]
 [   0    0    0 3040 1906]]

              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.00      0.00      0.00       209
           3       0.00      0.00      0.00      1597
           4       0.59      0.86      0.70      7229
           5       0.56      0.39      0.46      4946
```

```
    accuracy                           0.58     13982
   macro avg       0.23      0.25      0.23     13982
weighted avg       0.50      0.58      0.52     13982

metrics on train data
[[    0     0     0     1     1]
 [    0     0     0   252   236]
 [    0     0     0  2875   851]
 [    0     0     0 14542  2326]
 [    0     0     0  7158  4381]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         2
           2       0.00      0.00      0.00       488
           3       0.00      0.00      0.00      3726
           4       0.59      0.86      0.70     16868
           5       0.56      0.38      0.45     11539

    accuracy                           0.58     32623
   macro avg       0.23      0.25      0.23     32623
weighted avg       0.50      0.58      0.52     32623

==============================
<class 'sklearn.ensemble._gb.GradientBoostingClassifier'>
metrics on test data
[[   1    0    0    0    0]
 [   0  207    2    0    0]
 [   0    0 1570   25    2]
 [   0    0   11 7191   27]
 [   0    0    0    4 4942]]


              precision    recall  f1-score   support

           1       1.00      1.00      1.00         1
           2       1.00      0.99      1.00       209
           3       0.99      0.98      0.99      1597
           4       1.00      0.99      1.00      7229
           5       0.99      1.00      1.00      4946

    accuracy                           0.99     13982
   macro avg       1.00      0.99      0.99     13982
weighted avg       0.99      0.99      0.99     13982

metrics on train data
[[    2     0     0     0     0]
 [    0   487     1     0     0]
```

```
 [    0     0  3702    24     0]
 [    0     0     5 16821    42]
 [    0     0     1     0 11538]]


              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2
           2       1.00      1.00      1.00       488
           3       1.00      0.99      1.00      3726
           4       1.00      1.00      1.00     16868
           5       1.00      1.00      1.00     11539

    accuracy                           1.00     32623
   macro avg       1.00      1.00      1.00     32623
weighted avg       1.00      1.00      1.00     32623

==============================

summary_1=pd.DataFrame(model_summary_1).sort_values('f1_score_test',as
cending=False).drop('model_name_test',axis=1)

summary_1

             model_name_train  f1_score_train  recall_score_train  \
3           BaggingClassifier        0.999121            0.998902
7  GradientBoostingClassifier        0.998168            0.997727
1      DecisionTreeClassifier        1.000000            1.000000
2      RandomForestClassifier        0.999985            0.999988
5                  GaussianNB        0.991022            0.990428
4        KNeighborsClassifier        0.653556            0.645150
6                         SVC        0.230143            0.248355
0          LogisticRegression        0.170224            0.197773

   accuracy_score_train  f1_score_test  recall_score_test
accuracy_score_test
3              0.999448       0.995426           0.994129
0.995780
7              0.997762       0.994930           0.993492
0.994922
1              1.000000       0.992078           0.992480
0.993778
2              0.999969       0.794836           0.793304
0.995494
5              0.991570       0.788133           0.787123
0.990988
4              0.866321       0.588062           0.581788
0.800887
6              0.580051       0.230859           0.248963
0.580675
```

```
0                    0.485792        0.172151            0.199039
0.487698

plt.figure(figsize=(7,6))
sns.barplot(y=summary_1['model_name_train'],x=summary_1['f1_score_test
'])
plt.xticks(rotation=90)
plt.show()
```



Model selection for task 3 - to tag priority

- from the above graph it is found that the `DecissionTreeClassifier`,`bagging_classifier`,`gradiant boosting` performing well compared to other algorithms
- and it is performing well above 95 percentage so not using `optimization techniques` separatly
- im considering the `bagging_classifier`, `gradiant boosting` model over `DecisionTreeClassifier` as it performing better in more number of times compared to DecisionTree classifer
- will create the `GradientBoostingClassifier` model for further use

```python
#model creation
#model initialization
all_priority_model=GradientBoostingClassifier()

#fitting the model
all_priority_model.fit(X_train,y_train)

#predicting using the model
all_priority_pred=all_priority_model.predict(X_test)

#printing the confusion metrics and classification report
print('metrics on test data')
print('confusion matrix')
print(confusion_matrix(y_test,all_priority_pred))
print('\n')
print('classification report')
print(classification_report(y_test,all_priority_pred))
print('==='*10)
```

```
metrics on test data
confusion matrix
[[   1    0    0    0    0]
 [   0  207    2    0    0]
 [   0    0 1570   25    2]
 [   0    0   11 7191   27]
 [   0    0    0    4 4942]]


classification report
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         1
           2       1.00      0.99      1.00       209
           3       0.99      0.98      0.99      1597
           4       1.00      0.99      1.00      7229
           5       0.99      1.00      1.00      4946

    accuracy                           0.99     13982
   macro avg       1.00      0.99      0.99     13982
weighted avg       0.99      0.99      0.99     13982

==============================
```

The above is for the priority and next we'll build a model for segregation of those tickets based on the respective departments

Logic behind the function

1. first creating a dictionary with the name model_summary and initiating with null values with proper keys

2. function called `model_selection` will take model as parameter 3.initially the model will be `initiated` within the function and will be `stored in the variable called model`

3. model will be `fitted on x_train and y_train` 5.model will `first predict on test data` 6.after prediction all the evaluation `metric values will be appended to dictionary` with corresponding key values. 7.then it will `print the confusion matrix and classification report` of that model 8.the same `steps will also the performed` on train data ---

```
model_summary_3={'model_name_train':[],'f1_score_train':
[],'recall_score_train':[],'accuracy_score_train':[],
               'model_name_test':[],'f1_score_test':
[],'recall_score_test':[],'accuracy_score_test':[]}


def model_selction_3(model):

    #model initialization ,fitting and predicting
    print(model)
    model=model()
    model.fit(X_train,y_train)
    model_pred=model.predict(X_test)

    #appending the metrics to the dictionary created

model_summary_3['model_name_test'].append(model.__class__.__name__)

model_summary_3['f1_score_test'].append(f1_score(y_test,model_pred,ave
rage='macro'))

model_summary_3['recall_score_test'].append(recall_score(y_test,model_
pred,average='macro'))

model_summary_3['accuracy_score_test'].append(accuracy_score(y_test,mo
del_pred))
```

```
    #printing the confusion metrics and classification report
    print('metrics on test data')
    print(confusion_matrix(y_test,model_pred))
    print('\n')
    print(classification_report(y_test,model_pred))

    #predictions on train data
    model_pred1=model.predict(X_train)

    #appending the metrics to the dictionary created

model_summary_3['model_name_train'].append(model.__class__.__name__)

model_summary_3['f1_score_train'].append(f1_score(y_train,model_pred1,
average='macro'))

model_summary_3['recall_score_train'].append(recall_score(y_train,mode
l_pred1,average='macro'))

model_summary_3['accuracy_score_train'].append(accuracy_score(y_train,
model_pred1))

    #printing the confusion metrics and classification report
    print('metrics on train data')
    print(confusion_matrix(y_train,model_pred1))
    print('\n')
    print(classification_report(y_train,model_pred1))
    print('==='*10)

X_train, X_test, y_train, y_test = train_test_split(X1, y2,
test_size=0.3, random_state=42,stratify=y2)

for i in models:
    model_selction_3(i)
```

```
<class 'sklearn.linear_model._logistic.LogisticRegression'>
metrics on test data
[[9192    0   49    0    0   11    0    0    0    0  651]
 [   1    0    0    0    0    0    0    0    0    0    0]
 [1053    0   40    0    0    0    0    0    0    0    0]
 [  61    0    3    0    0    0    0    0    0    0    0]
 [  64    0    0    0    0    0    0    0    0    0    0]
 [ 106    0   25    0    0    2    0    0    0    0    0]
 [  32    0    0    0    0    0    0    0    0    0    0]
 [  46    0    0    0    0    0    0    0    0    0    0]
 [  91    0    9    0    0    0    0    0    0    0    0]
 [ 210    0    1    0    0    0    0    0    0    0    0]
 [2262    0   12    0    0    0    0    0    0    0   61]]


              precision    recall  f1-score   support
```

```
           1       0.70      0.93      0.80      9903
           2       0.00      0.00      0.00         1
           3       0.29      0.04      0.06      1093
           4       0.00      0.00      0.00        64
           5       0.00      0.00      0.00        64
           6       0.15      0.02      0.03       133
           7       0.00      0.00      0.00        32
           8       0.00      0.00      0.00        46
           9       0.00      0.00      0.00       100
          10       0.00      0.00      0.00       211
          11       0.09      0.03      0.04      2335

    accuracy                           0.66     13982
   macro avg       0.11      0.09      0.08     13982
weighted avg       0.53      0.66      0.58     13982

metrics on train data
[[    0     2     0     0     0     0     0     0     0     0     0
0]
 [    0 21436     0   102     0     0    28     0     0     0     0
1541]
 [    0     3     0     1     0     0     0     0     0     0     0
0]
 [    0  2467     0    82     0     0     1     0     0     0     0
0]
 [    0   148     0     2     0     0     0     0     0     0     0
0]
 [    0   148     0     0     0     0     0     0     0     0     0
0]
 [    0   254     0    50     0     0     5     0     0     0     0
0]
 [    0    75     0     0     0     0     0     0     0     0     0
0]
 [    0   106     0     0     0     0     0     0     0     0     0
0]
 [    0   223     0    10     0     0     0     0     0     0     0
0]
 [    0   491     0     1     0     0     0     0     0     0     0
0]
 [    0  5272     0    23     0     0     0     0     0     0     0
152]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.70      0.93      0.80     23107
           2       0.00      0.00      0.00         4
           3       0.30      0.03      0.06      2550
```

```
           4       0.00      0.00      0.00       150
           5       0.00      0.00      0.00       148
           6       0.15      0.02      0.03       309
           7       0.00      0.00      0.00        75
           8       0.00      0.00      0.00       106
           9       0.00      0.00      0.00       233
          10       0.00      0.00      0.00       492
          11       0.09      0.03      0.04      5447

    accuracy                           0.66     32623
   macro avg       0.10      0.08      0.08     32623
weighted avg       0.54      0.66      0.58     32623

==============================
<class 'sklearn.tree._classes.DecisionTreeClassifier'>
metrics on test data
[[   0    0    0    0    0    0    0    0    0    0    0    0]
 [   0 9824    0    0    0    0    0    0    0    0    0   79]
 [   0    0    1    0    0    0    0    0    0    0    0    0]
 [   1    0    0 1091    0    0    0    1    0    0    0    0]
 [   0    0    0    0   63    0    0    1    0    0    0    0]
 [   0    0    0    0    0   64    0    0    0    0    0    0]
 [   0    0    0    1    0    0  132    0    0    0    0    0]
 [   0    0    0    1    1    0    1   27    0    0    2    0]
 [   0    0    0    0    0    0    0    0   46    0    0    0]
 [   0    0    0    0    0    0    0    0    0  100    0    0]
 [   0    1    0    0    0    0    0    1    0    0  209    0]
 [   0   49    0    0    0    0    0    0    0    0    0 2286]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.99      0.99      0.99      9903
           2       1.00      1.00      1.00         1
           3       1.00      1.00      1.00      1093
           4       0.98      0.98      0.98        64
           5       1.00      1.00      1.00        64
           6       0.99      0.99      0.99       133
           7       0.90      0.84      0.87        32
           8       1.00      1.00      1.00        46
           9       1.00      1.00      1.00       100
          10       0.99      0.99      0.99       211
          11       0.97      0.98      0.97      2335

    accuracy                           0.99     13982
   macro avg       0.90      0.90      0.90     13982
weighted avg       0.99      0.99      0.99     13982

metrics on train data
```

```
[[    2     0     0     0     0     0     0     0     0     0     0
0]
 [    0 23107     0     0     0     0     0     0     0     0     0
0]
 [    0     0     4     0     0     0     0     0     0     0     0
0]
 [    0     0     0  2550     0     0     0     0     0     0     0
0]
 [    0     0     0     0   150     0     0     0     0     0     0
0]
 [    0     0     0     0     0   148     0     0     0     0     0
0]
 [    0     0     0     0     0     0   309     0     0     0     0
0]
 [    0     0     0     0     0     0     0    75     0     0     0
0]
 [    0     0     0     0     0     0     0     0   106     0     0
0]
 [    0     0     0     0     0     0     0     0     0   233     0
0]
 [    0     0     0     0     0     0     0     0     0     0   492
0]
 [    0     0     0     0     0     0     0     0     0     0     0
5447]]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00     23107
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00      2550
           4       1.00      1.00      1.00       150
           5       1.00      1.00      1.00       148
           6       1.00      1.00      1.00       309
           7       1.00      1.00      1.00        75
           8       1.00      1.00      1.00       106
           9       1.00      1.00      1.00       233
          10       1.00      1.00      1.00       492
          11       1.00      1.00      1.00      5447

    accuracy                           1.00     32623
   macro avg       1.00      1.00      1.00     32623
weighted avg       1.00      1.00      1.00     32623

==============================
<class 'sklearn.ensemble._forest.RandomForestClassifier'>
metrics on test data
[[9822    0    0    0    0    0    0    0    0    0   81]
 [   0    0    1    0    0    0    0    0    0    0    0]
```

```
 [   7    0 1086    0    0    0    0    0    0    0    0]
 [   0    0    1   63    0    0    0    0    0    0    0]
 [   0    0    0    0   64    0    0    0    0    0    0]
 [   1    0    3    0    0  129    0    0    0    0    0]
 [   4    0    3    0    1    2   21    0    1    0    0]
 [   0    0    0    0    0    0    0   46    0    0    0]
 [   1    0    0    0    0    0    0    0   99    0    0]
 [   1    0    1    0    0    2    0    0    0  207    0]
 [ 181    0    0    0    0    0    0    0    0    0 2154]]


              precision    recall  f1-score   support

           1       0.98      0.99      0.99      9903
           2       0.00      0.00      0.00         1
           3       0.99      0.99      0.99      1093
           4       1.00      0.98      0.99        64
           5       0.98      1.00      0.99        64
           6       0.97      0.97      0.97       133
           7       1.00      0.66      0.79        32
           8       1.00      1.00      1.00        46
           9       0.99      0.99      0.99       100
          10       1.00      0.98      0.99       211
          11       0.96      0.92      0.94      2335

    accuracy                           0.98     13982
   macro avg       0.90      0.86      0.88     13982
weighted avg       0.98      0.98      0.98     13982


metrics on train data
[[    2     0     0     0     0     0     0     0     0     0     0
0]
 [    0 23107     0     0     0     0     0     0     0     0     0
0]
 [    0     0     4     0     0     0     0     0     0     0     0
0]
 [    0     0     0  2550     0     0     0     0     0     0     0
0]
 [    0     0     0     0   150     0     0     0     0     0     0
0]
 [    0     0     0     0     0   148     0     0     0     0     0
0]
 [    0     0     0     0     0     0   309     0     0     0     0
0]
 [    0     0     0     0     0     0     0    75     0     0     0
0]
 [    0     0     0     0     0     0     0     0   106     0     0
0]
 [    0     0     0     0     0     0     0     0     0   233     0
```

```
0]
 [    0     0     0     0     0     0     0     0     0     0   492
0]
 [    0     0     0     0     0     0     0     0     0     0     0
5447]]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00     23107
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00      2550
           4       1.00      1.00      1.00       150
           5       1.00      1.00      1.00       148
           6       1.00      1.00      1.00       309
           7       1.00      1.00      1.00        75
           8       1.00      1.00      1.00       106
           9       1.00      1.00      1.00       233
          10       1.00      1.00      1.00       492
          11       1.00      1.00      1.00      5447

    accuracy                           1.00     32623
   macro avg       1.00      1.00      1.00     32623
weighted avg       1.00      1.00      1.00     32623

==============================
<class 'sklearn.ensemble._bagging.BaggingClassifier'>
metrics on test data
[[   0    0    0    0    0    0    0    0    0    0    0    0]
 [   0 9856    0    0    0    0    0    0    0    0    0   47]
 [   0    0    1    0    0    0    0    0    0    0    0    0]
 [   1    0    0 1092    0    0    0    0    0    0    0    0]
 [   0    0    0    0   63    0    0    1    0    0    0    0]
 [   0    0    0    0    0   64    0    0    0    0    0    0]
 [   0    0    0    1    0    0  132    0    0    0    0    0]
 [   0    1    0    0    0    1    2   26    0    0    2    0]
 [   0    0    0    0    0    0    0    0   46    0    0    0]
 [   0    0    0    0    0    0    0    0    0  100    0    0]
 [   0    1    0    0    0    0    0    1    0    0  209    0]
 [   0   58    0    0    0    0    0    0    0    0    0 2277]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.99      1.00      0.99      9903
           2       1.00      1.00      1.00         1
           3       1.00      1.00      1.00      1093
           4       1.00      0.98      0.99        64
```

```
           5       0.98      1.00      0.99        64
           6       0.99      0.99      0.99       133
           7       0.93      0.81      0.87        32
           8       1.00      1.00      1.00        46
           9       1.00      1.00      1.00       100
          10       0.99      0.99      0.99       211
          11       0.98      0.98      0.98      2335

    accuracy                           0.99     13982
   macro avg       0.91      0.90      0.90     13982
weighted avg       0.99      0.99      0.99     13982

metrics on train data
[[    2     0     0     0     0     0     0     0     0     0     0
0]
 [    0 23098     0     0     0     0     0     0     0     0     0
9]
 [    0     0     4     0     0     0     0     0     0     0     0
0]
 [    0     0     0  2550     0     0     0     0     0     0     0
0]
 [    0     0     0     0   150     0     0     0     0     0     0
0]
 [    0     0     0     0     0   148     0     0     0     0     0
0]
 [    0     0     0     0     0     0   309     0     0     0     0
0]
 [    0     1     0     1     0     1     0    72     0     0     0
0]
 [    0     0     0     0     0     0     0     0   106     0     0
0]
 [    0     0     0     0     0     0     0     0     1   232     0
0]
 [    0     0     0     0     0     0     0     0     0     0   492
0]
 [    0    15     0     0     0     0     0     0     0     0     0
5432]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00     23107
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00      2550
           4       1.00      1.00      1.00       150
           5       0.99      1.00      1.00       148
           6       1.00      1.00      1.00       309
           7       1.00      0.96      0.98        75
           8       0.99      1.00      1.00       106
```

```
           9       1.00      1.00      1.00       233
          10       1.00      1.00      1.00       492
          11       1.00      1.00      1.00      5447

    accuracy                           1.00     32623
   macro avg       1.00      1.00      1.00     32623
weighted avg       1.00      1.00      1.00     32623

==============================
<class 'sklearn.neighbors._classification.KNeighborsClassifier'>
metrics on test data
[[9498    0  108    2    7   17    2    8    7   15  239]
 [   0    0    1    0    0    0    0    0    0    0    0]
 [ 155    0  912    0   14    2    0    0    1    3    6]
 [   9    0    5   50    0    0    0    0    0    0    0]
 [  19    0    7    0   38    0    0    0    0    0    0]
 [  42    0    3    0    1   83    1    0    0    1    2]
 [  19    0    0    0    0    3    9    0    1    0    0]
 [   9    0    0    0    0    0    0   37    0    0    0]
 [  29    0    0    0    0    0    2    0   67    0    2]
 [  42    0    4    0    2    0    0    1    1  161    0]
 [ 361    0   29    0    2    1    0    4    2    2 1934]]


              precision    recall  f1-score   support

           1       0.93      0.96      0.95      9903
           2       0.00      0.00      0.00         1
           3       0.85      0.83      0.84      1093
           4       0.96      0.78      0.86        64
           5       0.59      0.59      0.59        64
           6       0.78      0.62      0.69       133
           7       0.64      0.28      0.39        32
           8       0.74      0.80      0.77        46
           9       0.85      0.67      0.75       100
          10       0.88      0.76      0.82       211
          11       0.89      0.83      0.86      2335

    accuracy                           0.91     13982
   macro avg       0.74      0.65      0.68     13982
weighted avg       0.91      0.91      0.91     13982

metrics on train data
[[    0     1     0     1     0     0     0     0     0     0     0
0]
 [    0 22549     0   136     2    16    27     6     7     8    21
335]
 [    0     4     0     0     0     0     0     0     0     0     0
0]
 [    0   306     0  2202     0    15     5     0     1     2     6
```

```
13]
 [    0   10    0    1  138    0    1    0    0    0    0
0]
 [    0   27    0   16    0  105    0    0    0    0    0
0]
 [    0   63    0   14    0    0  220    2    0    4    4
2]
 [    0   34    0    1    0    0    0   38    0    2    0
0]
 [    0   28    0    2    0    0    0    0   76    0    0
0]
 [    0   56    0    4    0    1    3    0    0  164    0
5]
 [    0   80    0   12    0    3    0    0    2    2  390
3]
 [    0  626    0   50    0    0    1    0    2    1    2
4765]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.95      0.98      0.96     23107
           2       0.00      0.00      0.00         4
           3       0.90      0.86      0.88      2550
           4       0.99      0.92      0.95       150
           5       0.75      0.71      0.73       148
           6       0.86      0.71      0.78       309
           7       0.83      0.51      0.63        75
           8       0.86      0.72      0.78       106
           9       0.90      0.70      0.79       233
          10       0.92      0.79      0.85       492
          11       0.93      0.87      0.90      5447

    accuracy                           0.94     32623
   macro avg       0.74      0.65      0.69     32623
weighted avg       0.94      0.94      0.94     32623

==============================
<class 'sklearn.naive_bayes.GaussianNB'>
metrics on test data
[[4645   27  983   25    0  282   17 1057   11   93 2763]
 [   0    1    0    0    0    0    0    0    0    0    0]
 [ 116    5  901    0    0    6    1   15    7    2   40]
 [   2    0    0   59    0    3    0    0    0    0    0]
 [   0    0    0    0   64    0    0    0    0    0    0]
 [  21    1   38    1    0   71    0    0    0    0    1]
 [  11    0    3    0    0    3   14    1    0    0    0]
 [   6    0    0    0    0    0    0   36    0    0    4]
 [  38    2    0    0    0   13    0    0   33    0   14]
```

```
 [  42    8   38    0    0    1    0  113    0    9    0]
 [ 490    0   32    0    0   10    0    8   10    9 1776]]


              precision    recall  f1-score   support

           1       0.86      0.47      0.61      9903
           2       0.02      1.00      0.04         1
           3       0.45      0.82      0.58      1093
           4       0.69      0.92      0.79        64
           5       1.00      1.00      1.00        64
           6       0.18      0.53      0.27       133
           7       0.44      0.44      0.44        32
           8       0.03      0.78      0.06        46
           9       0.54      0.33      0.41       100
          10       0.08      0.04      0.06       211
          11       0.39      0.76      0.51      2335

    accuracy                           0.54     13982
   macro avg       0.43      0.65      0.43     13982
weighted avg       0.73      0.54      0.58     13982


metrics on train data
[[    2     0     0     0     0     0     0     0     0     0     0
0]
 [    0 10784    65  2338    53     0   634    49  2432    34   246
6472]
 [    0     0     4     0     0     0     0     0     0     0     0
0]
 [    0   265    18  2082     0     0    13     1    45    16    15
95]
 [    0     6     0     0   140     0     4     0     0     0     0
0]
 [    0     2     0     0     0   146     0     0     0     0     0
0]
 [    0    59     1    80     3     0   163     2     0     0     1
0]
 [    0    23     1     3     0     0     3    39     5     1     0
0]
 [    0    13     0     2     0     0     0     0    82     0     0
9]
 [    0    86     6     4     0     0    30     1     1    63     1
41]
 [    0    99    16    92     0     0     4     0   270     0    11
0]
 [    0  1201     1    74     0     0    22     2    27    10    28
4082]]
```

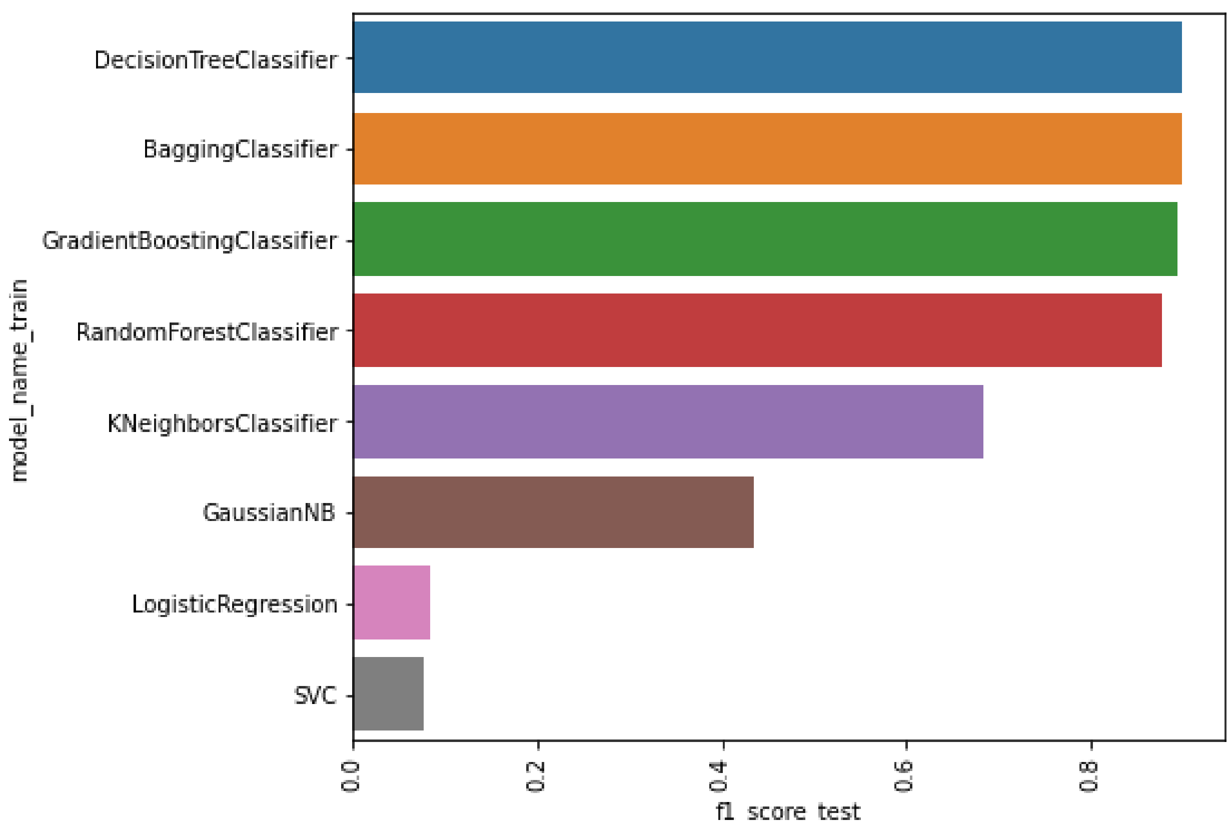
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       0.86      0.47      0.61     23107
           2       0.04      1.00      0.07         4
           3       0.45      0.82      0.58      2550
           4       0.71      0.93      0.81       150
           5       1.00      0.99      0.99       148
           6       0.19      0.53      0.28       309
           7       0.41      0.52      0.46        75
           8       0.03      0.77      0.06       106
           9       0.51      0.27      0.35       233
          10       0.04      0.02      0.03       492
          11       0.38      0.75      0.51      5447

    accuracy                           0.54     32623
   macro avg       0.47      0.67      0.48     32623
weighted avg       0.72      0.54      0.57     32623

==============================
<class 'sklearn.svm._classes.SVC'>
metrics on test data
[[9902    0    0    0    0    0    0    0    0    0    1]
 [   1    0    0    0    0    0    0    0    0    0    0]
 [1093    0    0    0    0    0    0    0    0    0    0]
 [  64    0    0    0    0    0    0    0    0    0    0]
 [  64    0    0    0    0    0    0    0    0    0    0]
 [ 133    0    0    0    0    0    0    0    0    0    0]
 [  32    0    0    0    0    0    0    0    0    0    0]
 [  46    0    0    0    0    0    0    0    0    0    0]
 [ 100    0    0    0    0    0    0    0    0    0    0]
 [ 211    0    0    0    0    0    0    0    0    0    0]
 [2335    0    0    0    0    0    0    0    0    0    0]]


              precision    recall  f1-score   support

           1       0.71      1.00      0.83      9903
           2       0.00      0.00      0.00         1
           3       0.00      0.00      0.00      1093
           4       0.00      0.00      0.00        64
           5       0.00      0.00      0.00        64
           6       0.00      0.00      0.00       133
           7       0.00      0.00      0.00        32
           8       0.00      0.00      0.00        46
           9       0.00      0.00      0.00       100
          10       0.00      0.00      0.00       211
          11       0.00      0.00      0.00      2335

    accuracy                           0.71     13982
```

```
   macro avg       0.06      0.09      0.08     13982
weighted avg       0.50      0.71      0.59     13982

metrics on train data
[[    0     2     0     0     0     0     0     0     0     0     0
0]
 [    0 23107     0     0     0     0     0     0     0     0     0
0]
 [    0     4     0     0     0     0     0     0     0     0     0
0]
 [    0  2550     0     0     0     0     0     0     0     0     0
0]
 [    0   150     0     0     0     0     0     0     0     0     0
0]
 [    0   148     0     0     0     0     0     0     0     0     0
0]
 [    0   309     0     0     0     0     0     0     0     0     0
0]
 [    0    75     0     0     0     0     0     0     0     0     0
0]
 [    0   106     0     0     0     0     0     0     0     0     0
0]
 [    0   233     0     0     0     0     0     0     0     0     0
0]
 [    0   492     0     0     0     0     0     0     0     0     0
0]
 [    0  5445     0     0     0     0     0     0     0     0     0
2]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.71      1.00      0.83     23107
           2       0.00      0.00      0.00         4
           3       0.00      0.00      0.00      2550
           4       0.00      0.00      0.00       150
           5       0.00      0.00      0.00       148
           6       0.00      0.00      0.00       309
           7       0.00      0.00      0.00        75
           8       0.00      0.00      0.00       106
           9       0.00      0.00      0.00       233
          10       0.00      0.00      0.00       492
          11       1.00      0.00      0.00      5447

    accuracy                           0.71     32623
   macro avg       0.14      0.08      0.07     32623
weighted avg       0.67      0.71      0.59     32623

==============================
```

```
<class 'sklearn.ensemble._gb.GradientBoostingClassifier'>
metrics on test data
[[   0    0    0    0    0    0    0    0    0    0    0    0]
 [   0 9824    0    0    0    0    0    0    0    0    0   79]
 [   0    0    1    0    0    0    0    0    0    0    0    0]
 [   0    2    0 1091    0    0    0    0    0    0    0    0]
 [   0    1    0    0   63    0    0    0    0    0    0    0]
 [   0    0    0    0    0   64    0    0    0    0    0    0]
 [   0    1    0    0    0    0  132    0    0    0    0    0]
 [   1    5    0    1    1    0    0   24    0    0    0    0]
 [   0    0    0    0    0    0    0    0   46    0    0    0]
 [   0    0    0    0    0    0    0    0    0  100    0    0]
 [   0    2    0    0    0    1    0    0    0    0  208    0]
 [   0  256    0    0    0    0    0    0    0    0    0 2079]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.97      0.99      0.98      9903
           2       1.00      1.00      1.00         1
           3       1.00      1.00      1.00      1093
           4       0.98      0.98      0.98        64
           5       0.98      1.00      0.99        64
           6       1.00      0.99      1.00       133
           7       1.00      0.75      0.86        32
           8       1.00      1.00      1.00        46
           9       1.00      1.00      1.00       100
          10       1.00      0.99      0.99       211
          11       0.96      0.89      0.93      2335

    accuracy                           0.97     13982
   macro avg       0.91      0.88      0.89     13982
weighted avg       0.97      0.97      0.97     13982

metrics on train data
[[    2     0     0     0     0     0     0     0     0     0     0
0]
 [    0 22907     0     0     0     0     0     0     0     0     0
200]
 [    0     0     4     0     0     0     0     0     0     0     0
0]
 [    0     5     0  2545     0     0     0     0     0     0     0
0]
 [    0     0     0     0   150     0     0     0     0     0     0
0]
 [    0     0     0     0     0   148     0     0     0     0     0
0]
 [    0     0     0     0     0     0   309     0     0     0     0
0]
```

```
 [    0     0     0     0     0     0     0    75     0     0     0
0]
 [    0     0     0     0     0     0     0     0   106     0     0
0]
 [    0     0     0     0     0     0     0     0     0   233     0
0]
 [    0     0     0     0     0     0     0     0     0     0   492
0]
 [    0   594     0     0     0     0     0     0     0     0     0
4853]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       0.97      0.99      0.98     23107
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00      2550
           4       1.00      1.00      1.00       150
           5       1.00      1.00      1.00       148
           6       1.00      1.00      1.00       309
           7       1.00      1.00      1.00        75
           8       1.00      1.00      1.00       106
           9       1.00      1.00      1.00       233
          10       1.00      1.00      1.00       492
          11       0.96      0.89      0.92      5447

    accuracy                           0.98     32623
   macro avg       0.99      0.99      0.99     32623
weighted avg       0.98      0.98      0.98     32623

==============================

summary_3=pd.DataFrame(model_summary_3).sort_values('f1_score_test',as
cending=False).drop('model_name_test',axis=1)

summary_3

             model_name_train  f1_score_train  recall_score_train  \
1      DecisionTreeClassifier        1.000000            1.000000
3           BaggingClassifier        0.997203            0.996047
7  GradientBoostingClassifier        0.992188            0.990028
2      RandomForestClassifier        1.000000            1.000000
4        KNeighborsClassifier        0.688076            0.647983
5                  GaussianNB        0.477642            0.672186
0          LogisticRegression        0.077313            0.083661
6                         SVC        0.069168            0.083364

   accuracy_score_train  f1_score_test  recall_score_test
accuracy_score_test
1              1.000000       0.900230           0.898361
```

```
0.990059
3                0.999142       0.900123           0.895781
0.991704
7                0.975508       0.894133           0.882766
0.974968
2                1.000000       0.877153           0.862681
0.979188
4                0.939429       0.684181           0.649042
0.914676
5                0.539435       0.433813           0.645678
0.544200
0                0.664409       0.084632           0.091451
0.664783
6                0.708365       0.075379           0.090900
0.708196

plt.figure(figsize=(7,6))
sns.barplot(y=summary_3['model_name_train'],x=summary_3['f1_score_test
'])
plt.xticks(rotation=90)
plt.show()
```



Model selection for task 3 - to tag departments

- from the above graph it is found that the `DecissionTreeClassifier`,`bagging_classifier`,`gradiant boosting` performing well compared to other algorithms

- and it is performing well above 95 percentage so not using `optimization techniques` separatly

- im considering the `bagging_classifier`, DecisionTreeClassifier model over `gradiant boosting` as it performing better in more number of times compared to DecisionTree classifer

- will create the `bagging_classifier` model for further use

```
#model creation
#model initialization
department_classification_model=BaggingClassifier()

#fitting the model
department_classification_model.fit(X_train,y_train)

#predicting using the model
department_classification_pred=department_classification_model.predict
(X_test)

#printing the confusion metrics and classification report
print('metrics on test data')
print('confusion matrix')
print(confusion_matrix(y_test,department_classification_pred))
print('\n')
print('classification report')
print(classification_report(y_test,department_classification_pred))
print('==='*10)
```

```
metrics on test data
confusion matrix
[[   0    0    0    0    0    0    0    0    0    0    0    0]
 [   0 9848    0    0    0    0    0    0    0    0    0   55]
 [   0    0    1    0    0    0    0    0    0    0    0    0]
 [   1    0    0 1092    0    0    0    0    0    0    0    0]
 [   0    0    0    0   63    0    0    1    0    0    0    0]
 [   0    0    0    0    0   64    0    0    0    0    0    0]
 [   0    0    0    1    0    0  132    0    0    0    0    0]
 [   0    0    0    1    1    0    2   26    0    0    2    0]
 [   0    0    0    0    0    0    0    0   46    0    0    0]
 [   0    0    0    0    0    0    0    0    0  100    0    0]
 [   0    1    0    0    0    0    0    1    0    1  208    0]
 [   0   60    0    0    0    0    0    0    0    0    0 2275]]
```

```
classification report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.99      0.99      0.99      9903
           2       1.00      1.00      1.00         1
           3       1.00      1.00      1.00      1093
           4       0.98      0.98      0.98        64
           5       1.00      1.00      1.00        64
           6       0.99      0.99      0.99       133
           7       0.93      0.81      0.87        32
           8       1.00      1.00      1.00        46
           9       0.99      1.00      1.00       100
          10       0.99      0.99      0.99       211
          11       0.98      0.97      0.98      2335

    accuracy                           0.99     13982
   macro avg       0.90      0.90      0.90     13982
weighted avg       0.99      0.99      0.99     13982

==============================
```

Task 4

Predict RFC (Request for change) and possible failure / misconfiguration of ITSM assets.

```
data_4=df.copy()

data_4.head()

   CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
0      11         57  162       0       4        4         4
0.601292
1       1         57   88       0       3        3         3
0.415050
2       1         10   92       0       4        3         4
0.517551
3       1         57   88       0       4        4         4
0.642927
4       1         57   88       0       4        4         4
0.345258

   Category  KB_number  No_of_Reassignments           Open_Time  \
0         1        553                   26 2012-05-02 13:32:00
1         1        611                   33 2012-12-03 15:44:00
2         3        339                    3 2012-03-29 12:36:00
3         1        611                   13 2012-07-17 11:49:00
```

```
4         1        611                   2 2012-10-08 11:01:00

        Resolved_Time          Close_Time  No_of_Related_Interactions
\
0 2013-04-11 13:50:00 2013-04-11 13:51:00                           1

1 2013-02-12 12:36:00 2013-02-12 12:36:00                           1

2 2014-01-13 15:12:00 2014-01-13 15:13:00                           1

3 2013-11-14 09:31:00 2013-11-14 09:31:00                           1

4 2013-08-11 13:55:00 2013-08-11 13:55:00                           1


   Handle_Time_hrs_conv
0           8256.316667
1           1700.866667
2          15722.616667
3          11637.700000
4           7370.900000
```

```
data_4['Category'].value_counts()
```

```
1    37748
3     8845
0       11
2        1
Name: Category, dtype: int64
```

```
data_4.loc[data_4['Category']==2]
```

```
       CI_Cat  CI_Subcat  WBS  Status  Impact  Urgency  Priority
number_cnt  \
24520       1         45  296       0       5        5         5
0.900155

       Category  KB_number  No_of_Reassignments           Open_Time  \
24520         2       1032                    0 2013-12-31 11:53:00

            Resolved_Time          Close_Time
No_of_Related_Interactions  \
24520 2014-07-01 14:46:00 2014-07-01 14:46:00
1

       Handle_Time_hrs_conv
24520           4370.883333
```

```
data_4.drop(data_4.loc[data_4['Category']==2].index,inplace=True)
```

```
X_4=data_4.drop(['Category','Open_Time','Resolved_Time','Close_Time'],
axis=1)
y_4=data_4['Category']
```

Logic behind the function

1. first creating a dictionary with the name model_summary and initiating with null values with proper keys

2. function called `model_selection` will take model as parameter 3.initially the model will be `initiated` within the function and will be `stored in the variable called model`

3. model will be `fitted on x_train and y_train` 5.model will `first predict on test data` 6.after prediction all the evaluation `metric values will be appended to dictionary` with corresponding key values. 7.then it will `print the confusion matrix and classification report` of that model 8.the same `steps will also the performed` on train data ---

```
model_summary_4={'model_name_train':[],'f1_score_train':
[],'recall_score_train':[],'accuracy_score_train':[],
               'model_name_test':[],'f1_score_test':
[],'recall_score_test':[],'accuracy_score_test':[]}


def model_selction_4(model):

    #model initialization ,fitting and predicting
    print(model)
    model=model()
    model.fit(X_train,y_train)
    model_pred=model.predict(X_test)

    #appending the metrics to the dictionary created

model_summary_4['model_name_test'].append(model.__class__.__name__)

model_summary_4['f1_score_test'].append(f1_score(y_test,model_pred,ave
rage='macro'))

model_summary_4['recall_score_test'].append(recall_score(y_test,model_
pred,average='macro'))

model_summary_4['accuracy_score_test'].append(accuracy_score(y_test,mo
del_pred))

    #printing the confusion metrics and classification report
    print('metrics on test data')
```

```python
    print(confusion_matrix(y_test,model_pred))
    print('\n')
    print(classification_report(y_test,model_pred))

    #predictions on train data
    model_pred1=model.predict(X_train)

    #appending the metrics to the dictionary created

model_summary_4['model_name_train'].append(model.__class__.__name__)

model_summary_4['f1_score_train'].append(f1_score(y_train,model_pred1,
average='macro'))

model_summary_4['recall_score_train'].append(recall_score(y_train,mode
l_pred1,average='macro'))

model_summary_4['accuracy_score_train'].append(accuracy_score(y_train,
model_pred1))

    #printing the confusion metrics and classification report
    print('metrics on train data')
    print(confusion_matrix(y_train,model_pred1))
    print('\n')
    print(classification_report(y_train,model_pred1))
    print('==='*10)

X_train, X_test, y_train, y_test = train_test_split(X_4, y_4,
test_size=0.3, random_state=42,stratify=y_4)

for i in models:
    model_selction_4(i)
```

```
<class 'sklearn.linear_model._logistic.LogisticRegression'>
metrics on test data
[[    0     3     0]
 [    0 11324     1]
 [    0  2654     0]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.81      1.00      0.89     11325
           3       0.00      0.00      0.00      2654

    accuracy                           0.81     13982
   macro avg       0.27      0.33      0.30     13982
weighted avg       0.66      0.81      0.72     13982

metrics on train data
```

```
[[    0     8     0]
 [    0 26415     8]
 [    0  6190     1]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         8
           1       0.81      1.00      0.89     26423
           3       0.11      0.00      0.00      6191

    accuracy                           0.81     32622
   macro avg       0.31      0.33      0.30     32622
weighted avg       0.68      0.81      0.72     32622

==============================
<class 'sklearn.tree._classes.DecisionTreeClassifier'>
metrics on test data
[[    3     0     0]
 [    0 11139   186]
 [    1   219  2434]]


              precision    recall  f1-score   support

           0       0.75      1.00      0.86         3
           1       0.98      0.98      0.98     11325
           3       0.93      0.92      0.92      2654

    accuracy                           0.97     13982
   macro avg       0.89      0.97      0.92     13982
weighted avg       0.97      0.97      0.97     13982

metrics on train data
[[    8     0     0]
 [    0 26423     0]
 [    0     0  6191]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00     26423
           3       1.00      1.00      1.00      6191

    accuracy                           1.00     32622
   macro avg       1.00      1.00      1.00     32622
weighted avg       1.00      1.00      1.00     32622

==============================
<class 'sklearn.ensemble._forest.RandomForestClassifier'>
```

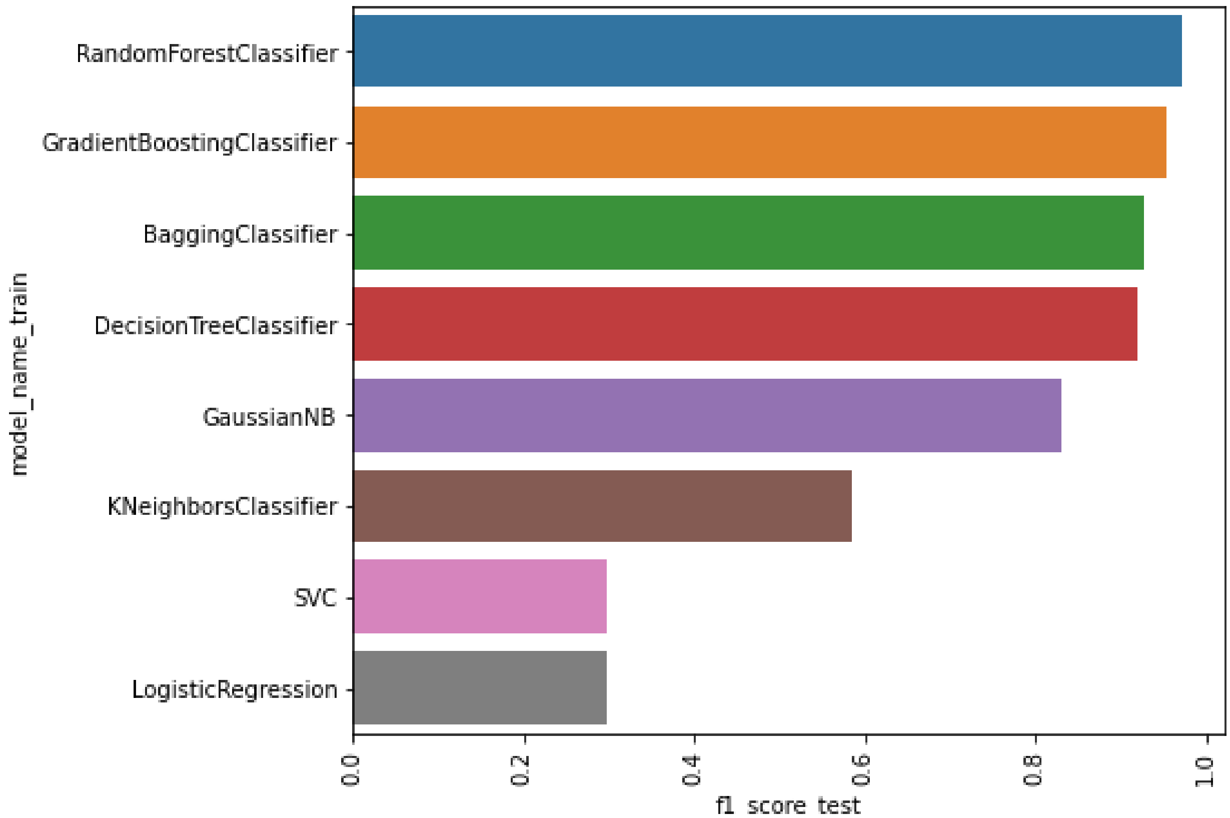
```
metrics on test data
[[    3     0     0]
 [    0 11228    97]
 [    0   244  2410]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       0.98      0.99      0.99     11325
           3       0.96      0.91      0.93      2654

    accuracy                           0.98     13982
   macro avg       0.98      0.97      0.97     13982
weighted avg       0.98      0.98      0.98     13982

metrics on train data
[[    8     0     0]
 [    0 26423     0]
 [    0     1  6190]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00     26423
           3       1.00      1.00      1.00      6191

    accuracy                           1.00     32622
   macro avg       1.00      1.00      1.00     32622
weighted avg       1.00      1.00      1.00     32622

==============================
<class 'sklearn.ensemble._bagging.BaggingClassifier'>
metrics on test data
[[    3     0     0]
 [    0 11236    89]
 [    1   232  2421]]


              precision    recall  f1-score   support

           0       0.75      1.00      0.86         3
           1       0.98      0.99      0.99     11325
           3       0.96      0.91      0.94      2654

    accuracy                           0.98     13982
   macro avg       0.90      0.97      0.93     13982
weighted avg       0.98      0.98      0.98     13982

metrics on train data
```

```
[[    8     0     0]
 [    0 26420     3]
 [    0    80  6111]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00     26423
           3       1.00      0.99      0.99      6191

    accuracy                           1.00     32622
   macro avg       1.00      1.00      1.00     32622
weighted avg       1.00      1.00      1.00     32622

==============================
<class 'sklearn.neighbors._classification.KNeighborsClassifier'>
metrics on test data
[[    0     3     0]
 [    0 10918   407]
 [    0   612  2042]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.95      0.96      0.96     11325
           3       0.83      0.77      0.80      2654

    accuracy                           0.93     13982
   macro avg       0.59      0.58      0.59     13982
weighted avg       0.93      0.93      0.93     13982

metrics on train data
[[    0     8     0]
 [    0 25830   593]
 [    0  1048  5143]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         8
           1       0.96      0.98      0.97     26423
           3       0.90      0.83      0.86      6191

    accuracy                           0.95     32622
   macro avg       0.62      0.60      0.61     32622
weighted avg       0.95      0.95      0.95     32622

==============================
<class 'sklearn.naive_bayes.GaussianNB'>
```

```
metrics on test data
[[   3    0    0]
 [   0 8830 2495]
 [   0  286 2368]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       0.97      0.78      0.86     11325
           3       0.49      0.89      0.63      2654

    accuracy                           0.80     13982
   macro avg       0.82      0.89      0.83     13982
weighted avg       0.88      0.80      0.82     13982

metrics on train data
[[    8     0     0]
 [    0 20685  5738]
 [    0   619  5572]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       0.97      0.78      0.87     26423
           3       0.49      0.90      0.64      6191

    accuracy                           0.81     32622
   macro avg       0.82      0.89      0.83     32622
weighted avg       0.88      0.81      0.82     32622

==============================
<class 'sklearn.svm._classes.SVC'>
metrics on test data
[[    0     3     0]
 [    0 11325     0]
 [    0  2654     0]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.81      1.00      0.90     11325
           3       0.00      0.00      0.00      2654

    accuracy                           0.81     13982
   macro avg       0.27      0.33      0.30     13982
weighted avg       0.66      0.81      0.72     13982

metrics on train data
```

```
[[    0     8     0]
 [    0 26423     0]
 [    0  6191     0]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         8
           1       0.81      1.00      0.90     26423
           3       0.00      0.00      0.00      6191

    accuracy                           0.81     32622
   macro avg       0.27      0.33      0.30     32622
weighted avg       0.66      0.81      0.72     32622

==============================
<class 'sklearn.ensemble._gb.GradientBoostingClassifier'>
metrics on test data
[[    3     0     0]
 [    0 11151   174]
 [    0   410  2244]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       0.96      0.98      0.97     11325
           3       0.93      0.85      0.88      2654

    accuracy                           0.96     13982
   macro avg       0.96      0.94      0.95     13982
weighted avg       0.96      0.96      0.96     13982

metrics on train data
[[    8     0     0]
 [    0 26026   397]
 [    0   859  5332]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       0.97      0.98      0.98     26423
           3       0.93      0.86      0.89      6191

    accuracy                           0.96     32622
   macro avg       0.97      0.95      0.96     32622
weighted avg       0.96      0.96      0.96     32622

==============================
```

```
summary_4=pd.DataFrame(model_summary_4).sort_values('f1_score_test',as
cending=False).drop('model_name_test',axis=1)

summary_4

              model_name_train  f1_score_train  recall_score_train  \
2       RandomForestClassifier        0.999967            0.999946
7   GradientBoostingClassifier        0.957023            0.948742
3            BaggingClassifier        0.997229            0.995655
1       DecisionTreeClassifier        1.000000            1.000000
5                   GaussianNB        0.834523            0.894286
4         KNeighborsClassifier        0.610493            0.602760
6                          SVC        0.298337            0.333333
0           LogisticRegression        0.298400            0.333286

   accuracy_score_train  f1_score_test  recall_score_test
accuracy_score_test
2              0.999969       0.972990           0.966499
0.975612
7              0.961498       0.953113           0.943384
0.958232
3              0.997456       0.926902           0.968116
0.976970
1              1.000000       0.920769           0.966894
0.970963
5              0.805132       0.831329           0.890643
0.801101
4              0.949451       0.585201           0.577822
0.926906
6              0.809975       0.298336           0.333333
0.809970
0              0.809760       0.298322           0.333304
0.809898

plt.figure(figsize=(7,6))
sns.barplot(y=summary_4['model_name_train'],x=summary_4['f1_score_test
'])
plt.xticks(rotation=90)
plt.show()
```



Model selection for task 4

- from the above graph it is found that the `RandomForestClassifier`,`bagging_classifier`,`gradiant boosting` performing well compared to other algorithms

- and it is performing well above 95 percentage so not using `optimization techniques` separatly

- im considering the `bagging_classifier`, RandomForestClassifier model over `gradiant boosting` as it performing better in more number of times compared to DecisionTree classifer

- will create the `bagging_classifier` model for further use

```
#model creation
#model initialization
category_classification_model=BaggingClassifier()

#fitting the model
category_classification_model.fit(X_train,y_train)

#predicting using the model
category_classification_pred=category_classification_model.predict(X_t
```

```
est)

#printing the confusion metrics and classification report
print('metrics on test data')
print('confusion matrix')
print(confusion_matrix(y_test,category_classification_pred))
print('\n')
print('classification report')
print(classification_report(y_test,category_classification_pred))
print('==='*10)
```

```
metrics on test data
confusion matrix
[[    3     0     0]
 [    0 11151   174]
 [    0   410  2244]]


classification report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       0.96      0.98      0.97     11325
           3       0.93      0.85      0.88      2654

    accuracy                           0.96     13982
   macro avg       0.96      0.94      0.95     13982
weighted avg       0.96      0.96      0.96     13982

==============================
```

Conlcusion

- In the task 1 we will consider Random Forest Classifier since it gave 100% accuracy.
- In the task 2 since it is a time series problem, we will consider sarima model it was performing very well in forcasting.
- In the task 3 we will consider 2 models which is for tagging priority and their respectinve departments. The models we considered for tagging priority and their respective departments are Gardient Boosting Classifier and Bagging Classifier respectively as the accuracy of the both model was 99%.
- In the task 4 we will consider the model Bagging Classifier since it gave 96% of accuracy compared to other models.

Risks and Challenges

- Stationarity Assumption: Many time series models assume stationarity, meaning that statistical properties like mean, variance, and autocorrelation structure remain constant over time. However, real-world data might exhibit trends, seasonality, or other non-stationary patterns, violating this assumption.

- Seasonality: Seasonal patterns can introduce periodic fluctuations in the data due to factors like weather, holidays, or other recurring events. Ignoring seasonality can lead to biased forecasts or misinterpretation of trends.
- Missing Values and Outliers: Time series data may contain missing values or outliers, which can distort analyses and model predictions if not handled properly. Imputation techniques or outlier detection methods are often used to address these issues.
- Overfitting: Overfitting occurs when a model captures noise or random fluctuations in the data rather than underlying patterns. This can lead to poor generalization performance, especially in complex models or with limited data.
- Data Quality: Data quality issues such as measurement errors, data entry mistakes, or inconsistencies can affect the reliability of time series analyses. Data cleaning and preprocessing techniques are essential to address these challenges.