copy-of-ticket-management

July 2, 2024

#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

[1]:
```
#basic modules
!pip install mysql-connector-python

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

##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,accurac
```

```
Collecting mysql-connector-python
  Downloading mysql_connector_python-9.0.0-cp310-cp310-manylinux_2_17_x86_64.whl
(19.3 MB)
                           19.3/19.3 MB
44.8 MB/s eta 0:00:00
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-9.0.0
```

[2]:
```
encoder=LabelEncoder()
```

#basic checks

[3]:
```
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
```

[4]:
```
df = pd.read_csv('Ticket.csv')
```

[5]:
```
df = df.replace('', pd.NA)
```

[6]:
```
pd.set_option('display.max_columns',None)
```

[7]:
```
df.head()
```

[7]:

	CI_Name	CI_Cat	CI_Subcat	WBS	Incident_ID	Status	Impact	Urgency	Priority	number_cnt	Category	KB_number	Alert_Status	No_of_Reassignments	Open_Time	Reopen_Time	Resolved_Time	Close_Time	Handle_Time_hrs	Closure_Code	No_of_Related_Interactions
0	SUB000508	subapplication	Web Based Application	WBS000162	IM0000004	Closed	4	4	4.0	0.601292	incident	KM0000553	closed	26.0	05-02-2012 13:32	NaN	04-11-2013 13:50	04-11-2013 13:51	3,87,16,91,111	Other	1.0
1	WBA000124	application	Web Based Application	WBS000088	IM0000005	Closed	3	3	3.0	0.415050	incident	KM0000611	closed	33.0	12-03-2012 15:44	02-12-2013 12:31	02-12-2013 12:36	02-12-2013 12:36	4,35,47,86,389	Software	1.0
2	DTA000024	application	Desktop Application	WBS000092	IM0000006	Closed	NS	3	NaN	0.517551	request for information	KM0000339	closed	3.0	29-03-2012 12:36	NaN	13-01-2014 15:12	13-01-2014 15:13	4,84,31,19,444	No error - works as designed	1.0
3	WBA000124	application	Web Based Application	WBS000088	IM0000011	Closed	4	4	4.0	0.642927	incident	KM0000611	closed	13.0	17-07-2012 11:49	NaN	14-11-2013 09:31	14-11-2013 09:31	4,32,18,33,333	Operator error	1.0
4	WBA000124	application	Web Based Application	WBS000088	IM0000012	Closed	4	4	4.0	0.345258	incident	KM0000611	closed	2.0	10-08-2012 11:01	NaN	08-11-2013 13:55	08-11-2013 13:55	3,38,39,03,333		

```
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
```

[8]:
```
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
```

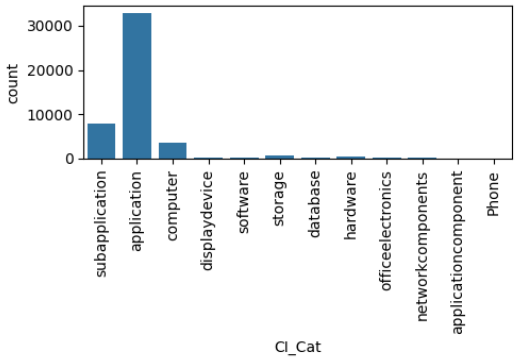
```
 24  Related_Change              560 non-null    object
dtypes: float64(6), object(19)
memory usage: 8.9+ MB
```

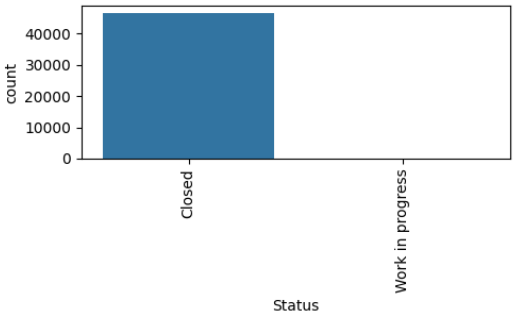
[9]:
```
exclude_columns␣
 ↪=['CI_Name','CI_Subcat','WBS','Incident_ID','number_cnt','KB_number','Open_Time','Reopen_Ti
    ␣
 ↪'Close_Time','Handle_Time_hrs','Related_Interaction','No_of_Related_Incidents','No_of_Relat
print(len([column for column in df.columns if column not in exclude_columns]))
print('\n')
print([column for column in df.columns if column not in exclude_columns])
```

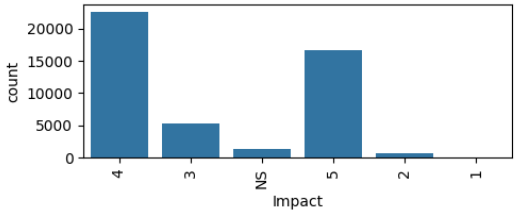
```
10


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

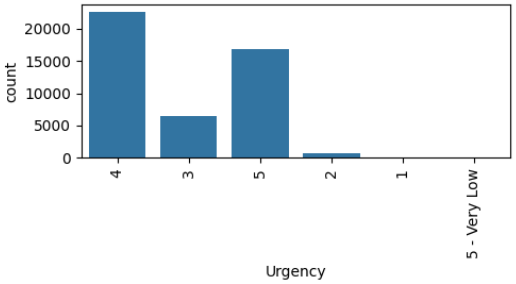
#EDA

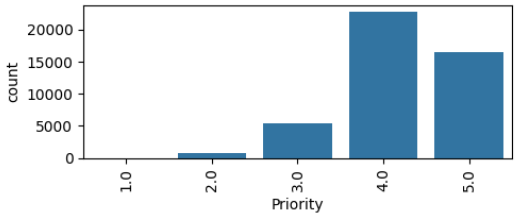
[10]:
```
pl_no=1
plt.figure(figsize=(10,18))
for i in [column for column in df.columns if column not in exclude_columns]:

  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()
```

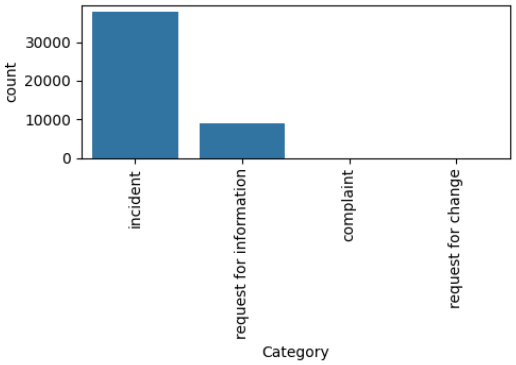

count
30000
20000
10000
0
subapplication
application
computer
displaydevice
software
storage
database
hardware
officeelectronics
networkcomponents
applicationcomponent
Phone
CI_Cat

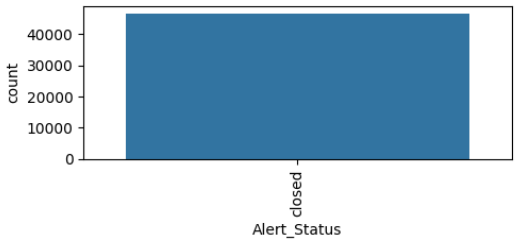

count
40000
30000
20000
10000
0
Closed
Work in progress
Status

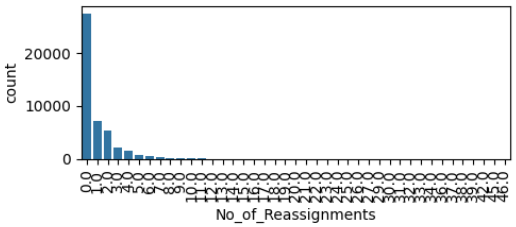

count
20000
15000
10000
5000
0
4
3
NS
5
2
1
Impact

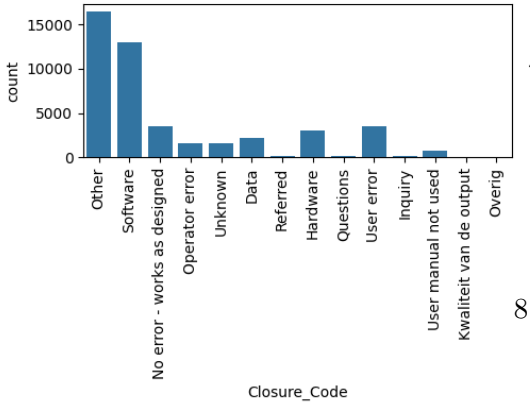

count
20000
15000
10000
5000
0
4
3
5
2
1
5 - Very Low
Urgency

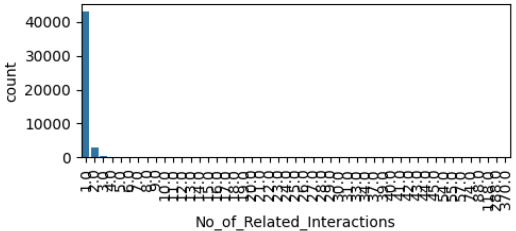

count
20000
15000
10000
5000
0
1.0
2.0
3.0
4.0
5.0
Priority


count
30000
20000
10000
0
incident
request for information
complaint
request for change
Category


count
40000
30000
20000
10000
0
closed
Alert_Status


count
20000
10000
0
No_of_Reassignments


count
15000
10000
5000
0
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
Overig
Closure_Code


count
40000
30000
20000
10000
0
No_of_Related_Interactions

0.1 ##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

---

[11]:
```
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**

---

[12]:
```
null_df=pd.DataFrame((df.isnull().sum()/len(df))*100,columns=['per'])
null_df['count']=df.isnull().sum()
null_df
```

[12]:

	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

---

[13]:
```
null_df[null_df['per']>50]
```

[13]:

	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

[14]:
```
df.drop(null_df[null_df['per']>50].index,axis=1,inplace=True)
```

[15]:
```
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

---

[16]:
```
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
  1. null value imputation
  2. label encoding
  3. force typecating
  4. dropping the unnessesory columns

  and other required preprocessing steps

0.2 df['CI_Name']

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

---

[17]:
```
df.drop('CI_Name',axis=1,inplace=True)
```

[18]:
```
df.head()
```

[18]:

	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
```

0.3 df['CI_Cat']

[19]:
```
df['CI_Cat'].value_counts()
```

[19]:
```
CI_Cat
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: count, dtype: int64
```

[20]:
```
df['CI_Cat'].isnull().sum()
```

[20]:
```
111
```

[21]:
```
df.loc[df['CI_Cat'].isnull(),'CI_Cat']='application'
```

[22]:
```
df['CI_Cat'].value_counts()
```

[22]:
```
CI_Cat
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: count, dtype: int64
```

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

[23]:
```
df['CI_Cat']=encoder.fit_transform(df['CI_Cat'])
```

[24]:
```
df.head()
```

[24]:

	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
```

0.4 df['CI_Subcat']

[25]:
```
df['CI_Subcat'].unique()
```

[25]:
```
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)
```

[26]:
```
df['CI_Subcat'].isnull().sum()
```

[26]:
```
111
```

[27]:
```
df['CI_Subcat'].mode()
```

[27]:
```
0    Server Based Application
Name: CI_Subcat, dtype: object
```

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

application

[28]:
```
df.loc[df['CI_Subcat'].isnull(),'CI_Subcat']=df['CI_Subcat'].mode()[0]
```

[29]:
```
df['CI_Subcat'].isnull().sum()
```

[29]:
```
0
```

- using `lable_encoder` to transform categorical to numerical columns

[30]:
```
df['CI_Subcat']=encoder.fit_transform(df['CI_Subcat'])
```

[31]:
```
df.head()
```

[31]:

	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

0.5 df['WBS']

[32]:
```
len(df['WBS'].unique())
```

[32]:
```
274
```

- extracting the unique numbers of the WBS system

[33]:
```
df['WBS']=df['WBS'].apply(lambda x: x[-3:])
```

[34]:
```
df['WBS']
```

[34]:
```
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
```

[35]:
```
df['WBS']=df['WBS'].astype(int)
```

[36]:
```
df.head()
```

[36]:

	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
```

0.6 df['Incident_ID']

[37]:
```
len(df['Incident_ID'].unique())
```

[37]:
```
46606
```

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

[38]:
```
df.drop('Incident_ID',axis=1,inplace=True)
```

[39]:
```
df.head()
```

[39]:
```
   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
```

0.7 df['Status']

[40]:
```
df['Status'].unique()
```

[40]:
```
array(['Closed', 'Work in progress'], dtype=object)
```

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

[41]:
```
df['Status'].isnull().sum()
```

[41]:
```
0
```

[42]:
```
df['Status']=encoder.fit_transform(df['Status'])
```

[43]:
```
df.head()
```

[43]:
```
   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
```

0.8 df['Impact']

[44]:
```
df['Impact'].value_counts()
```

[44]:
```
Impact
4     22556
5     16741
3      5234
NS     1380
2       692
1         3
Name: count, dtype: int64
```

[45]:
```
df['Impact'].mode()[0]
```

[45]:
```
'4'
```

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

[46]:
```
df.loc[df['Impact']=='NS','Impact']=df['Impact'].mode()[0]
```

[47]:
```
df.loc[df['Impact']=='NS']
```

[47]:
```
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: []
```

[48]:
```
df['Impact'].dtype
```

[48]:
```
dtype('O')
```

[49]:
```
df['Impact']=df['Impact'].astype(int)
```

[50]:
```
df['Impact'].unique()
```

[50]:
```
array([4, 3, 5, 2, 1])
```

0.9 df['Urgency']

[51]:
```
df['Urgency'].value_counts()
```

[51]:
```
Urgency
4                15526
5                12284
4                 7062
5                 4495
3                 4419
3                 2117
2                  538
2                  158
1                    5
1                    1
5 - Very Low         1
Name: count, dtype: int64
```

- as only 1 entry there in data dropping the column

[52]:
```
df.drop(df.loc[df['Urgency']=='5 - Very Low'].index,axis=0,inplace=True)
```

[53]:
```
df['Urgency'].value_counts()
```

[53]:
```
Urgency
4    15526
5    12284
4     7062
5     4495
3     4419
3     2117
2      538
2      158
```

```
1        5
1        1
Name: count, dtype: int64
```

[54]:
```
df.drop_duplicates()
```

[54]:

	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]
```

[55]:
```
df['Urgency']=df['Urgency'].astype(int)
```

[56]:
```
df['Urgency'].unique()
```

[56]:
```
array([4, 3, 5, 2, 1])
```

[57]:
```
df.shape
```

[57]:
```
(46605, 19)
```

[58]:
```
df.head()
```

[58]:
```
   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
```

0.10 df['Priority']

[59]:
```
df['Priority'].unique()
```

[59]:
```
array([ 4.,  3., nan,  5.,  2.,  1.])
```

[60]:
```
df['Priority'].value_counts()
```

[60]:
```
Priority
4.0    22717
5.0    16485
3.0     5323
2.0      697
1.0        3
Name: count, dtype: int64
```

- replacing the null values with mode

[61]:
```
df['Priority'].mode()
```

[61]:
```
0    4.0
Name: Priority, dtype: float64
```

[62]:
```
df.loc[df['Priority'].isna(),'Priority']=df['Priority'].mode()[0]
```

[63]:
```
df['Priority']=df['Priority'].astype(int)
```

[64]:
```
df.head()
```

[64]:

	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’]

[65]:
```
df['number_cnt']=df['number_cnt'].astype(float)
```

0.11 df['Category']

[66]:
```
df['Category'].unique()
```

[66]:
```
array(['incident', 'request for information', 'complaint',
       'request for change'], dtype=object)
```

- transforming the catergorical columns to numerical columns

[67]:
```
df['Category']=encoder.fit_transform(df['Category'])
```

[68]:
```
df.head()
```

[68]:

	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

0.12 df['Alert_Status']

[69]:
```
df['Alert_Status'].unique()
```

[69]:
```
array(['closed'], dtype=object)
```

[70]:
```
df['Alert_Status'].value_counts()
```

[70]:
```
Alert_Status
closed    46605
Name: count, dtype: int64
```

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

[71]:
```
df.drop('Alert_Status',axis=1,inplace=True)
```

[72]:
```
df.head()
```

[72]:

	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
```

0.13 df['KB_number']

[73]:
```
len(df['KB_number'].unique())
```

[73]:
```
1824
```

- extracting the last 4 numbers of column

[74]:
```
df['KB_number']=df['KB_number'].apply(lambda x: x[-4:])
```

[75]:
```
df['KB_number']=df['KB_number'].astype(int)
```

[76]:
```
df.head()
```

[76]:

	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
```

[77]:
```
len(df['KB_number'].unique())
```

[77]:
```
1824
```

0.14 df['No__of__Reassignments']

[78]:
```
len(df['No_of_Reassignments'].unique())
```

[78]:
```
42
```

[79]:
```
df['No_of_Reassignments'].mode()
```

[79]:
```
0    0.0
Name: No_of_Reassignments, dtype: float64
```

[80]:
```
df['No_of_Reassignments'].isnull().sum()
```

[80]:
```
1
```

- replacing the null values with mode i.e 0

[81]:
```
df.loc[df['No_of_Reassignments'].
↪isnull(),'No_of_Reassignments']=df['No_of_Reassignments'].mode()[0]
```

[82]:
```
df['No_of_Reassignments'].isnull().sum()
```

[82]:
```
0
```

[83]:
```
df['No_of_Reassignments']=df['No_of_Reassignments'].astype(int)
```

[84]:
```
df.head()
```

[84]:
```
   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
```

0.15 df['Open_Time']

[85]:
```
df['Open_Time'].isnull().sum()
```

[85]:
```
0
```

- converting the open time column to datetime format

[86]:
```
df['Open_Time'] = pd.to_datetime(df['Open_Time'], format="%d-%m-%Y %H:%M")
```

[87]:
```
df.head()
```

[87]:
```
    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 2012-02-05 13:32:00
1         1        611                   33 2012-03-12 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 2012-08-10 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
```

0.16 df['Resolved_Time']

[88]:
```
df['Resolved_Time'] = pd.to_datetime(df['Resolved_Time'], format="%d-%m-%Y %H:
↪%M")
```

[89]:
```
df.head()
```

[89]:
```
   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 2012-02-05 13:32:00
1         1        611                   33 2012-03-12 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 2012-08-10 11:01:00
```

```
        Resolved_Time        Close_Time Handle_Time_hrs  \
0 2013-11-04 13:50:00  04-11-2013 13:51  3,87,16,91,111
1 2013-12-02 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 2013-11-08 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
```

[90]:
```
df['Resolved_Time'].isnull().sum()
```

[90]:
```
1780
```

[91]:
```
df['Resolved_Time'].mode()[0]
```

[91]:
```
Timestamp('2013-10-10 12:53:00')
```

[92]:
```
df.loc[df['Resolved_Time'].isnull(),'Resolved_Time']=df['Resolved_Time'].
↪mode()[0]
```

[93]:
```
df['Resolved_Time'].isnull().sum()
```

[93]:
```
0
```

[94]:
```
df['Resolved_Time']=pd.to_datetime(df['Resolved_Time'])
```

[95]:
```
# data.drop('Resolved_Time',axis=1,inplace=True)
```

[96]:
```
df.head()
```

[96]:

	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 2012-02-05 13:32:00
1         1        611                   33 2012-03-12 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 2012-08-10 11:01:00

        Resolved_Time        Close_Time Handle_Time_hrs  \
0 2013-11-04 13:50:00  04-11-2013 13:51  3,87,16,91,111
1 2013-12-02 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 2013-11-08 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
```

0.17 df['Close_Time']

[97]:
```
df['Close_Time'].isnull().sum()
```

[97]:
```
0
```

[98]:
```
df['Close_Time']=pd.to_datetime(df['Close_Time'], format="%d-%m-%Y %H:%M")
```

[99]:
```
df.head()
```

[99]:
```
    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 2012-02-05 13:32:00
1         1        611                   33 2012-03-12 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 2012-08-10 11:01:00

         Resolved_Time          Close_Time Handle_Time_hrs  \
0 2013-11-04 13:50:00 2013-11-04 13:51:00  3,87,16,91,111
1 2013-12-02 12:36:00 2013-12-02 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 2013-11-08 13:55:00 2013-11-08 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
```

0.18 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

[100]:
```
df.drop('Handle_Time_hrs',axis=1,inplace=True)
```

[101]:
```
df['Handle_Time_hrs_conv']=abs(df['Close_Time']-df['Open_Time'])
```

[102]:
```
a=[]
for i in df['Handle_Time_hrs_conv'].index:
    a.append((df['Handle_Time_hrs_conv'][i].total_seconds())/3600)
```

[103]:
```
df['Handle_Time_hrs_conv']=a
```

[104]:
```
df.head()
```

[104]:
```
   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 2012-02-05 13:32:00
1         1        611                   33 2012-03-12 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 2012-08-10 11:01:00

         Resolved_Time          Close_Time                  Closure_Code  \
0 2013-11-04 13:50:00 2013-11-04 13:51:00                         Other
1 2013-12-02 12:36:00 2013-12-02 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 2013-11-08 13:55:00 2013-11-08 13:55:00                         Other

   No_of_Related_Interactions Related_Interaction  Handle_Time_hrs_conv
0                         1.0           SD0000007          15312.316667
1                         1.0           SD0000011          15116.866667
2                         1.0           SD0000017          15722.616667
3                         1.0           SD0000025          11637.700000
4                         1.0           SD0000029          10922.900000
```

0.19 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

[105]:
```
df.drop('Closure_Code',axis=1,inplace=True)
```

[106]:
```
df.head()
```

[106]:
```
   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 2012-02-05 13:32:00
1         1        611                   33 2012-03-12 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 2012-08-10 11:01:00

         Resolved_Time          Close_Time  No_of_Related_Interactions  \
0 2013-11-04 13:50:00 2013-11-04 13:51:00                         1.0
1 2013-12-02 12:36:00 2013-12-02 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 2013-11-08 13:55:00 2013-11-08 13:55:00                         1.0

  Related_Interaction  Handle_Time_hrs_conv
0           SD0000007          15312.316667
1           SD0000011          15116.866667
2           SD0000017          15722.616667
3           SD0000025          11637.700000
4           SD0000029          10922.900000
```

0.20 df['No__of__Related__Interactions']

[107]:
```
df['No_of_Related_Interactions'].isnull().sum()
```

[107]:
```
114
```

[108]:
```
len(df['No_of_Related_Interactions'].unique())
```

[108]:
```
50
```

[109]:
```
df['No_of_Related_Interactions'].mode()
```

[109]:
```
0    1.0
Name: No_of_Related_Interactions, dtype: float64
```

- replacing the null values with mode

[110]:
```
df.loc[df['No_of_Related_Interactions'].
↪isnull(),'No_of_Related_Interactions']=df['No_of_Related_Interactions'].
↪mode()[0]
```

[111]:
```
df['No_of_Related_Interactions'].isnull().sum()
```

[111]:
```
0
```

[112]:
```
df['No_of_Related_Interactions']=df['No_of_Related_Interactions'].astype(int)
```

[113]:
```
df.head()
```

[113]:
```
   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 2012-02-05 13:32:00
1         1        611                   33 2012-03-12 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 2012-08-10 11:01:00

         Resolved_Time          Close_Time  No_of_Related_Interactions  \
0 2013-11-04 13:50:00 2013-11-04 13:51:00                           1
1 2013-12-02 12:36:00 2013-12-02 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 2013-11-08 13:55:00 2013-11-08 13:55:00                           1

  Related_Interaction  Handle_Time_hrs_conv
0           SD0000007          15312.316667
1           SD0000011          15116.866667
2           SD0000017          15722.616667
3           SD0000025          11637.700000
4           SD0000029          10922.900000
```

0.21 df[‘Related_Interaction’]

[114]:
```
len(df['Related_Interaction'].unique())
```

[114]:
```
43059
```

[115]:
```
df.drop('Related_Interaction',axis=1,inplace=True)
```

0.22 Preprocessed dataset for machine learning

[116]:
```
df.head()
```

[116]:
```
   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 2012-02-05 13:32:00
1         1        611                   33 2012-03-12 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 2012-08-10 11:01:00

         Resolved_Time          Close_Time  No_of_Related_Interactions  \
0 2013-11-04 13:50:00 2013-11-04 13:51:00                           1
1 2013-12-02 12:36:00 2013-12-02 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 2013-11-08 13:55:00 2013-11-08 13:55:00                           1

   Handle_Time_hrs_conv
0          15312.316667
1          15116.866667
2          15722.616667
3          11637.700000
4          10922.900000
```

[117]:
```
df.shape
```

[117]:
```
(46605, 16)
```

0.23 # 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.

[118]:
```
# sns.pairplot(data=data)
```

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

[119]:
```
df.isnull().sum()
```

[119]:
```
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
```

[120]:
```
data=df.drop(['Open_Time','Resolved_Time','Close_Time'],axis=1)
```

[121]:
```
data.head()
```

[121]:

	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	15312.316667
1	15116.866667
2	15722.616667
3	11637.700000
4	10922.900000

[122]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 46605 entries, 0 to 46605
Data columns (total 13 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   CI_Cat                      46605 non-null  int64
 1   CI_Subcat                   46605 non-null  int64
 2   WBS                         46605 non-null  int64
 3   Status                      46605 non-null  int64
 4   Impact                      46605 non-null  int64
 5   Urgency                     46605 non-null  int64
 6   Priority                    46605 non-null  int64
 7   number_cnt                  46605 non-null  float64
```

```
 8   Category                    46605 non-null  int64
 9   KB_number                   46605 non-null  int64
 10  No_of_Reassignments         46605 non-null  int64
 11  No_of_Related_Interactions  46605 non-null  int64
 12  Handle_Time_hrs_conv        46605 non-null  float64
dtypes: float64(2), int64(11)
memory usage: 6.0 MB
```

[123]:
```
scaler=MinMaxScaler()
```

[124]:
```
X=data.drop(['Priority','Urgency'],axis=1)
```

[125]:
```
X.head()
```

[125]:

	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	15312.316667
1	33	1	15116.866667
2	3	1	15722.616667
3	13	1	11637.700000
4	2	1	10922.900000

[126]:
```
y=data['Priority'].map({1:1,2:1,3:0,4:0,5:0})
```

[127]:
```
y.value_counts()
```

[127]:
```
Priority
0    45905
1      700
Name: count, dtype: int64
```

0.24 train test split

[128]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42,stratify=y)
```

[129]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

0.25 scaling

[130]:
```
X_train_scaled=scaler.fit_transform(X_train)
 X_test_scaled=scaler.transform(X_test)
```

[131]:
```
X_train_scaled=pd.DataFrame(X_train_scaled,columns=X_train.columns)
 X_train_scaled.head()
```

[131]:

	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	0.010536
1	0.010927
2	0.000119
3	0.003129
4	0.004258

[132]:
```
X_test_scaled=pd.DataFrame(X_test_scaled,columns=X_test.columns)
 X_test_scaled.head()
```

[132]:

	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              0.000254
1              0.001528
2              0.004826
3              0.004684
4              0.000005
```

0.26 function for model selection task1

0.27 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 —

[133]:
```
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,average='macro'))
    model_summary['recall_score_test'].
↪append(recall_score(y_test,model_pred,average='macro'))
```

```
    model_summary['accuracy_score_test'].
↪append(accuracy_score(y_test,model_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,average='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,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)
```

[134]:
```
models=[LogisticRegression,DecisionTreeClassifier,RandomForestClassifier,
        ␣
↪BaggingClassifier,KNeighborsClassifier,GaussianNB,SVC,GradientBoostingClassifier]
```

[135]:
```
for i in models:
    model_selction_1(i)
```

```
<class 'sklearn.linear_model._logistic.LogisticRegression'>
metrics on test data
[[13738    34]
 [  184    26]]


              precision    recall  f1-score   support

           0       0.99      1.00      0.99     13772
           1       0.43      0.12      0.19       210

    accuracy                           0.98     13982
```

```
   macro avg       0.71      0.56      0.59     13982
weighted avg       0.98      0.98      0.98     13982

metrics on train data
[[32059    74]
 [  418    72]]


              precision    recall  f1-score   support

           0       0.99      1.00      0.99     32133
           1       0.49      0.15      0.23       490

    accuracy                           0.98     32623
   macro avg       0.74      0.57      0.61     32623
weighted avg       0.98      0.98      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]
 [    1   489]]
```

```
              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
[[13757    15]
 [   88   122]]


              precision    recall  f1-score   support

           0       0.99      1.00      1.00     13772
           1       0.89      0.58      0.70       210

    accuracy                           0.99     13982
   macro avg       0.94      0.79      0.85     13982
weighted avg       0.99      0.99      0.99     13982

metrics on train data
[[32108    25]
 [  155   335]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     32133
           1       0.93      0.68      0.79       490

    accuracy                           0.99     32623
   macro avg       0.96      0.84      0.89     32623
weighted avg       0.99      0.99      0.99     32623

==============================
<class 'sklearn.naive_bayes.GaussianNB'>
metrics on test data
[[13769     3]
 [   14   196]]
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13772
           1       0.98      0.93      0.96       210

    accuracy                           1.00     13982
   macro avg       0.99      0.97      0.98     13982
weighted avg       1.00      1.00      1.00     13982

metrics on train data
[[32124     9]
 [   23   467]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     32133
           1       0.98      0.95      0.97       490

    accuracy                           1.00     32623
   macro avg       0.99      0.98      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

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

[136]:
```
summary=pd.DataFrame(model_summary).
↪sort_values('f1_score_test',ascending=False).drop('model_name_test',axis=1)
```

[137]:
```
summary
```

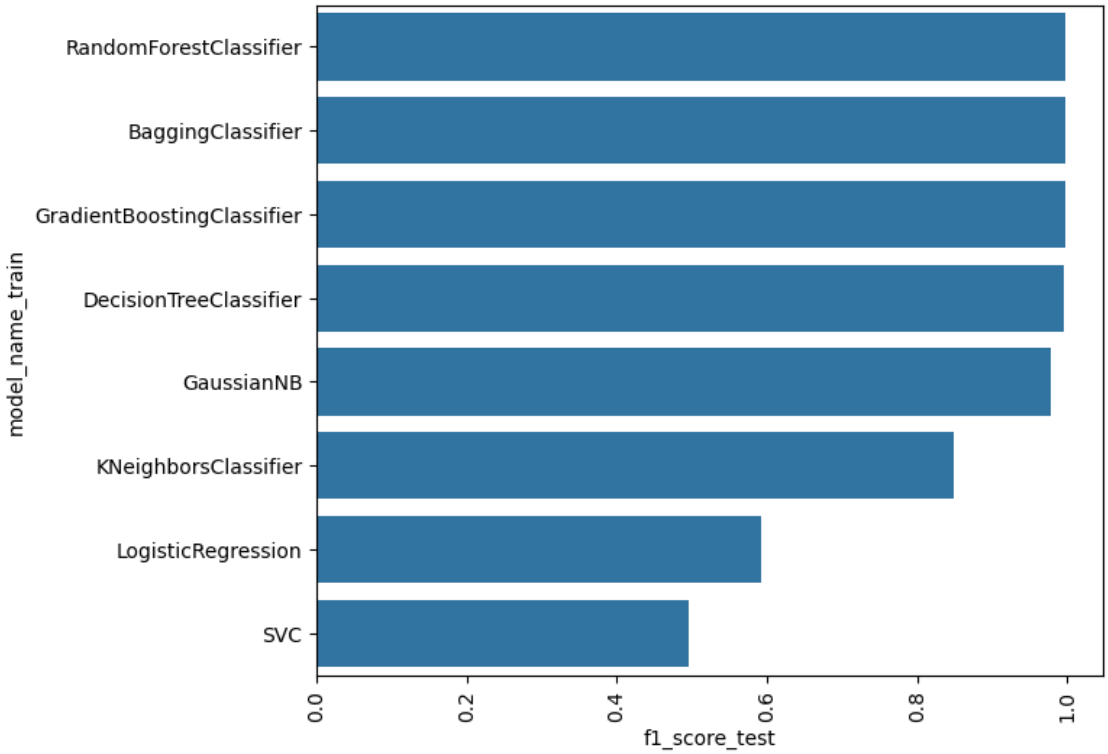
[137]:

	model_name_train	f1_score_train	recall_score_train	\
2	RandomForestClassifier	1.000000	1.000000	
3	BaggingClassifier	0.999481	0.998980	
7	GradientBoostingClassifier	1.000000	1.000000	
1	DecisionTreeClassifier	1.000000	1.000000	

5	GaussianNB	0.983188	0.976391
4	KNeighborsClassifier	0.892720	0.841448
0	LogisticRegression	0.609400	0.572318
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	0.999969	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	0.999019	0.978909	0.966558	0.998784
4	0.994482	0.849720	0.789932	0.992633
0	0.984919	0.592360	0.560670	0.984409
6	0.984980	0.496217	0.500000	0.984981

[138]:
```
plt.figure(figsize=(7,6))
sns.barplot(y=summary['model_name_train'],x=summary['f1_score_test'])
plt.xticks(rotation=90)
plt.show()
```



0.28 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

[139]:
```
#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

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

1 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.

[140]:
```
data_1=df.copy()
```

[141]:
```
data_1.head()
```

[141]:

	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	2012-02-05 13:32:00
1	1	611	33	2012-03-12 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	2012-08-10 11:01:00

	Resolved_Time	Close_Time	No_of_Related_Interactions
0	2013-11-04 13:50:00	2013-11-04 13:51:00	1
1	2013-12-02 12:36:00	2013-12-02 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	2013-11-08 13:55:00	2013-11-08 13:55:00	1

	Handle_Time_hrs_conv
0	15312.316667
1	15116.866667
2	15722.616667
3	11637.700000
4	10922.900000

[142]:
```
data_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 46605 entries, 0 to 46605
Data columns (total 16 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   CI_Cat                      46605 non-null  int64
 1   CI_Subcat                   46605 non-null  int64
 2   WBS                         46605 non-null  int64
 3   Status                      46605 non-null  int64
```

```
 4   Impact                       46605 non-null  int64
 5   Urgency                      46605 non-null  int64
 6   Priority                     46605 non-null  int64
 7   number_cnt                   46605 non-null  float64
 8   Category                     46605 non-null  int64
 9   KB_number                    46605 non-null  int64
 10  No_of_Reassignments          46605 non-null  int64
 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  int64
 15  Handle_Time_hrs_conv         46605 non-null  float64
dtypes: datetime64[ns](3), float64(2), int64(11)
memory usage: 7.1 MB
```

---

[143]:
```
timeseries_data=data_1.sort_values('Open_Time')
```

[144]:
```
timeseries_data.head()
```

[144]:

	CI_Cat	CI_Subcat	WBS	Status	Impact	Urgency	Priority	number_cnt	Category	KB_number	No_of_Reassignments	Open_Time	Resolved_Time	Close_Time	No_of_Related_Interactions	Handle_Time_hrs_conv
0	11	57	162	0	4	4	4	0.601292	1	553	26	2012-02-05 13:32:00	2013-11-04 13:50:00	2013-11-04 13:51:00	1	15312.316667
1	1	57	88	0	3	3	3	0.415050	1	611	33	2012-03-12 15:44:00	2013-12-02 12:36:00	2013-12-02 12:36:00	1	15116.866667
2	1	10	92	0	4	3	4	0.517551	3	339	3	2012-03-29 12:36:00	2014-01-13 15:12:00	2014-01-13 15:13:00	1	15722.616667
3	1	57	88	0	4	4	4	0.642927	1	611	13	2012-07-17 11:49:00	2013-11-14 09:31:00	2013-11-14 09:31:00	1	11637.700000
4	1	57	88	0	4	4	4	0.345258	1	611	2	2012-08-10 11:01:00	2013-11-08 13:55:00	2013-11-08 13:55:00	1	10922.900000

[145]:
```
forecast_data=timeseries_data[['CI_Cat','Open_Time']]
```

[146]:
```
forecast_data['Open_Time']=forecast_data['Open_Time'].dt.date
```

[147]:
```
forecast_data.head()
```

[147]:

	CI_Cat	Open_Time
0	11	2012-02-05
1	1	2012-03-12
2	1	2012-03-29
3	1	2012-07-17
4	1	2012-08-10

[148]:
```
pivot_table = forecast_data.pivot_table(index='Open_Time', columns='CI_Cat',␣
↪aggfunc='size')
```

[149]:
```
pd.set_option('display.max_rows',None)
```

[150]:
```
pivot_table
```

[150]:

CI_Cat	0	1	2	3	4	5	6	7	8	9	10
Open_Time											
2012-02-05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2012-03-12	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
2012-07-17	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2012-08-10	NaN	2.0	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
2012-09-03	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
2012-10-01	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2012-10-02	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
2012-12-05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2012-12-07	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2012-12-10	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
2013-01-15	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

2013-02-04	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-02-06	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-02-08	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
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
2013-03-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-03-04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-03-05	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-03-11	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-03-12	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
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
2013-04-03	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-04-04	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-04-05	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-04-09	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-04-10	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
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
2013-05-01	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
2013-05-06	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-05-07	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-05-10	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
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
2013-06-03	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-06-04	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-06-05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

2013-06-06	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-06-10	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-06-11	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-06-12	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
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
2013-07-01	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-07-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-07-03	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-07-04	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-07-05	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-07-08	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-07-09	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-07-10	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-07-11	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-07-12	NaN	3.0	NaN	NaN	NaN	NaN	NaN	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
2013-08-01	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-08-05	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-08-06	NaN	2.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-08-07	NaN	7.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-08-08	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-08-09	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-08-12	NaN	4.0	NaN	NaN	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
2013-09-02	NaN	7.0	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-09-03	NaN	8.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-09-04	NaN	8.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-09-05	NaN	5.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-09-06	NaN	7.0	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	15.0	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN
2013-09-11	NaN	24.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	1.0	1.0
2013-09-12	NaN	13.0	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
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
2013-10-01	NaN	286.0	NaN	55.0	3.0	1.0	10.0	NaN	NaN	1.0	4.0
2013-10-02	NaN	294.0	NaN	34.0	NaN	NaN	2.0	1.0	1.0	4.0	5.0
2013-10-03	NaN	333.0	NaN	29.0	NaN	2.0	4.0	NaN	1.0	3.0	5.0
2013-10-04	NaN	260.0	NaN	30.0	NaN	NaN	2.0	NaN	1.0	NaN	1.0
2013-10-05	NaN	4.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-10-06	NaN	3.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-10-07	NaN	347.0	NaN	29.0	1.0	7.0	2.0	NaN	2.0	1.0	4.0
2013-10-08	NaN	285.0	NaN	27.0	2.0	3.0	2.0	5.0	1.0	1.0	4.0
2013-10-09	NaN	247.0	NaN	32.0	NaN	2.0	1.0	1.0	1.0	1.0	2.0
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	227.0	NaN	22.0	NaN	NaN	3.0	3.0	1.0	1.0	5.0
2013-10-12	NaN	3.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN
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
2013-11-01	NaN	233.0	NaN	18.0	NaN	3.0	1.0	NaN	NaN	5.0	3.0
2013-11-02	NaN	3.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	2.0	NaN
2013-11-03	NaN	3.0	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	2.0
2013-11-04	NaN	321.0	NaN	33.0	NaN	2.0	2.0	2.0	NaN	2.0	16.0
2013-11-05	NaN	322.0	NaN	35.0	3.0	2.0	2.0	2.0	2.0	3.0	10.0
2013-11-06	NaN	328.0	1.0	38.0	NaN	2.0	4.0	NaN	4.0	7.0	7.0
2013-11-07	NaN	270.0	NaN	42.0	NaN	1.0	2.0	NaN	1.0	4.0	7.0
2013-11-08	NaN	195.0	NaN	26.0	NaN	1.0	4.0	NaN	2.0	4.0	5.0
2013-11-09	NaN	4.0	NaN	1.0	NaN	NaN	NaN	1.0	NaN	NaN	NaN
2013-11-10	NaN	5.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN
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	277.0	NaN	44.0	1.0	NaN	3.0	2.0	NaN	3.0	3.0
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
2013-12-01	NaN	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	NaN
2013-12-02	NaN	279.0	NaN	24.0	1.0	1.0	1.0	1.0	1.0	4.0	5.0

2013-12-03	NaN	282.0	NaN	43.0	NaN	1.0	2.0	4.0	2.0	NaN	7.0
2013-12-04	NaN	267.0	NaN	25.0	1.0	3.0	1.0	NaN	NaN	3.0	6.0
2013-12-05	NaN	229.0	1.0	29.0	NaN	10.0	1.0	NaN	6.0	2.0	6.0
2013-12-06	NaN	226.0	NaN	20.0	NaN	1.0	NaN	1.0	1.0	NaN	5.0
2013-12-07	NaN	4.0	NaN	1.0	NaN	NaN	1.0	NaN	NaN	NaN	NaN
2013-12-08	NaN	1.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2013-12-09	NaN	306.0	NaN	35.0	NaN	1.0	1.0	NaN	NaN	NaN	4.0
2013-12-10	NaN	283.0	NaN	40.0	3.0	3.0	2.0	2.0	1.0	1.0	6.0
2013-12-11	NaN	243.0	NaN	14.0	NaN	2.0	1.0	1.0	1.0	1.0	10.0
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	233.0	NaN	30.0	4.0	1.0	12.0	NaN	2.0	3.0	4.0
2014-01-03	NaN	182.0	NaN	20.0	NaN	NaN	6.0	NaN	3.0	1.0	9.0
2014-01-04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2014-01-05	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2014-01-06	NaN	280.0	NaN	29.0	7.0	NaN	5.0	1.0	3.0	3.0	9.0
2014-01-07	NaN	297.0	NaN	25.0	3.0	1.0	1.0	1.0	3.0	9.0	8.0
2014-01-08	NaN	249.0	NaN	33.0	2.0	3.0	3.0	1.0	1.0	4.0	9.0
2014-01-09	NaN	253.0	NaN	34.0	1.0	2.0	5.0	1.0	1.0	3.0	5.0
2014-01-10	NaN	224.0	NaN	16.0	4.0	1.0	10.0	NaN	NaN	1.0	4.0
2014-01-11	NaN	3.0	NaN	NaN	NaN	NaN	50.0	NaN	NaN	NaN	NaN
2014-01-12	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	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	6.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2014-02-02	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2014-02-03	NaN	345.0	NaN	29.0	6.0	NaN	4.0	NaN	1.0	4.0	7.0
2014-02-04	NaN	298.0	NaN	37.0	4.0	1.0	3.0	1.0	2.0	4.0	4.0
2014-02-05	NaN	316.0	NaN	23.0	4.0	NaN	2.0	2.0	NaN	2.0	9.0
2014-02-06	NaN	281.0	NaN	29.0	NaN	1.0	3.0	2.0	3.0	3.0	10.0
2014-02-07	NaN	247.0	NaN	21.0	6.0	2.0	4.0	1.0	NaN	NaN	9.0
2014-02-08	NaN	3.0	NaN	1.0	NaN	NaN	1.0	NaN	NaN	NaN	NaN
2014-02-09	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN
2014-02-10	NaN	296.0	NaN	29.0	3.0	2.0	5.0	1.0	1.0	4.0	5.0
2014-02-11	NaN	271.0	NaN	21.0	2.0	NaN	2.0	1.0	NaN	2.0	5.0
2014-02-12	NaN	228.0	NaN	23.0	3.0	4.0	2.0	4.0	1.0	1.0	4.0
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	6.0	NaN	NaN	NaN	NaN	1.0	NaN	NaN	1.0	NaN
2014-03-02	NaN	1.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	2.0
2014-03-03	NaN	185.0	NaN	24.0	4.0	NaN	1.0	NaN	NaN	4.0	6.0
2014-03-04	NaN	220.0	NaN	27.0	5.0	1.0	3.0	NaN	NaN	1.0	5.0
2014-03-05	NaN	201.0	NaN	26.0	2.0	2.0	8.0	NaN	NaN	NaN	6.0
2014-03-06	NaN	231.0	NaN	31.0	2.0	NaN	2.0	2.0	3.0	4.0	7.0
2014-03-07	NaN	199.0	NaN	17.0	1.0	1.0	2.0	2.0	NaN	6.0	1.0

2014-03-08	NaN	6.0	NaN	1.0	NaN	NaN	NaN	1.0	NaN	NaN	NaN
2014-03-09	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2014-03-10	NaN	232.0	NaN	31.0	4.0	6.0	10.0	NaN	NaN	2.0	10.0
2014-03-11	NaN	241.0	NaN	33.0	3.0	4.0	2.0	1.0	1.0	5.0	5.0
2014-03-12	NaN	221.0	NaN	24.0	1.0	NaN	17.0	NaN	NaN	4.0	4.0
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

CI_Cat	11
Open_Time	
2012-02-05	1.0
2012-03-12	NaN
2012-03-29	NaN
2012-07-17	NaN
2012-08-10	NaN
2012-08-15	NaN
2012-08-22	NaN
2012-08-29	NaN
2012-09-03	NaN
2012-09-21	NaN
2012-10-01	NaN
2012-10-02	NaN
2012-10-15	NaN
2012-10-18	NaN
2012-10-23	NaN
2012-11-21	1.0
2012-12-05	1.0
2012-12-07	NaN
2012-12-10	NaN
2012-12-24	NaN

```
2013-01-15   NaN
2013-01-22   NaN
2013-01-23   NaN
2013-01-30   1.0
2013-01-31   1.0
2013-02-04   NaN
2013-02-06   NaN
2013-02-08   NaN
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
2013-03-01   1.0
2013-03-04   1.0
2013-03-05   NaN
2013-03-11   NaN
2013-03-12   1.0
2013-03-15   NaN
2013-03-21   NaN
2013-03-26   1.0
2013-03-27   1.0
2013-04-03   NaN
2013-04-04   NaN
2013-04-05   NaN
2013-04-09   NaN
2013-04-10   NaN
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
2013-05-01   NaN
2013-05-03   NaN
2013-05-06   1.0
2013-05-07   NaN
2013-05-10   NaN
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
2013-06-03   1.0
2013-06-04   NaN
2013-06-05   1.0
2013-06-06   NaN
2013-06-10   NaN
2013-06-11   NaN
2013-06-12   NaN
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
2013-07-01   NaN
2013-07-02   1.0
2013-07-03   NaN
2013-07-04   1.0
2013-07-05   NaN
2013-07-08   NaN
2013-07-09   NaN
2013-07-10   NaN
2013-07-11   2.0
2013-07-12   1.0
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
2013-08-01   NaN
2013-08-05   NaN
2013-08-06   NaN
2013-08-07   NaN
2013-08-08   1.0
2013-08-09   NaN
```

```
2013-08-12   1.0
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
2013-09-02   3.0
2013-09-03   1.0
2013-09-04   1.0
2013-09-05   4.0
2013-09-06   2.0
2013-09-09   NaN
2013-09-10   3.0
2013-09-11   NaN
2013-09-12   6.0
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
2013-10-01  79.0
2013-10-02  71.0
2013-10-03  78.0
2013-10-04  51.0
2013-10-05   1.0
2013-10-06   1.0
2013-10-07  63.0
2013-10-08  77.0
2013-10-09  71.0
2013-10-10  66.0
```

```
2013-10-11  67.0
2013-10-12   NaN
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
2013-11-01  45.0
2013-11-02   NaN
2013-11-03   NaN
2013-11-04  76.0
2013-11-05  73.0
2013-11-06  67.0
2013-11-07  52.0
2013-11-08  38.0
2013-11-09   NaN
2013-11-10   1.0
2013-11-11  61.0
2013-11-12  60.0
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
2013-12-01   NaN
2013-12-02  73.0
2013-12-03  50.0
2013-12-04  61.0
2013-12-05  46.0
2013-12-06  41.0
2013-12-07   NaN
2013-12-08   NaN
2013-12-09  59.0
2013-12-10  37.0
2013-12-11  42.0
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  55.0
2014-01-03  62.0
2014-01-04   1.0
2014-01-05   NaN
2014-01-06  53.0
2014-01-07  57.0
2014-01-08  55.0
2014-01-09  61.0
2014-01-10  57.0
2014-01-11   1.0
2014-01-12   NaN
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   NaN
2014-02-02   NaN
2014-02-03  70.0
2014-02-04  91.0
2014-02-05  67.0
2014-02-06  82.0
2014-02-07  65.0
2014-02-08   NaN
2014-02-09   NaN
2014-02-10  75.0
2014-02-11  69.0
2014-02-12  68.0
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   1.0
2014-03-02   NaN
```

```
2014-03-03  58.0
2014-03-04  56.0
2014-03-05  53.0
2014-03-06  69.0
2014-03-07  51.0
2014-03-08   2.0
2014-03-09   NaN
2014-03-10  57.0
2014-03-11  51.0
2014-03-12  44.0
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
```

[151]:
```
final_df=pd.DataFrame(pivot_table)
```

[152]:
```
final_df.index=pd.to_datetime(final_df.index)
```

[153]:
```
final_df.fillna(0,inplace=True)
```

[154]:
```
len(final_df)
```

[154]: 331

[155]:
```
daily_data = final_df.resample('D', closed='right', label='right').asfreq()
```

[156]:
```
quaterly_data = daily_data.resample('Q').sum()
```

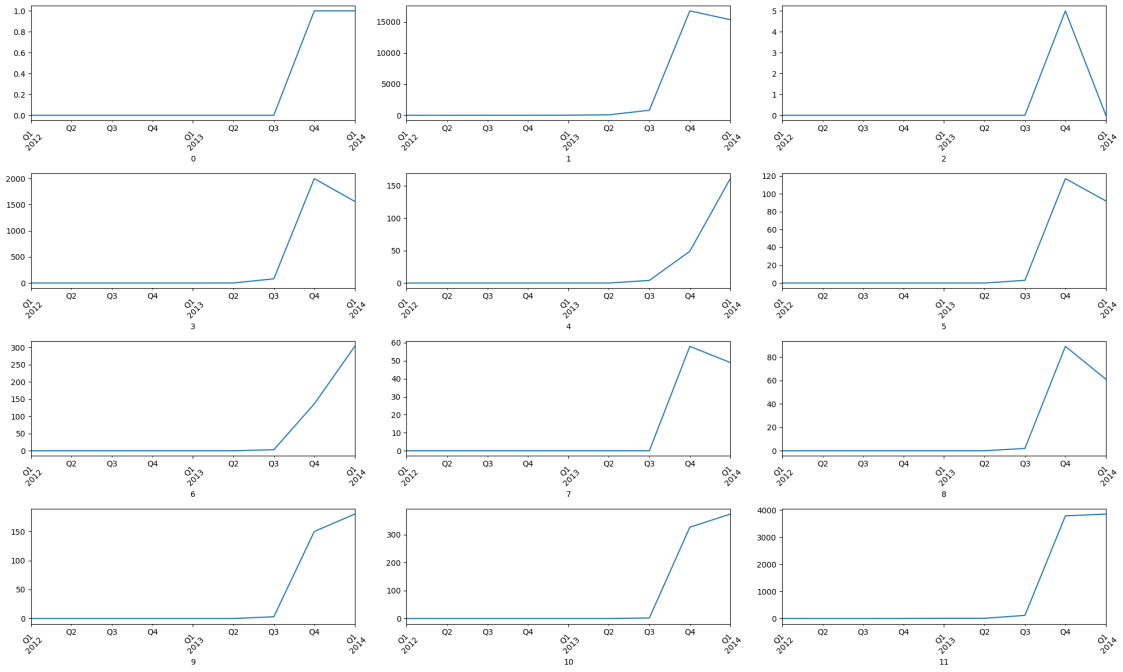
[157]:
```
quaterly_data
```

[157]:
```
CI_Cat          0        1    2       3      4      5      6     7     8      9   \
Open_Time
2012-03-31   0.0      2.0  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      8.0  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     16.0  0.0     0.0    0.0    0.0    0.0   0.0   0.0    0.0
2013-06-30   0.0     73.0  0.0     1.0    0.0    0.0    0.0   0.0   0.0    0.0
2013-09-30   0.0    811.0  0.0    80.0    4.0    3.0    3.0   0.0   2.0    3.0
2013-12-31   1.0  16754.0  5.0  1999.0   49.0  117.0  136.0  58.0  89.0  150.0
2014-03-31   1.0  15338.0  0.0  1563.0  161.0   92.0  303.0  49.0  61.0  180.0

CI_Cat          10      11
Open_Time
2012-03-31    0.0     1.0
2012-06-30    0.0     0.0
2012-09-30    0.0     0.0
2012-12-31    0.0     2.0
2013-03-31    0.0     8.0
2013-06-30    0.0     8.0
2013-09-30    2.0   115.0
2013-12-31  327.0  3792.0
2014-03-31  374.0  3856.0
```

[158]:
```
quaterly_data.shape
```

[158]:
```
(9, 12)
```

[159]:
```
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()
```



[160]:
```
quaterly_data
```

[160]:
```
CI_Cat          0         1    2         3      4      5      6     7     8      9   \
Open_Time
2012-03-31   0.0       2.0  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       8.0  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      16.0  0.0       0.0    0.0    0.0    0.0   0.0   0.0    0.0
2013-06-30   0.0      73.0  0.0       1.0    0.0    0.0    0.0   0.0   0.0    0.0
2013-09-30   0.0     811.0  0.0      80.0    4.0    3.0    3.0   0.0   2.0    3.0
2013-12-31   1.0   16754.0  5.0    1999.0   49.0  117.0  136.0  58.0  89.0  150.0
2014-03-31   1.0   15338.0  0.0    1563.0  161.0   92.0  303.0  49.0  61.0  180.0

CI_Cat          10      11
Open_Time
2012-03-31     0.0     1.0
2012-06-30     0.0     0.0
2012-09-30     0.0     0.0
2012-12-31     0.0     2.0
2013-03-31     0.0     8.0
2013-06-30     0.0     8.0
2013-09-30     2.0   115.0
2013-12-31   327.0  3792.0
2014-03-31   374.0  3856.0
```

2 Stationarity check

[161]:
```
from statsmodels.tsa.stattools import adfuller
```

[162]:
```
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 ")
```

[163]:
```
data_diff_1=quaterly_data.diff()
```

[164]:
```
data_diff_1.dropna()
```

[164]:

CI_Cat	0	1	2	3	4	5	6	7	8	9	\
Open_Time											
2012-06-30	0.0	-2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2012-09-30	0.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2012-12-31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2013-03-31	0.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2013-06-30	0.0	57.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
2013-09-30	0.0	738.0	0.0	79.0	4.0	3.0	3.0	0.0	2.0	3.0	
2013-12-31	1.0	15943.0	5.0	1919.0	45.0	114.0	133.0	58.0	87.0	147.0	
2014-03-31	0.0	-1416.0	-5.0	-436.0	112.0	-25.0	167.0	-9.0	-28.0	30.0	

CI_Cat	10	11
Open_Time		
2012-06-30	0.0	-1.0
2012-09-30	0.0	0.0
2012-12-31	0.0	2.0
2013-03-31	0.0	6.0
2013-06-30	0.0	0.0
2013-09-30	2.0	107.0
2013-12-31	325.0	3677.0
2014-03-31	47.0	64.0

[165]:
```
perform_adf_test(data_diff_1.dropna())
```

```
0 is  not stationary
```

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

[166]:
```
data_diff_2=quaterly_data.diff().diff()
```

[167]:
```
data_diff_3=quaterly_data.diff().diff().diff()
```

[168]:
```
perform_adf_test(data_diff_2.dropna())
```

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

2.1 # 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

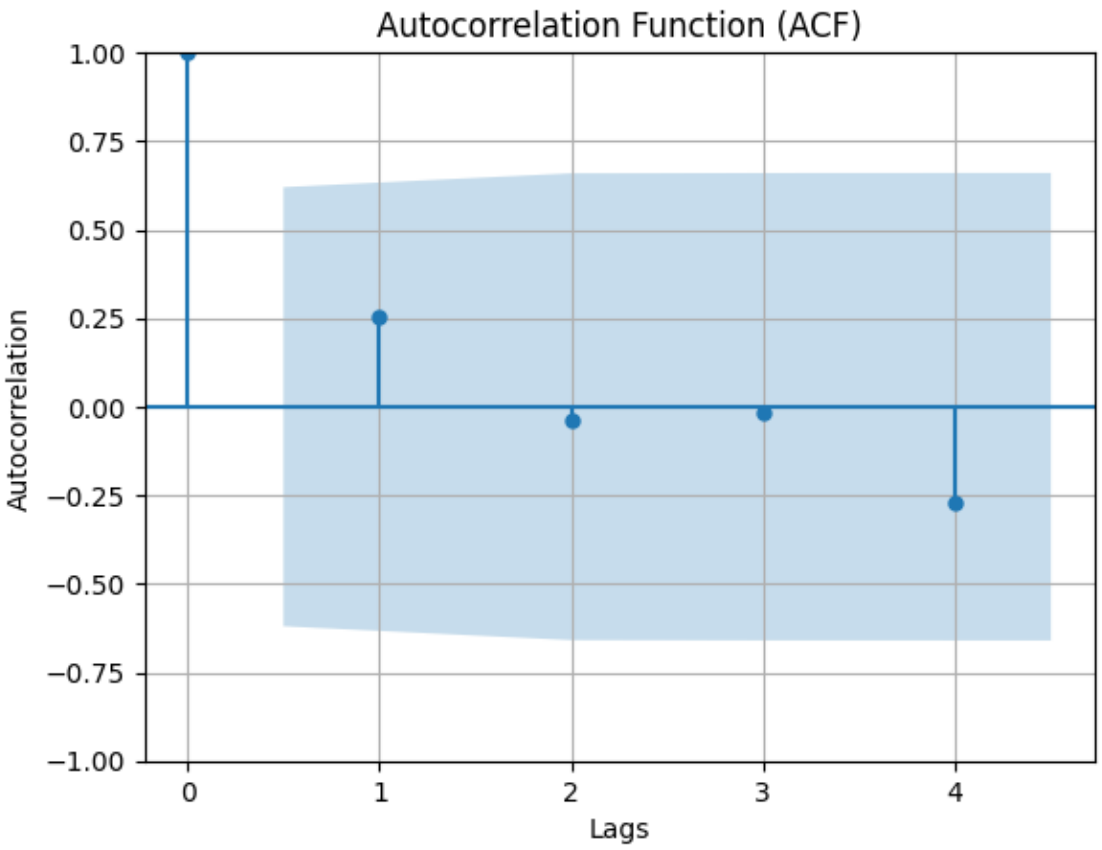
---

[169]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

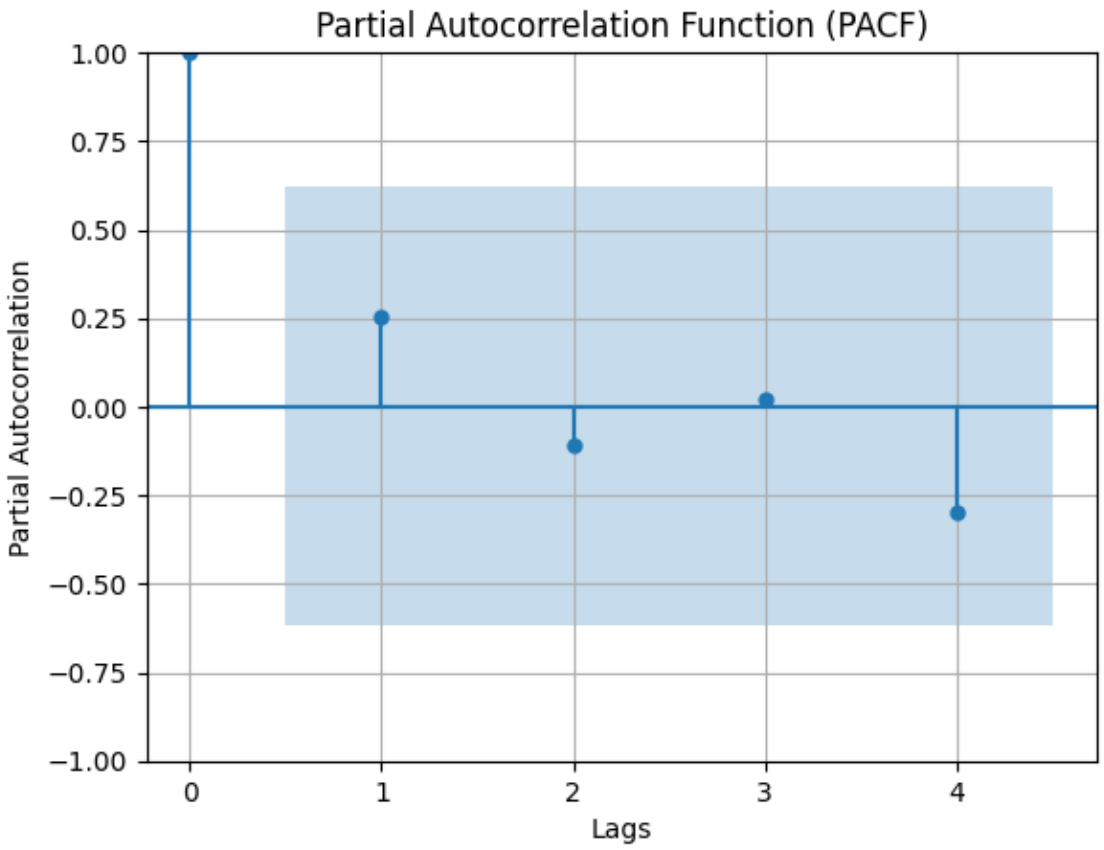
[170]:
```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import matplotlib.pyplot as plt

# Example data (replace with your actual data)
quaterly_data = [pd.Series(np.random.rand(10)) for _ in range(20)]  # Replace␣
 ↪with your actual data

# Determine the sample size
sample_size = len(quaterly_data[10])
```

```
# Set lags to be less than half the sample size
max_lags = sample_size // 2 - 1

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

```
<Figure size 1200x600 with 0 Axes>
```



<Figure size 1200x600 with 0 Axes>



2.2 # 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

2.3 2. q=1 —-> acf_plot

3 Arima model

3.1 Arima model forecasts and forecast plots

[171]:
```
import pandas as pd
import numpy as np
from statsmodels.tsa.arima.model import ARIMA

# Sample quarterly data creation (replace with your actual data)
data = {f'Column_{i}': np.random.rand(40) for i in range(10)}
quaterly_data = pd.DataFrame(data)  # Replace with your actual data
```

```
arima_forecast = {}
steps = 12

for column in quaterly_data.columns:
    model = ARIMA(quaterly_data[column], order=(1, 2, 1))  # ARIMA(1, 2, 1)␣
 ↪model
    model_fit = model.fit()
    forecast = model_fit.forecast(steps=steps)
    arima_forecast[column] = forecast

# Display the forecasts
for key, value in arima_forecast.items():
    print(f"Forecast for {key}:")
    print(value)
```

```
Forecast for Column_0:
40    0.243463
41    0.818967
42    0.376833
43    0.724945
44    0.459392
45    0.670380
46    0.511310
47    0.639609
48    0.544752
49    0.623186
50    0.567051
51    0.615415
Name: predicted_mean, dtype: float64
Forecast for Column_1:
40    0.185251
41    0.101386
42    0.132441
43    0.093919
44    0.097522
45    0.075620
46    0.069160
47    0.053351
48    0.043202
49    0.029626
50    0.018125
51    0.005368
Name: predicted_mean, dtype: float64
Forecast for Column_2:
40    0.276294
41    0.219935
```

```
42    0.230758
43    0.224590
44    0.222720
45    0.219762
46    0.217080
47    0.214328
48    0.211594
49    0.208855
50    0.206117
51    0.203379
Name: predicted_mean, dtype: float64
Forecast for Column_3:
40    0.395209
41    0.277063
42    0.314005
43    0.286832
44    0.286165
45    0.274540
46    0.267445
47    0.258478
48    0.250284
49    0.241771
50    0.233390
51    0.224954
Name: predicted_mean, dtype: float64
Forecast for Column_4:
40    0.378726
41    0.402905
42    0.393317
43    0.400881
44    0.399733
45    0.403010
46    0.404039
47    0.406210
48    0.407802
49    0.409687
50    0.411423
51    0.413235
Name: predicted_mean, dtype: float64
Forecast for Column_5:
40    0.548923
41    0.519111
42    0.521939
43    0.517116
44    0.514086
45    0.510636
46    0.507285
47    0.503910
```

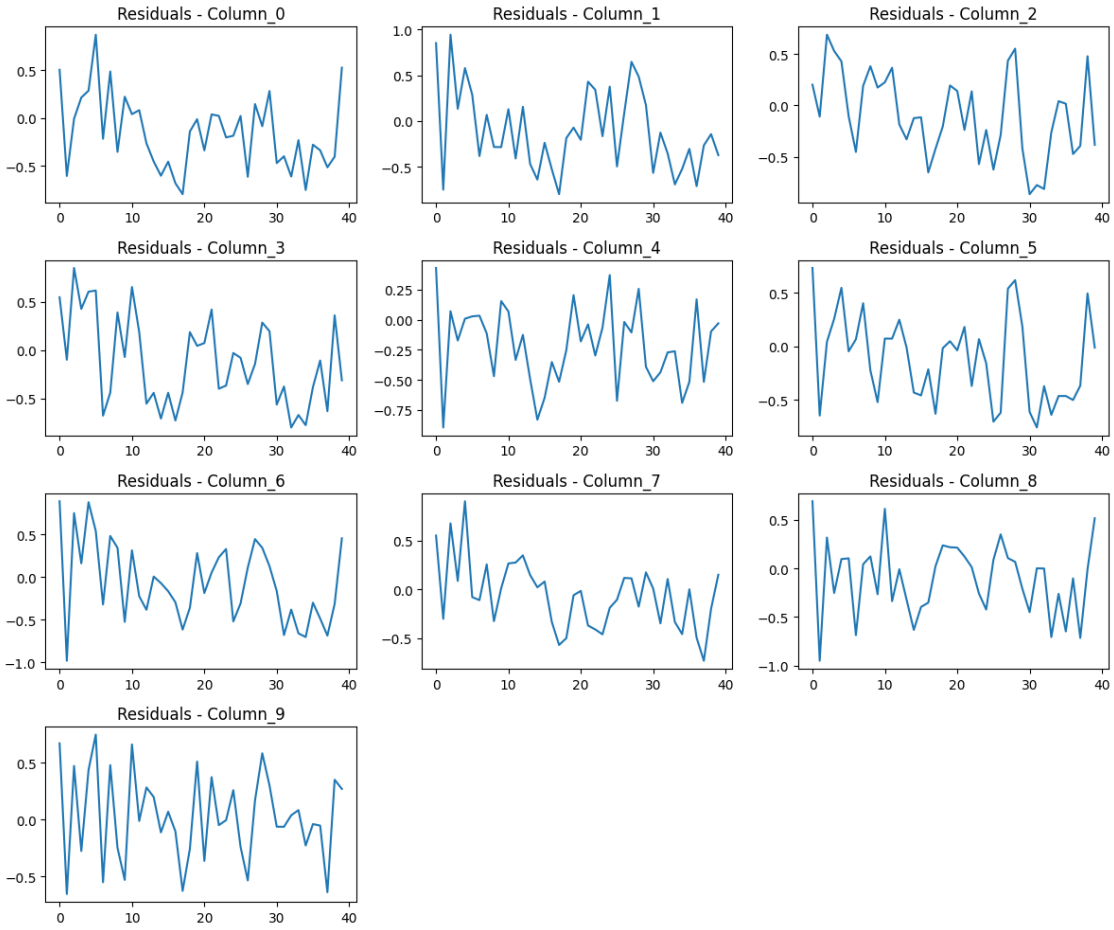
```
48    0.500541
49    0.497171
50    0.493801
51    0.490431
Name: predicted_mean, dtype: float64
Forecast for Column_6:
40    0.558513
41    0.711857
42    0.646020
43    0.676430
44    0.664576
45    0.671281
46    0.669836
47    0.671971
48    0.672533
49    0.673786
50    0.674735
51    0.675818
Name: predicted_mean, dtype: float64
Forecast for Column_7:
40    0.463817
41    0.510990
42    0.490611
43    0.494854
44    0.490123
45    0.488663
46    0.486010
47    0.483792
48    0.481416
49    0.479098
50    0.476758
51    0.474426
Name: predicted_mean, dtype: float64
Forecast for Column_8:
40    0.711993
41    0.847956
42    0.793507
43    0.831924
44    0.825050
45    0.840264
46    0.844706
47    0.854402
48    0.861535
49    0.869918
50    0.877691
51    0.885762
Name: predicted_mean, dtype: float64
Forecast for Column_9:
```

```
40    0.757914
41    0.745860
42    0.759104
43    0.759378
44    0.766301
45    0.769815
46    0.775077
47    0.779443
48    0.784269
49    0.788859
50    0.793569
51    0.798218
Name: predicted_mean, dtype: float64
```

3.2 # 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.

[172]:
```
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()
```



[173]:
```
arima_forecast_data=pd.DataFrame(arima_forecast)
```

[174]:
```
arima_forecast_data=arima_forecast_data.astype(int)
arima_forecast_data
```

[174]:

	Column_0	Column_1	Column_2	Column_3	Column_4	Column_5	Column_6	\
40	0	0	0	0	0	0	0	
41	0	0	0	0	0	0	0	
42	0	0	0	0	0	0	0	
43	0	0	0	0	0	0	0	
44	0	0	0	0	0	0	0	
45	0	0	0	0	0	0	0	
46	0	0	0	0	0	0	0	
47	0	0	0	0	0	0	0	
48	0	0	0	0	0	0	0	
49	0	0	0	0	0	0	0	
50	0	0	0	0	0	0	0	
51	0	0	0	0	0	0	0	

	Column_7	Column_8	Column_9
40	0	0	0
41	0	0	0
42	0	0	0
43	0	0	0
44	0	0	0
45	0	0	0
46	0	0	0
47	0	0	0
48	0	0	0
49	0	0	0
50	0	0	0
51	0	0	0

4 Sarimax model

[175]:
```
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
```

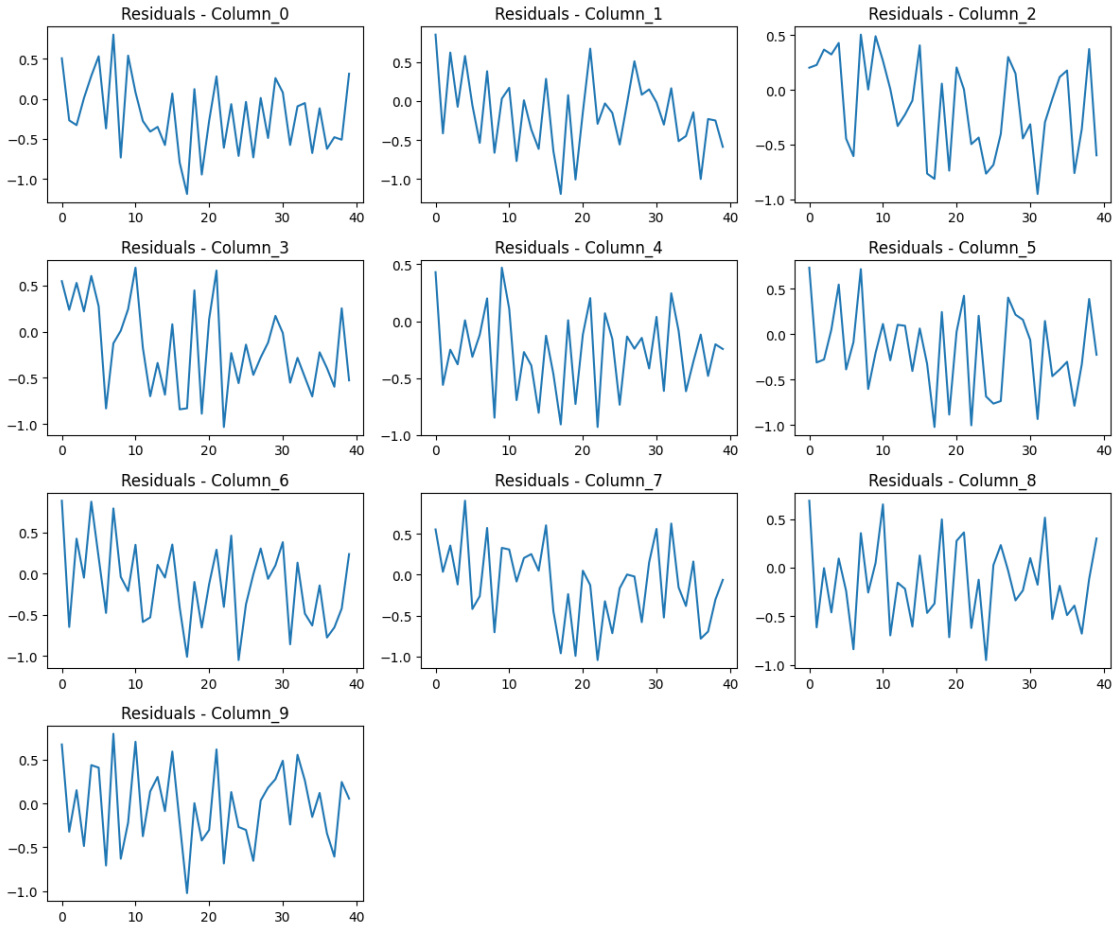
```
/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py:607:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

4.1 # 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.

[176]:
```
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()
```



[177]:
```
sarima_forecast_data=pd.DataFrame(sarima_forecast)
```

[178]:
```
sarima_forecast_data=sarima_forecast_data.astype(int)
sarima_forecast_data
```

[178]:

	Column_0	Column_1	Column_2	Column_3	Column_4	Column_5	Column_6	\
40	0	0	0	0	0	0	0	
41	0	0	0	0	0	0	0	
42	0	0	0	0	0	0	0	
43	0	0	0	0	0	0	0	
44	0	0	0	0	0	0	0	
45	0	0	0	0	0	0	0	
46	0	0	0	0	0	0	0	

```
47          0          0          0          0          0          0          0
48          0          0          0          0          0          0          0
49          0          0          0          0          0          0          0
50          0          0          0          0          0          0          0
51          0          0          0          0          0          0          0

    Column_7  Column_8  Column_9
40         0         0         0
41         0         0         0
42         0         0         0
43         0         0         1
44         0         0         0
45         0         0         0
46         0         0         0
47         0         0         0
48         0         0         0
49         0         0         0
50         0         0         0
51         0         0         0
```

4.2 # 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.
-

4.3 also evaluated the models based on their residuals, comparing to the other p,d,q combinations (1,2,1) is giving good result

4.4 # TASK 3

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

[179]:
```
data_3=df.copy()
```

[180]:
```
data_3.head()
```

[180]:
```
   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 2012-02-05 13:32:00
1         1        611                   33 2012-03-12 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 2012-08-10 11:01:00

        Resolved_Time          Close_Time  No_of_Related_Interactions  \
0 2013-11-04 13:50:00 2013-11-04 13:51:00                           1
1 2013-12-02 12:36:00 2013-12-02 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 2013-11-08 13:55:00 2013-11-08 13:55:00                           1

   Handle_Time_hrs_conv
0          15312.316667
1          15116.866667
2          15722.616667
3          11637.700000
4          10922.900000
```

[181]:
```
data_3=data_3.drop(['Open_Time','Resolved_Time','Close_Time'],axis=1)
```

[182]:
```
X1=data_3.drop(['Priority','CI_Cat','Urgency'],axis=1)
```

[183]:
```
X1.head()
```

[183]:
```
   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          15312.316667
1                   33                           1          15116.866667
2                    3                           1          15722.616667
3                   13                           1          11637.700000
4                    2                           1          10922.900000
```

[184]:
```
y1=data_3['Priority']
```

[185]:
```
y1.head()
```

[185]:
```
0    4
1    3
2    4
```

```
3    4
4    4
Name: Priority, dtype: int64
```

[186]:
```
y2=data_3['CI_Cat']
```

5 Function for model selection Task 3

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 —

[187]:
```
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,average='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,model_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,model_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)
```

[188]:
```
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3,␣
↪random_state=42,stratify=y1)
```

[189]:
```
for i in models:
    model_selction_2(i)
```

```
<class 'sklearn.linear_model._logistic.LogisticRegression'>
metrics on test data
[[   0    0    0    1    0]
 [   0    3    0   76  130]
 [   0    4    0 1264  329]
 [   0    3    0 5833 1393]
 [   0    0    0 2414 2532]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.30      0.01      0.03       209
           3       0.00      0.00      0.00      1597
           4       0.61      0.81      0.69      7229
           5       0.58      0.51      0.54      4946

    accuracy                           0.60     13982
   macro avg       0.30      0.27      0.25     13982
weighted avg       0.52      0.60      0.55     13982
```

```
metrics on train data
[[    0     0     0     2     0]
 [    0     2     0   174   312]
 [    0     6     0  2930   790]
 [    0     5     0 13584  3279]
 [    0     0     0  5704  5835]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         2
           2       0.15      0.00      0.01       488
           3       0.00      0.00      0.00      3726
           4       0.61      0.81      0.69     16868
           5       0.57      0.51      0.54     11539

    accuracy                           0.60     32623
   macro avg       0.27      0.26      0.25     32623
weighted avg       0.52      0.60      0.55     32623

==============================
<class 'sklearn.tree._classes.DecisionTreeClassifier'>
metrics on test data
[[   1    0    0    0    0]
 [   0  207    2    0    0]
 [   0    4 1564   27    2]
 [   0    0   16 7195   18]
 [   0    0    0   27 4919]]


              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      0.99      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    5 7200   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 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._bagging.BaggingClassifier'>
metrics on test data
[[   1    0    0    0    0]
 [   0  207    2    0    0]
 [   0    2 1567   26    2]
 [   0    0    9 7202   18]
 [   0    0    0    8 4938]]


              precision    recall  f1-score   support

           1       1.00      1.00      1.00         1
           2       0.99      0.99      0.99       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       0.99      0.99      0.99     13982
weighted avg       1.00      1.00      1.00     13982

metrics on train data
[[    2     0     0     0     0]
 [    0   487     1     0     0]
 [    0     0  3722     4     0]
 [    0     0     3 16859     6]
 [    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.neighbors._classification.KNeighborsClassifier'>
metrics on test data
[[   0    0    0    1    0]
 [   0  137   34   36    2]
 [   0   27 1059  396  115]
 [   0   20  292 6404  513]
 [   0    5   85  752 4104]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.72      0.66      0.69       209
           3       0.72      0.66      0.69      1597
           4       0.84      0.89      0.86      7229
           5       0.87      0.83      0.85      4946

    accuracy                           0.84     13982
   macro avg       0.63      0.61      0.62     13982
weighted avg       0.84      0.84      0.84     13982

metrics on train data
[[    0     0     1     1     0]
 [    0   355    63    55    15]
 [    0    45  2889   607   185]
 [    0    18   482 15557   811]
 [    0     6   155  1217 10161]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         2
           2       0.84      0.73      0.78       488
           3       0.80      0.78      0.79      3726
           4       0.89      0.92      0.91     16868
           5       0.91      0.88      0.89     11539

    accuracy                           0.89     32623
   macro avg       0.69      0.66      0.67     32623
weighted avg       0.89      0.89      0.89     32623

==============================
<class 'sklearn.naive_bayes.GaussianNB'>
metrics on test data
```

```
[[   0    1    0    0    0]
 [   0  206    3    0    0]
 [   0    4 1556   35    2]
 [   0    1    3 7151   74]
 [   0    0    0    4 4942]]


              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.97      0.99      0.98       209
           3       1.00      0.97      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     5  3651    70     0]
 [    0     6     5 16678   179]
 [    0     0     0    13 11526]]


              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  191   18]
 [   0    0    0 1240  357]
 [   0    0    0 6164 1065]
 [   0    0    0 2841 2105]]
```

```
              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      0.85      0.70      7229
           5       0.59      0.43      0.50      4946

    accuracy                           0.59     13982
   macro avg       0.24      0.26      0.24     13982
weighted avg       0.52      0.59      0.54     13982

metrics on train data
[[    0     0     0     1     1]
 [    0     0     0   452    36]
 [    0     0     0  2876   850]
 [    0     0     0 14499  2369]
 [    0     0     0  6564  4975]]


              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       0.60      0.43      0.50     11539

    accuracy                           0.60     32623
   macro avg       0.24      0.26      0.24     32623
weighted avg       0.52      0.60      0.54     32623

==============================
<class 'sklearn.ensemble._gb.GradientBoostingClassifier'>
metrics on test data
[[   1    0    0    0    0]
 [   0  207    2    0    0]
 [   0    0 1566   29    2]
 [   0    0    8 7187   34]
 [   0    0    1    2 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      0.99      0.99      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  3693    33     0]
 [    0     0     3 16811    54]
 [    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      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

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

[190]:
```
summary_1=pd.DataFrame(model_summary_1).
↪sort_values('f1_score_test',ascending=False).drop('model_name_test',axis=1)
```

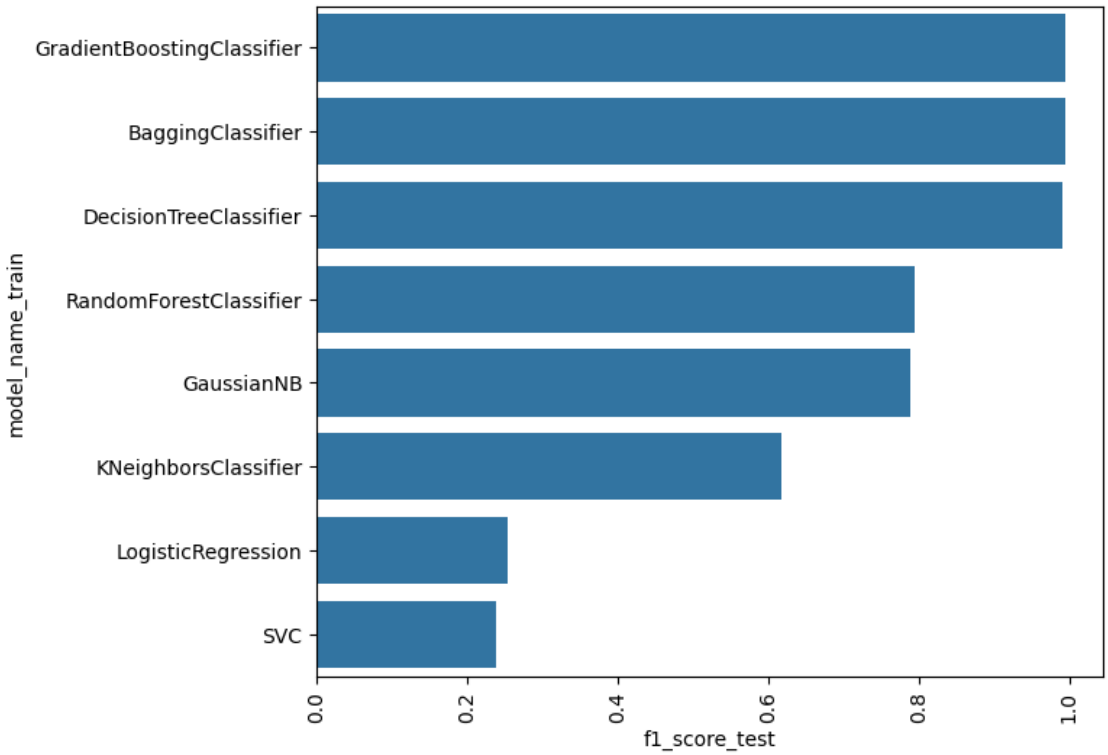
[191]:
```
summary_1
```

[191]:

	model_name_train	f1_score_train	recall_score_train	accuracy_score_train	f1_score_test	recall_score_test	accuracy_score_test
7	GradientBoostingClassifier	0.997797	0.997143	0.997211	0.994595	0.992921	0.994421
3	BaggingClassifier	0.999451	0.999269	0.999571	0.994095	0.993259	0.995208
1	DecisionTreeClassifier	1.000000	1.000000				
2	RandomForestClassifier	1.000000	1.000000				
5	GaussianNB	0.990461	0.990217				
4	KNeighborsClassifier	0.674015	0.661136				
0	LogisticRegression	0.247276	0.263017				
6	SVC	0.241220	0.258141				

1	1.000000	0.991768	0.991921	0.993134
2	1.000000	0.794759	0.793276	0.995423
5	0.991233	0.789520	0.789675	0.990917
4	0.887779	0.618261	0.606852	0.837076
0	0.595316	0.252773	0.266634	0.598484
6	0.596941	0.238731	0.255655	0.591403

[192]:
```
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()
```



5.1 ## 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

[193]:
```
#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 1566   29    2]
 [   0    0    8 7187   34]
 [   0    0    1    2 4943]]


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      0.99      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

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

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

5.3 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 —

[194]:
```
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,average='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,model_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,model_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)
```

[195]:
```
X_train, X_test, y_train, y_test = train_test_split(X1, y2, test_size=0.3,␣
 ↪random_state=42,stratify=y2)
```

[196]:
```
for i in models:
    model_selction_3(i)
```

```
<class 'sklearn.linear_model._logistic.LogisticRegression'>
metrics on test data
[[9171    0   56    0    0    0    0    0    0    0  676]
 [   1    0    0    0    0    0    0    0    0    0    0]
 [1091    0    2    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]
 [  99    0    1    0    0    0    0    0    0    0    0]
 [ 211    0    0    0    0    0    0    0    0    0    0]
 [2254    0   18    0    0    0    0    0    0    0   63]]


              precision    recall  f1-score   support

           1       0.70      0.93      0.80      9903
           2       0.00      0.00      0.00         1
           3       0.03      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       0.09      0.03      0.04      2335

    accuracy                           0.66     13982
   macro avg       0.07      0.09      0.08     13982
weighted avg       0.51      0.66      0.57     13982

metrics on train data
[[    0     2     0     0     0     0     0     0     0     0     0     0]
 [    0 21342     0   131     0     0     0     0     0     0     0  1634]
 [    0     3     0     1     0     0     0     0     0     0     0     0]
 [    0  2547     0     3     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  5249     0    35     0     0     0     0     0     0     0   163]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.70      0.92      0.79     23107
           2       0.00      0.00      0.00         4
           3       0.02      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       0.09      0.03      0.05      5447

    accuracy                           0.66     32623
   macro avg       0.07      0.08      0.07     32623
weighted avg       0.51      0.66      0.57     32623

==============================
<class 'sklearn.tree._classes.DecisionTreeClassifier'>
metrics on test data
[[   0    0    0    0    0    0    0    0    0    0    0    0]
 [   0 9813    0    0    0    0    0    0    0    1    0   89]
```

```
[   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    0    1    0    2   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   64    0    0    0    0    0    0    0    0    0 2271]]
```

	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	0.84	0.89	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	0.96	0.97	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
[[9821    0    0    0    0    0    0    0    0    0   82]
 [   0    1    0    0    0    0    0    0    0    0    0]
 [   7    0 1086    0    0    0    0    0    0    0    0]
 [   1    0    0   63    0    0    0    0    0    0    0]
 [   0    0    0    0   64    0    0    0    0    0    0]
 [   2    0    3    0    0  128    0    0    0    0    0]
 [   5    0    2    0    1    2   21    0    1    0    0]
 [   0    0    0    0    0    0    0   46    0    0    0]
 [   0    0    0    0    0    0    0    0  100    0    0]
 [   1    0    1    0    0    2    0    0    0  207    0]
 [ 188    0    0    0    0    0    0    0    0    0 2147]]


              precision    recall  f1-score   support

           1       0.98      0.99      0.99      9903
           2       1.00      1.00      1.00         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      0.96      0.97       133
           7       1.00      0.66      0.79        32
           8       1.00      1.00      1.00        46
           9       0.99      1.00      1.00       100
          10       1.00      0.98      0.99       211
          11       0.96      0.92      0.94      2335

    accuracy                           0.98     13982
```

```
   macro avg       0.99      0.95      0.97     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 9854    0    0    0    0    0    0    0    0    0   49]
 [   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    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   67    0    0    0    0    0    0    0    0    0 2268]]
```

	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	0.90	0.81	0.85	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	0.97	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

```
metrics on train data
[[    2     0     0     0     0     0     0     0     0     0     0     0]
 [    0 23103     0     0     0     0     0     0     0     0     0     4]
 [    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     1     0    74     0     0     0     0]
 [    0     0     0     0     0     0     0     0   106     0     0     0]
 [    0     1     0     0     0     0     0     0     0   232     0     0]
 [    0     0     0     0     0     0     0     0     0     0   492     0]
 [    0    24     0     0     0     0     0     0     0     0     0  5423]]
```

	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      0.99      0.99        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.neighbors._classification.KNeighborsClassifier'>
metrics on test data
[[9647    0   61    1    6    7    1    6    8   11  155]
 [   0    0    1    0    0    0    0    0    0    0    0]
 [ 122    0  949    0   11    4    0    0    1    2    4]
 [   6    0    2   56    0    0    0    0    0    0    0]
 [   8    0    3    0   53    0    0    0    0    0    0]
 [  22    0    7    0    1  100    1    0    0    2    0]
 [  14    0    1    0    0    3   12    0    2    0    0]
 [   5    0    0    0    0    0    0   41    0    0    0]
 [  19    0    0    0    0    1    1    0   76    0    3]
 [  26    0    2    0    0    0    0    1    2  180    0]
 [ 280    0   16    0    0    0    0    4    2    2 2031]]


              precision    recall  f1-score   support

           1       0.95      0.97      0.96      9903
           2       0.00      0.00      0.00         1
           3       0.91      0.87      0.89      1093
           4       0.98      0.88      0.93        64
           5       0.75      0.83      0.79        64
           6       0.87      0.75      0.81       133
           7       0.80      0.38      0.51        32
           8       0.79      0.89      0.84        46
           9       0.84      0.76      0.80       100
          10       0.91      0.85      0.88       211
          11       0.93      0.87      0.90      2335

    accuracy                           0.94     13982
   macro avg       0.79      0.73      0.75     13982
weighted avg       0.94      0.94      0.94     13982

metrics on train data
[[    0     2     0     0     0     0     0     0     0     0     0     0]
 [    0 22742     0    75     1     5    14     5     6     4    13   242]
```

```
 [    0     2     1     1     0     0     0     0     0     0     0     0]
 [    0   216     0  2309     0     6     6     0     0     2     4     7]
 [    0     7     0     1   140     0     1     1     0     0     0     0]
 [    0    15     0     7     0   126     0     0     0     0     0     0]
 [    0    31     0    10     0     0   258     2     0     3     5     0]
 [    0    23     0     1     0     0     1    49     0     1     0     0]
 [    0    23     0     0     0     0     0     0    82     0     0     1]
 [    0    35     0     1     0     0     3     0     0   187     1     6]
 [    0    50     0     5     0     0     1     0     0     3   432     1]
 [    0   467     0    23     0     0     0     0     6     4     1  4946]]
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	2
1	0.96	0.98	0.97	23107
2	1.00	0.25	0.40	4
3	0.95	0.91	0.93	2550
4	0.99	0.93	0.96	150
5	0.92	0.85	0.88	148
6	0.91	0.83	0.87	309
7	0.86	0.65	0.74	75
8	0.87	0.77	0.82	106
9	0.92	0.80	0.86	233
10	0.95	0.88	0.91	492
11	0.95	0.91	0.93	5447
accuracy			0.96	32623
macro avg	0.86	0.73	0.77	32623
weighted avg	0.96	0.96	0.96	32623

```
==============================
<class 'sklearn.naive_bayes.GaussianNB'>
metrics on test data
[[3886   69  932   23    0  283    8 2899  100   44 1659]
 [   0    1    0    0    0    0    0    0    0    0    0]
 [ 118   16  848    0    0   39    1   18   17    8   28]
 [   3    1    0   60    0    0    0    0    0    0    0]
 [   0    0    0    0   64    0    0    0    0    0    0]
 [  24    2   32    1    0   70    3    0    0    0    1]
 [   9    0    3    0    0    2   14    2    1    1    0]
 [   2    0    0    0    0    0    0   44    0    0    0]
 [  23    6    0    0    0   11    0    0   45    2   13]
 [  33   31    1    0    0   13    0  123    1    9    0]
 [ 432    0   30    0    0    8    0   85   31   12 1737]]
```

	precision	recall	f1-score	support

```
           1       0.86      0.39      0.54      9903
           2       0.01      1.00      0.02         1
           3       0.46      0.78      0.58      1093
           4       0.71      0.94      0.81        64
           5       1.00      1.00      1.00        64
           6       0.16      0.53      0.25       133
           7       0.54      0.44      0.48        32
           8       0.01      0.96      0.03        46
           9       0.23      0.45      0.31       100
          10       0.12      0.04      0.06       211
          11       0.51      0.74      0.60      2335

    accuracy                           0.48     13982
   macro avg       0.42      0.66      0.42     13982
weighted avg       0.74      0.48      0.54     13982

metrics on train data
[[   2    0    0    0    0    0    0    0    0    0    0    0]
 [   0 8943  175 2252   51    0  617   31 6712  266   85 3975]
 [   0    0    4    0    0    0    0    0    0    0    0    0]
 [   0  279   41 1953    0    0   75    0   59   33   28   82]
 [   0    2    1    0  142    0    5    0    0    0    0    0]
 [   0    3    0    0    0  145    0    0    0    0    0    0]
 [   0   58   10   67    2    0  154   14    0    1    3    0]
 [   0   11    3    3    0    0    2   45    9    2    0    0]
 [   0    6    0    0    0    0    0    0   99    0    0    1]
 [   0   43   13    2    0    0   30    1    1   96   13   34]
 [   0   83   69    1    2    0   30    0  292    3   12    0]
 [   0 1034    1   64    0    0   20    2  216   59   26 4025]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       0.85      0.39      0.53     23107
           2       0.01      1.00      0.02         4
           3       0.45      0.77      0.57      2550
           4       0.72      0.95      0.82       150
           5       1.00      0.98      0.99       148
           6       0.17      0.50      0.25       309
           7       0.48      0.60      0.54        75
           8       0.01      0.93      0.03       106
           9       0.21      0.41      0.28       233
          10       0.07      0.02      0.04       492
          11       0.50      0.74      0.59      5447

    accuracy                           0.48     32623
```

```
   macro avg       0.46      0.69      0.47     32623
weighted avg       0.74      0.48      0.54     32623

==============================
<class 'sklearn.svm._classes.SVC'>
metrics on test data
[[9903    0    0    0    0    0    0    0    0    0    0]
 [   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  5447     0     0     0     0     0     0     0     0     0     0]]


              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       0.00      0.00      0.00      5447

    accuracy                           0.71     32623
   macro avg       0.06      0.08      0.07     32623
weighted avg       0.50      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 9815    0    0    0    0    0    0    0    0    1   87]
 [   0    0    1    0    0    0    0    0    0    0    0    0]
 [   0    3    0 1090    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    3    1    0    0   22    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    0    0    1    0    1    0    0    0    0  209    0]
 [   0  243    0    0    0    0    0    0    0    0    0 2092]]


              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      0.69      0.81        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      0.90      0.93      2335

    accuracy                           0.98     13982
   macro avg       0.91      0.88      0.89     13982
weighted avg       0.98      0.98      0.97     13982

metrics on train data
[[    2     0     0     0     0     0     0     0     0     0     0     0]
 [    0 22880     0     0     0     0     0     0     0     0     0   227]
 [    0     0     4     0     0     0     0     0     0     0     0     0]
 [    0     7     0  2543     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   567     0     0     0     0     0     0     0     0     0  4880]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       0.98      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      0.90      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

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

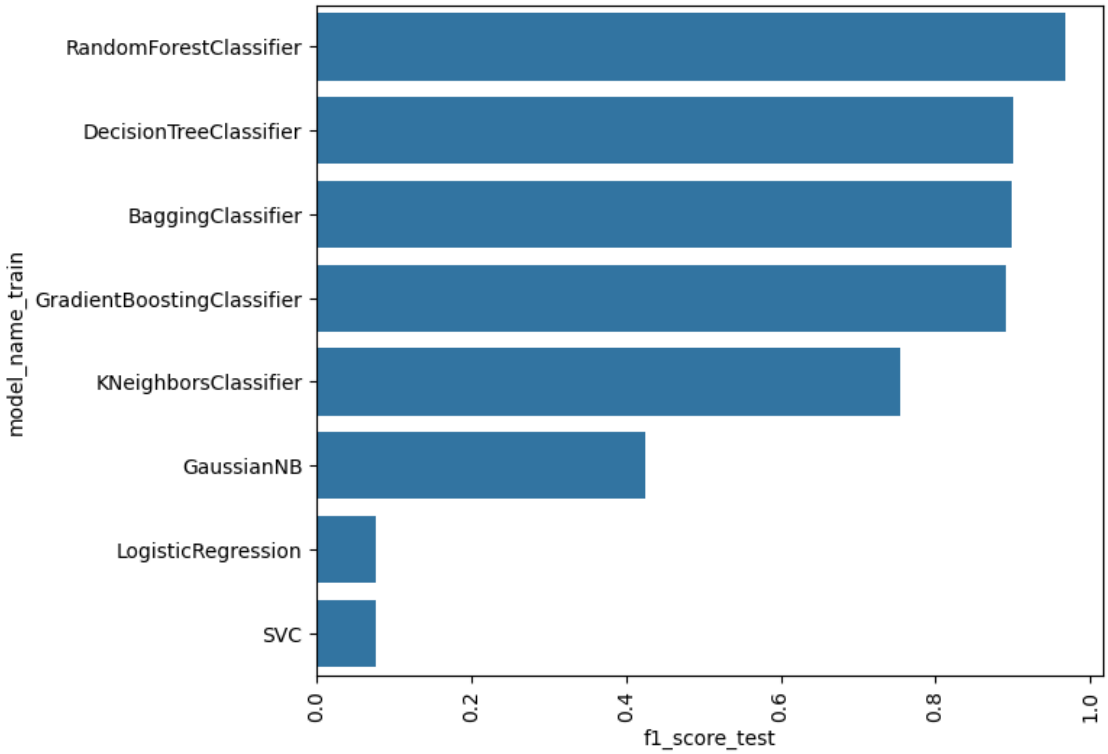
[197]:
```
summary_3=pd.DataFrame(model_summary_3).
↪sort_values('f1_score_test',ascending=False).drop('model_name_test',axis=1)
```

[198]:
```
summary_3
```

[198]:

	model_name_train	f1_score_train	recall_score_train	accuracy_score_train	f1_score_test	recall_score_test	accuracy_score_test
2	RandomForestClassifier	1.000000	1.000000	1.000000	0.968078	0.953534	0.978687
1	DecisionTreeClassifier	1.000000	1.000000	1.000000	0.900216	0.897810	0.988271
3	BaggingClassifier	0.998714	0.998150	0.999080	0.898655	0.895367	0.990845
7	GradientBoostingClassifier	0.992183	0.990278	0.975447	0.890596	0.878265	0.975111
4	KNeighborsClassifier	0.772945	0.731241	0.958587	0.753734	0.731509	0.940137
5	GaussianNB	0.470814	0.690583	0.478803	0.424749	0.660240	0.484766
0	LogisticRegression	0.070095	0.079560	0.659289	0.076318	0.086809	0.660564
6	SVC	0.069104	0.083333	0.708304	0.075384	0.090909	0.708268

[199]:
```
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()
```



5.4 ## 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

[200]:
```
#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 9862    0    0    0    0    0    0    0    0    0   41]
 [   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    0    0    0    2   28    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   57    0    0    0    0    0    0    0    0    0 2278]]


classification report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.99      1.00      1.00      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       1.00      1.00      1.00        64
           6       0.99      0.99      0.99       133
           7       0.93      0.88      0.90        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

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

5.5 # Task 4

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

[201]:
```
data_4=df.copy()
```

[202]:
```
data_4.head()
```

[202]:
```
   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 2012-02-05 13:32:00
1         1        611                   33 2012-03-12 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 2012-08-10 11:01:00

        Resolved_Time          Close_Time  No_of_Related_Interactions  \
0 2013-11-04 13:50:00 2013-11-04 13:51:00                           1
1 2013-12-02 12:36:00 2013-12-02 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 2013-11-08 13:55:00 2013-11-08 13:55:00                           1

   Handle_Time_hrs_conv
0          15312.316667
1          15116.866667
2          15722.616667
3          11637.700000
4          10922.900000
```

[203]:
```
data_4['Category'].value_counts()
```

[203]:
```
Category
1    37748
3     8845
0       11
2        1
Name: count, dtype: int64
```

[204]:
```
data_4.loc[data_4['Category']==2]
```

[204]:
```
       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 2014-01-07 14:46:00 2014-01-07 14:46:00                           1

       Handle_Time_hrs_conv
24520            170.883333
```

[205]:
```
data_4.drop(data_4.loc[data_4['Category']==2].index,inplace=True)
```

[206]:
```
X_4=data_4.drop(['Category','Open_Time','Resolved_Time','Close_Time'],axis=1)
y_4=data_4['Category']
```

5.6 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 —

[207]:
```
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,average='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,model_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_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,model_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)
```

[208]:
```
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)
```

[209]:
```
for i in models:
    model_selction_4(i)
```

```
<class 'sklearn.linear_model._logistic.LogisticRegression'>
metrics on test data
[[    0     3     0]
 [    0 11076   249]
 [    0  2556    98]]


              precision    recall  f1-score   support
```

```
           0       0.00      0.00      0.00         3
           1       0.81      0.98      0.89     11325
           3       0.28      0.04      0.07      2654

    accuracy                           0.80     13982
   macro avg       0.36      0.34      0.32     13982
weighted avg       0.71      0.80      0.73     13982

metrics on train data
[[    0     8     0]
 [    0 25867   556]
 [    0  5980   211]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         8
           1       0.81      0.98      0.89     26423
           3       0.28      0.03      0.06      6191

    accuracy                           0.80     32622
   macro avg       0.36      0.34      0.32     32622
weighted avg       0.71      0.80      0.73     32622

==============================
<class 'sklearn.tree._classes.DecisionTreeClassifier'>
metrics on test data
[[    3     0     0]
 [    0 11127   198]
 [    2   228  2424]]


              precision    recall  f1-score   support

           0       0.60      1.00      0.75         3
           1       0.98      0.98      0.98     11325
           3       0.92      0.91      0.92      2654

    accuracy                           0.97     13982
   macro avg       0.83      0.97      0.88     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 11226    99]
 [    0   251  2403]]


              precision    recall  f1-score   support

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

    accuracy                           0.97     13982
   macro avg       0.98      0.97      0.97     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._bagging.BaggingClassifier'>
metrics on test data
[[    3     0     0]
```

```
 [    0 11230    95]
 [    1   231  2422]]


              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 26416     7]
 [    0    75  6116]]


              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 11040   285]
 [    0   420  2234]]


              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.96      0.97      0.97     11325
           3       0.89      0.84      0.86      2654

    accuracy                           0.95     13982
   macro avg       0.62      0.61      0.61     13982
weighted avg       0.95      0.95      0.95     13982
```

```
metrics on train data
[[    2     6     0]
 [    0 25985   438]
 [    0   707  5484]]
```

	precision	recall	f1-score	support
0	1.00	0.25	0.40	8
1	0.97	0.98	0.98	26423
3	0.93	0.89	0.91	6191
accuracy			0.96	32622
macro avg	0.97	0.71	0.76	32622
weighted avg	0.96	0.96	0.96	32622

```
==============================
<class 'sklearn.naive_bayes.GaussianNB'>
metrics on test data
[[   3    0    0]
 [   0 8834 2491]
 [   0  287 2367]]
```

	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 20698  5725]
 [    0   620  5571]]
```

	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 11126   199]
 [    0   416  2238]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       0.96      0.98      0.97     11325
```

```
           3       0.92      0.84      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 25965   458]
 [    0   891  5300]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       0.97      0.98      0.97     26423
           3       0.92      0.86      0.89      6191

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

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

[210]:
```
summary_4=pd.DataFrame(model_summary_4).
↪sort_values('f1_score_test',ascending=False).drop('model_name_test',axis=1)
```

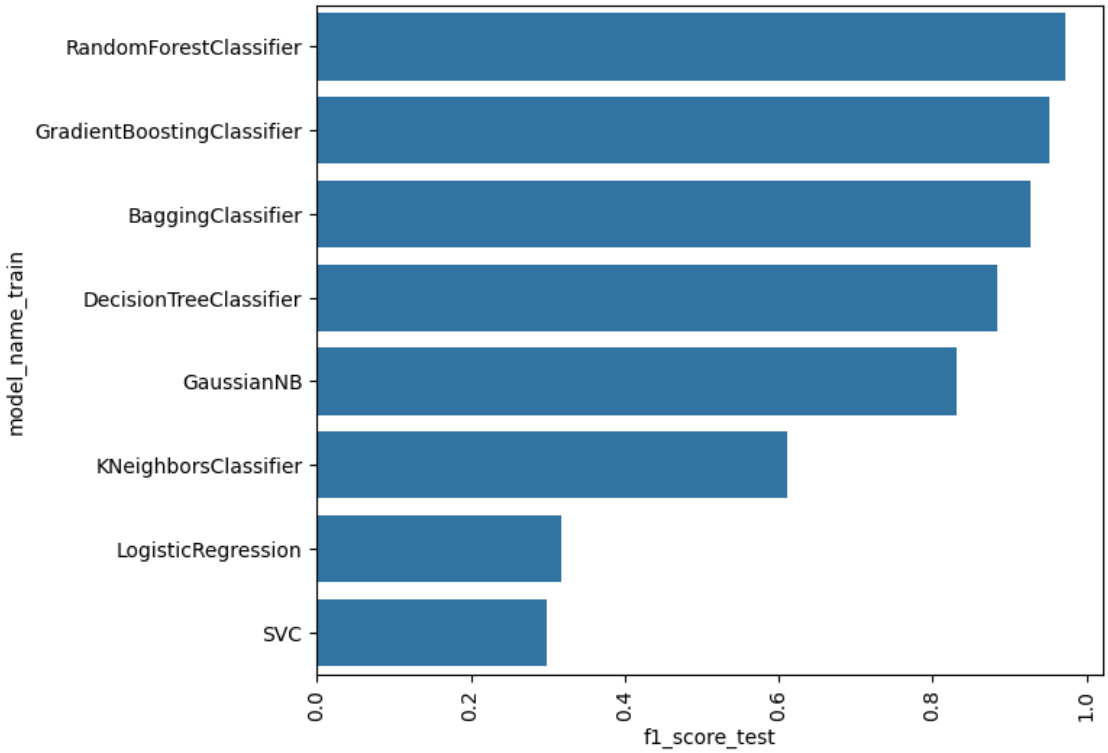
[211]:
```
summary_4
```

[211]:

	model_name_train	f1_score_train	recall_score_train	accuracy_score_train	f1_score_test	recall_score_test	accuracy_score_test
2	RandomForestClassifier	1.000000	1.000000	1.000000	0.972256	0.965561	0.974968
7	GradientBoostingClassifier	0.953928	0.946249	0.958648	0.950768	0.941895	0.956015
3	BaggingClassifier	0.997264	0.995874	0.997486	0.926533	0.968065	0.976613
1	DecisionTreeClassifier	1.000000	1.000000	1.000000	0.883365	0.965285	0.969389
5	GaussianNB	0.834751	0.894396	0.805499	0.831441	0.890635	0.801316
4	KNeighborsClassifier	0.761269	0.706409	0.964717	0.610882	0.605528	0.949363
0	LogisticRegression	0.316120	0.337680	0.799399	0.317604	0.338313	0.799170
6	SVC	0.298337	0.333333				

```
6              0.809975        0.298336              0.333333              0.809970
```

[212]:
```
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()
```



5.7 ## 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

[213]:
```
#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_test)

#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 11240    85]
 [    0   240  2414]]


classification report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       0.98      0.99      0.99     11325
           3       0.97      0.91      0.94      2654

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

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

5.8 # 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.

5.9 # 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.