Comcast_complaint

October 29, 2020

[2]:
```
import pandas as pd
import numpy as np
import matplotlib as mp
import scipy
import scipy.stats
import tensorflow as tf
#import tensorflow_hub as hub
import json
import pickle
import urllib

from sklearn.preprocessing import LabelBinarizer

print(tf.__version__)
```

```
2.1.0
```

Import data into Python environment.

[3]:
```
b=pd.read_csv("Comcast_telecom_complaints_data.csv") #import dataset
# 1 Import
a=b
```

[4]:
```
#import datetime as dt #import datetime
import matplotlib.pyplot as plt
a.dtypes
```

[4]:
```
Ticket #                        object
Customer Complaint              object
Date                            object
Date_month_year                 object
Time                            object
Received Via                    object
City                            object
State                           object
Zip code                         int64
Status                          object
Filing on Behalf of Someone     object
```

```
dtype: object
```

Convert into datetime datatype

[5]:
```
a['Datetime']=a['Date_month_year']+" "+a['Time']
a['Datetime']=pd.to_datetime(a['Datetime'])
#apply(lambda row: np.nan if type(row)==type(0.0)\
#else(dt.datetime.strptime(row,"%d-%m-%y")))
a.Datetime
```

[5]:
```
0       2015-04-22 15:53:50
1       2015-08-04 10:22:56
2       2015-04-18 09:55:47
3       2015-07-05 11:59:35
4       2015-05-26 13:25:26
                  …
2219    2015-02-04 09:13:18
2220    2015-02-06 13:24:39
2221    2015-09-06 17:28:41
2222    2015-06-23 23:13:30
2223    2015-06-24 22:28:33
Name: Datetime, Length: 2224, dtype: datetime64[ns]
```

[6]:
```
a.dtypes
```

[6]:
```
Ticket #                             object
Customer Complaint                   object
Date                                 object
Date_month_year                      object
Time                                 object
Received Via                         object
City                                 object
State                                object
Zip code                              int64
Status                               object
Filing on Behalf of Someone          object
Datetime                     datetime64[ns]
dtype: object
```

[7]:
```
a['Date_month_year_dt']=pd.to_datetime(a['Date_month_year'])
```

[8]:
```
a.dtypes
```

[8]:
```
Ticket #                             object
Customer Complaint                   object
Date                                 object
Date_month_year                      object
```

```
Time                                 object
Received Via                         object
City                                 object
State                                object
Zip code                              int64
Status                               object
Filing on Behalf of Someone          object
Datetime                     datetime64[ns]
Date_month_year_dt           datetime64[ns]
dtype: object
```

Set index as date time check how it works

[9]:
```
a=a.set_index(a['Datetime'])
```

[10]:
```
a.head()
```

[10]:
```
                     Ticket #  \
Datetime
2015-04-22 15:53:50   250635
2015-08-04 10:22:56   223441
2015-04-18 09:55:47   242732
2015-07-05 11:59:35   277946
2015-05-26 13:25:26   307175

                                                  Customer Complaint  \
Datetime
2015-04-22 15:53:50                       Comcast Cable Internet Speeds
2015-08-04 10:22:56       Payment disappear - service got disconnected
2015-04-18 09:55:47                                    Speed and Service
2015-07-05 11:59:35  Comcast Imposed a New Usage Cap of 300GB that …
2015-05-26 13:25:26         Comcast not working and no service to boot

                         Date Date_month_year         Time  \
Datetime
2015-04-22 15:53:50  22-04-15       22-Apr-15   3:53:50 PM
2015-08-04 10:22:56  04-08-15       04-Aug-15  10:22:56 AM
2015-04-18 09:55:47  18-04-15       18-Apr-15   9:55:47 AM
2015-07-05 11:59:35  05-07-15       05-Jul-15  11:59:35 AM
2015-05-26 13:25:26  26-05-15       26-May-15   1:25:26 PM

                           Received Via      City     State  Zip code  Status  \
Datetime
2015-04-22 15:53:50  Customer Care Call  Abingdon  Maryland     21009  Closed
2015-08-04 10:22:56            Internet   Acworth   Georgia     30102  Closed
2015-04-18 09:55:47            Internet   Acworth   Georgia     30101  Closed
2015-07-05 11:59:35            Internet   Acworth   Georgia     30101    Open
```

```
2015-05-26 13:25:26            Internet   Acworth   Georgia     30101  Solved

                    Filing on Behalf of Someone            Datetime  \
Datetime
2015-04-22 15:53:50                          No 2015-04-22 15:53:50
2015-08-04 10:22:56                          No 2015-08-04 10:22:56
2015-04-18 09:55:47                         Yes 2015-04-18 09:55:47
2015-07-05 11:59:35                         Yes 2015-07-05 11:59:35
2015-05-26 13:25:26                          No 2015-05-26 13:25:26

                    Date_month_year_dt
Datetime
2015-04-22 15:53:50         2015-04-22
2015-08-04 10:22:56         2015-08-04
2015-04-18 09:55:47         2015-04-18
2015-07-05 11:59:35         2015-07-05
2015-05-26 13:25:26         2015-05-26
```

[11]:
```
a['2015']
```

[11]:
```
                    Ticket #  \
Datetime
2015-04-22 15:53:50   250635
2015-08-04 10:22:56   223441
2015-04-18 09:55:47   242732
2015-07-05 11:59:35   277946
2015-05-26 13:25:26   307175
…                      …
2015-02-04 09:13:18   213550
2015-02-06 13:24:39   318775
2015-09-06 17:28:41   331188
2015-06-23 23:13:30   360489
2015-06-24 22:28:33   363614

                                                 Customer Complaint  \
Datetime
2015-04-22 15:53:50                       Comcast Cable Internet Speeds
2015-08-04 10:22:56       Payment disappear - service got disconnected
2015-04-18 09:55:47                                    Speed and Service
2015-07-05 11:59:35  Comcast Imposed a New Usage Cap of 300GB that …
2015-05-26 13:25:26         Comcast not working and no service to boot
…                                                                    …
2015-02-04 09:13:18                                Service Availability
2015-02-06 13:24:39         Comcast Monthly Billing for Returned Modem
2015-09-06 17:28:41                            complaint about comcast
2015-06-23 23:13:30             Extremely unsatisfied Comcast customer
2015-06-24 22:28:33               Comcast, Ypsilanti MI Internet Speed
```

```
                        Date Date_month_year         Time  \
Datetime
2015-04-22 15:53:50  22-04-15       22-Apr-15   3:53:50 PM
2015-08-04 10:22:56  04-08-15       04-Aug-15  10:22:56 AM
2015-04-18 09:55:47  18-04-15       18-Apr-15   9:55:47 AM
2015-07-05 11:59:35  05-07-15       05-Jul-15  11:59:35 AM
2015-05-26 13:25:26  26-05-15       26-May-15   1:25:26 PM
…                       …              …          …
2015-02-04 09:13:18  04-02-15       04-Feb-15   9:13:18 AM
2015-02-06 13:24:39  06-02-15       06-Feb-15   1:24:39 PM
2015-09-06 17:28:41  06-09-15       06-Sep-15   5:28:41 PM
2015-06-23 23:13:30  23-06-15       23-Jun-15  11:13:30 PM
2015-06-24 22:28:33  24-06-15       24-Jun-15  10:28:33 PM

                           Received Via        City     State  Zip code  \
Datetime
2015-04-22 15:53:50  Customer Care Call    Abingdon  Maryland     21009
2015-08-04 10:22:56            Internet     Acworth   Georgia     30102
2015-04-18 09:55:47            Internet     Acworth   Georgia     30101
2015-07-05 11:59:35            Internet     Acworth   Georgia     30101
2015-05-26 13:25:26            Internet     Acworth   Georgia     30101
…                                   …        …       …       …
2015-02-04 09:13:18  Customer Care Call  Youngstown   Florida     32466
2015-02-06 13:24:39  Customer Care Call   Ypsilanti  Michigan     48197
2015-09-06 17:28:41            Internet   Ypsilanti  Michigan     48197
2015-06-23 23:13:30  Customer Care Call   Ypsilanti  Michigan     48197
2015-06-24 22:28:33  Customer Care Call   Ypsilanti  Michigan     48198

                     Status Filing on Behalf of Someone            Datetime  \
Datetime
2015-04-22 15:53:50  Closed                          No 2015-04-22 15:53:50
2015-08-04 10:22:56  Closed                          No 2015-08-04 10:22:56
2015-04-18 09:55:47  Closed                         Yes 2015-04-18 09:55:47
2015-07-05 11:59:35    Open                         Yes 2015-07-05 11:59:35
2015-05-26 13:25:26  Solved                          No 2015-05-26 13:25:26
…                     …                         …                 …
2015-02-04 09:13:18  Closed                          No 2015-02-04 09:13:18
2015-02-06 13:24:39  Solved                          No 2015-02-06 13:24:39
2015-09-06 17:28:41  Solved                          No 2015-09-06 17:28:41
2015-06-23 23:13:30  Solved                          No 2015-06-23 23:13:30
2015-06-24 22:28:33    Open                         Yes 2015-06-24 22:28:33

                    Date_month_year_dt
Datetime
2015-04-22 15:53:50         2015-04-22
2015-08-04 10:22:56         2015-08-04
```

```
2015-04-18 09:55:47         2015-04-18
2015-07-05 11:59:35         2015-07-05
2015-05-26 13:25:26         2015-05-26
…                                  …
2015-02-04 09:13:18         2015-02-04
2015-02-06 13:24:39         2015-02-06
2015-09-06 17:28:41         2015-09-06
2015-06-23 23:13:30         2015-06-23
2015-06-24 22:28:33         2015-06-24

[2224 rows x 13 columns]
```

[12]:
```
a['2015-12']
```

[12]:
```
                     Ticket #  \
Datetime
2015-12-06 21:59:40   338519
2015-12-06 15:59:57   337489
2015-12-05 21:06:01   286768
2015-12-04 15:22:03   231273
2015-12-04 15:45:26   231292
…                      …
2015-12-06 13:18:20   336982
2015-12-04 19:39:11   231419
2015-12-06 11:49:17   336674
2015-12-06 14:45:21   337251
2015-12-06 18:35:59   338192

                                                  Customer Complaint  \
Datetime
2015-12-06 21:59:40  ISP Charging for arbitrary data limits with ov…
2015-12-06 15:59:57                    Comcast not refunding my credit
2015-12-05 21:06:01     not getting what I am paying for with internet
2015-12-04 15:22:03  Comcast's Monopolistic Practices and Data Capping
2015-12-04 15:45:26                           Comcast data cap "trials"
…                                                                    …
2015-12-06 13:18:20                                  Comcast monopoly
2015-12-04 19:39:11     over sold/ over billed for short term services
2015-12-06 11:49:17                           Data caps for cable ISP;s
2015-12-06 14:45:21                                Unfair Pricing
2015-12-06 18:35:59    Speed throttling, speeds not at promised output

                         Date Date_month_year         Time  \
Datetime
2015-12-06 21:59:40  06-12-15       06-Dec-15   9:59:40 PM
2015-12-06 15:59:57  06-12-15       06-Dec-15   3:59:57 PM
2015-12-05 21:06:01  05-12-15       05-Dec-15   9:06:01 PM
```

```
2015-12-04 15:22:03  04-12-15       04-Dec-15   3:22:03 PM
2015-12-04 15:45:26  04-12-15       04-Dec-15   3:45:26 PM
…                       …              …          …
2015-12-06 13:18:20  06-12-15       06-Dec-15   1:18:20 PM
2015-12-04 19:39:11  04-12-15       04-Dec-15   7:39:11 PM
2015-12-06 11:49:17  06-12-15       06-Dec-15  11:49:17 AM
2015-12-06 14:45:21  06-12-15       06-Dec-15   2:45:21 PM
2015-12-06 18:35:59  06-12-15       06-Dec-15   6:35:59 PM

                          Received Via         City          State  Zip code  \
Datetime
2015-12-06 21:59:40            Internet      Acworth        Georgia     30101
2015-12-06 15:59:57            Internet   Alpharetta        Georgia     30005
2015-12-05 21:06:01  Customer Care Call      Andover      Minnesota     55304
2015-12-04 15:22:03            Internet      Atlanta        Georgia     30315
2015-12-04 15:45:26            Internet      Atlanta        Georgia     30316
…                                 …             …            …       …
2015-12-06 13:18:20            Internet      Tukwila     Washington     98188
2015-12-04 19:39:11  Customer Care Call      Wenonah     New Jersey      8090
2015-12-06 11:49:17  Customer Care Call  Westminster       Colorado     80021
2015-12-06 14:45:21  Customer Care Call       Woburn  Massachusetts      1801
2015-12-06 18:35:59  Customer Care Call    Yorkville       Illinois     60560

                    Status Filing on Behalf of Someone            Datetime  \
Datetime
2015-12-06 21:59:40  Solved                          No 2015-12-06 21:59:40
2015-12-06 15:59:57  Solved                          No 2015-12-06 15:59:57
2015-12-05 21:06:01  Solved                          No 2015-12-05 21:06:01
2015-12-04 15:22:03  Closed                          No 2015-12-04 15:22:03
2015-12-04 15:45:26  Closed                          No 2015-12-04 15:45:26
…                     …                         …                 …
2015-12-06 13:18:20  Solved                          No 2015-12-06 13:18:20
2015-12-04 19:39:11  Closed                          No 2015-12-04 19:39:11
2015-12-06 11:49:17  Solved                          No 2015-12-06 11:49:17
2015-12-06 14:45:21  Solved                          No 2015-12-06 14:45:21
2015-12-06 18:35:59    Open                         Yes 2015-12-06 18:35:59

                    Date_month_year_dt
Datetime
2015-12-06 21:59:40         2015-12-06
2015-12-06 15:59:57         2015-12-06
2015-12-05 21:06:01         2015-12-05
2015-12-04 15:22:03         2015-12-04
2015-12-04 15:45:26         2015-12-04
…                                 …
2015-12-06 13:18:20         2015-12-06
2015-12-04 19:39:11         2015-12-04
```

```
2015-12-06 11:49:17         2015-12-06
2015-12-06 14:45:21         2015-12-06
2015-12-06 18:35:59         2015-12-06

[65 rows x 13 columns]
```

[13]:
```
a.sort_index(inplace=True)
```

[14]:
```
a.tail()
```

[14]:
```
                     Ticket #  \
Datetime
2015-12-06 21:18:18   338467
2015-12-06 21:46:12   338507
2015-12-06 21:51:40   338510
2015-12-06 21:59:40   338519
2015-12-06 23:52:11   338606

                                                 Customer Complaint  \
Datetime
2015-12-06 21:18:18                        Cable internet unavailable
2015-12-06 21:46:12       Comcast Internet, cable, and phone outtages
2015-12-06 21:51:40                                           Comcast
2015-12-06 21:59:40  ISP Charging for arbitrary data limits with ov…
2015-12-06 23:52:11                        Internet connection outage

                        Date Date_month_year         Time  \
Datetime
2015-12-06 21:18:18  06-12-15       06-Dec-15   9:18:18 PM
2015-12-06 21:46:12  06-12-15       06-Dec-15   9:46:12 PM
2015-12-06 21:51:40  06-12-15       06-Dec-15   9:51:40 PM
2015-12-06 21:59:40  06-12-15       06-Dec-15   9:59:40 PM
2015-12-06 23:52:11  06-12-15       06-Dec-15  11:52:11 PM

                           Received Via       City       State  Zip code  \
Datetime
2015-12-06 21:18:18            Internet  San Mateo  California     94402
2015-12-06 21:46:12  Customer Care Call  Clarkston    Michigan     48346
2015-12-06 21:51:40            Internet   Muskegon    Michigan     49445
2015-12-06 21:59:40            Internet    Acworth     Georgia     30101
2015-12-06 23:52:11  Customer Care Call  Clarkston    Michigan     48346

                     Status Filing on Behalf of Someone            Datetime  \
Datetime
2015-12-06 21:18:18  Closed                          No 2015-12-06 21:18:18
2015-12-06 21:46:12  Solved                          No 2015-12-06 21:46:12
2015-12-06 21:51:40  Solved                          No 2015-12-06 21:51:40
```

```
2015-12-06 21:59:40  Solved                          No 2015-12-06 21:59:40
2015-12-06 23:52:11  Solved                          No 2015-12-06 23:52:11

                    Date_month_year_dt
Datetime
2015-12-06 21:18:18         2015-12-06
2015-12-06 21:46:12         2015-12-06
2015-12-06 21:51:40         2015-12-06
2015-12-06 21:59:40         2015-12-06
2015-12-06 23:52:11         2015-12-06
```

[15]:
```
a.head()
```

[15]:
```
                    Ticket #          Customer Complaint      Date  \
Datetime
2015-01-04 00:18:47   211255          Comcast harassment  04-01-15
2015-01-04 10:43:20   211472               comcast cable  04-01-15
2015-01-04 10:47:35   211478                     Comcast  04-01-15
2015-01-04 12:01:06   211677  Comcast refusal of service  04-01-15
2015-01-04 12:28:58   211775            Horrible Service  04-01-15

                    Date_month_year         Time        Received Via  \
Datetime
2015-01-04 00:18:47       04-Jan-15  12:18:47 AM  Customer Care Call
2015-01-04 10:43:20       04-Jan-15  10:43:20 AM  Customer Care Call
2015-01-04 10:47:35       04-Jan-15  10:47:35 AM           Internet
2015-01-04 12:01:06       04-Jan-15  12:01:06 PM  Customer Care Call
2015-01-04 12:28:58       04-Jan-15  12:28:58 PM  Customer Care Call

                                 City         State  Zip code  Status  \
Datetime
2015-01-04 00:18:47         Schaumburg      Illinois     60193  Closed
2015-01-04 10:43:20           Lockport      Illinois     60441  Closed
2015-01-04 10:47:35  North Huntingdon  Pennsylvania     15642  Closed
2015-01-04 12:01:06              Wayne  Pennsylvania     19087  Closed
2015-01-04 12:28:58         Mckeesport  Pennsylvania     15132  Closed

                    Filing on Behalf of Someone            Datetime  \
Datetime
2015-01-04 00:18:47                           No 2015-01-04 00:18:47
2015-01-04 10:43:20                           No 2015-01-04 10:43:20
2015-01-04 10:47:35                           No 2015-01-04 10:47:35
2015-01-04 12:01:06                           No 2015-01-04 12:01:06
2015-01-04 12:28:58                           No 2015-01-04 12:28:58

                    Date_month_year_dt
Datetime
```

```
2015-01-04 00:18:47          2015-01-04
2015-01-04 10:43:20          2015-01-04
2015-01-04 10:47:35          2015-01-04
2015-01-04 12:01:06          2015-01-04
2015-01-04 12:28:58          2015-01-04
```

Set index as date month year dt and plot daily graph

[16]:
```
a=a.set_index(a['Date_month_year_dt'])
```

[17]:
```
#create new columns month day and day of year
a["month"]=a.apply(lambda row: row.Datetime.month,axis=1)#(row.ClosedDate.year))
```

Create a new month coluumn

[18]:
```
a.month
```

[18]:
```
Date_month_year_dt
2015-01-04     1
2015-01-04     1
2015-01-04     1
2015-01-04     1
2015-01-04     1
               ..
2015-12-06    12
2015-12-06    12
2015-12-06    12
2015-12-06    12
2015-12-06    12
Name: month, Length: 2224, dtype: int64
```

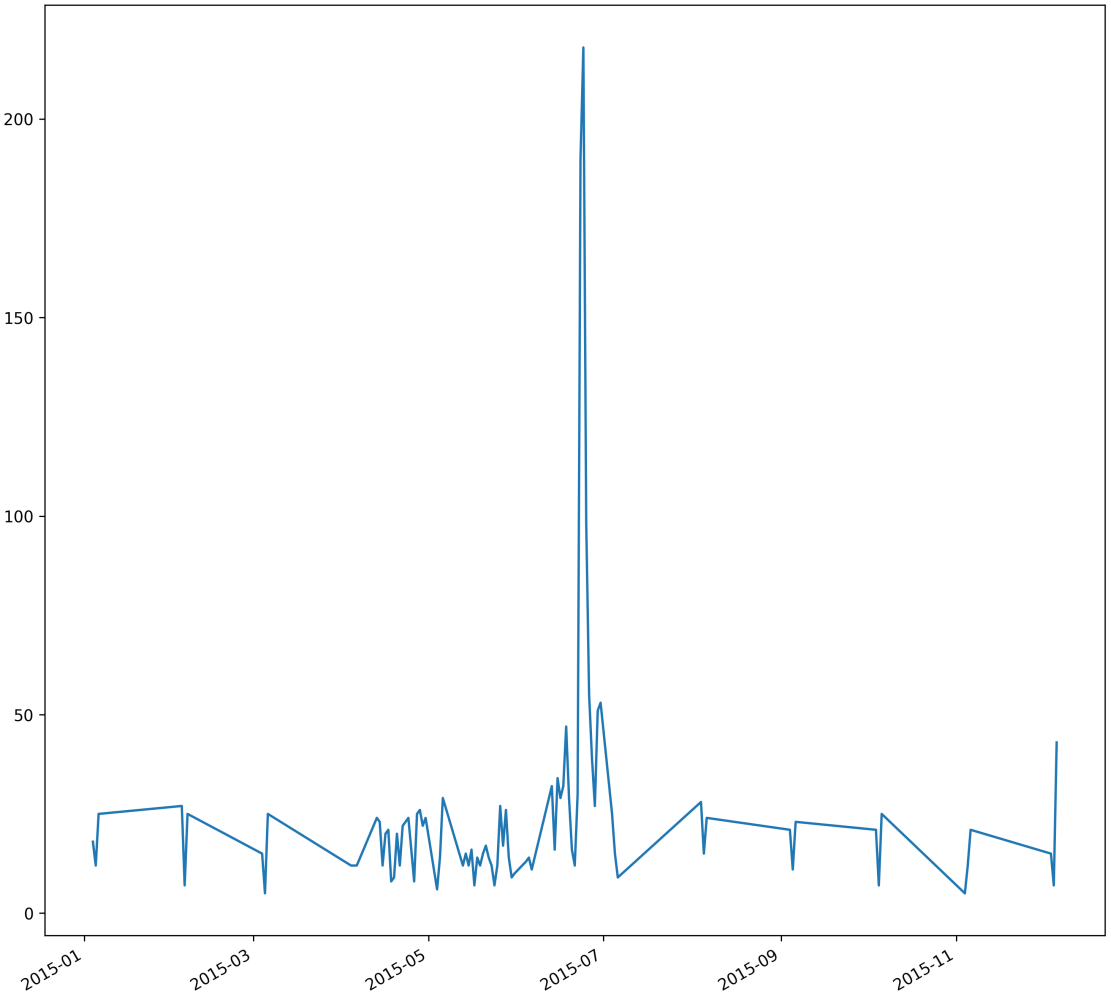
[19]:
```
a['Date_month_year_dt'].value_counts()[:20]
```

[19]:
```
2015-06-24    218
2015-06-23    190
2015-06-25     98
2015-06-26     55
2015-06-30     53
2015-06-29     51
2015-06-18     47
2015-12-06     43
2015-06-27     39
2015-06-15     34
2015-06-17     32
2015-06-13     32
2015-06-22     30
2015-06-19     29
2015-05-06     29
```

```
2015-06-16     29
2015-08-04     28
2015-02-04     27
2015-06-28     27
2015-05-26     27
Name: Date_month_year_dt, dtype: int64
```

Get daily plot

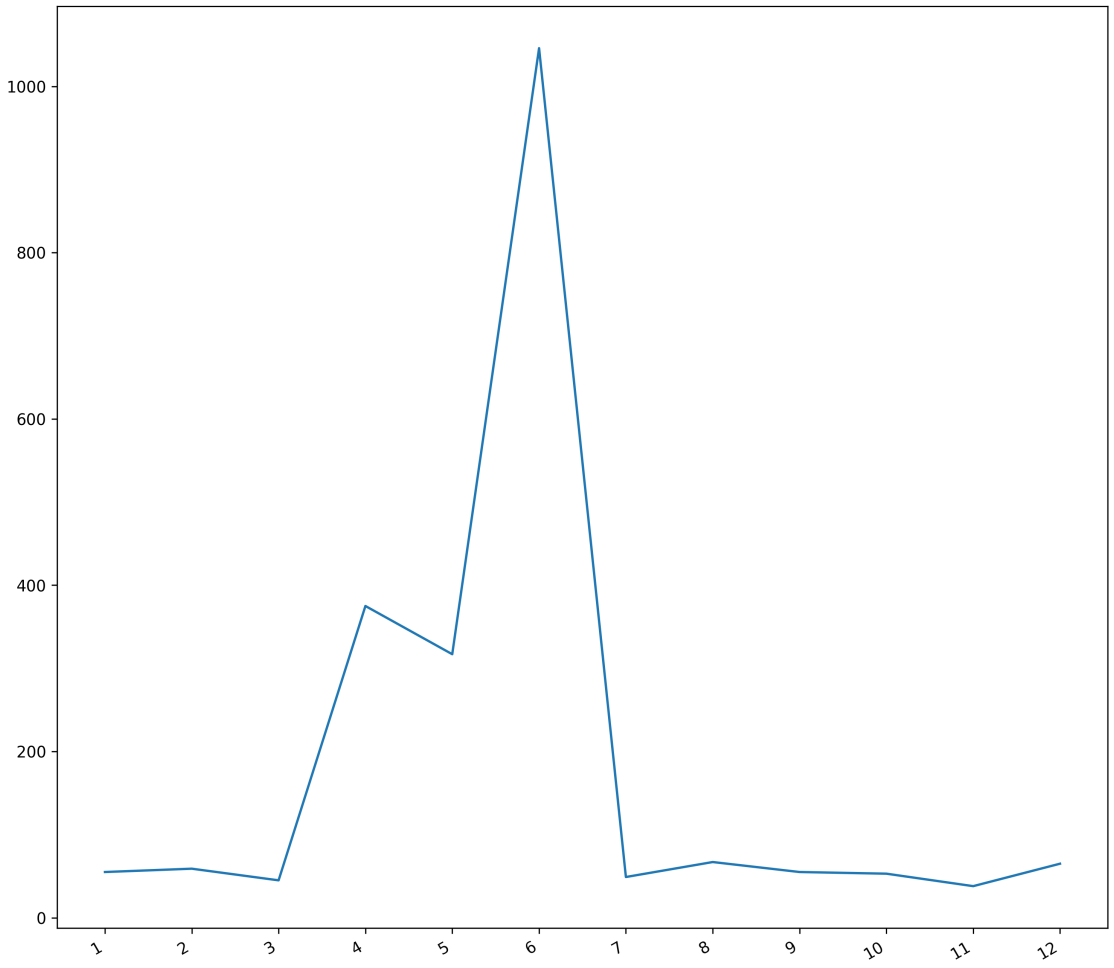
[20]:
```
a['Date_month_year_dt'].value_counts().index#.plot();
a['Date_month_year_dt'].value_counts().sort_index()
# make up some data
x = a['Date_month_year_dt'].value_counts().sort_index().index
y =a['Date_month_year_dt'].value_counts().sort_index()
plt.figure(num=None, figsize=(12, 12), dpi=300, facecolor='w', edgecolor='k')

# plot
plt.plot(x,y)
# beautify the x-labels
plt.gcf().autofmt_xdate()
plt.show()
#a['Date_month_year_dt'].value_counts().plot();
```



More commplaints happened between montth 6 and 7 (june and july) Now plot monthly data

[21]:
```
a.sort_values('month')
a['month'].value_counts().sort_index()
# make up some data
x = a['month'].value_counts().sort_index().index
y =a['month'].value_counts().sort_index()
# plot

plt.figure(num=None, figsize=(12, 12), dpi=300, facecolor='w', edgecolor='k')
plt.plot(x,y)

# beautify the x-labels
plt.gcf().autofmt_xdate()
plt.xticks(x)
plt.show()
```

```
#we have major spike in complaints on June month
```



Create new columnn for day of month and plot data for that(results same as plot 1

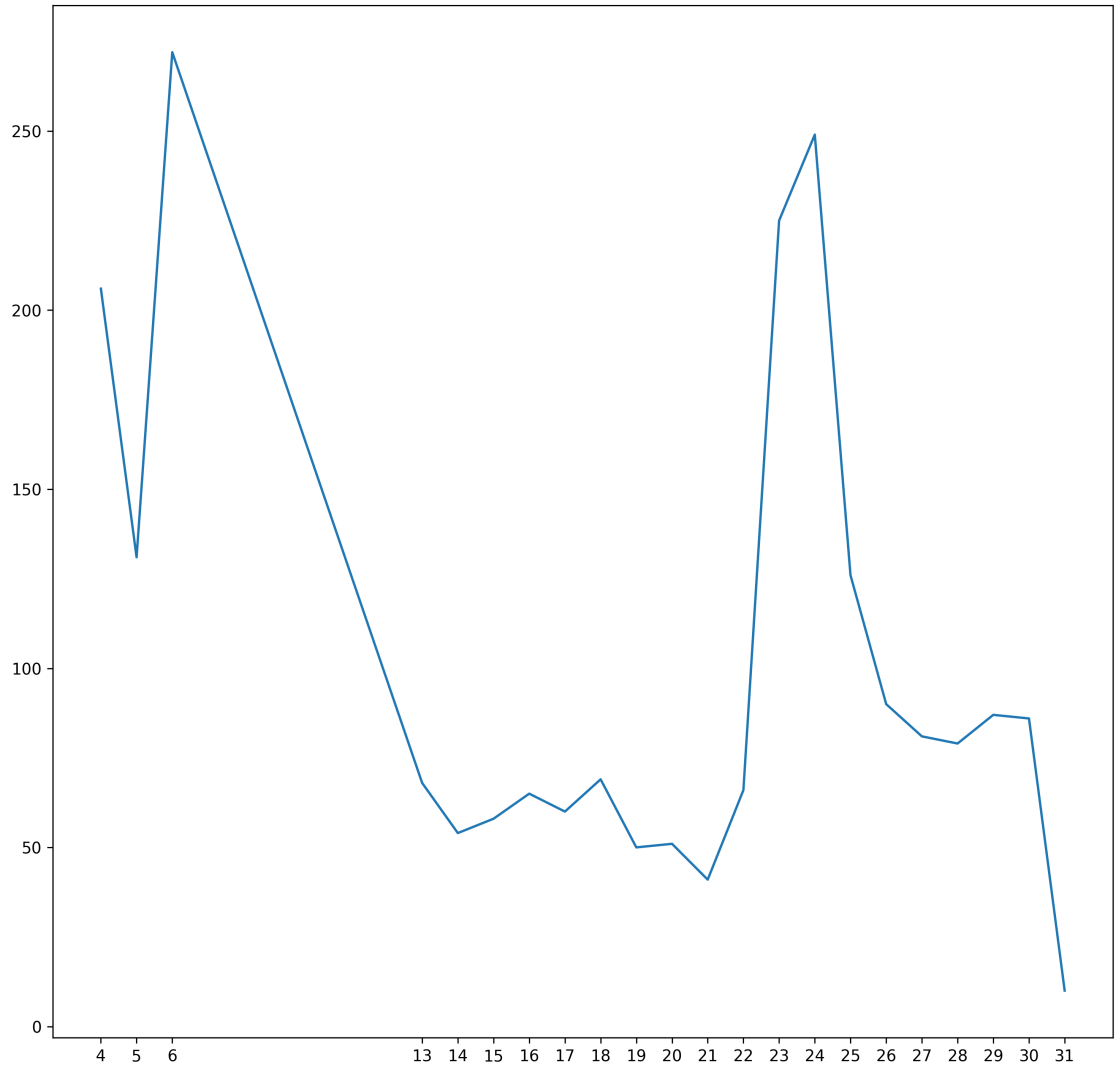
[238]:
```
a["day"]=a.apply(lambda row: row.Datetime.day,axis=1)#(row.ClosedDate.year))
a['day']
```

[238]:
```
Date_month_year_dt
2015-01-04    4
2015-01-04    4
2015-01-04    4
2015-01-04    4
2015-01-04    4
              ..
2015-12-06    6
2015-12-06    6
2015-12-06    6
```

```
2015-12-06    6
2015-12-06    6
Name: day, Length: 2224, dtype: int64
```

- Provide the trend chart for the number of complaints at monthly and daily granularity levels.

[23]:
```
a.sort_values('day')
a['day'].value_counts().sort_index()
# make up some data
x = a['day'].value_counts().sort_index().index
y =a['day'].value_counts().sort_index()

plt.figure(num=None, figsize=(12, 12), dpi=300, facecolor='w', edgecolor='k')
# plot
plt.plot(x,y)

# beautify the x-labels
#plt.gcf().autofmt_xdate()
plt.xticks(x)
plt.show()
```



first week and last 10 days of month seem to have more complaints

[24]:
```
a.columns
#we do not have complaint type in our data
```

[24]:
```
Index(['Ticket #', 'Customer Complaint', 'Date', 'Date_month_year', 'Time',
       'Received Via', 'City', 'State', 'Zip code', 'Status',
       'Filing on Behalf of Someone', 'Datetime', 'Date_month_year_dt',
       'month', 'day'],
      dtype='object')
```

[25]:
```
a.columns=a.columns.str.replace(' ','')
#remove spaces from columns for ease
a.columns
```

```
classes=pd.read_excel("Classifier.xlsx") #import datasetb=pd.
↪read_csv("Comcast_telecom_complaints_data.csv") #import dataset
```

We manually labeled some of the data with complaint type this will be used to train and test our model to commpute complaint type

We will see that this is inaccuurate(max around 70% accuracy) and later at end we will try NLP Using LDA with unsuperwised learning which would also eleminate manual labelling effort taken in this method however lets see the output of supervised methods and check accuracy levels for this particular problem with different models and algorithms

[26]:
```
from sklearn.utils import shuffle
classes = shuffle(classes)
classes
classes.columns=classes.columns.str.replace(' ','')
classes.columns
```

[26]:
```
Index(['CustomerComplaint', 'Type'], dtype='object')
```

[27]:
```
classes.head
#drop the complaint types that are rare/amb
classes['Type'].replace({'suupport':'support'},inplace=True)
#classes=classes[classes["Type"]!='suupport']
classes['Type'].replace({'health':'misc'},inplace=True)

classes['Type'].replace({'others':'misc'},inplace=True)
#classes=classes[classes["Type"]!='misc']

classes.Type.unique()
```

[27]:
```
array(['payment and charges', 'Data usage limit', 'customer sentiment',
       'service termination/ not setup right', 'service not working',
       'misc', 'cable', 'support', 'Internet Speed',
       'Customer Care Feedback', 'Blockage', 'Unwanted service'],
      dtype=object)
```

[28]:
```
classes.Type.unique()
```

[28]:
```
array(['payment and charges', 'Data usage limit', 'customer sentiment',
       'service termination/ not setup right', 'service not working',
       'misc', 'cable', 'support', 'Internet Speed',
       'Customer Care Feedback', 'Blockage', 'Unwanted service'],
      dtype=object)
```

[29]:
```
classes.Type.value_counts()
```

[29]:
```
payment and charges                  113
Internet Speed                        53
```

```
Data usage limit                          50
customer sentiment                        33
misc                                      32
service not working                       30
support                                   25
service termination/ not setup right      19
Blockage                                  17
Customer Care Feedback                    13
Unwanted service                           9
cable                                      5
Name: Type, dtype: int64
```

[30]:
```
from io import StringIO
col = [ 'Type','CustomerComplaint' ]
df=classes
df = df[col]
df = df[pd.notnull(df['Type'])]

df['category_id'] = df['Type'].factorize()[0]

category_id_df = df[['Type', 'category_id']].drop_duplicates().
 ↪sort_values('category_id')
category_to_id = dict(category_id_df.values)
id_to_category = dict(category_id_df[['category_id', 'Type']].values)
print(id_to_category)
df.head()
```

```
{0: 'payment and charges', 1: 'Data usage limit', 2: 'customer sentiment', 3:
'service termination/ not setup right', 4: 'service not working', 5: 'misc', 6:
'cable', 7: 'support', 8: 'Internet Speed', 9: 'Customer Care Feedback', 10:
'Blockage', 11: 'Unwanted service'}
```

[30]:

	Type	CustomerComplaint	category_id
179	payment and charges	hidden fees, dropped internet connection	0
32	Data usage limit	Comcast using a Data Cap to take however much …	1
124	customer sentiment	Comcast (Xfinity) Monopolistic Billing Practices	2
125	customer sentiment	Terrible waiting times	2
233	service termination/ not setup right	disconnection of service	3

[31]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2',␣
 ↪encoding='latin-1', ngram_range=(1, 2), stop_words='english')
features = tfidf.fit_transform(df.CustomerComplaint).toarray()
labels = df.category_id
features.shape
```

[31]:
```
(399, 53)
```

[32]:
```
from sklearn.feature_selection import chi2
import numpy as np
N = 2
#print(category_to_id.items())
for Type, category_id in sorted(category_to_id.items()):
  print(Type)
  features_chi2 = chi2(features, labels == category_id)
  indices = np.argsort(features_chi2[0])
  feature_names = np.array(tfidf.get_feature_names())[indices]
  unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
  bigrams = [v for v in feature_names if len(v.split(' ')) == 2]
  print("# '{}':".format(Type))
  print("  . Most correlated unigrams:\n. {}".format('\n. '.join(unigrams[-N:
 ↪])))
  print("  . Most correlated bigrams:\n. {}".format('\n. '.join(bigrams[-N:])))
```

```
Blockage
# 'Blockage':
  . Most correlated unigrams:
. contract
. hbo
  . Most correlated bigrams:
. comcast data
. comcast internet
Customer Care Feedback
# 'Customer Care Feedback':
  . Most correlated unigrams:
. poor
. customer
  . Most correlated bigrams:
. comcast service
. customer service
Data usage limit
# 'Data usage limit':
  . Most correlated unigrams:
. cap
. data
  . Most correlated bigrams:
```

```
. data cap
. comcast data
Internet Speed
# 'Internet Speed':
  . Most correlated unigrams:
. throttling
. speed
  . Most correlated bigrams:
. slow internet
. internet speed
Unwanted service
# 'Unwanted service':
  . Most correlated unigrams:
. services
. service
  . Most correlated bigrams:
. comcast data
. comcast internet
cable
# 'cable':
  . Most correlated unigrams:
. connection
. cable
  . Most correlated bigrams:
. comcast internet
. customer service
customer sentiment
# 'customer sentiment':
  . Most correlated unigrams:
. monopoly
. complaint
  . Most correlated bigrams:
. comcast internet
. comcast xfinity
misc
# 'misc':
  . Most correlated unigrams:
. comcast
. xfinity
  . Most correlated bigrams:
. comcast service
. comcast xfinity
payment and charges
# 'payment and charges':
  . Most correlated unigrams:
. pricing
. billing
  . Most correlated bigrams:
```

```
. billing practices
. comcast billing
service not working
# 'service not working':
  . Most correlated unigrams:
. service
. problem
  . Most correlated bigrams:
. service issues
. internet service
service termination/ not setup right
# 'service termination/ not setup right':
  . Most correlated unigrams:
. service
. failure
  . Most correlated bigrams:
. comcast internet
. internet service
support
# 'support':
  . Most correlated unigrams:
. internet
. issues
  . Most correlated bigrams:
. service issues
. comcast internet
```

[33]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
X_train, X_test, y_train, y_test = train_test_split(df['CustomerComplaint'],␣
 ↪df['Type'], random_state = 0, test_size=0.2)
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
clf = MultinomialNB().fit(X_train_tfidf, y_train)
```

[34]:
```
print(clf.predict(count_vect.transform(["This company refuses to provide me␣
 ↪verification and validation of debt per my right under the FDCPA. I do not␣
 ↪believe this debt is mine."])))
```

```
['payment and charges']
```

[35]:
```
print(clf.predict(count_vect.transform(["This company is not giving proper info␣
 ↪onn data cap limit"])))
```

```
['Data usage limit']
```

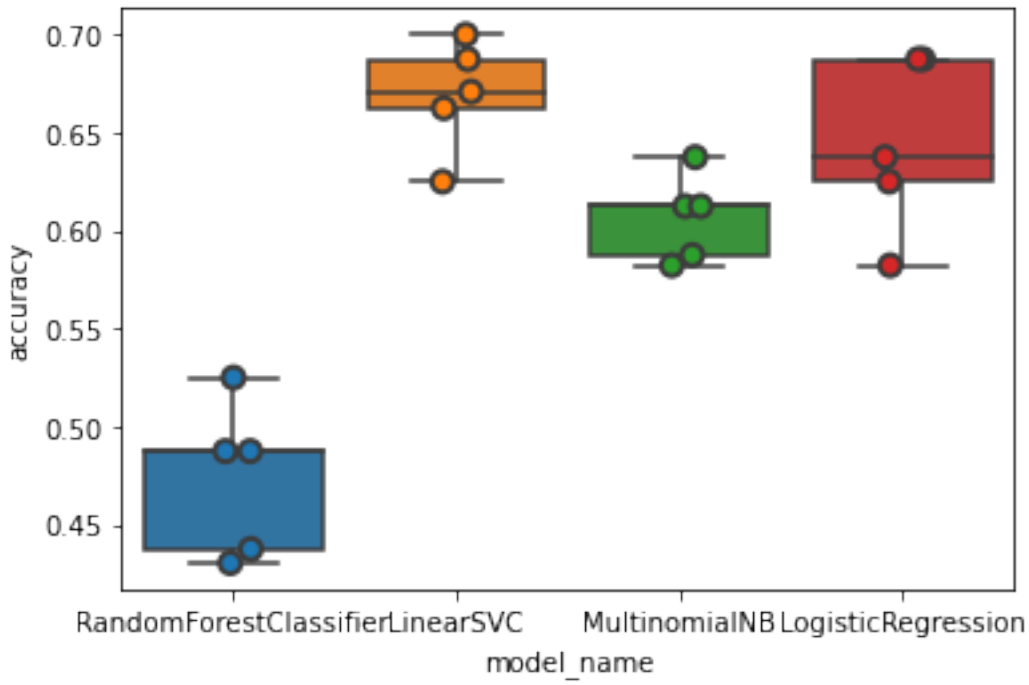
[36]:
```
print(clf.predict(count_vect.transform(["HBOgo ps4"])))
```

```
['payment and charges']
```

[37]:
```
print(clf.predict(count_vect.transform(["slow mbps"])))
```

```
['Internet Speed']
```

[38]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.model_selection import cross_val_score
models = [
    RandomForestClassifier(n_estimators=200, max_depth=3, random_state=0),
    LinearSVC(),
    MultinomialNB(),
    LogisticRegression(random_state=0),
]
CV = 5
cv_df = pd.DataFrame(index=range(CV * len(models)))
entries = []
for model in models:
  model_name = model.__class__.__name__
  accuracies = cross_val_score(model, features, labels, scoring='accuracy',␣
 ↪cv=CV)
  for fold_idx, accuracy in enumerate(accuracies):
    entries.append((model_name, fold_idx, accuracy))
cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])
import seaborn as sns
sns.boxplot(x='model_name', y='accuracy', data=cv_df)
sns.stripplot(x='model_name', y='accuracy', data=cv_df,
              size=8, jitter=True, edgecolor="gray", linewidth=2)
plt.show()
```



Comparing different algorithms and models we find Linear SVC isperforming around 66-70% accurately but this is not enough for our particular use case and this consumed sufficient time for manual labelling

[39]:
```
cv_df.groupby('model_name').accuracy.mean()
```

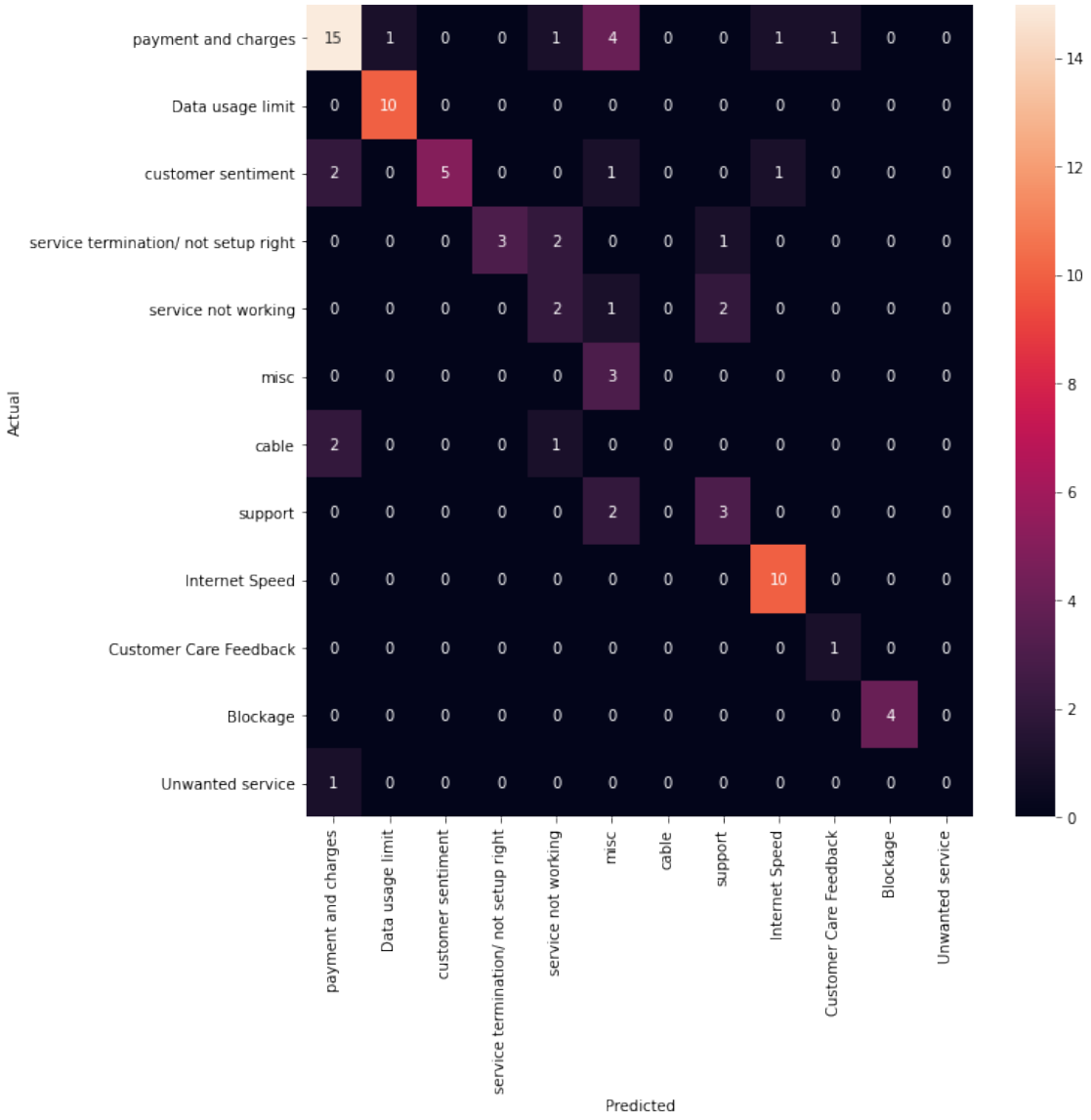
[39]:
```
model_name
LinearSVC                 0.669177
LogisticRegression        0.643956
MultinomialNB             0.606456
RandomForestClassifier    0.473576
Name: accuracy, dtype: float64
```

[ ]:
```

```

[ ]:
```

```

[40]:
```
model = LinearSVC()
X_train, X_test, y_train, y_test, indices_train, indices_test =␣
 ↪train_test_split(features, labels, df.index, test_size=0.2, random_state=0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import confusion_matrix
conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(10,10))
```

```
sns.heatmap(conf_mat, annot=True, fmt='d',
            xticklabels=category_id_df.Type.values, yticklabels=category_id_df.
 ↪Type.values)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



[41]:
```
#Cv
#from sklearn.cross_validation import train_test_split

#Classifier imports
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC

# Performance metrics
from sklearn.metrics import accuracy_score, classification_report


tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2',␣
 ↪encoding='latin-1', ngram_range=(1, 2), stop_words='english')
features = tfidf.fit_transform(df.CustomerComplaint).toarray()
labels = df.Type
features.shape

x1,x2,y1,y2, indices_train, indices_test = train_test_split(features, labels,␣
 ↪df.index, test_size=0.25, random_state=0)
# Initialize our classifiers
gnb = GaussianNB()
KNN = KNeighborsClassifier(n_neighbors=1)
MNB = MultinomialNB()
BNB = BernoulliNB()
LR = LogisticRegression()
SDG = SGDClassifier()
SVC = SVC()
LSVC = LinearSVC()
NSVC = NuSVC()

# Train our classifier and test predict
gnb.fit(x1, y1)
y2_GNB_model = gnb.predict(x2)
print("GaussianNB Accuracy :", accuracy_score(y2, y2_GNB_model))

KNN.fit(x1,y1)
y2_KNN_model = KNN.predict(x2)
print("KNN Accuracy :", accuracy_score(y2, y2_KNN_model))

#MNB.fit(x1,y1)
#y2_MNB_model = MNB.predict(x2)
#print("MNB Accuracy :", accuracy_score(y2, y2_MNB_model))

BNB.fit(x1,y1)
y2_BNB_model = BNB.predict(x2)
print("BNB Accuracy :", accuracy_score(y2, y2_BNB_model))

LR.fit(x1,y1)
y2_LR_model = LR.predict(x2)
print("LR Accuracy :", accuracy_score(y2, y2_LR_model))
```

```
SDG.fit(x1,y1)
y2_SDG_model = SDG.predict(x2)
print("SDG Accuracy :", accuracy_score(y2, y2_SDG_model))

SVC.fit(x1,y1)
y2_SVC_model = SVC.predict(x2)
print("SVC Accuracy :", accuracy_score(y2, y2_SVC_model))

LSVC.fit(x1,y1)
y2_LSVC_model = LSVC.predict(x2)
print("LSVC Accuracy :", accuracy_score(y2, y2_LSVC_model))

#NSVC.fit(x1,y1)
#y2_NSVC_model = NSVC.predict(x2)
#print("NSVC Accuracy :", accuracy_score(y2, y2_NSVC_model))
```

```
GaussianNB Accuracy : 0.5
KNN Accuracy : 0.56
BNB Accuracy : 0.49
LR Accuracy : 0.62
SDG Accuracy : 0.65
SVC Accuracy : 0.63
LSVC Accuracy : 0.66
```

[42]:
```
model.fit(features, labels)
N = 2
for Product, category_id in sorted(category_to_id.items()):
  indices = np.argsort(model.coef_[category_id])
  feature_names = np.array(tfidf.get_feature_names())[indices]
  unigrams = [v for v in reversed(feature_names) if len(v.split(' ')) == 1][:N]
  bigrams = [v for v in reversed(feature_names) if len(v.split(' ')) == 2][:N]
  print("# '{}':".format(Product))
  print("  . Top unigrams:\n       . {}".format('\n       . '.join(unigrams)))
  print("  . Top bigrams:\n       . {}".format('\n       . '.join(bigrams)))
```

```
# 'Blockage':
  . Top unigrams:
       . failure
       . service
  . Top bigrams:
       . internet service
       . comcast xfinity
# 'Customer Care Feedback':
  . Top unigrams:
       . problem
       . poor
  . Top bigrams:
```

```
        . service issues
        . internet service
# 'Data usage limit':
  . Top unigrams:
        . customer
        . poor
  . Top bigrams:
        . customer service
        . service issues
# 'Internet Speed':
  . Top unigrams:
        . billing
        . pricing
  . Top bigrams:
        . comcast billing
        . internet speeds
# 'Unwanted service':
  . Top unigrams:
        . issues
        . internet
  . Top bigrams:
        . comcast internet
        . data usage
# 'cable':
  . Top unigrams:
        . complaint
        . practices
  . Top bigrams:
        . comcast internet
        . comcast xfinity
# 'customer sentiment':
  . Top unigrams:
        . data
        . overage
  . Top bigrams:
        . comcast data
        . data cap
# 'misc':
  . Top unigrams:
        . cable
        . connection
  . Top bigrams:
        . customer service
        . comcast data
# 'payment and charges':
  . Top unigrams:
        . hbo
        . contract
```

```
  . Top bigrams:
       . customer service
       . billing practices
# 'service not working':
  . Top unigrams:
       . services
       . xfinity
  . Top bigrams:
       . comcast throttling
       . data cap
# 'service termination/ not setup right':
  . Top unigrams:
       . speed
       . speeds
  . Top bigrams:
       . slow internet
       . comcast throttling
# 'support':
  . Top unigrams:
       . customer
       . xfinity
  . Top bigrams:
       . comcast service
       . comcast xfinity
```

[ ]:

[43]:
```
a.dtypes
```

[43]:
```
Ticket#                          object
CustomerComplaint                object
Date                             object
Date_month_year                  object
Time                             object
ReceivedVia                      object
City                             object
State                            object
Zipcode                           int64
Status                           object
FilingonBehalfofSomeone          object
Datetime                 datetime64[ns]
Date_month_year_dt       datetime64[ns]
month                             int64
day                               int64
dtype: object
```

[44]:
```
a.columns

a['Type']=a.CustomerComplaint.apply(lambda row: str(LSVC.predict(tfidf.
 ↪transform([row]))).lstrip('[\'').rstrip('\']'))
```

[45]:
```
print(model.predict(tfidf.transform(["slow mbps"])))
```

```
['Internet Speed']
```

[46]:
```
print(LSVC.predict(tfidf.transform(["hidden bill"])))
```

```
['payment and charges']
```

[47]:
```
print(a['Type'].unique())
```

```
['misc' 'payment and charges' 'service termination/ not setup right'
 'service not working' 'Internet Speed' 'Customer Care Feedback'
 'Data usage limit' 'support' 'customer sentiment' 'Blockage']
```

[48]:
```
frequencytype = pd.crosstab(index=a["Type"],columns="occur_count")
frequencytype

#Provide a table with the frequency of complaint types.
```

[48]:

col_0	occur_count
Type	
Blockage	41
Customer Care Feedback	84
Data usage limit	241
Internet Speed	258
customer sentiment	91
misc	442
payment and charges	659
service not working	139
service termination/ not setup right	63
support	206

[49]:
```
a.Type.unique()
```

[49]:
```
array(['misc', 'payment and charges',
       'service termination/ not setup right', 'service not working',
       'Internet Speed', 'Customer Care Feedback', 'Data usage limit',
       'support', 'customer sentiment', 'Blockage'], dtype=object)
```

Which complaint types are maximum i.e., around internet, network issues, or across any other domains.

Maximum complaints are from payments and charges reg. billing, misc charges, etc

payment and charges

Followed by internet speed and data usage limit, support and service not working

Internet Speed

Data usage limit

suppoort

service not working

Create a new categorical variable with value as Open and Closed. Open & Pending is to be categorized as Open and Closed & Solved is to be categorized as Closed.

[50]:
```
a['CompState']=a.Status.apply(lambda temp: "Open" if temp in ["Open",'Pending']
 ↪else "Closed")
a.State = a.State.str.upper()
a.State.unique()
```

[50]:
```
array(['ILLINOIS', 'PENNSYLVANIA', 'GEORGIA', 'ALABAMA', 'CALIFORNIA',
       'TENNESSEE', 'FLORIDA', 'COLORADO', 'WASHINGTON', 'DELAWARE',
       'NEW JERSEY', 'INDIANA', 'MISSISSIPPI', 'MICHIGAN', 'VIRGINIA',
       'RHODE ISLAND', 'OREGON', 'NEW MEXICO', 'SOUTH CAROLINA',
       'MARYLAND', 'DISTRICT OF COLUMBIA', 'TEXAS', 'UTAH',
       'MASSACHUSETTS', 'NEVADA', 'MINNESOTA', 'ARIZONA', 'LOUISIANA',
       'MISSOURI', 'NEW YORK', 'NEW HAMPSHIRE', 'WEST VIRGINIA', 'KANSAS',
       'KENTUCKY', 'OHIO', 'NORTH CAROLINA', 'CONNECTICUT', 'ARKANSAS',
       'VERMONT', 'MAINE', 'MONTANA', 'IOWA'], dtype=object)
```

[229]:
```
a['CompState'].unique()
#a[['CompState','Status']]
#- Create a new categorical variable with value as Open and Closed. Open &
 ↪Pending is to be categorized as Open and Closed & Solved is to be
 ↪categorized as Closed.
#a.columns
```

[229]:
```
array(['Closed', 'Open'], dtype=object)
```

[230]:
```
a.head()
```

[230]:

Date_month_year_dt	Ticket#	CustomerComplaint	Date
2015-01-04	211255	Comcast harassment	04-01-15
2015-01-04	211472	comcast cable	04-01-15
2015-01-04	211478	Comcast	04-01-15
2015-01-04	211677	Comcast refusal of service	04-01-15
2015-01-04	211775	Horrible Service	04-01-15

```
                    Date_month_year         Time         ReceivedVia  \
Date_month_year_dt
2015-01-04               04-Jan-15  12:18:47 AM  Customer Care Call
2015-01-04               04-Jan-15  10:43:20 AM  Customer Care Call
2015-01-04               04-Jan-15  10:47:35 AM           Internet
2015-01-04               04-Jan-15  12:01:06 PM  Customer Care Call
2015-01-04               04-Jan-15  12:28:58 PM  Customer Care Call

                                 City         State  Zipcode  Status  \
Date_month_year_dt
2015-01-04                Schaumburg      ILLINOIS    60193  Closed
2015-01-04                  Lockport      ILLINOIS    60441  Closed
2015-01-04          North Huntingdon  PENNSYLVANIA    15642  Closed
2015-01-04                    Wayne  PENNSYLVANIA    19087  Closed
2015-01-04               Mckeesport  PENNSYLVANIA    15132  Closed

                    FilingonBehalfofSomeone            Datetime  \
Date_month_year_dt
2015-01-04                               No 2015-01-04 00:18:47
2015-01-04                               No 2015-01-04 10:43:20
2015-01-04                               No 2015-01-04 10:47:35
2015-01-04                               No 2015-01-04 12:01:06
2015-01-04                               No 2015-01-04 12:28:58

                    Date_month_year_dt  month  day  \
Date_month_year_dt
2015-01-04                  2015-01-04      1    4
2015-01-04                  2015-01-04      1    4
2015-01-04                  2015-01-04      1    4
2015-01-04                  2015-01-04      1    4
2015-01-04                  2015-01-04      1    4

                                                Type CompState
Date_month_year_dt
2015-01-04                                      misc    Closed
2015-01-04                         payment and charges    Closed
2015-01-04                                      misc    Closed
2015-01-04          service termination/ not setup right    Closed
2015-01-04                         service not working    Closed
```

[231]:
```
a[a["CompState"]!="Closed"].head()
```

[231]:
```
                    Ticket#                    CustomerComplaint      Date  \
Date_month_year_dt
2015-01-05           268362  tInternet Service Provider Complaint  05-01-15
2015-01-05           268773     Poor service from Comcast Xfinity  05-01-15
2015-01-05           268789                         about comcast  05-01-15
```

```
2015-01-06          316257              Comcast Data Usage Meter  06-01-15
2015-01-06          317426    internet availability for students  06-01-15

                   Date_month_year         Time         ReceivedVia  \
Date_month_year_dt
2015-01-05               05-Jan-15  11:51:11 AM  Customer Care Call
2015-01-05               05-Jan-15   2:01:54 PM            Internet
2015-01-05               05-Jan-15   2:04:56 PM            Internet
2015-01-06               06-Jan-15   9:44:42 AM  Customer Care Call
2015-01-06               06-Jan-15   4:59:16 PM            Internet

                            City        State  Zipcode Status  \
Date_month_year_dt
2015-01-05                Renton   WASHINGTON    98055   Open
2015-01-05               Milford     DELAWARE    19963   Open
2015-01-05          Richmond Hill      GEORGIA    31324   Open
2015-01-06              Senatobia  MISSISSIPPI    38668   Open
2015-01-06             Lake Tapps   WASHINGTON    98391   Open

                   FilingonBehalfofSomeone            Datetime  \
Date_month_year_dt
2015-01-05                              No 2015-01-05 11:51:11
2015-01-05                              No 2015-01-05 14:01:54
2015-01-05                              No 2015-01-05 14:04:56
2015-01-06                              No 2015-01-06 09:44:42
2015-01-06                             Yes 2015-01-06 16:59:16

                   Date_month_year_dt  month  day                Type  \
Date_month_year_dt
2015-01-05                 2015-01-05      1    5  customer sentiment
2015-01-05                 2015-01-05      1    5                misc
2015-01-05                 2015-01-05      1    5                misc
2015-01-06                 2015-01-06      1    6    Data usage limit
2015-01-06                 2015-01-06      1    6             support

                   CompState
Date_month_year_dt
2015-01-05              Open
2015-01-05              Open
2015-01-05              Open
2015-01-06              Open
2015-01-06              Open
```

[52]:
```
#- Provide state wise status of complaints in a stacked bar chart. Use the␣
 ↪categorized variable from Q3. Provide insights on:
```

```
df_plot = a.groupby(['CompState', 'State']).size().reset_index().
↪pivot(columns='CompState', index='State', values=0)
```

Provide state wise status of complaints in a stacked bar chart. Use the categorized variable from Q3. Provide insights on:

[53]:
```
a.State.value_counts().to_frame()[:15]
```

[53]:

	State
GEORGIA	288
FLORIDA	240
CALIFORNIA	220
ILLINOIS	164
TENNESSEE	143
PENNSYLVANIA	130
MICHIGAN	115
WASHINGTON	98
COLORADO	80
MARYLAND	78
NEW JERSEY	75
TEXAS	71
MASSACHUSETTS	61
VIRGINIA	60
INDIANA	59

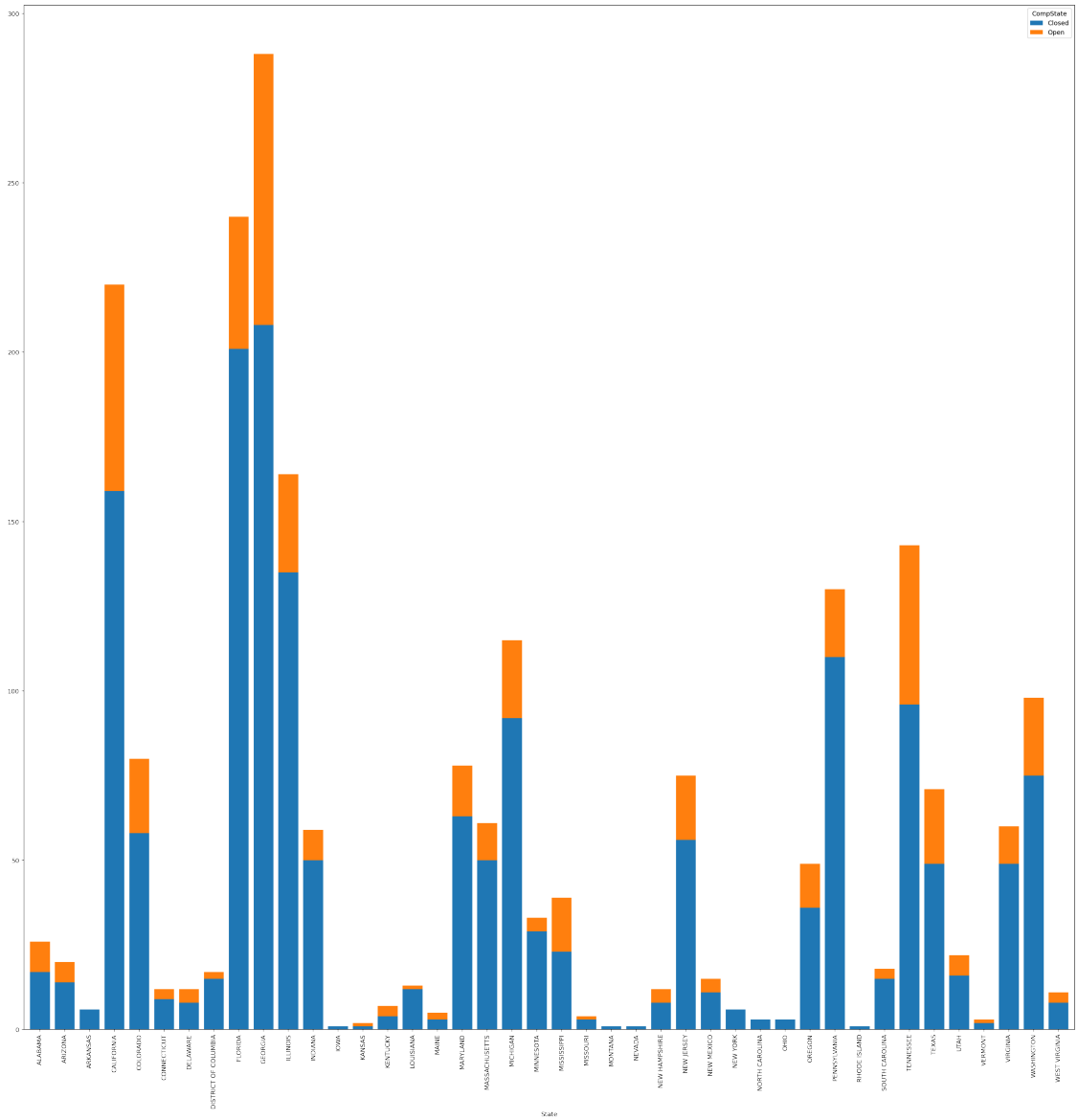
[54]:
```
df_plot.fillna(0.0,inplace=True)
df_plot.head
```

[54]:
```
<bound method NDFrame.head of CompState            Closed  Open
State
ALABAMA                 17.0   9.0
ARIZONA                 14.0   6.0
ARKANSAS                 6.0   0.0
CALIFORNIA             159.0  61.0
COLORADO                58.0  22.0
CONNECTICUT              9.0   3.0
DELAWARE                 8.0   4.0
DISTRICT OF COLUMBIA    15.0   2.0
FLORIDA                201.0  39.0
GEORGIA                208.0  80.0
ILLINOIS               135.0  29.0
INDIANA                 50.0   9.0
IOWA                     1.0   0.0
KANSAS                   1.0   1.0
KENTUCKY                 4.0   3.0
LOUISIANA               12.0   1.0
MAINE                    3.0   2.0
```

```
MARYLAND                63.0  15.0
MASSACHUSETTS           50.0  11.0
MICHIGAN                92.0  23.0
MINNESOTA               29.0   4.0
MISSISSIPPI             23.0  16.0
MISSOURI                 3.0   1.0
MONTANA                  1.0   0.0
NEVADA                   1.0   0.0
NEW HAMPSHIRE            8.0   4.0
NEW JERSEY              56.0  19.0
NEW MEXICO              11.0   4.0
NEW YORK                 6.0   0.0
NORTH CAROLINA           3.0   0.0
OHIO                     3.0   0.0
OREGON                  36.0  13.0
PENNSYLVANIA           110.0  20.0
RHODE ISLAND             1.0   0.0
SOUTH CAROLINA          15.0   3.0
TENNESSEE               96.0  47.0
TEXAS                   49.0  22.0
UTAH                    16.0   6.0
VERMONT                  2.0   1.0
VIRGINIA                49.0  11.0
WASHINGTON              75.0  23.0
WEST VIRGINIA            8.0   3.0>
```

- Provide state wise status of complaints in a stacked bar chart. Use the categorized variable from Q3. Provide insights on:

[55]:
```
import matplotlib.pyplot as plt
fig = plt.figure(dpi=140)
df_plot.plot(kind='bar', stacked=True,figsize=(30,30),ax = plt.gca(), width=0.8)
plt.rcParams.update({'font.size':30})
```



[56]:
```
df_plot['Total']=df_plot.Closed+df_plot.Open
```

[57]:
```
df_plot.head()
```

[57]:

CompState	Closed	Open	Total
State			
ALABAMA	17.0	9.0	26.0
ARIZONA	14.0	6.0	20.0
ARKANSAS	6.0	0.0	6.0
CALIFORNIA	159.0	61.0	220.0
COLORADO	58.0	22.0	80.0

Which state has the maximum complaints GEORGIA
Which state has the highest percentage of unresolved complaints KANSAS

[234]:
```
 #   Which state has the maximum complaints
print(df_plot[df_plot.Total==df_plot.Total.max()])
#Georgia
#  Which state has the highest percentage of unresolved complaints
df_plot['PercentUnresolved']=100* (df_plot.Open / df_plot.Total)

df_plot[df_plot.PercentUnresolved==df_plot.PercentUnresolved.max()]
#df_plot['PercentUnresolved']
```

```
CompState  Closed  Open  Total  PercentUnresolved
State
GEORGIA     208.0  80.0  288.0          27.777778
```

[234]:
```
CompState  Closed  Open  Total  PercentUnresolved
State
KANSAS        1.0   1.0    2.0              50.0
```

but as kansas has only 2 complaints and kentucky has only 7 total complaints
we don't have balance in amount of data avilable
we can also look out for Mississippi amd maine for any data driven decisions to check why complaints stay open in certain states

[235]:
```
df_plot.sort_values(by=['PercentUnresolved','Total'],ascending=False)
```

[235]:

State	CompState	Closed	Open	Total	PercentUnresolved
KANSAS		1.0	1.0	2.0	50.000000
KENTUCKY		4.0	3.0	7.0	42.857143
MISSISSIPPI		23.0	16.0	39.0	41.025641
MAINE		3.0	2.0	5.0	40.000000
ALABAMA		17.0	9.0	26.0	34.615385
DELAWARE		8.0	4.0	12.0	33.333333
NEW HAMPSHIRE		8.0	4.0	12.0	33.333333
VERMONT		2.0	1.0	3.0	33.333333
TENNESSEE		96.0	47.0	143.0	32.867133
TEXAS		49.0	22.0	71.0	30.985915
ARIZONA		14.0	6.0	20.0	30.000000
GEORGIA		208.0	80.0	288.0	27.777778
CALIFORNIA		159.0	61.0	220.0	27.727273
COLORADO		58.0	22.0	80.0	27.500000
UTAH		16.0	6.0	22.0	27.272727
WEST VIRGINIA		8.0	3.0	11.0	27.272727
NEW MEXICO		11.0	4.0	15.0	26.666667
OREGON		36.0	13.0	49.0	26.530612
NEW JERSEY		56.0	19.0	75.0	25.333333

```
CONNECTICUT              9.0   3.0   12.0          25.000000
MISSOURI                 3.0   1.0    4.0          25.000000
WASHINGTON              75.0  23.0   98.0          23.469388
MICHIGAN                92.0  23.0  115.0          20.000000
MARYLAND                63.0  15.0   78.0          19.230769
VIRGINIA                49.0  11.0   60.0          18.333333
MASSACHUSETTS           50.0  11.0   61.0          18.032787
ILLINOIS               135.0  29.0  164.0          17.682927
SOUTH CAROLINA          15.0   3.0   18.0          16.666667
FLORIDA                201.0  39.0  240.0          16.250000
PENNSYLVANIA           110.0  20.0  130.0          15.384615
INDIANA                 50.0   9.0   59.0          15.254237
MINNESOTA               29.0   4.0   33.0          12.121212
DISTRICT OF COLUMBIA    15.0   2.0   17.0          11.764706
LOUISIANA               12.0   1.0   13.0           7.692308
ARKANSAS                 6.0   0.0    6.0           0.000000
NEW YORK                 6.0   0.0    6.0           0.000000
NORTH CAROLINA           3.0   0.0    3.0           0.000000
OHIO                     3.0   0.0    3.0           0.000000
IOWA                     1.0   0.0    1.0           0.000000
MONTANA                  1.0   0.0    1.0           0.000000
NEVADA                   1.0   0.0    1.0           0.000000
RHODE ISLAND             1.0   0.0    1.0           0.000000
```

[60]:
```
a.columns
#df=a[a.
recvia_cnt=a.ReceivedVia.value_counts()
total=recvia_cnt[0]+recvia_cnt[1]
recvia_pct=100*(recvia_cnt/total)
recvia_pct
```

[60]:
```
Customer Care Call    50.314748
Internet              49.685252
Name: ReceivedVia, dtype: float64
```

The percentage of complaints resolved till date, which were received through the Internet and customer care calls.

[236]:
```
int_sub=a[a.ReceivedVia=="Internet"]
cnt=int_sub.CompState.value_counts()
total=cnt[0]+cnt[1]

print(cnt)
print(total)
print(100*cnt[0]/total)
#76.28 % of complaints made from internet are resolved
```

```
Closed    843
```

```
Open      262
Name: CompState, dtype: int64
1105
76.289592760181
```

The percentage of complaints resolved till date, which were received through the Internet are 76.28 %

[237]:
```
int_sub=a[a.ReceivedVia=="Customer Care Call"]
cnt=int_sub.CompState.value_counts()
total=cnt[0]+cnt[1]

print(cnt)
print(total)
print(100*cnt[0]/total)

#77.211 % of complaints made from customer care call are resolved
```

```
Closed    864
Open      255
Name: CompState, dtype: int64
1119
77.21179624664879
```

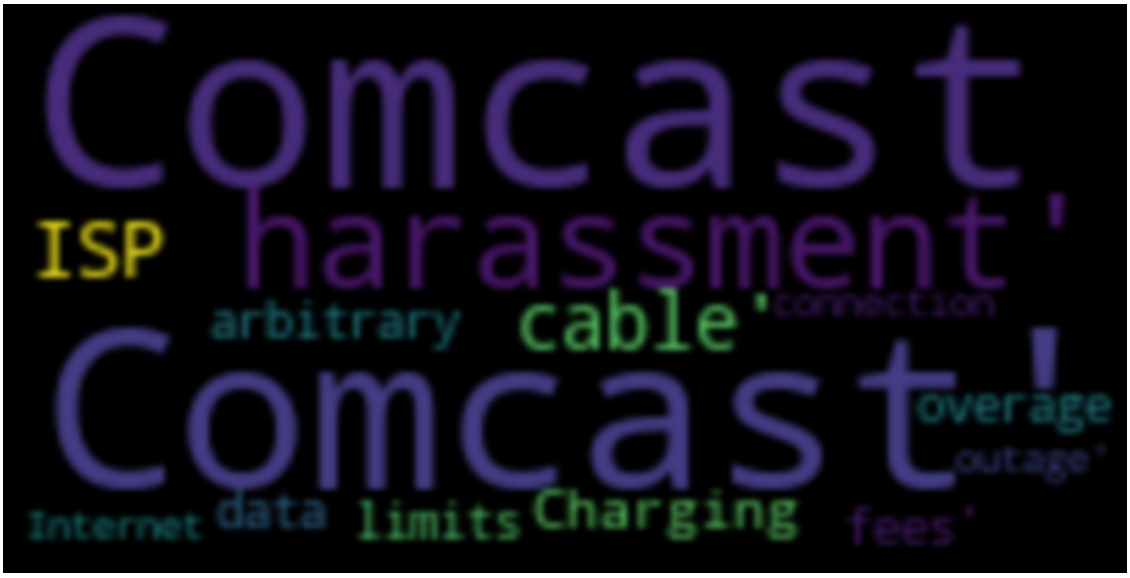
The percentage of complaints resolved till date, which were received through customer care calls. are 77.21%

[63]:
```
#lets get bbetter with complaint type uusing nlp
from wordcloud import WordCloud, STOPWORDS
```

[64]:
```
data=a.CustomerComplaint.values
```

[65]:
```
wc=WordCloud(width=200,height=100,background_color='black',stopwords=STOPWORDS).
 ↪generate(str(data))
```

[66]:
```
fig=plt.figure(figsize=(20,29),facecolor='k',edgecolor='w')
plt.imshow(wc,interpolation='bicubic')
plt.axis("off")
plt.tight_layout()
plt.show()
```



we will use LDA TO GET ACCURATE CATEGORIZATION

[67]:
```
from nltk.corpus import stopwords
```

[68]:
```
from nltk.stem.wordnet import WordNetLemmatizer
```

[69]:
```
import string
```

[70]:
```
stop = set(stopwords.words('english'))
```

[71]:
```
exclude = set(string.punctuation)
```

[72]:
```
lemmma= WordNetLemmatizer() #base word conversion for bbetter tuning and␣
↪performance
```

[73]:
```
def clean(doc):
    stop_free=" ".join([i for i in doc.lower().split() if i not in stop])
    punc_free="".join([char for char in stop_free if char not in exclude])
    normalisation = " ".join(lemmma.lemmatize(word) for word in punc_free.
↪split(' '))
    return normalisation
```

[74]:
```
document=a['CustomerComplaint'].to_list()
```

[75]:
```
doc_clean=[clean(docu).split() for docu in document ]
```

[76]:
```
doc_clean[:10]
```

[76]:
```
[['comcast', 'harassment'],
 ['comcast', 'cable'],
 ['comcast'],
 ['comcast', 'refusal', 'service'],
 ['horrible', 'service'],
 ['billing'],
 ['unable', 'get', 'touch', 'anyone', 'power', 'cancel', 'service'],
 ['fraudulent', 'claim', 'reported', 'collection', 'agency'],
 ['internet', 'service'],
 ['comcast', 'lied', 'pricing', 'installation']]
```

[77]:
```
import gensim
```

[78]:
```
from gensim import corpora
```

[79]:
```
dictionary=corpora.Dictionary(doc_clean)
```

[80]:
```
print(dictionary)
```

```
Dictionary(1412 unique tokens: ['comcast', 'harassment', 'cable', 'refusal',
'service']…)
```

[81]:
```
doc_word_freqcies=[dictionary.doc2bow(term) for term in doc_clean]
doc_word_freqcies[:30]
```

[81]:
```
[[(0, 1), (1, 1)],
 [(0, 1), (2, 1)],
 [(0, 1)],
 [(0, 1), (3, 1), (4, 1)],
 [(4, 1), (5, 1)],
 [(6, 1)],
 [(4, 1), (7, 1), (8, 1), (9, 1), (10, 1), (11, 1), (12, 1)],
 [(13, 1), (14, 1), (15, 1), (16, 1), (17, 1)],
 [(4, 1), (18, 1)],
 [(0, 1), (19, 1), (20, 1), (21, 1)],
 [(0, 1),
  (22, 1),
  (23, 1),
  (24, 1),
  (25, 1),
  (26, 2),
  (27, 1),
  (28, 1),
  (29, 1),
  (30, 1)],
 [(0, 1), (2, 1)],
 [(31, 1)],
```

```
 [(0, 1), (4, 1), (32, 1), (33, 1), (34, 1)],
 [(6, 1), (35, 1)],
 [(36, 1), (37, 1)],
 [(4, 1), (38, 1)],
 [(0, 1), (39, 1), (40, 1), (41, 1), (42, 1), (43, 1), (44, 1)],
 [(18, 1),
  (45, 1),
  (46, 1),
  (47, 1),
  (48, 1),
  (49, 1),
  (50, 1),
  (51, 2),
  (52, 1),
  (53, 1),
  (54, 1),
  (55, 1)],
 [(0, 1), (48, 1), (56, 1)],
 [(48, 1), (51, 1)],
 [(4, 1), (57, 1), (58, 1), (59, 1)],
 [(0, 1)],
 [(0, 1), (4, 1), (60, 1), (61, 1)],
 [(0, 1)],
 [(6, 1), (16, 1)],
 [(4, 1), (12, 1), (62, 1), (63, 1), (64, 1), (65, 1)],
 [(0, 1), (66, 1), (67, 1), (68, 1), (69, 1), (70, 1)],
 [(35, 1), (62, 1)],
 [(16, 1), (71, 1), (72, 1), (73, 1), (74, 1)]]
```

[82]:
```
from gensim.models import LdaModel
```

[150]:
```
model=LdaModel(doc_word_freqcies,num_topics=5,id2word=dictionary,passes=1000)
```

[151]:
```
types= model.show_topics()
for t in types:
    print(t)
    print('----------------')
```

```
(0, '0.231*"comcast" + 0.107*"service" + 0.081*"internet" + 0.063*"billing" +
0.030*"customer" + 0.027*"issue" + 0.026*"complaint" + 0.018*"charge" +
0.015*"cable" + 0.012*"problem"')
----------------
(1, '0.035*"fee" + 0.034*"charged" + 0.033*"charging" + 0.030*"bill" +
0.023*"service" + 0.017*"modem" + 0.014*"month" + 0.013*"lower" + 0.013*"never"
+ 0.012*"rental"')
----------------
(2, '0.127*"comcast" + 0.122*"data" + 0.101*"cap" + 0.025*"slow" +
```

```
0.022*"comcastxfinity" + 0.019*"usage" + 0.018*"throttling" + 0.013*"email" +
0.011*"overage" + 0.009*"help"')
----------------
(3, '0.058*"service" + 0.055*"comcast" + 0.051*"practice" + 0.043*"billing" +
0.031*"unfair" + 0.017*"hbo" + 0.015*"deceptive" + 0.014*"issue" + 0.011*"call"
+ 0.011*"false"')
----------------
(4, '0.133*"internet" + 0.095*"speed" + 0.063*"comcast" + 0.035*"service" +
0.016*"xfinity" + 0.016*"connection" + 0.015*"pricing" + 0.014*"paying" +
0.013*"price" + 0.013*"high"')
----------------
```

[152]:
```
diction={}
for i in range(5):
    words=model.show_topic(i,topn=20)
    #print(words)
    diction["Topic number" + "{}".format(i)]=[i[0] for i in words]


pd.DataFrame(diction)
```

[152]:

	Topic number0	Topic number1	Topic number2	Topic number3	Topic number4
0	comcast	fee	comcast	service	internet
1	service	charged	data	comcast	speed
2	internet	charging	cap	practice	comcast
3	billing	bill	slow	billing	service
4	customer	service	comcastxfinity	unfair	xfinity
5	issue	modem	usage	hbo	connection
6	complaint	month	throttling	deceptive	pricing
7	charge	lower	email	issue	paying
8	cable	never	overage	call	price
9	problem	rental	help	false	high
10	poor	owned	charge	go	advertised
11	failure	cable	promised	switch	without
12	unauthorized	charge	limit	sale	issue
13	bad	provided	pay	lack	access
14	account	overcharged	cancellation	complaint	business
15	phone	back	credit	misleading	connectivity
16	horrible	lied	plan	bait	poor
17	tv	tv	failure	ps4	day
18	fraud	technician	changed	xfinitycomcast	inconsistent
19	provide	failing	content	business	bill

[153]:
```
import pyLDAvis.gensim
```

[154]:
```
Vis=pyLDAvis.gensim.
 ↪prepare(model,doc_word_freqcies,dictionary,sort_topics=False)
```

[155]:
```
pyLDAvis.display(Vis)
#topic 2 and 4  is billing practice
#topic 3 is data cap issues
#topic 5 IS internet  speed
#topic 1 is internet and cable service

#
# billing practice and fairness has highest frequency
#topic data cap is 2nd most frequent
# topic internet speed is 3rd most frequent
```

[155]:
```
<IPython.core.display.HTML object>
```

billing practice and fairness has highest frequency #topic data cap is 2nd most frequent # topic internet speed is 3rd most frequent

[156]:
```
# Create Corpus: Term Document Frequency
corpus = [dictionary.doc2bow(text) for text in doc_clean]
```

[199]:
```
def format_topics_sentences(ldamodel=None, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row_list in enumerate(ldamodel[corpus]):
        row = row_list[0] if ldamodel.per_word_topics else row_list
        # print(row)
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each␣
↪document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.
↪Series([int(topic_num), round(prop_topic,4), topic_keywords]),␣
↪ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution',␣
↪'Topic_Keywords']

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
    return(sent_topics_df)
```

```
df_topic_sents_keywords = format_topics_sentences(ldamodel=model,␣
 ↪corpus=corpus, texts=doc_clean)

# Format
df_dominant_topic = df_topic_sents_keywords.reset_index()
df_dominant_topic.columns = ['Document_No', 'Dominant_Topic',␣
 ↪'Topic_Perc_Contrib', 'Keywords', 'Text']
df_dominant_topic.head(10)
```

[199]:

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	0.0	0.7320	comcast, service, internet, billing, customer,…	[comcast, harassment]
1	1	0.0	0.7307	comcast, service, internet, billing, customer,…	[comcast, cable]
2	2	0.0	0.5961	comcast, service, internet, billing, customer,…	[comcast]
3	3	3.0	0.4545	service, comcast, practice, billing, unfair, h…	[comcast, refusal, service]
4	4	0.0	0.7322	comcast, service, internet, billing, customer,…	[horrible, service]
5	5	0.0	0.5973	comcast, service, internet, billing, customer,…	[billing]
6	6	0.0	0.8992	comcast, service, internet, billing, customer,…	[unable, get, touch, anyone, power, cancel, se…
7	7	2.0	0.6700	comcast, data, cap, slow, comcastxfinity, usag…	[fraudulent, claim, reported, collection, agency]
8	8	0.0	0.7302	comcast, service, internet, billing, customer,…	[internet, service]
9	9	0.0	0.3921	comcast, service, internet, billing, customer,…	[comcast, lied, pricing, installation]

[201]:
```
vc=df_dominant_topic.Dominant_Topic.value_counts()
vc
```

[201]:
```
0.0    937
4.0    452
2.0    416
3.0    313
1.0    106
Name: Dominant_Topic, dtype: int64
```

[192]:
```
#topic 2 and 4  is billing practice
#topic 3 is data cap issues
#topic 5 IS internet  speed
#topic 1 is internet and cable service
dic={1.0:"internet and cable service",2.0:"billing practice",3.0:"data cap␣
 ↪issues",4.0:"billing practice",5.0:"internet speed",0.0:"no type"}
vc=df_dominant_topic.Dominant_Topic.value_counts()
```

[202]:
```
dt=df_dominant_topic[["Dominant_Topic"]]
```

[203]:
```
dt
```

[203]:
```
      Dominant_Topic
0                0.0
1                0.0
2                0.0
3                3.0
4                0.0
…                …
2219             0.0
2220             0.0
2221             0.0
2222             2.0
2223             4.0

[2224 rows x 1 columns]
```

[207]:
```
dt.Dominant_Topic=dt.Dominant_Topic.apply(lambda row: dic[row])
```

```
/usr/local/lib/python3.7/site-packages/pandas/core/generic.py:5303:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self[name] = value
```

[217]:
```
pd.DataFrame(dt.Dominant_Topic.value_counts())
```

[217]:

	Dominant_Topic
no type	937
billing practice	868
data cap issues	313
internet and cable service	106

complaint type with its count/ frequency

Max complaints are on billing practices

[ ]:
```

```