In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

from pprint import pprint
```

In [2]:

```
df=pd.read_csv('Comcast_telecom_complaints_data.csv' ,parse_dates=['Date_month_ye
df.head(3)
```

Out[2]:

	Ticket #	Customer Complaint	Date	Date_month_year	Time
0	250635	Comcast Cable Internet Speeds	22-04-15	2015-04-22	2019-10-20 15:53:50
1	223441	Payment disappear - service got disconnected	04-08-15	2015-08-04	2019-10-20 10:22:56
2	242732	Speed and Service	18-04-15	2015-04-18	2019-10-20 09:55:47

In [3]:

```
df.dtypes
```

Out[3]:

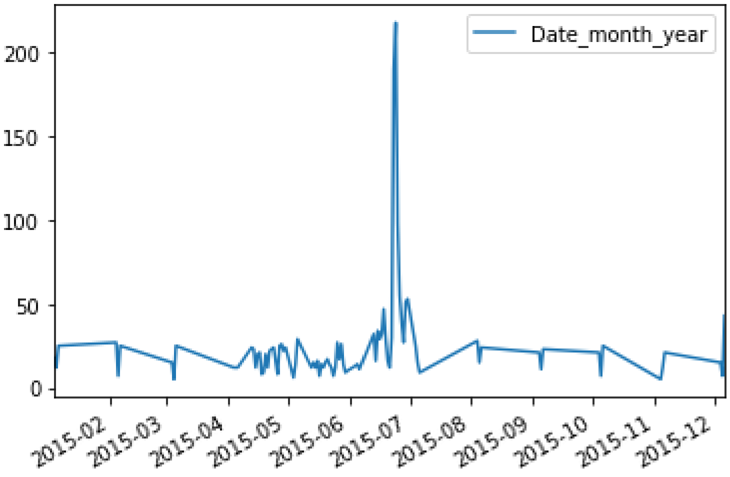
```
Ticket #                                object
Customer Complaint                      object
Date                                    object
Date_month_year                 datetime64[ns]
Time                            datetime64[ns]
Received Via                            object
City          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

Problem 1.1:

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

In [4]:
```
#trends of number of complaints at daily
df_days=pd.DataFrame(df['Date_month_year'].value_counts())
df_days.sort_index().plot()
```

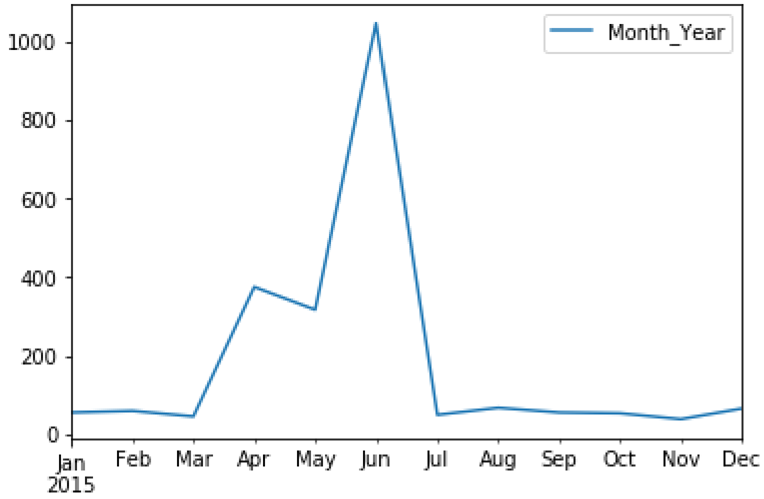
Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15d7dca7be0>
```



In [5]:
```
#trends of number of complaints at monthly
df['Month_Year']=df['Date_month_year'].dt.to_period('M')
df_month=pd.DataFrame(df['Month_Year'].value_counts())
```

In [6]:
```
df_month.sort_index().plot()
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15d00dec710>
```



Problem 1.2

Provide a table with the frequency of complaint types. Which complaint types are maximum i.e., arour

this is a problem of Topic Modeling which is a branch of NLP

In [7]:
```
data_text = df[['Customer Complaint']]
data_text['index'] = data_text.index
documents = data_text
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCo
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: http://pandas.pydata.org/pandas-docs/stable,
ng.html#indexing-view-versus-copy)
```

In [8]:
```
print(len(documents))
print(documents[:5])
```

```
2224
                                    Customer Complaint  index
0                        Comcast Cable Internet Speeds      0
1       Payment disappear - service got disconnected      1
2                                    Speed and Service      2
3  Comcast Imposed a New Usage Cap of 300GB that ...      3
4          Comcast not working and no service to boot      4
```

In [9]:
```
# Gensim
import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess
from gensim.models import CoherenceModel

# wordnet for lemmatization
from nltk.stem.wordnet import WordNetLemmatizer

# Plotting tools
import pyLDAvis
import pyLDAvis.gensim  # don't skip this
```

In [10]:
```
# Enable logging for gensim - optional
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=log

import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning)
```

In [11]:
```
# NLTK Stop words
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['from', 'subject', 're', 'edu', 'use'])
```

In [12]:

```
import re
data=documents['Customer Complaint'].values.tolist()
# Remove new line characters
data = [re.sub('\s+', ' ', sent) for sent in data]

# Remove distracting single quotes
data = [re.sub("\'", "", sent) for sent in data]

pprint(data[:5])
```

```
['Comcast Cable Internet Speeds',
 'Payment disappear - service got disconnected',
 'Speed and Service',
 'Comcast Imposed a New Usage Cap of 300GB that punishes streaming.',
 'Comcast not working and no service to boot']
```

In [13]:

```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # deacc

data_words = list(sent_to_words(data))

print(data_words[:5])
```

```
[['comcast', 'cable', 'internet', 'speeds'], ['payment', 'disappear', 'service',
'cap', 'of', 'gb', 'that', 'punishes', 'streaming'], ['comcast', 'not', 'working',
```

In [14]:

```
# Build the bigram and trigram models
bigram = gensim.models.Phrases(data_words, min_count=5, threshold=100) # higher t
trigram = gensim.models.Phrases(bigram[data_words], threshold=100)

# Faster way to get a sentence clubbed as a trigram/bigram
bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)

# See trigram example
print(trigram_mod[bigram_mod[data_words[0]]])
```

```
['comcast', 'cable', 'internet', 'speeds']
```

In [16]:

```
# Define functions for stopwords, bigrams, trigrams and lemmatization
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not in stop_wor

def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

def make_trigrams(texts):
    return [trigram_mod[bigram_mod[doc]] for doc in texts]

def lemmatization(text):
    texts_out=" ".join(lemma.lemmatize(word) for word in text)
    return texts_out
```

In [17]:
```
# Remove Stop Words
data_words_nostops = remove_stopwords(data_words)

# Form Bigrams
data_words_bigrams = make_bigrams(data_words_nostops)
#create WordNetLemmatizer
lemma=WordNetLemmatizer()
# Do lemmatization keeping only noun, adj, vb, adv
data_lemmatized = [lemmatization(text).split() for text in data_words_bigrams]
print(data_lemmatized[:5])
```

```
[['comcast', 'cable', 'internet', 'speed'], ['payment', 'disappear', 'service', 'g
'gb', 'punishes', 'streaming'], ['comcast', 'working', 'service', 'boot']]
```

In [18]:
```
# Create Dictionary
id2word = corpora.Dictionary(data_lemmatized)

# Create Corpus
texts = data_lemmatized

# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]

# View
print(corpus[:10])
```

```
[[(0, 1), (1, 1), (2, 1), (3, 1)], [(4, 1), (5, 1), (6, 1), (7, 1), (8, 1)], [(3,
1), (8, 1), (16, 1), (17, 1)], [(18, 1), (19, 1), (20, 1), (21, 1), (22, 1), (23,
1), (29, 1), (30, 1)], [(1, 1), (31, 1), (32, 1)], [(1, 1), (33, 1), (34, 1), (35,
```

In [19]:
```
id2word[0]
```

Out[19]:
```
'cable'
```

In [20]:
```
# Human readable format of corpus (term-frequency)
[[(id2word[id], freq) for id, freq in cp] for cp in corpus[:1]]
```

Out[20]:
```
[[('cable', 1), ('comcast', 1), ('internet', 1), ('speed', 1)]]
```

In [21]:
```
lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                           id2word=id2word,
                                           num_topics=10,
                                           update_every=1,
                                           chunksize=100,
                                           passes=10,
                                           alpha='auto',
                                           per_word_topics=True)
```

In [22]:
```
# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

```
[(0,
  '0.099*"price" + 0.061*"switch" + 0.060*"bait" + 0.049*"refund" + '
  '0.044*"availability" + 0.040*"account" + 0.026*"plan" + 0.021*"misleading" '
  '+ 0.019*"bandwidth" + 0.018*"gb"'),
 (1,
  '0.332*"billing" + 0.083*"false" + 0.041*"overcharge" + 0.033*"significant" '
  '+ 0.032*"reimburse" + 0.032*"admit" + 0.021*"unauthorized" + 0.017*"fraud" '
  '+ 0.012*"resolution" + 0.012*"improper"'),
 (2,
  '0.072*"charged" + 0.058*"fee" + 0.054*"usage" + 0.045*"contract" + '
  '0.037*"paying" + 0.030*"failure" + 0.029*"charging" + 0.024*"year" + '
  '0.022*"rate" + 0.021*"mb"'),
 (3,
  '0.118*"billing" + 0.107*"customer" + 0.077*"charge" + 0.075*"practice" + '
  '0.067*"unfair" + 0.058*"cable" + 0.042*"poor" + 0.028*"connectivity" + '
  '0.027*"modem" + 0.026*"quality"'),
 (4,
  '0.185*"bill" + 0.044*"fraudulent" + 0.030*"show" + 0.029*"incorrect" + '
  '0.026*"cramming" + 0.018*"promotion" + 0.018*"claim" + 0.014*"said" + '
  '0.014*"people" + 0.014*"way"'),
 (5,
  '0.364*"internet" + 0.127*"speed" + 0.121*"data" + 0.111*"cap" + '
  '0.027*"slow" + 0.023*"high" + 0.017*"connection" + 0.014*"intermittent" + '
  '0.013*"mi" + 0.013*"install"'),
 (6,
  '0.055*"equipment" + 0.045*"advertising" + 0.041*"promised" + 0.040*"phone" '
  '+ 0.033*"business" + 0.031*"returned" + 0.030*"check" + 0.029*"overage" + '
  '0.028*"miss" + 0.027*"email"'),
 (7,
  '0.606*"comcast" + 0.060*"issue" + 0.052*"complaint" + 0.047*"xfinity" + '
  '0.029*"pricing" + 0.020*"problem" + 0.011*"cost" + 0.011*"lied" + '
  '0.009*"tv" + 0.007*"outage"'),
 (8,
  '0.477*"service" + 0.029*"monthly" + 0.026*"without" + 0.023*"payment" + '
  '0.021*"extremely" + 0.018*"terrible" + 0.018*"day" + 0.017*"help" + '
  '0.016*"get" + 0.016*"advertised"'),
 (9,
  '0.143*"throttling" + 0.049*"access" + 0.044*"monopoly" + 0.043*"output" + '
  '0.034*"blocking" + 0.029*"hbo_go" + 0.028*"time" + 0.024*"p" + 0.019*"isp" '
  '+ 0.017*"ordered"')]
```

How to interpret this?
Topic 0 is a represented as '0.099*"price" + 0.061*"switch" + 0.060*"bait" + 0.049*"refund" + 0.044*"a
0.018*"gb"'
It means the top 10 keywords that contribute to this topic are: ‘price’, ‘switch’, ‘bait’.. and so on and th

Compute Model Perplexity and Coherence Score

In [23]:
```
# Compute Perplexity
print('\nPerplexity: ', lda_model.log_perplexity(corpus))  # a measure of how goo

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model, texts=data_lemmatized, dict
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```

Perplexity:  -6.0493189615045155

Coherence Score:  0.6437856159124845
```

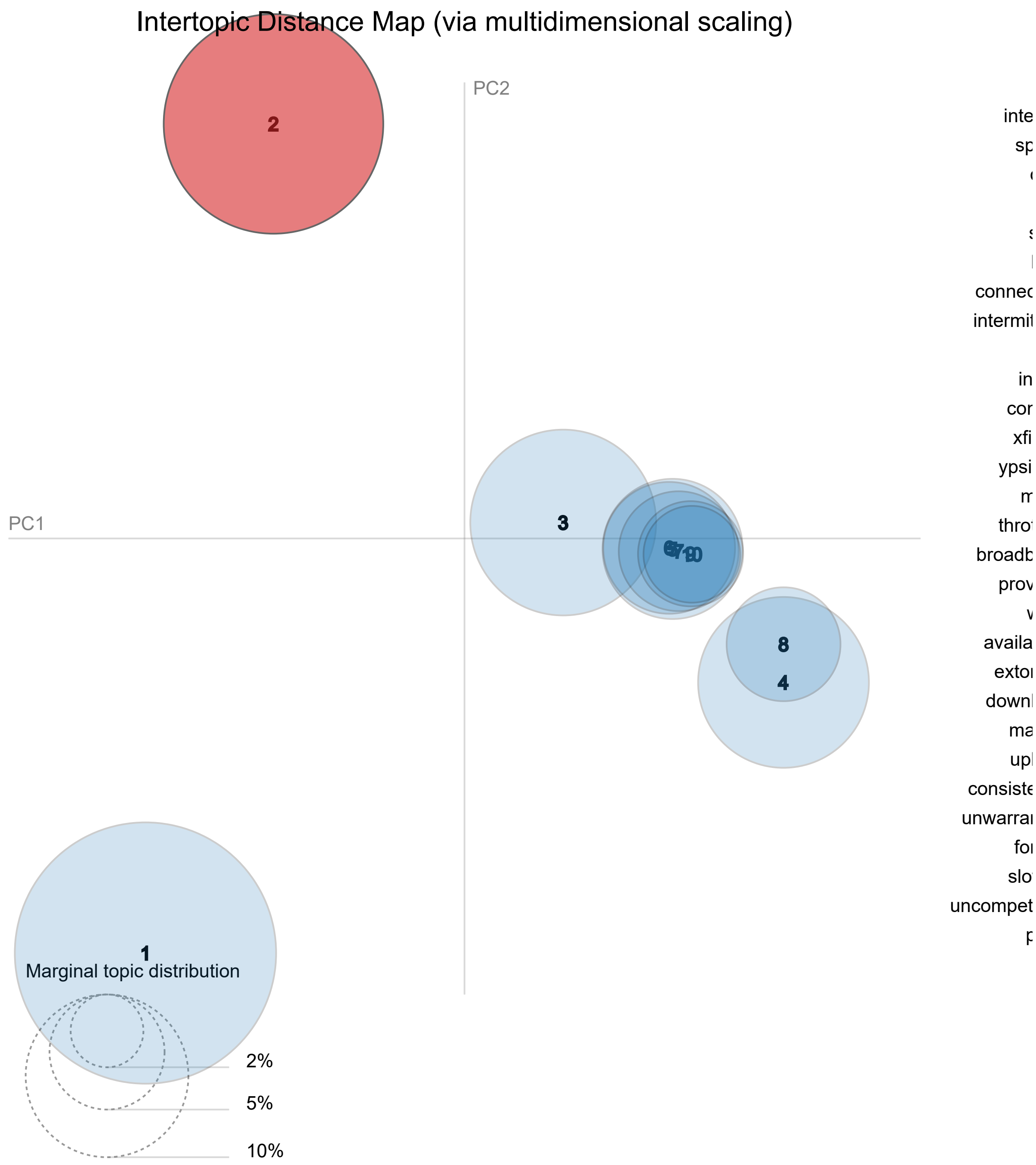
In [24]:

```
# Visualize the topics
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(lda_model, corpus, id2word)
vis
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pyLDAvis\_prepare.py:257: FutureWarning
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  return pd.concat([default_term_info] + list(topic_dfs))
```

Out[24]:

Selected Topic: 2 | Previous Topic | Next Topic | Clear Topic



Problem 1.3

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

```
In [63]: df['Status'].unique()
```

```
Out[63]: array(['Closed', 'Open', 'Solved', 'Pending'], dtype=object)
```

In [66]:
```
df['ModifiedStatus']=["Open" if status=='Open' or status=='Pending' else "Closed"
df['ModifiedStatus'].head()
```

Out[66]:
```
0    Closed
1    Closed
2    Closed
3      Open
4    Closed
Name: ModifiedStatus, dtype: object
```

Problem 1.4

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

- A. Which state has the maximum complaints
- B. Which state has the highest percentage of unresolved complaints

In [75]:
```
#A. which state has the maximum complaints
df_statewise_complaints=df.groupby(["State"]).size().sort_values(ascending=False)
df_statewise_complaints.head(10)
```

Out[75]:

	State	Count
0	Georgia	288
1	Florida	240
2	California	220
3	Illinois	164
4	Tennessee	143
5	Pennsylvania	130
6	Michigan	115
7	Washington	98
8	Colorado	80
9	Maryland	78

In [76]:
```
#state having maximum complaints
df_statewise_complaints['State'][0]
```

Out[76]:
```
'Georgia'
```

In [77]:

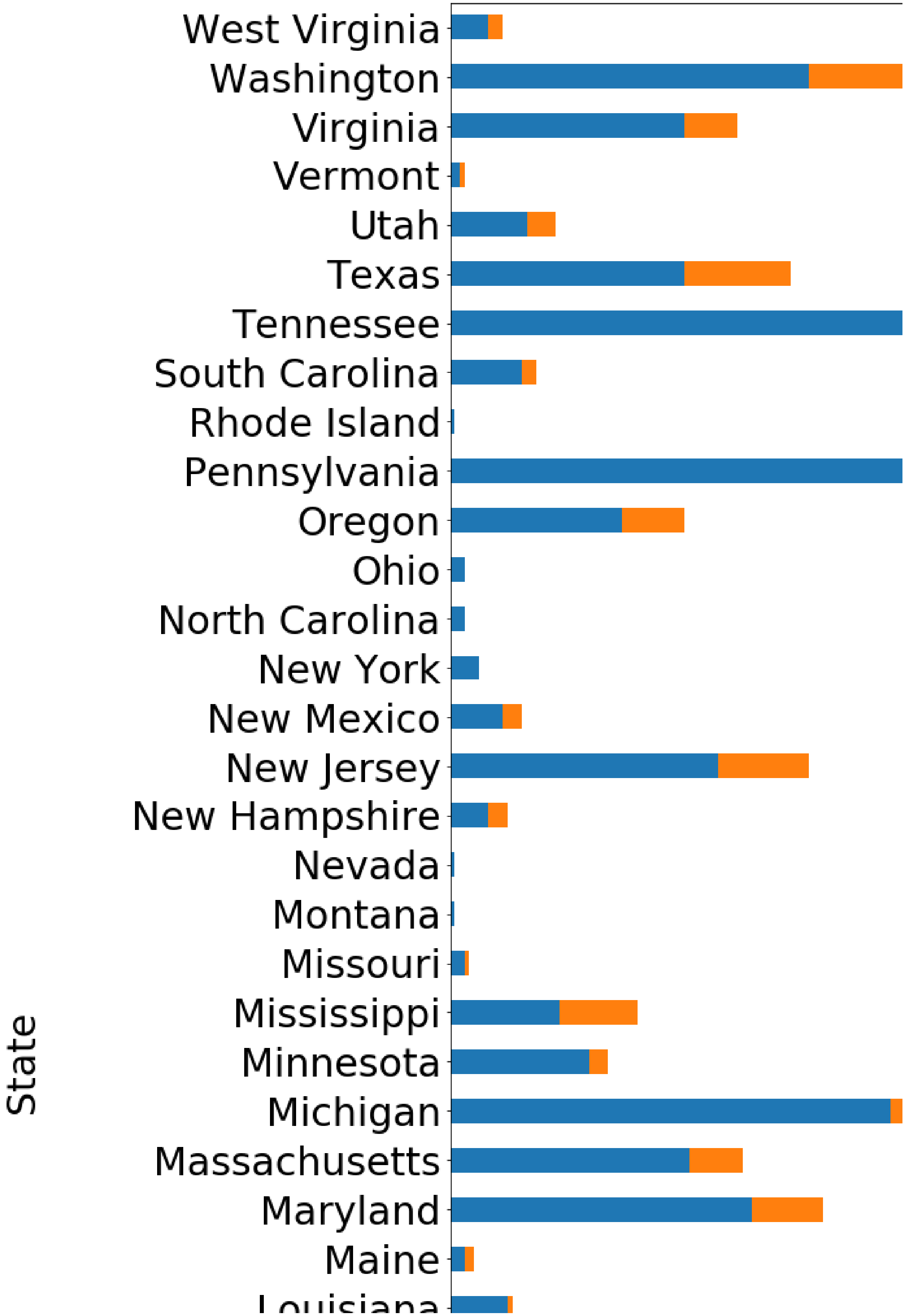
```
#B.Which state has the highest percentage of unresolved complaints
status_complaints = df.groupby(["State", "ModifiedStatus"]).size().unstack()
status_complaints
```

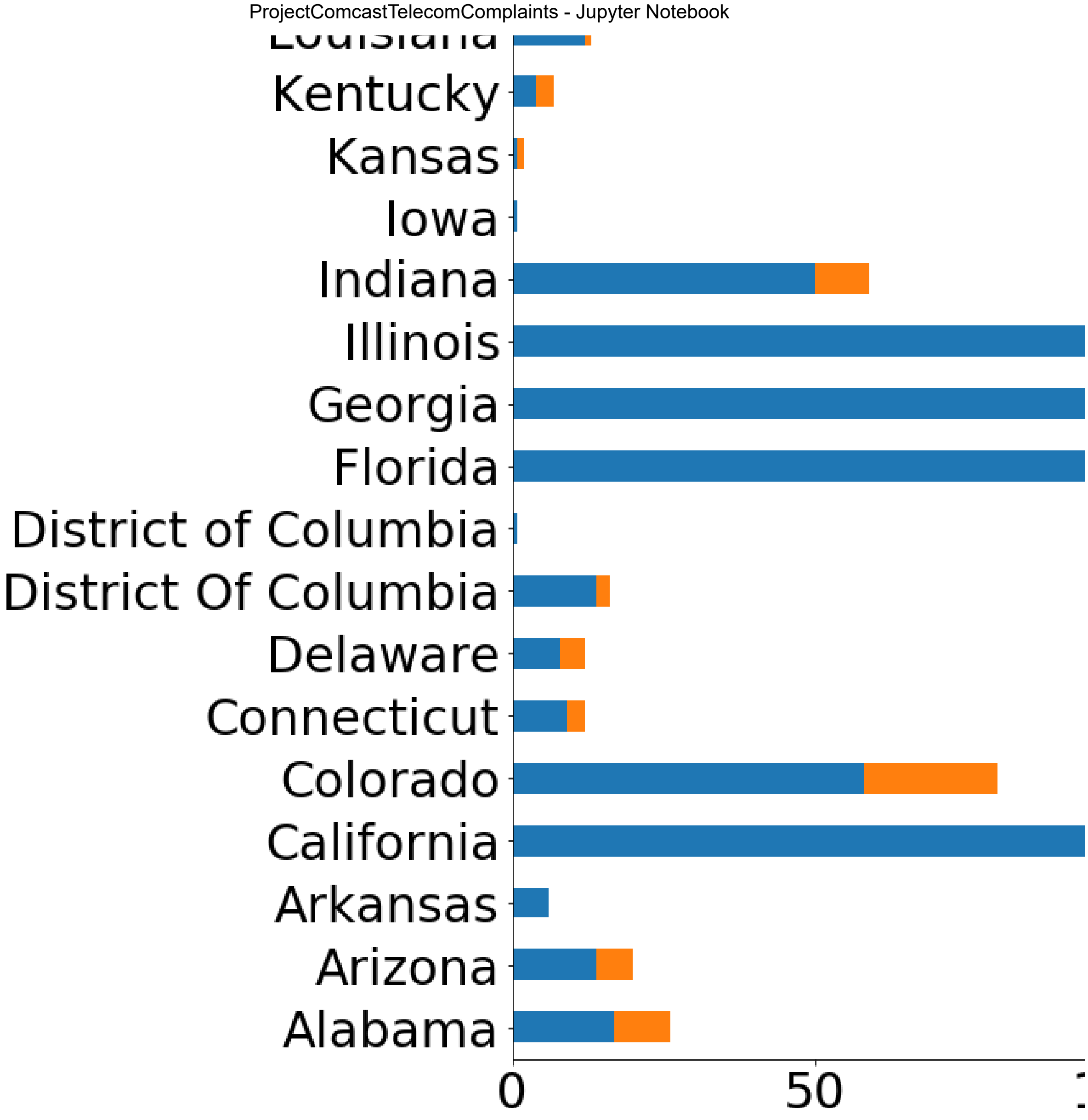
Out[77]:

ModifiedStatus	Closed	Open
State		
Alabama	17.0	9.0
Arizona	14.0	6.0
Arkansas	6.0	NaN
California	159.0	61.0
Colorado	58.0	22.0
Connecticut	9.0	3.0
Delaware	8.0	4.0
District Of Columbia	14.0	2.0
District of Columbia	1.0	NaN
Florida	201.0	39.0
Georgia	208.0	80.0
Illinois	135.0	29.0
Indiana	50.0	9.0
Iowa	1.0	NaN
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	NaN
Nevada	1.0	NaN
New Hampshire	8.0	4.0
New Jersey	56.0	19.0
New Mexico	11.0	4.0
New York	6.0	NaN
North Carolina	3.0	NaN
Ohio	3.0	NaN

ModifiedStatus	Closed	Open
State		
Oregon	36.0	13.0
Pennsylvania	110.0	20.0
Rhode Island	1.0	NaN
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

In [92]:
```
status_complaints=status_complaints.fillna(0)
status_complaints.plot(kind="barh", figsize=(20,30), stacked=True)
plt.rcParams.update({"font.size": 30})
```





```
In [91]: status_complaints.loc[status_complaints['Open'].idxmax()].name
```

```
Out[91]: 'Georgia'
```

Problem 1.4

Provide the percentage of complaints resolved till date, which were received through the Internet and

In [96]:

```
status_received_via = df.groupby(["Received Via", "ModifiedStatus"]).size().unsta
status_received_via['% of Resolved']=100*status_received_via['Closed']/(status_re
status_received_via
```

Out[96]:

ModifiedStatus	Closed	Open	% of Resolved
Received Via			
Customer Care Call	864	255	77.211796
Internet	843	262	76.289593