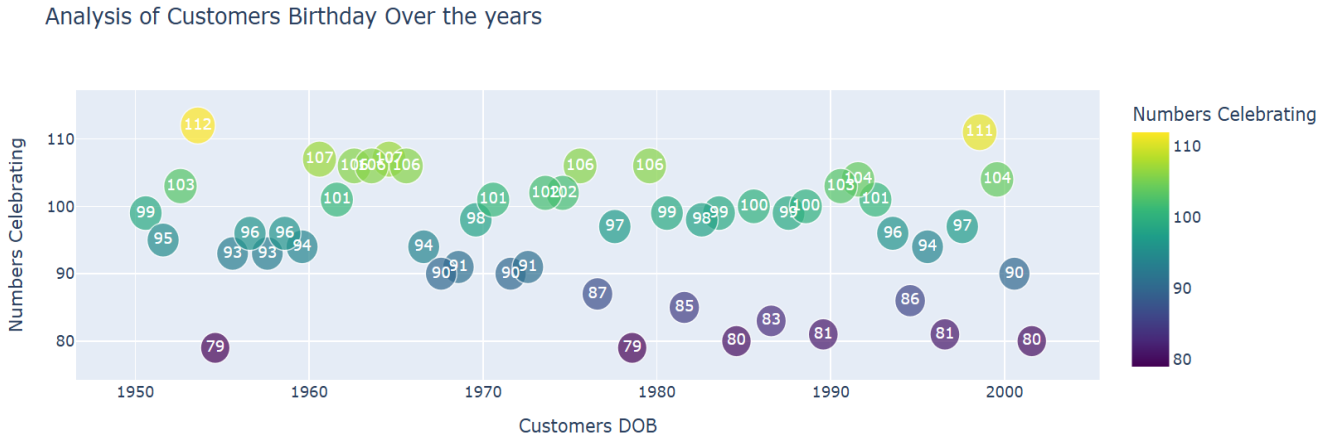
Mitigating Misclassification Risk in Customer Sentiment Intensity Due to Data Entry and Representation Defects to Enhance CRM Practice for JC Penney Retail Shop

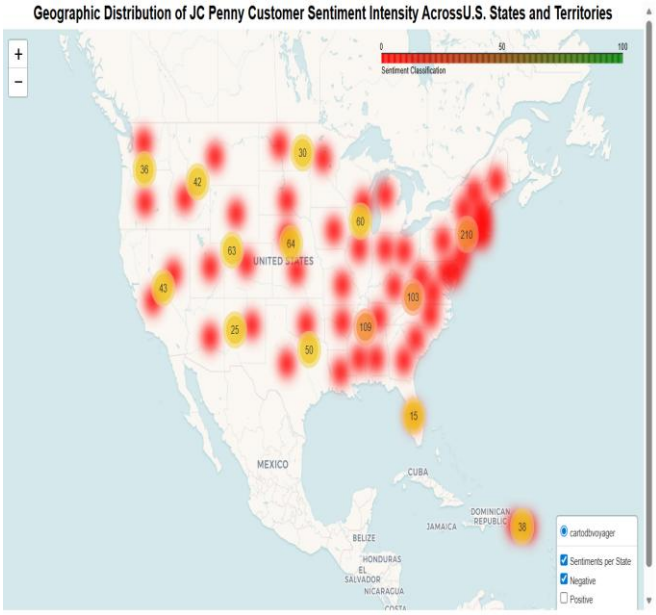
1. Introduction:

JC Penney was established in 1902, many decades before the current explosion of digital evolution. Then, the word of mouth was a trusted means for the spread of customer feedback and recommendation of goods and services. But today, customers increasingly rely on digital customer reviews and feedback to make purchasing decision. Digital customer reviews and feedback is now regarded as the modern word of mouth, widely available and mostly trusted in shopping perception concerning the quality of products and reliability. Given the right analytic environment, it also becomes a vital material for effective customer relationship management (CRM) as they can provide a direct pulse on customer sentiments and offer actionable insights that business can leverage to improve products, services and overall customer loyalty.
As a result, this document provides a report of analytical study into JC Penney historical data ranging from 08/08/1950 to 26/07/2001 with a view to extract actionable insights from the spread of the intensity of customer sentiments to enhance CRM practice in the current digital retail environment.

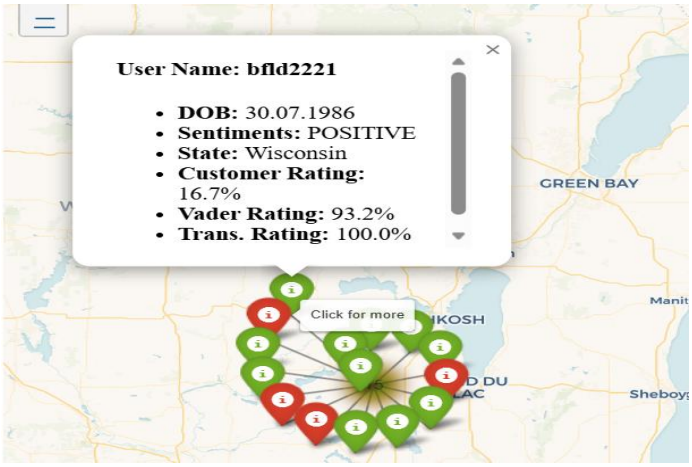


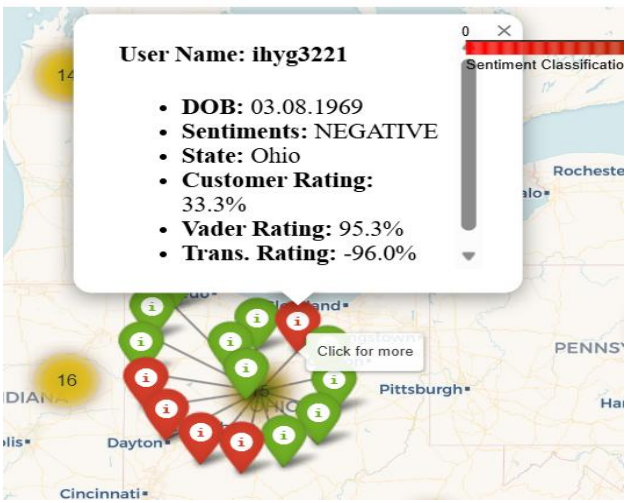
Research into the data reveals that the organisation has a record of 5000 customers but with 4999 Unique Usernames registered on their system during the period under review because of duplication of data. From additional studies it was gathered that their customer outreach is spread over 647 stores across 49 stores in the US and Pueto Rico.
The chart above shows the distribution of the numbers of customers' birth date per month over the years amounting to a total of 52 birth dates with a cluster of customers having their birthdays on the same day. While the maximum cluster was found on 07/08/1953 with 112 birth dates, the minimum numbers wason 08/08/1950 with 79 numbers of birth dates. The most recent cluster of birth date was on 26/07/2001 with 80 in numbers. The general trend shows that there is no consistent increase in the numbers of birth dates per month and as a matter of fact, there is a sharp fall in numbers between July 1988(111) and July 2001(80) which lead to the deduction that there is no observable growth in customer loyalty over a period of 50 decades which is not a representative of a business growth. This is true because an increase in the numbers of users will increase the numbers of birth dates in the system. As a result of this problem, additional analysis is made into the customer feedback data of the organisation with a view to excavate the pulse and the spread of the intensity of customer sentiments at the state level and among individual customers to provide datacentric business intelligence for customer relationship management . The result is presented in a demographical map showing positive and negative spread of sentiments at state level and among individuals and a Vader rating chart that classifies the sentiments into positive, neutral and negative by state. See screen shots on page 2.

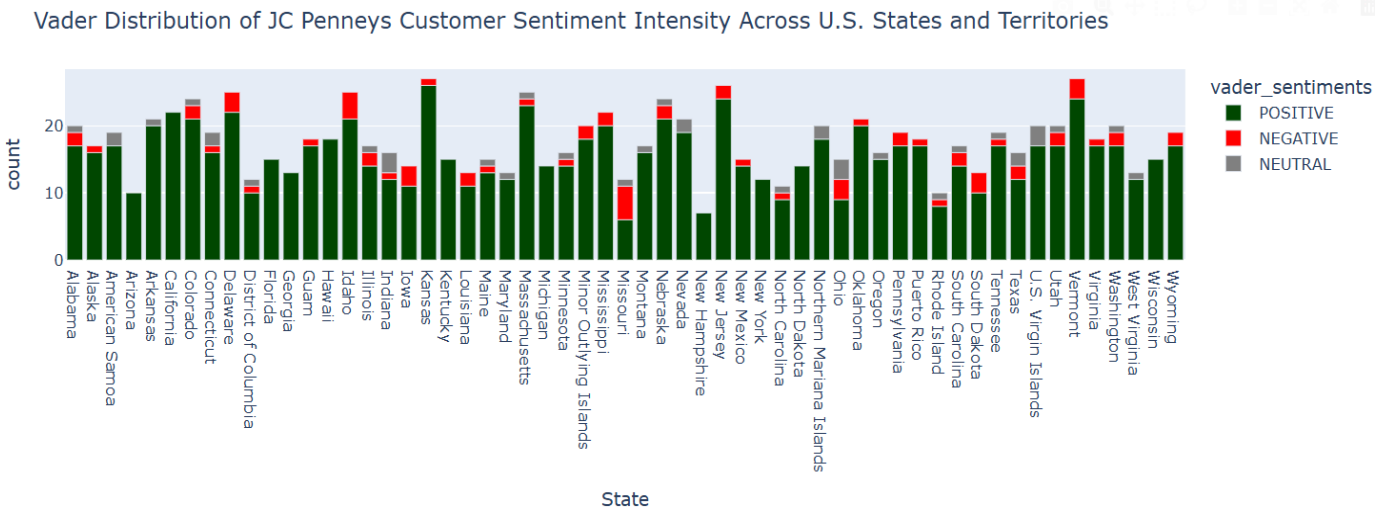
Spread of Positive and Negative Sentiments at State level



Spread of Positive and Negative Sentiments at Individual level







2. Data Quality Analysis

(a). Null values and empty list

In the products.csv, the description feature contains 543 null values while the price feature contains 2166 null values which made them unsuitable for computational calculation

(b).Duplicated Username

	Username	DOB	State	Reviewed
731	dqft3311	28.07.1995	Tennessee	[5f280fb338485cfc30678998a42f0a55]
2619	dqft3311	03.08.1969	New Mexico	[571b86d307f94e9e8d7919b551c6bb52]

Jcpenney_reviewers.json contains the duplicated user name 'dqft3311' as shown in the table above which is registered to two different users from different state and with different date of birth.

In the reviews.csv table shown on the right, this single duplicated username generated 17 rows of transactional data and there is no additional data to show whether they belong to the user from Tennessee or New Mexico. This makes the transactions by this username unsuitable for business due to conflict of interest and misleading outcome. Duplicated data is also a precursor to threat of cyber vulnerability as it compromises the integrity of the database.

	Uniq_id	Username	Score	Review
4331	e5bdf53f2374569526c9f4d55afdd88e	dqft3311	0	We wanted something to warm us from below not ...
6991	5f280fb338485cfc30678998a42f0a55	dqft3311	2	This dress is very pretty and flattering to al...
8142	6c3c7832675727669a52f794cdec743d	dqft3311	0	The shoes run big and fit very loose. I wear a...
8499	d7aafcd766f15954d797ebf5d6e12416	dqft3311	2	It is beautiful with the rest of the inspire t...
13739	9f140ea52aee99d65d94753c4e5ed3d6	dqft3311	1	These shorts fit perfectly. Just the right fab...
14350	6685782ab295f5033f5668bdfe1fad5b	dqft3311	0	I purchased two pairs of these jeans just to t...
21465	fe5786ae802f2b3e6537addec8a9c586	dqft3311	1	The swim skirt does exactly what I wanted it t...
22703	e82b806a6ef47f90a2ac7a68415d53c7	dqft3311	3	Bought these for the granddaughter. They are s...
22792	571b86d307f94e9e8d7919b551c6bb52	dqft3311	0	I dont like the texture of these pants. the fi...
25368	ed10255d35c36629911e40f5111c2b40	dqft3311	2	This is really nice luggage, planing to take o...
28177	2779bf6d8384905f6b7989f2ca929edd	dqft3311	1	Im a curvy women. It fit in all the right plac...
30772	625a19390f3d7d322d82d092e0e86978	dqft3311	4	I bought these same swim trunks in two colors ...
33402	a28cd5756bc08b2d9be5de8554aecd9f	dqft3311	3	This top cling in the unlikeliest of places lo...
33509	5c917c9049c4411af6026d957c173f72	dqft3311	2	Compared to most we shopped, this comforter is...
34077	0ef145f5ba1a3f4f7e3df731ddeaec32	dqft3311	4	Good quality jeans, a little pricey if not on ...
34407	c92e087518e7b42a577676e101c9267f	dqft3311	5	When we have our next snow I will be sporting ...
36136	aee5a83ed09df2b0625a48f507f0d631	dqft3311	0	I purchased this refrigerator about a month ag...

(c). The size of the unique id column

Each row of the unique_id feature in the jcpenney_products.json dataset contains 32 characters which amounts to 255, 425 million characters in the entire feature as a result of which it takes a lot of storage space in the entire database. In addition, it appears as foreign keys in reviews.csv and products.csv datasets. Within the jcpenney_products.json dataset, it is also used as the input elements of the lists in the row content of the 'Bought With' feature to represent association with a primary transaction. This create a multiplier effect for the requirement of larger storage as the use of the unique id continues to scale which can lead to poor database performance, storage inefficiency and difficulty in human usability heuristics. This development suggest that if each length of the unique id is reduced by 50%, the organisation can save 50% on storage cost improved database efficiency and human readability.

(d). Complexities in data format used for the storage

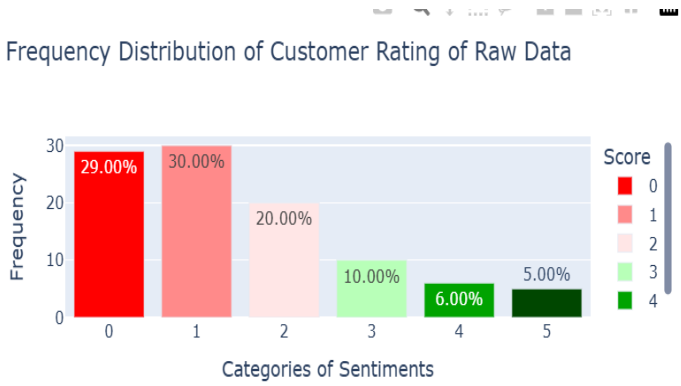
The Review feature of the jcpenney_reviewers.json dataset may not be improperly formatted but the nature of its architectural design presents a complex analytical problem to the entire data frame. This is because as column, it first presents as a class of Pandas Series, with an object data type and then each row of the column contain a data structure of class list some of which are multi-dimensional. This combination of attributes makes it difficult for a novice data scientist to process thereby requiring a highly skilled analytical skill to process and extract business insight at a higher cost to the organisation.

```
Reviewed column:
Type :<class 'pandas.core.series.Series'>
Data Type: object
Row content:<class 'list'>
```

Analysis and misclassification of customers sentiments

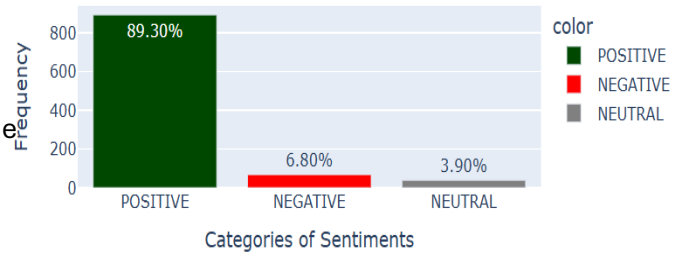
Percentage classification of raw customer rating

Percentage classification of raw customer rating as entered in the data base before analysis reveals that only 5% is absolutely positive while 29% negative. However, the class intervals indicate that up to 11% are more likely to be positive while up to 59% are more likely to be negative and about 30% neutral



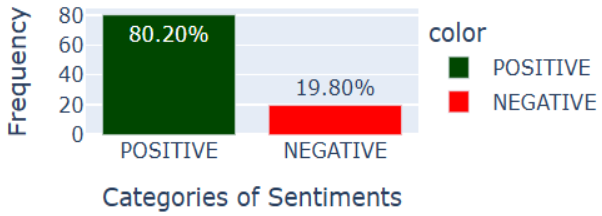
Percentage of classification of customer sentiments Vader rating

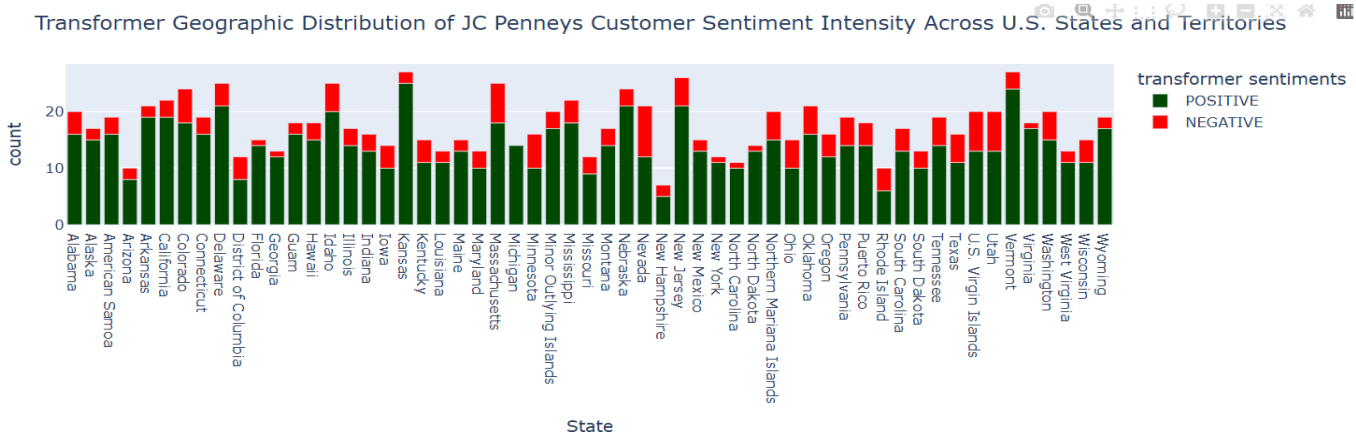
Vader rating showing that percentage of positive Sentiments is 89.30% while negative sentiments is 6.80%. This is a sharp contrast from the values of the raw data customer score above



Percentage of classification of customer sentiments by hugging face transformer

Similarly, classification made by hugging face transformer reveals that the positive customer sentiment is 80.20% while the negative sentiment is 19.80%. The below chart is showing the state wise distribution of the frequency of the positive and negative customer sentiments also confirming that there are more positive sentiments than negative

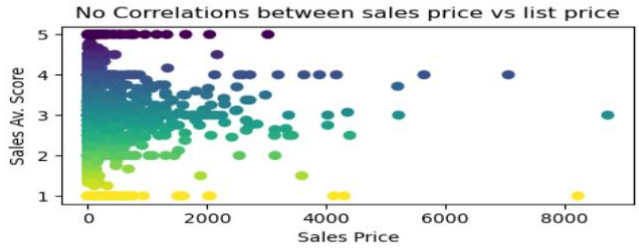




Recommendations

The variation of the classification of positive and negative on page four reveals the extent of misclassification of customer sentiments in the dataset. As it is evident in the analysis of customer birth dates, where there is a decline in the numbers of active loyal customer, JC Penney has been experiencing a customer churn which may be because of failure to address genuine customer concern due to misclassification of customer sentiments at the point of data entry.

In addition, the lack of correlation between the average score of customers review and the sales price is an indication of lack of conversion of customers sentiments into actionable business values.



As a result of the outcome of this analytic process, the following recommendations are made to the JC Penney's management to help boost their business performance and to ensure effective CRM :

1. Birth Dates:

The management should capitalize on the birth date data to drive proactive marketing campaign and automate personalized engagement a few days before their birthday with a view to offer exclusive individualized discounts, product recommendation, social engagements (online or in person) with groups of customers that celebrates their birthday the same month. This will help to build emotional connection and hence ensure customer loyalty and increasing returning clientele

2. Data entry policy:

The organisation should establish rules, procedures and standards to govern how data is collected, managed and maintained to mitigate data corruption. For instance, an integer data should not be entered into a feature designed for text data and vice versa, features designed for unique values should disallow duplicated data at the next attempt to reinput the same. Empty data structures like strings, arrays etc should be configured contain at least one element otherwise should be unacceptable by the system. The length and the size of data to be held by each column should be defined and maintained. In addition, the use of programmatic approach can help to maintain this policies in the database.

3. Normalization of database:

The database should be normalized to the third normal form (3NF) to eliminate transitive dependencies as observed in the datasets provided especially with the unique ID features. The normalization will also help to achieve consistent data entry, updating and deletion as the database scales over time

4. Automated score reclassification at the point of entry:

This implies the use of technology such as hugging face transformer to recalculate the scores provided by customers to synchronize with the tokens of the sentiments of their feedback to be stored in an alternate feature within the data table. By so doing the management can keep abreast of the score entered and the actual category of their sentiments to mitigate customer churn and promote efficient fraternity and proactive CRM practice

5. Lenth of unique id feature:

As a primary key, the unique id is central to the performance of the database engine. The size of current unique id if reduced by 50% size will reduce the increasing cost of storage as the database scales with time. JC Penney should adopt the Universally Unique Lexicographically Sortable Identifier (ULID) Code which is a 26-character Unique id standard or the Universal Product Code for North America which is a 13-character code.

6. The use skilled data scientist:

Finally, to take advantage of the current word of mouth digital landscape, the organisation should engage the use skilled data scientist to extract valuable insights from customers sentiments with the view to extract actionable business intelligence to impact on profitability and strategies.

Appendix

1. Business Understanding

JCPenney is a North American department store that was founded in the year 1902 at Kemmerer Wyoming by James Cash Penney. It's headquaters is at Plano in Texas where it is registered as J.C. Penney Corporation, Inc. to trade in goods and services. In goods, JCPenney is a merchant of clothing, foot wears for men, children, women (including plus size), home products such as beddings, bath, toiletories, kitchen wares and windows accessories. Other accessories in the collection of their products include handbags, jewelries and beuaty products. In addition, they are retailers of branded product such as Nike, Levi, Stanford men's tailored clothing etc. They offer services that include styling salon, optical centers, custom decorating etc. To reach its numerous customer base, the organisation has approximately 647 physical stores outlets in America and Puetorico and maintains an appreciable online presence using its e-commerce platform (jcpenney.com, not currently available in the UK) through which it is able to market goods and services. On the average, it is reported that it reached an of approximate of 26 million online view and a staggering 65 million per month during peak holiday season. Owing to this huge online presence and face to face physical contact with customers and clienteles especially during shopping and provision of services such as salon styling, the organisation is therefore required to maintain a business success strategy that entails customer satisfaction through excellent services, feedback with a view to mitigate intense negative feedback and churn risk. In addition, to maintain a healthy business to cutomer interaction, it is important that the database system void of data corruption to ensure that accurate customer data are used for personalised customer to business interaction, reliable, secure and accurate decision making. Sequel to the above business understanding, the **objective** of this research is defined as follows:

1. To identify the spread of the intensity of customer sentiment across United State and Territory
2. To identify weeknesses in the collection and storage of data with a view to mititgate weaknessess in database performace due to defective data entry

Data to Use:

1. all columns of reviews.csv data
2. Price and Av_Score columns from products.csv data
3. all columns from jcpenney_reviewers.json
4. all columns of jcpenney_reviewers.json
5. users.csv excluded due to similarities with jcpenney_reviewers.json

2. Importing libraries and loading data

[1]:
```
##### import libraries for numerical calculations, data and datetime processing
import numpy as np
import pandas as pd
import datetime as dt

# imported for visualization
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import folium

#from folium.plugins import FastMarkerCluster
from folium.plugins import MarkerCluster
from folium.plugins import HeatMap
import branca.element as be
import branca.colormap as cm

# for processing text data, build model that generates vader score for intensity of customer ratings
import re
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer # vader score processing
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from tqdm.notebook import tqdm

# to build huggy face ai model and make comparison with other librariew
from transformers import pipeline

import warnings
warnings.filterwarnings("ignore") # to eliminate the warning pages from the final output
```

[2]:
```
#### Load Dataframe
csv_reviews_data = pd.read_csv('data/reviews.csv')
csv_products_data = pd.read_csv('data/products.csv')
csv_user_data = pd.read_csv('data/users.csv')

# The 'lines=True' parameter is used here for json's file to mitigate ValueError: Trailing data
json_reviewers_data = pd.read_json('data/jcpenney_reviewers.json', lines = True)
json_products_data = pd.read_json('data/jcpenney_products.json', lines=True)
```

[3]:
```
states_lat_long_df=pd.read_csv('data/states_data.csv')#Added: Latitude and Longitude data from another source
state_code_df = pd.read_csv('data/state_code.csv') # data obtained from online
```

3. Data Understanding and Preparation

- data type of jcpenney_products.json list_price numeric columns is set to object which makes it unsuitale for numeric calculation. presence of empty string in the same column
- the jcpenney_products.json sale_price column which is expected to be a numeric data contains object datatype and its not properly formatted as a -result of which the current state of the column is not fit for purpose
- From the products.csv the description column contains 543 null values while the price column contains 2166 null values which made them unsuitable for calculation.
- The Reviewed column of jcpenney_reviewers.json dataset is improperly formatted as a result of which the datatype of the column is an object, each row contain a list or a multi-dimensional list. 971 rows of the column contains an empty list which disguises them as not null values but does not provide any useful contribution to the dataframe
- duplicated user name: dqft3311 jcpenney_reviewers.json. In reviews.csv' where it is a foreign key, it generated 17 rows of data which is not suitable for analysis
- The size of the unique id column in jcpenney_products.json. It is 32 character long and it's used multiple time as an element in the Bought With column
- additional data preparation are carried out under the feature engineering section below

(i). Handling unclean data

(a). null and empty strings

[4]:
```
print(f'Data type of list price column: {json_products_data['list_price'].dtypes}')# expected data type = float
```

```
Data type of list price column: object
```

[5]:
```
# To reveal empty strings in the rows of list_price column
try:
    json_products_data['list_price'].astype(float)
except ValueError as e:
    print(e)
```

```
could not convert string to float: ''
```

[6]:
```
json_products_data['sale_price'].dtypes
```

[6]:
```
dtype('O')
```

[7]:
```
# to reveal unclean data format in the rows of the sale_price column
try:
    json_products_data['sale_price'].astype(float)
except ValueError as e:
    print(e)
```

```
could not convert string to float: '12.07-60.42'
```

[8]:
```
def count_empty_string(data):
    """
    Helper function to count numbers of rows that contain empty strings in a column
    """
    count = 0
    for i in data:
        if i == '':
            count += 1
    return count

print(count_empty_string(json_products_data['list_price']))
```

```
2166
```

[9]:
```
# Focus is on the price column because SKU and Description are not relevant to this research
csv_products_data.isna().sum()
```

[9]:
```
Uniq_id          0
SKU             67
Name             0
Description    543
Price         2166
Av_Score         0
dtype: int64
```

Observation: numbers of empty string in json_products_data['list_price'] = 2166 also numbers of null values in csv_products_data = 2166 (equal amount).

[10]:
```
# They also have equal numbers of rows.
csv_products_data.shape
```

[10]:
```
(7982, 6)
```

[11]:
```
json_products_data.shape
```

[11]:
```
(7982, 15)
```

[12]:
```
# To ascertain if the index number where json_products_data contains empty data is the same as where csv_products_data contains null data
# Returns True if all of the 2166 rows has matching index numbers
np.unique(json_products_data[json_products_data['list_price']=='']['list_price'].index == csv_products_data[csv_products_data['Price'].isna()].index)
```

[12]:
```
array([ True])
```

[13]:
```
def handle_unclean(data_1, data_2):
    """To replace null values and empty strings with the mean value of each column. It first converts the datatypes of the columns into an object
    datatypes to ensure that they are suitable for a string method operations. The mean of each column is obtained by dividing the sum of each list by
    its. The program then sets the new value of the current location where the data is an empty string or a null value to mean of the column
    data_1, data_2 = column of a pandas dataframe
    """
    list_data_1 = []
    list_data_2 = []
    if data_1.dtypes != data_1.astype('O'):
        data_1 = data_1.astype(str)      # converts the values of the row into a string so as to use string method e.g i.isnumeric() below
    if data_2.dtypes != data_2.astype('O'):
        data_2 = data_2.astype(str)
    for i,j in zip(data_1,data_2):
        if i.isnumeric():
            list_data_1.append(float(i))
        if j.isnumeric():
            list_data_2.append(float(j))
    for i in range(len(data_1)):
        if (data_1[i]=='') or (data_2[i]==''):
            data_1[i]=str(np.sum(pd.to_numeric(list_data_1, errors='coerce'))/len(data_1))# assign mean value to every row that contain empty string
            data_2[i]=str(np.sum(pd.to_numeric(list_data_2, errors='coerce'))/len(data_2))
        elif(data_1[i]==np.nan) or (data_2[i]==np.nan):
            data_1[i]=str(np.sum(pd.to_numeric(list_data_1, errors='coerce'))/len(data_1))# assign mean value to every row that contain null value
            data_2[i]=str(np.sum(pd.to_numeric(list_data_2, errors='coerce'))/len(data_2))
    return [data_1, data_2]
json_products_data['list_price'],csv_products_data['Price'] = handle_unclean(json_products_data['list_price'], csv_products_data['Price'])
```

[14]:
```
print(f'Numbers of empty strings: {json_products_data[json_products_data['list_price']=='']['list_price'].shape[0]}')#to ascertain if the empty
                                                                                                                    #strings has been removed
```

```
Numbers of empty strings: 0
```

[15]:
```
print(f'Numbers of null value: {csv_products_data['Price'].isna().sum()}') # to ascertain if the null values has been removed
#csv_products_data['Price'].astype(float)
```

```
Numbers of null value: 0
```

[16]:
```
json_products_data['list_price']=json_products_data['list_price'].astype(float)# convert column to float to be suitable for numeric calculation
```

[17]:
```
json_products_data['sale_price']=json_products_data['sale_price'].str.extract(r'(.{4})', expand=False).astype(float)#to purify unclean data to a format
                                                                                                                    #that is suitable for conversion to float datatype
```

(b). empty lists in column rows

[18]:
```
json_reviewers_data.isna().sum() # ascertain of there is null value in dataframe
```

[18]:
```
Username    0
DOB         0
State       0
Reviewed    0
```

[19]:
```
json_reviewers_data['Reviewed'][0:4] # to have visual appraisal of first four rows of the Reviewed column
```

[19]:
```
0                  [cea76118f6a9110a893de2b7654319c0]
1                  [fa04fe6c0dd5189f54fe600838da43d3]
2                                                  []
3    [f129b1803f447c2b1ce43508fb822810, 3b0c9bc0be6...
Name: Reviewed, dtype: object
```

[20]:
```
json_reviewers_data['Reviewed']
```

[20]:
```
0                     [cea76118f6a9110a893de2b7654319c0]
1                     [fa04fe6c0dd5189f54fe600838da43d3]
2                                                     []
3       [f129b1803f447c2b1ce43508fb822810, 3b0c9bc0be6...
4                                                     []
                              ...
4995                  [d6cd506246bd17afa611b6a06236713c]
4996                  [97de1506cd0bcbe50f2797cd0588eb81]
4997    [799d62906019d910fa744987da184ae7, b8f5deb7b02...
4998                  [6250b1d691cd3842f05b87736f2fadbf]
4999                                                  []
Name: Reviewed, Length: 5000, dtype: object
```

[21]:
```
# To show the complex nature of the combination of datatypes in the Reviewed column
print(f'Reviewed column:\nType :{type(json_reviewers_data['Reviewed'])}\nData Type: {json_reviewers_data['Reviewed'].dtypes}\nRow content:{type(json_revi
```

```
Reviewed column:
Type :<class 'pandas.core.series.Series'>
Data Type: object
Row content:<class 'list'>
```

[22]:
```
rows_with_empty_list=json_reviewers_data[json_reviewers_data['Reviewed'].apply(len) == 0] # we target rows that has a len of 0
print(f'Numbers of rows with empty list: {rows_with_empty_list.shape[0]}')
```

```
Numbers of rows with empty list: 971
```

[23]:
```
print(f'Alternative code to find numbers of rows of with empty list: {json_reviewers_data['Reviewed'].duplicated(keep=False).sum()}')# this is because th
                                                                                                                    # empty lists are duplicated rows in the column
```

```
Alternative code to find numbers of rows of with empty list: 971
```

[24]:
```
percentage_rows_with_empty_list=(rows_with_empty_list.shape[0]/json_reviewers_data.shape[0])*100
print(f'Percentage of rows with empty list: {percentage_rows_with_empty_list}%')
```

```
Percentage of rows with empty list: 19.42%
```

(c). duplicated data:

[25]:
```
print(f' Numbers of Username duplicated rows: {json_reviewers_data.Username.duplicated(keep=False).sum()}')
```

```
 Numbers of Username duplicated rows: 2
```

[26]:
```
duplicated_username = json_reviewers_data[json_reviewers_data['Username'].duplicated(keep=False)] # To have a visual appraisal of the duplicated username
duplicated_username
```

[26]:

	Username	DOB	State	Reviewed
731	dqft3311	28.07.1995	Tennessee	[5f280fb338485cfc30678998a42f0a55]
2619	dqft3311	03.08.1969	New Mexico	[571b86d307f94e9e8d7919b551c6bb52]

[27]:
```
#passing the duplicated username from json_reviewers_data to csv_reviews_data to extract username generated by the username
csv_duplicated_in_username = csv_reviews_data[csv_reviews_data['Username']=='dqft3311']
print(f'Volume of data generated by duplicated username in reviews.csv data:\n{csv_duplicated_in_username.shape[0]} Rows\n{csv_duplicated_in_username[:1]
```

```
Volume of data generated by duplicated username in reviews.csv data:
17 Rows
                             Uniq_id  Username  Score  \
4331  e5bdf53f2374569526c9f4d55afdd88e  dqft3311      0

                                          Review
4331  We wanted something to warm us from below not ...
```

[28]:
```
csv_duplicated_in_username.shape[0]
```

[28]:
```
17
```

[29]:
```
# considering that the duplicated  username has a negative multiplier effect, we drop it from both the json_reviewers_data and csv_reviews_data
```

(d). size of unique_id column

[30]:
```
print(f'Numbers of characters per row of unique id: {len(csv_reviews_data['Uniq_id'][0])}') # to show the number of characters in each row of Uniq_id
```

```
Numbers of characters per row of unique id: 32
```

[31]:
```
print(f'Total numbers of characters in Unique_id Columns: {json_products_data['uniq_id'].shape[0] * len(json_products_data['uniq_id'][0])}')
```

```
Total numbers of characters in Unique_id Columns: 255424
```

[32]:
```
# visual appraisal of the size of column and the multiplier effect on other dependent columns in the storage
json_products_data[['uniq_id','Bought With']][:3]
```

[32]:

	uniq_id	Bought With
0	b6c0b6bea69c722939585baeac73c13d	[898e42fe937a33e8ce5e900ca7a4d924, 8c02c262567...
1	93e5272c51d8cce02597e3ce67b7ad0a	[bc9ab3406dcaa84a123b9da862e6367d, 18eb69e8fc2...
2	013e320f2f2ec0cf5b3ff5418d688528	[3ce70f519a9cfdd85cdbdecd358e5347, b0295c96d2b...

[33]:
```
json_products_data['Bought With'].dtypes
```

[33]:
```
dtype('O')
```

[34]:
```
#visual appraisal of the size of column and the multiplier effect on other column on storage
json_products_data['Bought With'][0]
```

[34]:
```
['898e42fe937a33e8ce5e900ca7a4d924',
 '8c02c262567a2267cd207e35637feb1c',
 'b62dd54545cdc1a05d8aaa2d25aed996',
 '0da4c2dcc8cfa0e71200883b00d22b30',
 '90c46b841e2eeece992c57071387899c']
```

[35]:
```
#visual appraisal of the size of column and the multiplier effect on other column and storage
json_reviewers_data['Reviewed'][7]
```

[35]:
```
['76242d553d8bd2b004eab36da5853016',
 '918493871af32c4ab1ca6aaa530501df',
 'f991a21d9058a37c30567a880dfd3f7e']
```

4. Engineering Features ¶

(a). Create products_df, reviewers_df and reviews_df

[36]:
```
# Creating new dataframe from result obtained from handling unclean data
products_df=pd.merge(csv_products_data[['Uniq_id','Av_Score']].reset_index(),
         json_products_data[['list_price','sale_price','average_product_rating','total_number_reviews']].reset_index()).drop(columns='index')
products_df.head(2)
```

[36]:

	Uniq_id	Av_Score	list_price	sale_price	average_product_rating	total_number_reviews
0	b6c0b6bea69c722939585baeac73c13d	2.625	41.09	24.1	2.625	8
1	93e5272c51d8cce02597e3ce67b7ad0a	3.000	41.09	24.1	3.000	8

[37]:
```
# Delete username == dqft3311 from jcpenneys_reviewers.json data and save data to create reviewers dataframe
reviewers_df = json_reviewers_data.drop(json_reviewers_data[json_reviewers_data['Username'] == 'dqft3311'].index)
# delete Reviewed column from dataframe
reviewers_df.drop(columns=['Reviewed'], inplace=True)

# Delete username == dqft3311 from reviews.csv data and save data to create reviews dataframe
reviews_df=csv_reviews_data.drop(csv_reviews_data[csv_reviews_data['Username'] == 'dqft3311'].index)
```

- Detailed research into the dataset indicates that the reveiwers_df is similar to csv_user_data. A quick example is shown by the following two line of codes. As a result, we exclude the csv_use_data to avoid duplication of data.

[38]:
```
reviewers_df.head(2)
```

[38]:

	Username	DOB	State
0	bkpn1412	31.07.1983	Oregon
1	gqjs4414	27.07.1998	Massachusetts

[39]:
```
csv_user_data.iloc[0:2]
```

[39]:

	Username	DOB	State
0	bkpn1412	31.07.1983	Oregon
1	gqjs4414	27.07.1998	Massachusetts

(b). Create a 'State' feature in the review_df

[40]:
```
# To create a 'State' feature in the review_df by matching the usernames in the reviewers_df column using for loop
# Using the for loop is a slow method. The map() function provides a faster approach but the following lines of code is used to
# demonstrate the use of the zip() function. The map() function is used in creating a birthday feature below to show an alternative method to achieve
# the same result
state_review_list = []
for u in reviews_df['Username']:
    for x,y in zip(reviewers_df['Username'],reviewers_df['State']):
        if u == x:      #if the usernames in both coloumns are the same ..
            state_review_list.append(y)
reviews_df['State']=state_review_list
reviews_df.head(3)
```

[40]:

	Uniq_id	Username	Score	Review	State
0	b6c0b6bea69c722939585baeac73c13d	fsdv4141	2	You never have to worry about the fit...Alfred...	American Samoa
1	b6c0b6bea69c722939585baeac73c13d	krpz1113	1	Good quality fabric. Perfect fit. Washed very ...	Virginia
2	b6c0b6bea69c722939585baeac73c13d	mbmg3241	2	I do not normally wear pants or capris that ha...	Northern Mariana Islands

(c). Create DOB feature on reviews_df by map() from reviewers_df

[41]:
```
# Create a mapping series by using the Username column because it is common to both dataframes
dob_lookup =reviewers_df.set_index('Username')['DOB']
# create the date of birth column on the reviews dataframe
reviews_df['DOB']=reviews_df['Username'].map(dob_lookup)
```

[42]:
```
reviews_df.isna().sum()
```

[42]:
```
Uniq_id     0
Username    0
Score       0
Review      0
State       0
DOB         0
dtype: int64
```

(d). Observation:

According to the construct of json_reviewers_data, each Username is allowed to make multiple reviews hence, the Reviewed column is designed to contain a list of as many unique_ids of items that are reviewed by the user. However, because each Username represent a distinct living individual, it becomes questionable when duplicated Username exists in a database. Although from our findings above, only one Username(dqft3311) is found to be duplicated with two entries at index 731 and 2619 which show different DOBs originating from Tenessee and New Mexico. The multiplier effect of this is that when the Username is passed on to csv_reviews_data, it revealed 17 different entries by the same Username but at this time it is not possible to tell if the entry was made by the user from Tennessee or New Mexico. Such duplication has the tendency to lead to flawed Analytics, defective data integrity, operational inefficiency, faulty decision-making and a precusor to cyber vulnerablity. The length of the Uniq_id column and the repeated mode of storage is a source of conern.Findings from the precedding programs revealed that each row of the uniq_id contain 32 character string, as the number of rows equals to 7982 the total numbers of string characters contained is equal to twenty five million and five thousand and four hundred and twenty four (255424) characters. This suggests that if each row is reduced by 50%, the organisation can save 50% on storage cost, and improve human readability, speed of processing and mitigate the disadvantage of the multiplier effect when the volume of the data increases. 32 characters of Uniq_id that serves the purpose of only identifying a unique product appears to be too long and has proven to require significantly more storage space in the primary table and foreign table. While long unique IDs, such as Universally

Unique Identifier(UUIDs), offer advantages in distributed systems for global uniqueness without centralized coordination, jcpenney is currently recognised as an american store hence the use of such long Unique_id come at a cost of local database performance, storage efficiency, increased overhead, more difficult technical requirements to debug and added complexities of human usability heurisritics.

(e).Data to delete:

(i). *The Reviewed column of the jcpenney_reviewers.json data.*

Justification to delete:

- a. It contains 971 rows of empty list which is a corrupted data
- b. Deleting the column instead of the rows with empty list will preserve all entries in Username column which is most relevant to this project
- c. The information contained in the Reviewed column is still available in the reviews.csv in a different format.

(ii). *Contents of the Username dqft3311*

Justification to delete:

- The username is duplicated. Deleting it will ensure data integrity, improve performance of the model, enhance storage capacity and help to mitigate threat of cyber security in the database as a whole

(iii). *Unique Id*

Justification: The length of the column is predicted to slow down data process during nltk data processing and values of the rows has no direct bearing to customer sentiments

(iii). *SKU*

Justification: Is predicted to have similar underlying pattern with unique id and not relevant to customers sentiments

(iii). *name_title*

Justification: Not relevant to model

Others include product_url', 'product_image_urls', 'brand','Bought With' - are not relevant to customers sentiments

features to use:

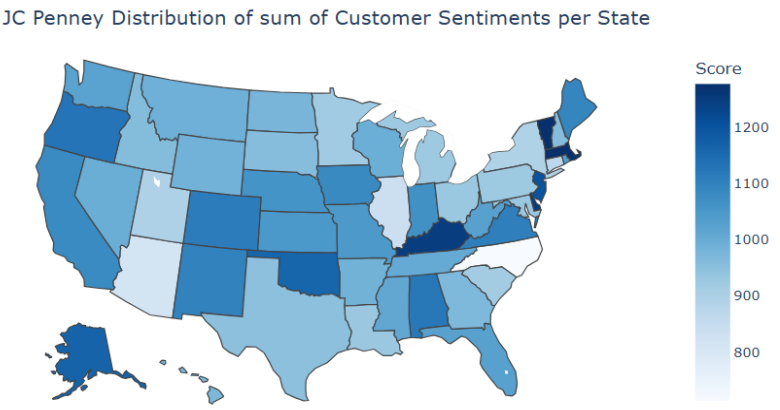
- reviews_df
- reviewers_df
- products_df

3. Visualization of raw data

(a). Total sum of customer sentiments per state

43]:
```
score_per_state_df=reviews_df.groupby(['State'], as_index=False)['Score'].sum()
score_per_state_df=pd.concat([score_per_state_df,state_code_df['Code']],axis=1)
```

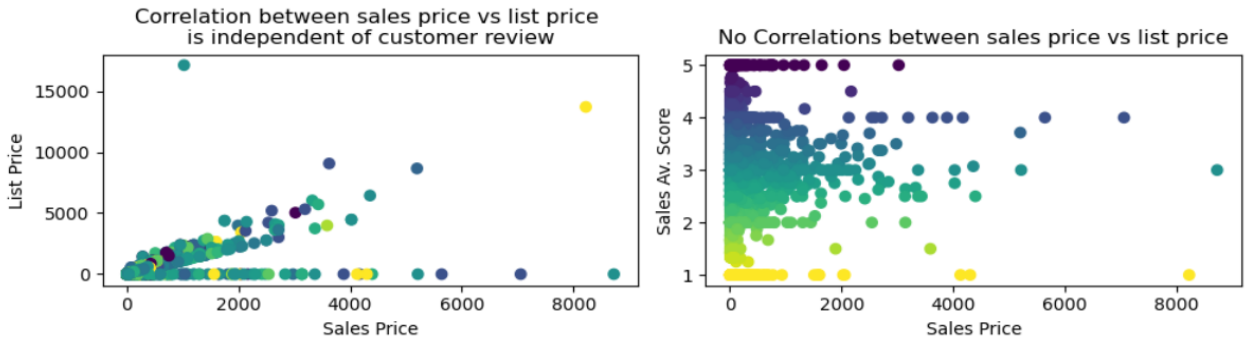
[44]:
```
fig = px.choropleth(
    score_per_state_df,
    locations='Code',
    color='Score',
    locationmode="USA-states",
    scope="usa",
    color_continuous_scale='Blues',
    hover_name='Score',
    title='JC Penney Distribution of sum of Customer Sentiments per State'
)

fig.update_layout(
    margin={"r":0.2,"t":40,"l":0.5,"b":0},
    coloraxis_showscale=True
)
fig.show()
```



(b). Correlation analysis of numeric values

To ascertain if a relationship exists between the average customer rating score and the list and sale price

[45]:
```
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(10, 3))
ax1.scatter(products_df['sale_price'], products_df['list_price'], c=products_df['Av_Score'], cmap='viridis_r')
ax1.set_xlabel('Sales Price')
ax1.set_ylabel('List Price')
ax1.set_title('Correlation between sales price vs list price \nis independent of customer review')
ax2.scatter(products_df['sale_price'], products_df['Av_Score'], c=products_df['Av_Score'], cmap='viridis_r')
ax2.set_xlabel('Sales Price')
ax2.set_ylabel('Sales Av. Score')
ax2.set_title('No Correlations between sales price vs list price')
plt.tight_layout()
plt.show()
```



(c). Analysis of customers' birthday

[46]:
```
print(f'Head count of users birthday: {reviewers_df.shape[0]}')
print(f'Total number of users with share birthday with other users: {reviewers_df['DOB'].astype(str).duplicated(keep=False).sum()}')
print(f'Total number of birthday celebrations in a year: {len(reviewers_df['DOB'].unique())}')
```

```
Head count of users birthday: 4998
Total number of users with share birthday with other users: 4998
Total number of birthday celebrations in a year: 52
```

[47]:
```
reviewers_df['DOB'].dtypes
```

[47]:
```
dtype('O')
```

[48]:
```
# Converts date to datetime to enable visualisation of birthday over a period of time.
reviewers_df['DOB'] = pd.to_datetime(reviewers_df['DOB'])
```

[49]:
```
def extract_date_data(data, d):
    '''
    Helper function to extract feeder data for create dataframes customers birthday
    data = dataframe
    d = target column e.g date column
    '''
    unique_date_list = []
    unique_date_dict = {}
    count = 0
    for i in data[d].unique():
        unique_date_list.append(data[data[d] == i]) # extract dataframe and add to the last index of the list
        unique_date_dict[i] = len(unique_date_list[count])
        count += 1
    return[unique_date_list, unique_date_dict]
```

[50]:
```
date_data = extract_date_data(reviewers_df, 'DOB') # reviewers dataframe and the data of birth column to the helper function
date_data[1].values()
```

[50]:
```
dict_values([99, 111, 99, 98, 80, 95, 106, 101, 91, 97, 104, 99, 90, 99, 93, 94, 96, 94, 100, 83, 103, 106, 97, 103, 107, 79, 107, 98, 112, 106, 101, 102, 90, 106, 81, 100, 81, 85, 90, 79, 80, 91, 87, 93, 94, 106, 96, 86, 102, 104, 101, 96])
```

[51]:
```
# create customers birthday dataframe
birthdays_df = pd.DataFrame(date_data[1].values(),date_data[1].keys()).reset_index().rename(columns={'index':'Customers DOB', 0:'Numbers Celebrating'})
birthdays_df.head(1)
```

[51]:

	Customers DOB	Numbers Celebrating
0	1983-07-31	99

[52]:
```
birthdays_df.min() # show the first customer's birthday on record
```

[52]:
```
Customers DOB          1950-08-08 00:00:00
Numbers Celebrating                    79
dtype: object
```

[52]:
```
birthdays_df.min() # show the first customer's birthday on record
```

[52]:
```
Customers DOB          1950-08-08 00:00:00
Numbers Celebrating                     79
dtype: object
```

[53]:
```
birthdays_df[birthdays_df['Customers DOB']=='1953-08-07']
```

[53]:

	Customers DOB	Numbers Celebrating
28	1953-08-07	112

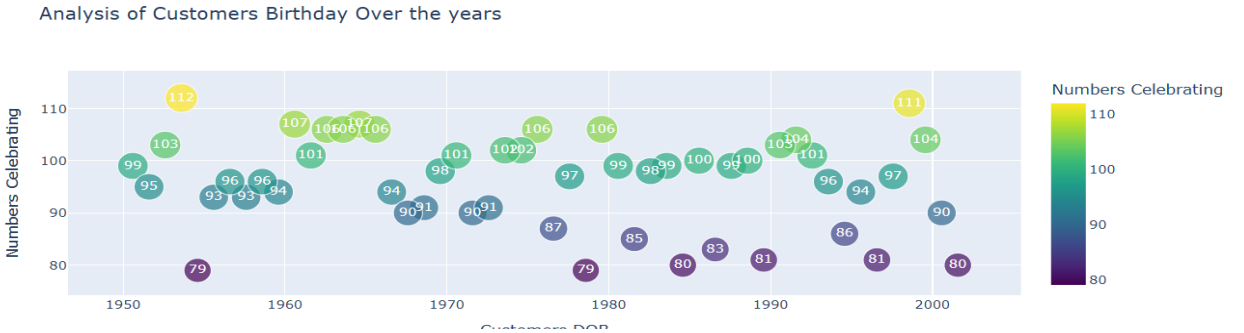
[54]:
```
birthdays_df['Customers DOB'].max() # the last customers birthday on recored
```

[54]:
```
Timestamp('2001-07-26 00:00:00')
```

[55]:
```
birthdays_df.dtypes
```

[55]:
```
Customers DOB          datetime64[ns]
Numbers Celebrating             int64
dtype: object
```

[56]:
```
fig= px.scatter(birthdays_df, x='Customers DOB', y='Numbers Celebrating', color='Numbers Celebrating',
                size='Numbers Celebrating',text='Numbers Celebrating', title='Analysis of Customers Birthday Over the years',color_continuous_scale=px.cc
fig.update_traces(textfont_color='#FFF')
fig.update_layout(
    width=1100, # width in pixels
    height=400 # height in pixels

)
fig.show()
```



(d). Analysis of frequency of customer sentiments

[57]:
```
frequency_sentiments = pd.DataFrame(reviews_df['Score'].value_counts()).reset_index()
frequency_sentiments.sort_values(by='Score', inplace=True)# To ensure that the bar charts appears in the order of the values of the 'Score' column
```

[58]:
```
#frequency_sentiments
```

[59]:
```
frequency_sentiments['Score']=frequency_sentiments['Score'].astype(str) # ensure that each Score is classified as a category
frequency_sentiments.dtypes
```

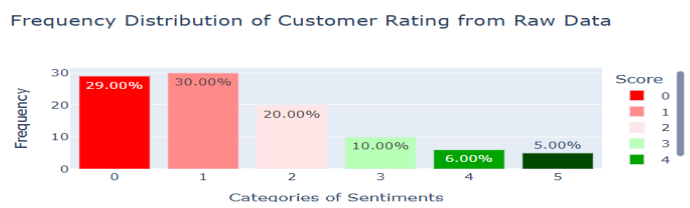
[59]:
```
Score     object
count      int64
dtype: object
```

[60]:
```
frequency_sentiments_list =[]
for d in frequency_sentiments['count']:
    frequency_sentiments_list.append((round(d/sum(frequency_sentiments['count']),2))*100)

frequency_sentiments['count(%)']=frequency_sentiments_list
frequency_sentiments.head(1)
```

[60]:

	Score	count	count(%)
1	0	11260	29.0

[61]:
```
colors_list = ['#FF0000','#FF8A8A','#FFE6E6','#B8FFB8','#00A300','#004700']
fig = px.bar(frequency_sentiments, y='count(%)', x='Score', #text_auto='.4s',
             text=[f"{i:.2f}%" for i in frequency_sentiments['count(%)']],
             color='Score',color_discrete_sequence=colors_list,
             title="Frequency Distribution of Customer Rating from Raw Data")
fig.update_layout(
    xaxis_title="Categories of Sentiments",
    yaxis_title="Frequency",
    width=600, # width in pixels
    height=300 # height in pixels
)
fig.show()
```



4. Data modeling

(i). Linear Regression model to show relationship between scores and price

The heatmap below shows that there is no correlation as a result of which the linear regression model is considered unsuitable for classification

[62]:
```
plt.figure(figsize=(8, 2))
sns.heatmap(products_df[['total_number_reviews','list_price','sale_price','Av_Score']].corr(), annot=True, linewidths=.5,cmap="YlGnBu", linecolor='black'
plt.title('No Correlation')
plt.show()
```

(ii). VADER(Valence Aware Dictionary and sEntiment Reasoner)

The customer comments in the reviews_df['Review'] is used to provide data to build this model.

[63]:
```
vader_sent_analyzer = SentimentIntensityAnalyzer() # initialize vader sentiments analyser for use to determine the intensity of the
                                                   # negativity, neutrality and positivity of customer sentiments on the Review column
vader_sent_analyzer # to ascertain that vader is initialize in the progra
```

[63]:
```
<nltk.sentiment.vader.SentimentIntensityAnalyzer at 0x15d52d53b60>
```

[64]:
```
porter = PorterStemmer() # To reduces words to their base or root form by removing prefixes and suffixes
lemma = WordNetLemmatizer()

def processing_data(data):
    """
    This function will perform the following functions 1. str.lower() 2. remove all special characters 3. word tokenize, 4. stemmer, 5. lemmatize to
    provide a clean text for the model
    """
    holding_list = []
    for i in data:
        lower_word_tokenized = str(word_tokenize(i.lower()))
        remove_special_char = re.sub(r'[^\w\s]',"",lower_word_tokenized)
        if not remove_special_char in stopwords.words('english'): # remove stop words
            stem_data = (porter.stem(remove_special_char)) #apply stemer to data
            holding_list.append(lemma.lemmatize(stem_data)) # append lematizer to holding list
    return holding_list
```

[65]:
```
# Use the processing_data() function above to clean the Review column and create a new column of clean text
reviews_df['clean text'] = processing_data(reviews_df['Review'])
reviews_df['clean text'].dtypes
```

[65]:
```
dtype('O')
```

[66]:
```
# Vader helper function
def get_polarity_scores(data, column):
    """
    run sentiment analyzer on all the rows text of the clean text data and return a pandas dataframe. The output is a dataframe
    that contains the polority scores of the Cleaned Review column.
    """
    data_structure = []
    for row in data[column]:
        data_structure.append(vader_sent_analyzer.polarity_scores(row))
    return pd.DataFrame(data_structure) # to return a new dataframe that contain the polarity scores
```

[67]:
```
# run vader sentiment analyser to generate polarity scores to show the intensity
# of customers sentiments
polarity_scores_df = get_polarity_scores(reviews_df,'clean text') # to save the output dataframe from the function in a variable
polarity_scores_df.head(2)
```

[67]:

	neg	neu	pos	compound
0	0.000	0.494	0.506	0.9546
1	0.131	0.264	0.606	0.8408

[68]:
```
# Merge polarity_scores_df with reviews_df by creating a dummy index column for use as a common column to both dataframes
score_output_df = pd.merge(polarity_scores_df.reset_index(),reviews_df.reset_index(), on='index')[['Username','DOB','State','neg','neu','pos','compound',
score_output_df.head(2)
# an alternative line of code to use to merge the dataframes is pd.concat([reviews_df,polarity_scores_df],axis=1) however this is proned to generating
# Null values in its output thereby may require additional lines of codes which is not desirable
```

[68]:

	Username	DOB	State	neg	neu	pos	compound	Score	clean text
0	fsdv4141	31.07.1980	American Samoa	0.000	0.494	0.506	0.9546	2	you never have to worry about the fit alfred ...
1	krpz1113	30.07.1987	Virginia	0.131	0.264	0.606	0.8408	1	good quality fabric perfect fit washed very ...

[69]:
```
def vader_sentiment_helper(data):
    vader_list = []
    for val in data:
        if val >= 0.05:
            vader_list.append('POSITIVE')
        elif -0.05 < val < 0.05:
            vader_list.append('NEUTRAL')
        elif val <= -0.05:
            vader_list.append('NEGATIVE')
    return vader_list
score_output_df['vader_sentiments'] = vader_sentiment_helper(score_output_df['compound'])
score_output_df.head(1)
```

[69]:

	Username	DOB	State	neg	neu	pos	compound	Score	clean text	vader_sentiments
0	fsdv4141	31.07.1980	American Samoa	0.0	0.494	0.506	0.9546	2	you never have to worry about the fit alfred ...	POSITIVE

[70]:
```
score_output_df[score_output_df['Score']==0].head(2)
```

[70]:

	Username	DOB	State	neg	neu	pos	compound	Score	clean text	vader_sentiments
3	zeqg1222	28.07.1994	Connecticut	0.0	0.553	0.447	0.9292	0	i love these capris they fit true to size and...	POSITIVE
10	ixlo1324	01.08.1976	Guam	0.0	0.733	0.267	0.9492	0	i do not normally wear pants or capris that ha...	POSITIVE

[71]:
```
score_output_df[score_output_df['Score']==0]['clean text'][24]# zero customer score but positve sentiment and positive vader classification
```

[71]:
```
'you never have to worry about the fit  alfred dunner clothing sizes are true to size and fits perfectly  great value for the money '
```

[72]:
```
score_output_df[score_output_df['vader_sentiments']=='NEGETIVE'].head(2)
```

[72]:

Username	DOB	State	neg	neu	pos	compound	Score	clean text	vader_sentiments

[73]:
```
score_output_df[score_output_df['vader_sentiments']=='NEGATIVE']['clean text'][38973] # Negative vader score but positive customer sentiments
```

[73]:
```
'love this vacuum  it took literally 5 minutes to assemble and once i started to use it  my carpets never looked cleaner  easy to empty  the pet hair tools work better than expected '
```

[74]:
```
score_output_df.iloc[38973]
```

[74]:
```
Username                                                pwyr3321
DOB                                                   31.07.1981
State                                  Northern Mariana Islands
neg                                                       0.111
neu                                                       0.786
pos                                                       0.103
compound                                                -0.1147
Score                                                         1
clean text          love this vacuum  it took literally 5 minutes ...
vader_sentiments                                         NEGATIVE
Name: 38973, dtype: object
```

(iii). Hugging face transformer

[75]:
```
score_output_df.head(1)
```

[75]:

	Username	DOB	State	neg	neu	pos	compound	Score	clean text	vader_sentiments
0	fsdv4141	31.07.1980	American Samoa	0.0	0.494	0.506	0.9546	2	you never have to worry about the fit alfred ...	POSITIVE

[76]:
```
hugging_face_clf = pipeline("sentiment-analysis",device=-1, max_length=512,truncation=True)
hugging_face_clf
```

```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst-2-english and revision 714eb0f (https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english).
Using a pipeline without specifying a model name and revision in production is not recommended.
Device set to use cpu
```

[76]:
```
<transformers.pipelines.text_classification.TextClassificationPipeline at 0x15d5c797380>
```

[77]:
```
# run hugging face classifier on the clean data to classify sentiments
# extract 1000 rows of data due Iterative Generation because the model runs into an iterative token by token loop which is inherently slower and make
# the task too long to complete
clf_list = []
for i in range(1000):
    clf_list.append(hugging_face_clf(score_output_df['clean text'][i])[0])
```

[78]:
```
trans_df = pd.DataFrame(clf_list).rename(columns={'score':'trans_score','label':'transformer sentiments'})
```

[79]:
```
# to add negative sign(-) to hugging face transformer rating where the sentiments is NEGATIVE for distinctions between negative and positive
# classification in the output map
conversion_list=[]
for sentiments,scores in zip(trans_df['transformer sentiments'], trans_df['trans_score']):
    if sentiments == 'NEGATIVE':
        scores= round((-1 * scores),2)
        conversion_list.append(scores)
    elif sentiments == 'POSITIVE':
        conversion_list.append(scores)
trans_df['transformer_score']=conversion_list
```

Result

[80]:
```
score_output_df['customer sentiments(%)']=round((score_output_df['Score']/6)*100,2)
score_output_df['vader sentiments(%)']=round((score_output_df['compound']*100),2)
trans_df['transformer sentiments(%)']=round((trans_df['transformer_score']*100),2)
```

[81]:
```
output_1000_rows = score_output_df[['Username','DOB','State','vader_sentiments','customer sentiments(%)','vader sentiments(%)','clean text']][:1000]
output_1000_rows.head(1)
```

[81]:

	Username	DOB	State	vader_sentiments	customer sentiments(%)	vader sentiments(%)	clean text
0	fsdv4141	31.07.1980	American Samoa	POSITIVE	33.33	95.46	you never have to worry about the fit alfred ...

[82]:
```
result = pd.concat([output_1000_rows,trans_df[['transformer sentiments','transformer sentiments(%)']]],axis=1)
result.head(1)
```

[82]:

	Username	DOB	State	vader_sentiments	customer sentiments(%)	vader sentiments(%)	clean text	transformer sentiments	transformer sentiments(%)
0	fsdv4141	31.07.1980	American Samoa	POSITIVE	33.33	95.46	you never have to worry about the fit alfred ...	POSITIVE	99.99

5. Evaluation and deployment

- How do the insights you obtained help the company, and how can should they be adopted in their business? If modeling techniques have been adopted, are their use scientifically sound and how should they be mantained?

[83]:
```
vader_counts=pd.DataFrame(result.groupby('vader_sentiments', as_index=False)['vader_sentiments'].value_counts())
# alternatively, the dataframe.reset_index() method can be used to move the index as a column and replace the as_index=False parameter
vader_counts.sort_values(by='count', ascending=False, inplace=True)
# the sorted values can be saved in a varible as an alternative to the inplace=True parameter as shown below
vader_counts.head(1)
```

[83]:

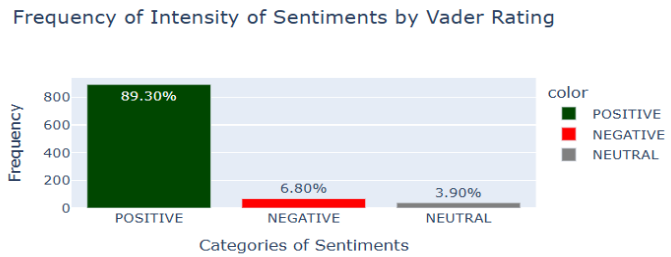
	vader_sentiments	count
2	POSITIVE	893

[84]:
```
vader_percent_list = []
for d in vader_counts['count']:
    vader_percent_list.append((d/sum(vader_counts['count']))*100)
vader_counts['count(%)']=vader_percent_list
vader_counts.head(1)
```

[84]:

	vader_sentiments	count	count(%)
2	POSITIVE	893	89.3

[85]:
```
colors_list = ['#004700','#FF0000','grey']
fig = px.bar(x=vader_counts['vader_sentiments'], y=vader_counts['count'], color=vader_counts['vader_sentiments'], color_discrete_sequence=colors_list,
             text=[f"{i:.2f}%" for i in vader_counts['count(%)']],
             title='Frequency of Intensity of Sentiments by Vader Rating')
fig.update_layout(xaxis_title="Categories of Sentiments",yaxis_title="Frequency",width=600, height=300)
fig.show()
```



[86]:
```
transformer_counts = pd.DataFrame(result.groupby('transformer sentiments')['transformer sentiments'].value_counts().reset_index())
transformer_counts=transformer_counts.sort_values(by='count', ascending=False)# sorted values saved in a variable
transformer_counts.head(1)
```

[86]:

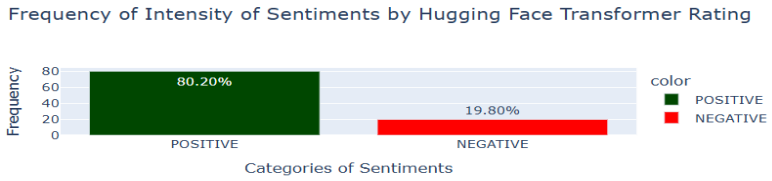
	transformer sentiments	count
1	POSITIVE	802

[87]:
```
transformer_percent_list = []
for d in transformer_counts['count']:
    transformer_percent_list.append((d/sum(transformer_counts['count']))*100)
transformer_counts['count(%)']=transformer_percent_list
transformer_counts.head(1)
```

[87]:

	transformer sentiments	count	count(%)
1	POSITIVE	802	80.2

[88]:
```
colors_list = ['#004700','#FF0000']
fig = px.bar(x=transformer_counts['transformer sentiments'], y=transformer_counts['count(%)'], color=transformer_counts['transformer sentiments'],
             color_discrete_sequence=colors_list,
             text=[f"{i:.2f}%" for i in transformer_counts['count(%)']],
             title='Frequency of Intensity of Sentiments by Hugging Face Transformer Rating')
fig.update_layout(xaxis_title="Categories of Sentiments",yaxis_title="Frequency",width=700, height=250)
fig.show()
```



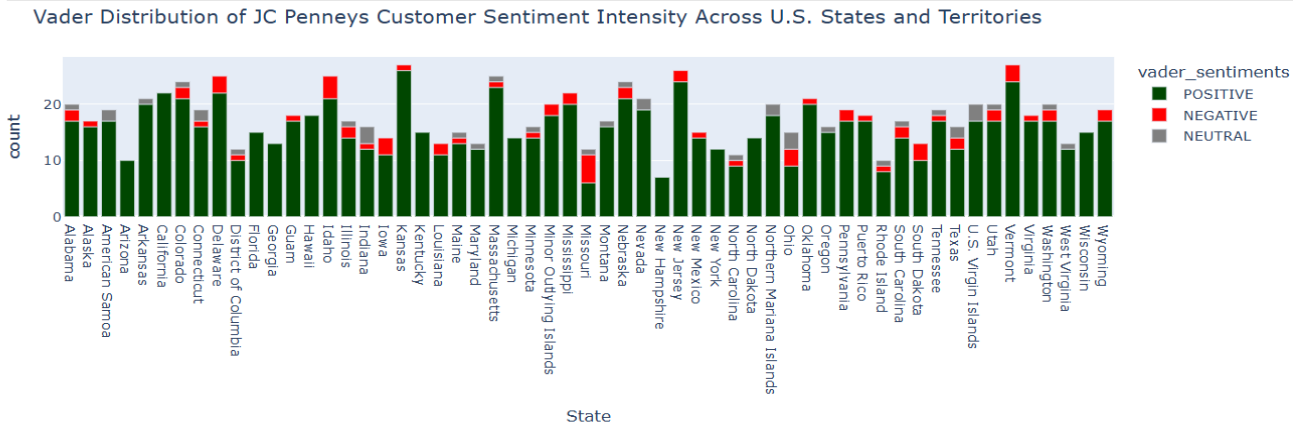
[89]:
```
result.head(1)
```

[89]:

	Username	DOB	State	vader_sentiments	customer sentiments(%)	vader sentiments(%)	clean text	transformer sentiments	transformer sentiments(%)
0	fsdv4141	31.07.1980	American Samoa	POSITIVE	33.33	95.46	you never have to worry about the fit alfred ...	POSITIVE	99.99

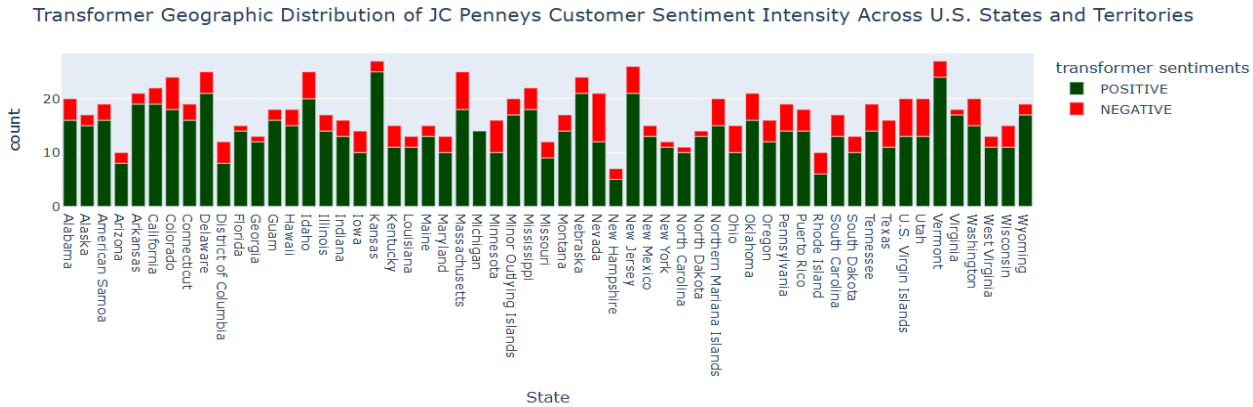
[90]:
```
vader_state_counts=pd.DataFrame(result.groupby('State', as_index=False)['vader_sentiments'].value_counts())
```

[91]:
```
colors_list = ['#004700','#FF0000','grey']
fig = px.bar(vader_state_counts, x="State", y="count", color='vader_sentiments',color_discrete_sequence=colors_list)
fig.update_layout(
    width=1110, # Set chart width in pixels
    height=400,  # Set chart height in pixels
    title='Vader Distribution of JC Penneys Customer Sentiment Intensity Across U.S. States and Territories'
)
fig.show()
```



2]:
```
transformer_state_counts=pd.DataFrame(result.groupby('State', as_index=False)['transformer sentiments'].value_counts())
```

3]:
```
fig = px.bar(transformer_state_counts, x="State", y="count", color='transformer sentiments',color_discrete_sequence=colors_list)
fig.update_layout(
    width=1110, # Set chart width in pixels
    height=400,  # Set chart height in pixels
    title='Transformer Geographic Distribution of JC Penneys Customer Sentiment Intensity Across U.S. States and Territories'
)
fig.show()
```



(5b).Folium Map

[94]:
```
states_lat_long_df.head(1)
```

[94]:

	code	latitude	longitude	state
0	AK	63.588753	-154.493062	Alaska

[95]:
```
# Create a dictionary lookup values for the columns in the dataframe
code_lookup = pd.Series(states_lat_long_df['code'].values, index=states_lat_long_df['state']).to_dict()
state_lookup = pd.Series(states_lat_long_df['state'].values, index=states_lat_long_df['state']).to_dict()
latitude_lookup = pd.Series(states_lat_long_df['latitude'].values, index=states_lat_long_df['state']).to_dict()
longitude_lookup = pd.Series(states_lat_long_df['longitude'].values, index=states_lat_long_df['state']).to_dict()
```

[96]:
```
import itertools
dict(itertools.islice(code_lookup.items(),3))
```

[96]:
```
{'Alaska': 'AK', 'Alabama': 'AL', 'Arkansas': 'AR'}
```

[97]:
```
# Apply the values to the result dataframe to create corresponding columns of geographical data for visualization
result['code'] = result['State'].map(code_lookup)
result['latitude'] = result['State'].map(latitude_lookup)
result['longitude'] = result['State'].map(longitude_lookup)
```

[98]:
```
#result.isna().sum()
```

[99]:
```
# List out the states that has not been mapped to latitude and longitude
result.loc[result.isna().any(axis=1)]['State'].unique()
# an alternative method to achieve the above is shown below by finding the difference in the list of the state columns from both dataframe
```

[99]:
```
array(['American Samoa', 'Northern Mariana Islands',
       'Minor Outlying Islands', 'Guam', 'U.S. Virgin Islands'],
      dtype=object)
```

[100]:
```
# Difference in list
list(set(result['State'].unique()) - set(states_lat_long_df['state'].unique()))
```

[100]:
```
['Northern Mariana Islands',
 'U.S. Virgin Islands',
 'Guam',
 'American Samoa',
 'Minor Outlying Islands']
```

[101]:
```
# Source additional Latitude and longitude data for the concerned states
lat_long_dict = {'code':['MO','MP','GU','VI','AS'],
                 'state': ['Minor Outlying Islands','Northern Mariana Islands','Guam','U.S. Virgin Islands','American Samoa'],
                 'latitude':[28.2075,15.190000,13.444304,18.036736,-14.275632],
                 'longitude':[-177.35,145.750000,144.793732,-64.801704,-170.702042]
}
added_lat_long_df=pd.DataFrame(lat_long_dict)
```

[102]:
```
# add the new data to the dataframe of latitude and longitude data
latitude_longitude_data = pd.concat([added_lat_long_df,states_lat_long_df],axis=0, ignore_index=True)
latitude_longitude_data.isna().sum()
```

[102]:
```
code         0
state        0
latitude     0
longitude    0
dtype: int64
```

[103]:
```
#latitude_longitude_data
```

[104]:
```
# Map the new data to the result_df uisng the map() method
code_lookup = pd.Series(latitude_longitude_data['code'].values, index=latitude_longitude_data['state']).to_dict()
state_lookup = pd.Series(latitude_longitude_data['state'].values, index=latitude_longitude_data['state']).to_dict()
latitude_lookup = pd.Series(latitude_longitude_data['latitude'].values, index=latitude_longitude_data['state']).to_dict()
longitude_lookup = pd.Series(latitude_longitude_data['longitude'].values, index=latitude_longitude_data['state']).to_dict()

# Apply the values to the result dataframe to create corresponding columns of geographical data for visualization
result['code'] = result['State'].map(code_lookup)
result['latitude'] = result['State'].map(latitude_lookup)
result['longitude'] = result['State'].map(longitude_lookup)
```

[105]:
```
# save result data as csv to currrent directory
result.to_csv('result_df.csv')
```

dataframe for heatmap

[106]:
```
vader_state_counts.head(2)
```

[106]:

	State	vader_sentiments	count
0	Alabama	POSITIVE	17
1	Alabama	NEGATIVE	2

[107]:
```
# Map the latitude and longitude new data to the result_df uisng the map() method
latitude_lookup = pd.Series(latitude_longitude_data['latitude'].values, index=latitude_longitude_data['state']).to_dict()
longitude_lookup = pd.Series(latitude_longitude_data['longitude'].values, index=latitude_longitude_data['state']).to_dict()

# Apply the values to the result dataframe to create corresponding columns of geographical data for visualization

transformer_state_counts['latitude'] = transformer_state_counts['State'].map(latitude_lookup)
transformer_state_counts['longitude'] = transformer_state_counts['State'].map(longitude_lookup)
```

[108]:
```
transformer_state_counts.head(1)
```

[108]:

	State	transformer sentiments	count	latitude	longitude
0	Alabama	POSITIVE	16	32.318231	-86.902298

[109]:
```
#transformer heatmap data

negative_df=transformer_state_counts[transformer_state_counts['transformer sentiments']=='NEGATIVE'].reset_index().drop(columns='index')
negative_df.rename(columns={'transformer sentiments':'negative_clf','count':'neg_count'}, inplace=True)
```

[110]:
```
positive_df=transformer_state_counts[transformer_state_counts['transformer sentiments']=='POSITIVE'].reset_index().drop(columns='index')
positive_df.rename(columns={'transformer sentiments':'positive_clf','count':'pos_count'}, inplace=True)
```

[116]:
```
heatmap_df = pd.concat([positive_df,negative_df[['negative_clf','neg_count']]], axis=1)
heatmap_df.isna().sum()
```

[116]:
```
State          0
positive_clf   0
pos_count      0
latitude       0
longitude      0
negative_clf   1
neg_count      1
dtype: int64
```

[117]:
```
# assign missing values to wyoming at index 56
heatmap_df.loc[56,'negative_clf']='NEGATIVE'
heatmap_df.loc[56,'neg_count']=2.0
```

[118]:
```
def color_marker(sentiments):
    """Help function to assign color to marker
    """
    if sentiments == 'NEGATIVE':
        return 'red'
    elif sentiments == 'POSITIVE':
        return 'green'
```

[114]:
```
# Convert DOB datatype to object for better rendering on the map
result['DOB']=result['DOB'].astype(str)
```

[115]:
```
# Initialize the map and set centroid of map start location to south dakota because of the spread of the data
customer_sentiment_map = folium.Map(location=[39.011902,-98.484246],tiles='CartoDB Voyager',zoom_start=5)#CartoDB Positron

# Add title
title = '''
    <h3 align="center" style="font-size:20px; color:black;">
    <b>Geographic Distribution of JC Penny Customer Sentiment Intensity AcrossU.S. States and Territories
    </b></h3>
'''
customer_sentiment_map.get_root().html.add_child(folium.Element(title))

# Legend
sentiment_color = cm.LinearColormap(["red","green"], vmin=0, vmax=100) #
sentiment_color.caption = 'Sentiment Classification' # Add a title to legend
sentiment_color.tick_labels = [0, 50, 100]
sentiment_color.add_to(customer_sentiment_map)

#initialize cluster object to locate each sentiments for each state
marker_cluster = MarkerCluster(name='Sentiments per State').add_to(customer_sentiment_map)
# Add the HeatMap layer for negative sentitments
heat_data_2 = heatmap_df[['latitude','longitude','neg_count']].values.tolist()
neg=HeatMap(
    name='Negative',
    data=heat_data_2,
    radius=15,
    blur=15,
    min_opacity=0.9,
    gradient={0.9:'red'}
    ).add_to(customer_sentiment_map)
```

```
# Add the HeatMap layer for positive sentiments
heat_data = heatmap_df[['latitude','longitude','pos_count']].values.tolist()
pos=HeatMap(
    name='Positive',
    data=heat_data,
    radius=15,
    blur=15,
    min_opacity=0.9,
    gradient={0.9:'green'}
    ).add_to(customer_sentiment_map)

# Use for loop to add to add markers with popups and colors based on the value in each of the rows in the columns provided
for i,r in result.iterrows():
    # Define the 5 pieces of information to display in the popup using HTML
    html_popup = f"""
    <h4>User Name: {r['Username']}</h4>
    <ul>
    <li><b>DOB:</b> {r['DOB']}</li>
    <li><b>Sentiments:</b> {r['transformer sentiments']}</li>
    <li><b>State:</b> {r['State']}</li>
    <li><b>Customer Rating:</b> {float(r['customer sentiments(%)']):.1f}%</li>
    <li><b>Vader Rating:</b> {float(r['vader sentiments(%)']):.1f}%</li>
    <li><b>Trans. Rating:</b> {float(r['transformer sentiments(%)']):.1f}%</li>
    </ul>
    """

    #IFrame to help render popoup
    iframe = be.IFrame(html=html_popup, width=240, height=180)
    popup = folium.Popup(iframe, max_width=2750)
    # Set marker color to negative or positive based on value in ['transformer sentiments'] column
    color = color_marker(r['transformer sentiments'])
    # Create the Marker object to render numbers of sentiments in each state
    marker_clusters=folium.Marker(
        location=[r['latitude'], r['longitude']],
        popup=popup,
        tooltip='Click for more',
        icon=folium.Icon(color=color, icon='info-sign', prefix='glyphicon') #Using a glyphicon
    ).add_to(marker_cluster)
layer_control=folium.LayerControl(collapsed=False, position='bottomright',show=True).add_to(customer_sentiment_map)
# save map in html format
customer_sentiment_map.save('jc_penny_sentiments_analysis.html')
customer_sentiment_map
```