000_data_inspection

June 23, 2020

1 Imports

```
In [1]: import gzip
        import os
        import pandas as pd
        import numpy as np
```

For visuals:

```
In [2]: import matplotlib.pyplot as plt
        %matplotlib inline
        import warnings
        warnings.filterwarnings('ignore')

In [3]: #%pip install python-decouple

In [4]: from decouple import config

In [5]: API_USERNAME = config('USER')

In [6]: API_KEY = config('PLOTLY_API_KEY')

In [7]: import chart_studio

In [8]: chart_studio.tools.set_credentials_file(username=API_USERNAME, api_key=API_KEY)

In [9]: import chart_studio.plotly as py
        import plotly.offline
        from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

In [11]: import cufflinks as cf
         cf.go_offline()
         # Configure cufflings
         cf.set_config_file(offline=False, world_readable=True, theme='pearl')
```

1.1 Loading the Data

The dataset chosen for this project belongs to the Amazon product Data (McAuley, 2014). The complete dataset contains product reviews and metadata from Amazon, including 142.8 million reviews ranging between May 1996 and July 2014. Relatively, smaller datasets are available for class projects. The chosen dataset concerns book reviews and is one such smaller dataset. The way that smaller datasets were generated relies on extracting the 5-core, such that each of the remaining users and items have k reviews each. In graph theory, 5-core refers to a graph, where every subgraph has a vertex of degree3 at most k: that is, some vertex in the subgraph is connected with at least k other vertices. A initial description of the dataset on book reviews appears below.

The below functions are provided directly from the Amazon Review Data link by the author and it is used to load the 5-cores book reviews as a `panda dataframe`.

```
In [12]: def parse(path):
             g = gzip.open(path, 'rb')
             for l in g:
                 yield eval(l)

In [13]: def getDF(path):
             i = 0
             df = {}
             for d in parse(path):
                 df[i] = d
                 i += 1
             return pd.DataFrame.from_dict(df, orient='index')

In [14]: df = getDF('../data/raw/reviews_Books_5.json.gz')
```

I used the below snippet to monitor the memory requirements for the loading.

```
In [15]: # %pip install memory_profiler
         # %load_ext memory_profiler
         # %memit
```

Below you can see the fields loaded and a count of the values per field;

```
In [16]: df.count()

Out[16]: reviewerID        8898041
         asin              8898041
         reviewerName      8872495
         helpful           8898041
         reviewText        8898041
         overall           8898041
         summary           8898041
         unixReviewTime    8898041
         reviewTime        8898041
         dtype: int64
```

A sample of the overal data appears next:

```
In [17]: df[0:10]
```

Out[17]:

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime
0	A10000012B7CGYKOMPQ4L	000100039X	Adam	[0, 0]	Spiritually and mentally inspiring! A book tha...	5.0	Wonderful!	1355616000
1	A2S166WSCFIFP5	000100039X	adead_poet@hotmail.com "adead_poet@hotmail.com"	[0, 2]	This is one my must have books. It is a master...	5.0	close to god	1071100800
2	A1BM81XB4QHOA3	000100039X	Ahoro Blethends "Seriously"	[0, 0]	This book provides a reflection that you can a...	5.0	Must Read for Life Afficianados	1390003200
3	A1MOSTXNIO5MPJ	000100039X	Alan Krug	[0, 0]	I first read THE PROPHET in college back in th...	5.0	Timeless for every good and bad time in your l...	1317081600
4	A2XQ5LZHTD4AFT	000100039X	Alaturka	[7, 9]	A timeless classic. It is a very demanding an...	5.0	A Modern Rumi	1033948800
5	A3V1MKC2BVWY48	000100039X	Alex Dawson	[0, 0]	Reading this made my mind feel like a still po...	5.0	This book will bring you peace	1390780800
6	A12387207U8U24	000100039X	Alex	[0, 0]	As you read, Gibran's poetry brings spiritual ...	5.0	Graet Work	1206662400
7	A29TRDMK51GKZR	000100039X	Alpine Plume	[0, 0]	Deep, moving dramatic verses of the heart and ...	5.0	Such Beauty	1383436800
8	A3FI0744PG1WYG	000100039X	Always Reading "tkm"	[0, 0]	This is a timeless classic. Over the years I'...	5.0	The Prophet	1390953600
9	A2LBBQHYLEHM7P	000100039X	Amazon Customer "Full Frontal Nerdity"	[0, 0]	An amazing work. Realizing extensive use of Bi...	5.0		

```
9                                   A Modern Classic      1379808000

    reviewTime
0  12 16, 2012
1  12 11, 2003
2  01 18, 2014
3  09 27, 2011
4   10 7, 2002
5  01 27, 2014
6  03 28, 2008
7   11 3, 2013
8  01 29, 2014
9  09 22, 2013
```

1.2 Column Fields of Interest

In general, the loaded dataframe, include 7 fields: *`reviewerID: AString(probably a hashText) that uniquely identifies the user that submitted the review.` *asin: ASIN stands for **Amazon Standard Identification Number**. Almost every product on **Amazon** has its own **ASIN**, a unique code used to identify it. For books, the **ASIN** is the same as the book's **ISBN** number. *reviewerName: The name of the reviewer. *helpful: Amazon has implemented an interface that allows customers to vote on whether a particular review has been helpful or unhelpful. This is captured by this field, which represents a rating of the review, e.g. if[2,3] --> 2/3. *reviewText: The actual review provided by the reviewer. *overall: The product's rating attributed by the same reviewer. *summary: A summary of the review. *unixReviewTime: Time of the review (unix time). *reviewTime': Time of the review (raw).

Of these fields, for the purposes of this project we care to keep the `reviewerID`, `asin`, `reviewText`, `overall` and `helpful`. Specifically, we keep `reviewerID` only to merge it with `asin` and create unique identifier (`key`) per review, e.g.:

```
key = reviewerID:"A10000012B7CGYKOMPQ4L" + asin:"000100039X"
```

`asin` is obviously necessary to identify the distinct books in the dataset, while the rest are necessary for the analysis (`overall`, `reviewText`) and for evaluation (`helpful`) purposes.

Data Inspection The inspection begins with a distribution of the review ratings, which appears in the below figure. As it is evident, close to half of the book reviews are rated with 5 stars, a quarter of them with 4 and the remaining quarter is distributed amongst the 3, 2 and 1 ratings. In order to get a clearer idea of how books are rated in overall, rather than look at the distribution of ratings amongst all reviews, the distribution of average book ratings was also generated and appears in Figure 5. The chart was generated to show 100 bins. The highly skewed to the right distribution confirms that the majority of books have reviews between 4 and 5 stars. As such, one should expect that this will be reflected in the text that accompanies the respective reviews, which means that negative reviews will be more difficult to find.

```
In [18]: # Number of reviews:
         number_of_reviews=len(df)
         my_number_string = '{:0,.0f}'.format(number_of_reviews)
```

```
        print('Number of Reviews: ' + my_number_string + '.')
```

```
Number of Reviews: 8,898,041.
```

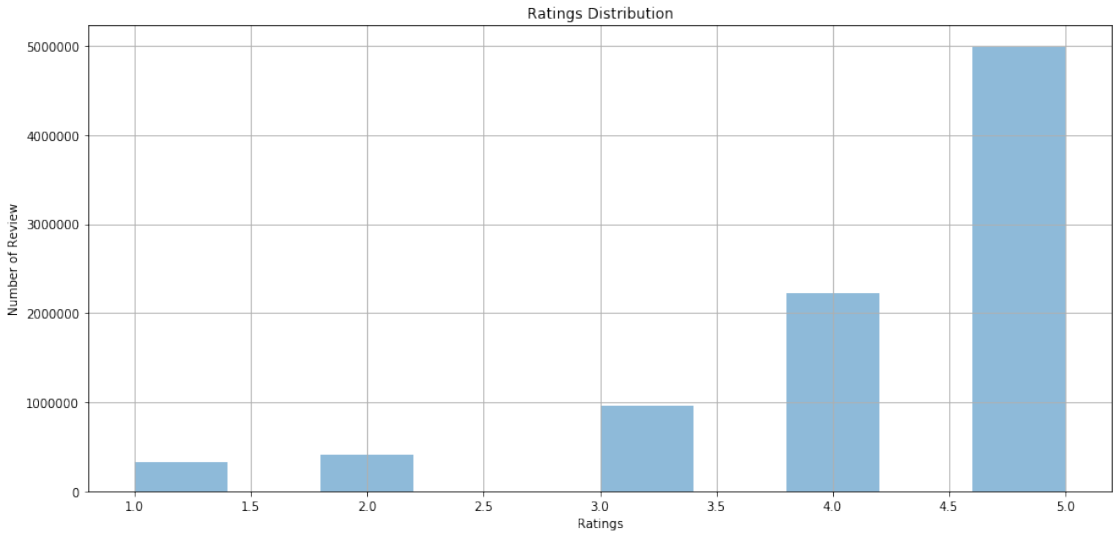
```
In [19]: # Unique number of items:
         unique_books=len(df['asin'].unique())
         my_number_string = '{:0,.0f}'.format(unique_books)

         print('Number of Books: ' + my_number_string + '.')
```

```
Number of Books: 367,982.
```

1.2.1 Distribution of ratings amongst all reviews

```
In [20]: # Distribution of Ratings (too many to plot with plotly)
         fig = df['overall'].plot.hist(alpha=0.5, title='Ratings Distribution', figsize=(15,7)
         fig.set_xlabel("Ratings")
         fig.set_ylabel("Number of Review")
```

```
Out[20]: <matplotlib.text.Text at 0x7fad65efc8d0>
```




```
In [21]: df10 = df[['overall','asin']]
```

```
In [22]: df11 = pd.DataFrame(df10.groupby(['asin'])['overall'].mean())
```

1.2.2 Distribution of Average Book Ratings

```
In [23]: len(df11)
```

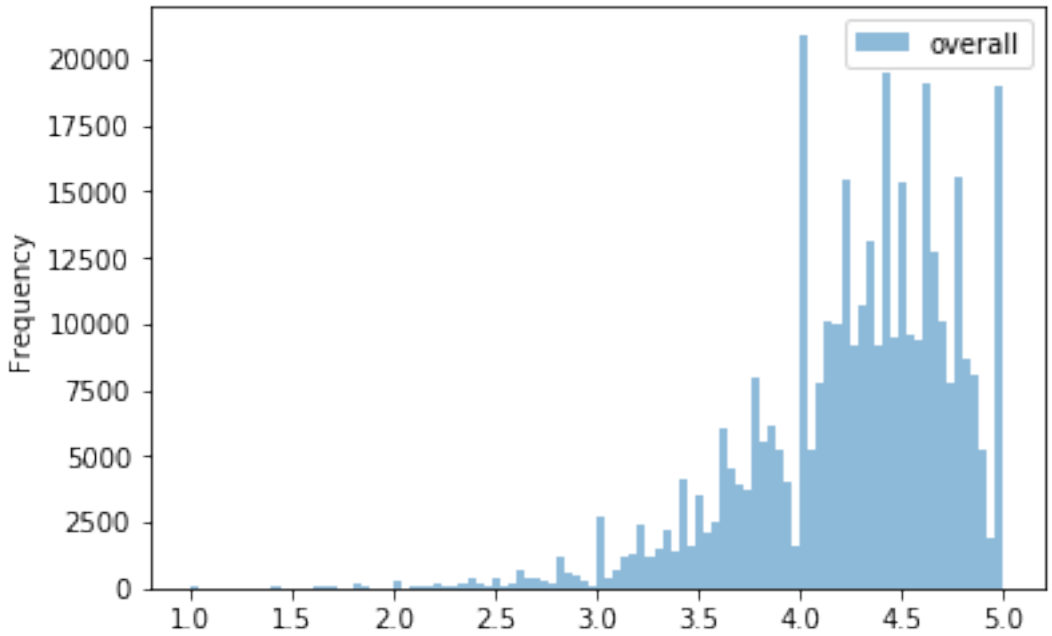
```
Out[23]: 367982
```

```
In [24]: df11 = df11.reset_index()
         df11.head()
```

```
Out[24]:          asin   overall
         0  000100039X  4.674757
         1  0001055178  3.555556
         2  0001473123  4.625000
         3  0001473727  5.000000
         4  0001473905  4.666667
```

```
In [25]: #df11['overall'].iplot(kind='histogram', bins=100, xTitle='Rating (0-5)',yTitle='Numb
         df11.plot.hist(alpha=0.5,bins=100)
```

```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x7fadf0778e80>
```



1.2.3 Books per Year

```
In [26]: df20 = df[['asin','reviewTime']]
```

```
In [27]: def get_year(reviewTime):
             day_month_year_list = reviewTime.split(',')
```

```
            if(len(day_month_year_list)==2):
                return day_month_year_list[1]
            else:
                return fillna(0)
```

```
In [28]: df20['reviewYear'] = pd.DataFrame(df20['reviewTime'].apply(lambda time: get_year(time)
```

```
In [29]: df20.head()
```

Out[29]:

	asin	reviewTime	reviewYear
0	000100039X	12 16, 2012	2012
1	000100039X	12 11, 2003	2003
2	000100039X	01 18, 2014	2014
3	000100039X	09 27, 2011	2011
4	000100039X	10 7, 2002	2002

```
In [30]: books_per_year = pd.DataFrame(df20.groupby(['reviewYear']).size())
```

```
In [31]: books_per_year.columns = ['counts']
```

```
In [32]: %jupyter notebook --NotebookApp.iopub_data_rate_limit=1.0e10
```

```
ERROR:root:Line magic function `%jupyter` not found.
```

```
In [33]: books_per_year.iplot(kind='bar', xTitle='Years', yTitle='Number of Reviews', title='Nu
```

```
In [34]: df30 = df[['asin','reviewTime', 'overall']]
```

```
In [35]: df30['reviewYear'] = pd.DataFrame(df30['reviewTime'].apply(lambda time: get_year(time)
```

```
In [36]: df30.head()
```

Out[36]:

	asin	reviewTime	overall	reviewYear
0	000100039X	12 16, 2012	5.0	2012
1	000100039X	12 11, 2003	5.0	2003
2	000100039X	01 18, 2014	5.0	2014
3	000100039X	09 27, 2011	5.0	2011
4	000100039X	10 7, 2002	5.0	2002

```
In [37]: books_per_rating_per_year = df30.groupby(['reviewYear','overall']).size().reset_index
```

```
In [38]: books_per_rating_per_year[0:10]
```

Out[38]:

	reviewYear	overall	counts
0	1996	1.0	1
1	1996	2.0	2
2	1996	3.0	1
3	1996	4.0	6

```
4       1996      5.0      15
5       1997      1.0      80
6       1997      2.0     132
7       1997      3.0     174
8       1997      4.0     466
9       1997      5.0    1189
```

```
In [39]: pivot_df = books_per_rating_per_year.pivot(index='reviewYear', columns='overall', valu

In [40]: pivot_df.iplot(kind='bar', barmode='stack', xTitle='Years', yTitle='Number of Reviews
```

1.2.4 Helpfulness

```
In [41]: df40 = df[['asin', 'helpful']]

In [42]: # Create new Column for the enumerator
         df40 = df40.assign(enum = df40['helpful'].apply(lambda enum_denom:enum_denom[0]))

In [43]: # Create new Column for the denominator
         df40 = df40.assign(denom = df40['helpful'].apply(lambda enum_denom:enum_denom[1]))

In [44]: # Filter on the denom
         df40 = df40.loc[df40['denom'] != 0]

In [45]: df40[0:15]
```

Out[45]:

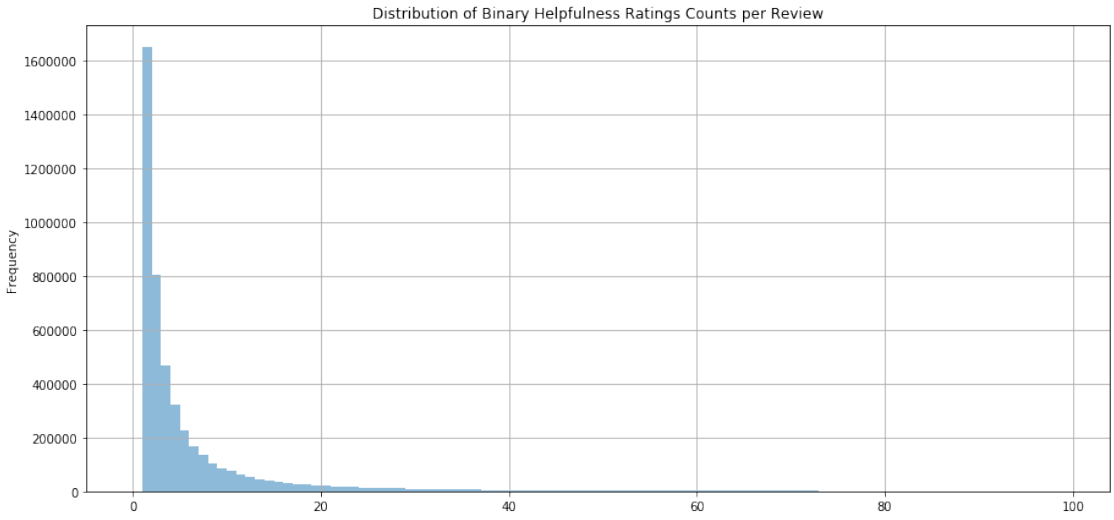
	asin	helpful	enum	denom
1	000100039X	[0, 2]	0	2
4	000100039X	[7, 9]	7	9
14	000100039X	[1, 1]	1	1
15	000100039X	[1, 1]	1	1
17	000100039X	[3, 5]	3	5
18	000100039X	[1, 1]	1	1
19	000100039X	[3, 3]	3	3
21	000100039X	[2, 3]	2	3
22	000100039X	[1, 4]	1	4
23	000100039X	[2, 9]	2	9
25	000100039X	[5, 6]	5	6
26	000100039X	[1, 2]	1	2
31	000100039X	[1, 1]	1	1
33	000100039X	[0, 2]	0	2
34	000100039X	[81, 92]	81	92

```
In [46]: len(df40)
```

Out[46]: 4756837

```
In [47]: bin_values = np.arange(start=0,stop=100,step=1)
         df40['denom'].plot.hist(alpha=0.5, bins=bin_values, figsize=(15,7), grid=True, title=
```

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x7fad931d5390>



In [48]:
```
# Focus on [10,100] range of rating per review
df40 = df40.loc[df40['denom'] > 15]
df40 = df40.loc[df40['denom'] < 100]
len(df40)
```

Out[48]: 439769

In [49]:
```
df50 = df40.assign(percentage = df40['enum']/df40['denom'])
df50['percentage'].iplot(kind='histogram', title='Distribution of Helpfulness Percenta
```

In [50]:
```
df50.head()
```

Out[50]:

	asin	helpful	enum	denom	percentage
34	000100039X	[81, 92]	81	92	0.880435
106	000100039X	[17, 20]	17	20	0.850000
121	000100039X	[0, 56]	0	56	0.000000
123	000100039X	[10, 28]	10	28	0.357143
133	000100039X	[19, 25]	19	25	0.760000

In [51]:
```
threshold = 0.7
df60 = df50.loc[df50['percentage'] > threshold]
```

In [52]:
```
len(df60)
```

Out[52]: 295941

In [53]:
```
# END OF FILE
```