How Do I Purchase the Table Computer that Provides the Greatest Value from Walmart.com?

Prashant B. Bhuyan

May 21, 2015

Problem Statement:

A person is in the market to buy a tablet. That person decides to go to Walmart.com to explore her options. She soon realizes that there are nearly a dozen top brands with seemingly endless configurations. The primary factor that would influence this person's decision to purchase a tablet at Walmart is value for her hard earned money. But how to quickly distinguish the product that offers the best value?

Solution:

To help this consumer choose the tablet that offers her the best value we can create a an interactive visualization that groups products by brand and model depicts the relationship between aggregated aggregated ratings, sale prices and number of reviews.

Products that have high ratings, low sale prices and lots of reviews can be interpreted as providing customers with a solidly great value for their money.

On the contrary, products that have low ratings, high sale prices and lots of reviews can be interpreted as providing customers with a solidly poor value for their money.

Implementation:

Luckily, Walmart Labs provides an array of free APIs (requires an api key) that we can write to in order to search products by name and retrieve product details and review statistics.

Below is a function for retrieving data from the Walmart API. We'll use this for querying the Search API for Item Ids as well as for querying the Review API for review statistics for each item.

In [202]:
```
# get data from specified api and return content (json)
# we use this function to get content from Walmart's search api and then
again from Walmart's review api.
def get_data(web_url):
    request = urllib2.Request(web_url)
    page = urllib2.urlopen(request)
    content = json.loads(page.read())
    # soup = BeautifulSoup(page.read())
    page.close()
    return content
```

We use this function above to retrieve raw data on various products that we are interested in (see snippet below). For example, if we query apple ipad we get data related on all different apple ipad models sold at Walmart such as Ipad Mini and Ipad Air.

```
    # use Walmart Search API to search for all items related to the foll
owing top selling brands (apple ipad, samsung galaxy, acer iconia, asus
tablet, toshiba tablet, hp tablet)
    # use the get_data() function from above- returns content in json fo
rmat.
    apple_ipadmini = get_data("http://api.walmartlabs.com/v1/search?apiK
ey=k33eueajm253wrp7uh75jv2m&lsPublisherId={Your%20LinkShare%20Publishe
r%20Id}&query=ipad")
    samsung_galaxy_minitab = get_data("http://api.walmartlabs.com/v1/sea
rch?apiKey=k33eueajm253wrp7uh75jv2m&lsPublisherId={Your%20LinkShare%20Pu
blisher%20Id}&query=galaxytablet")
    acer_iconia = get_data("http://api.walmartlabs.com/v1/search?apiKe
y=k33eueajm253wrp7uh75jv2m&lsPublisherId={Your%20LinkShare%20Publisher%2
0Id}&query=acericonia")
    asus_tablet = get_data("http://api.walmartlabs.com/v1/search?apiKe
y=k33eueajm253wrp7uh75jv2m&lsPublisherId={Your%20LinkShare%20Publisher%2
0Id}&query=asustablet")
    toshiba_tablet = get_data("http://api.walmartlabs.com/v1/search?apiK
ey=k33eueajm253wrp7uh75jv2m&lsPublisherId={Your%20LinkShare%20Publishe
r%20Id}&query=toshibatablet")
    hp_tablet = get_data("http://api.walmartlabs.com/v1/search?apiKey=k3
3eueajm253wrp7uh75jv2m&lsPublisherId={Your%20LinkShare%20Publisher%20I
d}&query=hptablet")
```

The Walmart search api returns product details, including item ids, in json format as below:

```
'**********Original Json Content Containing Querys, Items (Walmart Search
API)**********'


{u'a': {u'addToCartUrl': u'http://c.affil.walmart.com/t/api02?l=http%3A%2
F%2Fwww.walmart.com%2Fcatalog%2Fselect_product.gsp%3Fproduct_id%3D3309310
1%26add_to_cart%3D1%26qty%3D1%26affp1%3DQ-aX3XXuPRCvByNYrVsCtNsB9110qHqSw
NSMATqlBMI%26affilsrc%3Dapi%26veh%3Daff%26wmlspartner%3Dreadonlyapi',
        u'affiliateAddToCartUrl': u'http://linksynergy.walmart.com/fs-bi
n/click?id={Your LinkShare Publisher Id}&offerid=223073.7200&type=14&cati
d=8&subid=0&hid=7200&tmpid=1082&RD_PARM1=http%253A%252F%252Fwww.walmart.c
om%252Fcatalog%252Fselect_product.gsp%253Fproduct_id%253D33093101%2526ad
d_to_cart%253D1%2526qty%253D1%2526affp1%253DQ-aX3XXuPRCvByNYrVsCtNsB9110q
HqSwNSMATqlBMI%2526affilsrc%253Dapi',
        u'availableOnline': True,
        u'bundle': False,
        u'categoryNode': u'3944_3951_1078084',
        u'categoryPath': u'Electronics/Computers/Tablet PCs',
        u'customerRating': u'4.719',
        u'customerRatingImage': u'http://i2.walmartimages.com/i/CustRatin
g/4_7.gif',
        u'itemId': 33093101,
        u'longDescription': u'&lt;br&gt;&lt;b&gt;Apple iPad mini 16GB wit
h Wi-Fi (Space Gray or Silver):&lt;/b&gt;&lt;ul&gt;&lt;li&gt;The Apple iP
ad mini with WiFi has a 7.9-inch LED-backlit display&lt;/li&gt;&lt;li&g
t;A5 chip&lt;/li&gt;&lt;li&gt;5MP iSight camera with 1080p HD video recor
ding&lt;/li&gt;&lt;li&gt;FaceTime camera&lt;/li&gt;&lt;li&gt;The Apple iP
ad mini with WiFi features up to 10 hours of battery life *&lt;/li&gt;&l
t;li&gt;Built-in Wi-Fi (802.11a/b/g/n)&lt;/li&gt;&lt;li&gt;Over 275,000 a
pps on the App Store **&lt;/li&gt;&lt;li&gt;The Apple iPad 16GB (Space Gr
ay or Silver) runs on iOS 6 and iCloud&lt;/li&gt;&lt;li&gt;Cellular data
service on Wi-Fi + Cellular models (sold separately)&lt;/li&gt;&lt;/ul&g
t;&lt;br&gt;* Battery life varies by use and configuration. See www.appl
e.com/batteries for more information.&lt;br&gt;** App count refers to the
total number of apps worldwide.',
        u'marketplace': False,
        u'modelNumber': u'MF432LL/A',
        u'msrp': 249.0,
        u'name': u'Apple iPad mini 16GB Wi-Fi',
        u'numReviews': 1661,
        u'parentItemId': 33093101,
        u'productTrackingUrl': u'http://linksynergy.walmart.com/fs-bin/cl
ick?id={Your LinkShare Publisher Id}&offerid=223073.7200&type=14&cati
d=8&subid=0&hid=7200&tmpid=1082&RD_PARM1=http%253A%252F%252Fwww.walmart.c
```

The primary variables that we are interested in collecting are Product Name, Avg Rating, Mean Sale Price and Total Reviews for tablets grouped by brand and model. In order to accomplish this we must first grab review statistics for each item via another API called the Walmart Reviews API. But first we need to create an array of item ids. We accomplish this as follows:

```
    # parse the json content returned from the item search above based o
n 'query'.
    apple_ipadmini_query = apple_ipadmini["query"]
    samsung_galaxy_minitab_query = samsung_galaxy_minitab["query"]
    acer_iconia_query = acer_iconia["query"]
    asus_tablet_query = asus_tablet["query"]
```

```
    toshiba_tablet_query = toshiba_tablet["query"]
    hp_tablet_query = hp_tablet["query"]

    # parse the json content returned from the item search above based o
n 'items'.
    apple_ipadmini_items = apple_ipadmini["items"]
    samsung_galaxy_minitab_items = samsung_galaxy_minitab["items"]
    acer_iconia_items = acer_iconia["items"]
    asus_tablet_items = asus_tablet["items"]
    toshiba_tablet_items = toshiba_tablet["items"]
    hp_tablet_items = hp_tablet["items"]
```

```
    # zip parsed query and item values
    ipad_mini_ids_zipped = zip(apple_ipadmini_query, apple_ipadmini_item
s)
    galaxy_mini_ids_zipped = zip(samsung_galaxy_minitab_query, samsung_g
alaxy_minitab_items)
    acer_iconia_ids_zipped = zip(acer_iconia_query, acer_iconia_items)
    asus_tablet_ids_zipped = zip(asus_tablet_query, asus_tablet_items)
    toshiba_tablet_ids_zipped = zip(toshiba_tablet_query, toshiba_table
t_items)
    hp_tablet_ids_zipped = zip(hp_tablet_query, hp_tablet_items)
```

```
    # create new dicts for each product category
    # print our new dicts for each product category
    ipad_mini_ids_dict = dict(ipad_mini_ids_zipped)
    # pprint(ipad_mini_ids_dict)
    galaxy_mini_ids_dict = dict(galaxy_mini_ids_zipped)
    # pprint(galaxy_mini_ids_dict)
    acer_iconia_ids_dict = dict(acer_iconia_ids_zipped)
    # pprint(acer_iconia_ids_dict)
    asus_tablet_ids_dict = dict(asus_tablet_ids_zipped)
    # pprint(asus_tablet_ids_dict)
    toshiba_tablet_ids_dict = dict(toshiba_tablet_ids_zipped)
    # pprint(toshiba_tablet_ids_dict)
    hp_tablet_ids_dict = dict(hp_tablet_ids_zipped)
    # pprint(hp_tablet_ids_dict)

    # unpack item ids for each product category and store them into the
item_ids array defined below.
    item_ids = []

    for k,v in ipad_mini_ids_dict.items():
        item_ids.append(v['itemId'])

    for k,v in galaxy_mini_ids_dict.items():
        item_ids.append(v['itemId'])

    for k,v in acer_iconia_ids_dict.items():
        item_ids.append(v['itemId'])

    for k,v in asus_tablet_ids_dict.items():
        item_ids.append(v['itemId'])

    for k,v in toshiba_tablet_ids_dict.items():
        item_ids.append(v['itemId'])
```

```
    for k,v in hp_tablet_ids_dict.items():
        item_ids.append(v['itemId'])
```

Now we can iterate through the list of item ids that we created above and call a function to replace the 8 character substring that represents the item id for each item. We iteratively paste a new substring into the review api url for each item and then call the review api and store each item's review statistics accoringly. This is accomplished with the code below:

The function to replace item id substrings in each review api url:

```
In [203]: # this function replaces the item id string in a given url.  This is nec
          essary to iterate through each
          # of the items to collect data related to ratings, sale prices, total nu
          mber of reviews for each item.
          def replace_id_url(url,item_id):
              s = url
              new_url = s.replace(s[38:46],item_id)
              return new_url
```

The Walmart review api url:

```
    # this is the Walmart Review API
    ratings_url = "http://api.walmartlabs.com/v1/reviews/33093101?apiKe
y=k33eueajm253wrp7uh75jv2m&lsPublisherId={Your%20LinkShare%20Publisher%2
0Id}&format=json"
```

Below is the json output of parsed review content for each item id:

```
'**********Parsed Review Content (Walmart Review API)**********'


[{u'averageOverallRating': u'5.0',
  u'overallRatingRange': u'5',
  u'ratingDistributions': [{u'count': u'2', u'ratingValue': u'5'}],
  u'totalReviewCount': u'2'},
 {u'averageOverallRating': u'4.66',
  u'overallRatingRange': u'5',
  u'ratingDistributions': [{u'count': u'2', u'ratingValue': u'1'},
                           {u'count': u'3', u'ratingValue': u'3'},
                           {u'count': u'13', u'ratingValue': u'4'},
                           {u'count': u'61', u'ratingValue': u'5'}],
  u'totalReviewCount': u'79'},
 {u'averageOverallRating': u'4.72',
  u'overallRatingRange': u'5',
  u'ratingDistributions': [{u'count': u'53', u'ratingValue': u'1'},
                           {u'count': u'10', u'ratingValue': u'2'},
                           {u'count': u'25', u'ratingValue': u'3'},
                           {u'count': u'178', u'ratingValue': u'4'},
                           {u'count': u'1402', u'ratingValue': u'5'}],
  u'totalReviewCount': u'1668'},
 {u'averageOverallRating': u'4.74',
  u'overallRatingRange': u'5',
```

Next we extract the key variables that we are interested in and create a dataframe called df3 as below:

```
    # convert the key metric arrays into pandas data frames.
    product_names_df = pd.DataFrame(product_names)
    product_avg_overall_ratings_df = pd.DataFrame(product_avg_overall_ra
tings)
    product_sale_prices_df = pd.DataFrame(product_sale_prices)
    product_num_reviews_df = pd.DataFrame(product_num_reviews)
```

```
    # concat each of the key metric data frames into one data frame call
ed df3.
    df1 = pd.concat([product_names_df, product_avg_overall_ratings_df],
axis = 1)
    df2 = pd.concat([df1, product_sale_prices_df], axis = 1)
    df3 = pd.concat([df2, product_num_reviews_df], axis = 1)

    # rename columns of df3.
    df3.columns = ['ProductName', 'AvgRating', 'SalePrice', 'TotalReview
s']
```

But we still have to group this data by brand and model and aggregate key metrics to get average sale price, mean average rating, and total number of reviews for each product group. The problem is that the key variables in df3 are not typecast to the proper data type (float, float and int) for (avgrating, saleprice and totalreviews), respectively.

To properly type cast the data we have to split df3, apply the proper type cast and then re-combine the variables and create a new data frame called new_df3:

```
    # typecast columns AvgRating, SalePrice, TotalReviews as type float,
int.
    df3_avgratings_typecast = df3['AvgRating'].astype('float')
    df3_saleprice_typecast = df3['SalePrice'].astype('float')
    df3_totalreviews_typecast = df3['TotalReviews'].astype('int')

    # create a new df3 data frame- concat typecasted columns with produc
t names
    new_df = pd.concat([product_names_df, df3_avgratings_typecast], axis
= 1)
    new_df2 = pd.concat([new_df, df3_saleprice_typecast], axis = 1)
    new_df3 = pd.concat([new_df2, df3_totalreviews_typecast], axis = 1)

    # re-name columns of new_df3
    new_df3.columns = ['ProductName', 'AvgRating', 'SalePrice', 'TotalRe
views']
```

Now the hard part. We have to group items by product name and model using string matching. We have to store the grouped names in a separate array - and this is just a list of the strings that we are searching for. Each string that we search for, ipad mini, for example can return several items- a list. As such we have to give that product group a name and that name is just the string that we are matching. In this case that would be 'ipad mini'. We store those names in an array called grped_names. For the grouping of the items

by matching sub string we apply the following code:

```
    # define an array to store grouped names (Brand+Model of Tablet).
    grped_names = []

    # group items by product name and model using string matching.  Then
store the matched string
    # into the array grped_names.

    # apple ipad air products
    apple_ipadair_prods = new_df3.loc[new_df3['ProductName'].str.contain
s("Apple iPad Air")]
    grped_names.append("Apple iPad Air")

    # apple ipad mini products
    apple_ipadmini_prods = new_df Save current document (⌘S) Name'].str.contai
ns("Apple iPad mini")]
    grped_names.append("Apple iPad mini")

    # samsung galaxy tab pro products
    samsung_galaxy_tab_pro_prods = new_df3.loc[new_df3['ProductName'].st
r.contains("Samsung Galaxy Tab Pro")]
    grped_names.append("Samsung Galaxy Tab Pro")

    # samsung galaxy tab 3 products
    samsung_galaxy_tab_3_prods = new_df3.loc[new_df3['ProductName'].st
r.contains("Samsung Galaxy Tab 3")]
    grped_names.append("Samsung Galaxy Tab 3")
```

```
    # samsung galaxy tab S products
    samsung_galaxy_tab_S_prods = new_df3.loc[new_df3['ProductName'].st
r.contains("Samsung Galaxy Tab S")]
    grped_names.append("Samsung Galaxy Tab S")

    # samsung galaxy tab 4 products
    samsung_galaxy_tab_4_prods = new_df3.loc[new_df3['ProductName'].st
r.contains("Samsung Galaxy Tab 4")]
    grped_names.append("Samsung Galaxy Tab 4")

    # acer iconia products
    acer_iconia_prods = new_df3.loc[new_df3['ProductName'].str.contain
s("Acer Iconia")]
    grped_names.append("Acer Iconia")

    # acer ICONIA One products
    acer_ICONIA_One_prods = new_df3.loc[new_df3['ProductName'].str.conta
ins("Acer ICONIA ONE")]
    grped_names.append("Acer ICONIA ONE")

    # asus memo pad produts
    asus_memo_pad_prods = new_df3.loc[new_df3['ProductName'].str.contain
s("Asus MeMO Pad")]
    grped_names.append("Asus MeMO Pad")

    # asus transformer pad products
    asus_transformer_pad_prods = new_df3.loc[new_df3['ProductName'].st
r.contains("ASUS Transformer Pad")]
    grped_names.append("ASUS Transformer Pad")

    # asus t100TAB1GR products
    asus_t100tab1gr_prods = new_df3.loc[new_df3['ProductName'].str.conta
ins("Asus T100TA-B1-GR")]
    grped_names.append("Asus T100TA-B1-GR")

    # asus 8 Tablet products
    asus_8_prods = new_df3.loc[new_df3['ProductName'].str.contains("ASUS
8")]
    grped_names.append("ASUS 8")
```

```
    grped_names.append("ASUS 8")

    # asus vivotab products
    asus_vivotab_prods = new_df3.loc[new_df3['ProductName'].str.contain
s("Asus VivoTab")]
    grped_names.append("Asus VivoTab")

    # toshiba encore products
    toshiba_encore_prods = new_df3.loc[new_df3['ProductName'].str.contai
ns("Toshiba Encore")]
    grped_names.append("Toshiba Encore")

    # toshiba excite products
    toshiba_excite_prods = new_df3.loc[new_df3['ProductName'].str.contai
ns("Toshiba Excite")]
    grped_names.append("Toshiba Excite")

    # toshiba satellite products
```

```
    toshiba_satellite_prods = new_df3.loc[new_df3['ProductName'].str.con
tains("Toshiba Satellite")]
    grped_names.append("Toshiba Satellite")

    # hp slate products
    hp_slate_prods = new_df3.loc[new_df3['ProductName'].str.contains("HP
Slate")]
    grped_names.append("HP Slate")
```

To aggregate key metrics we call this function that passes a data frame object and returns an array:

```
# this function passes a dataframe and returns an array of metrics.
# below we pass in a dataframe for a particular brand and model of table
t.
def get_metrics(df):

    mean_rating = np.mean(df['AvgRating'].astype('float'))
    mean_salepx = np.mean(df['SalePrice'].astype('float'))
    totalreviews = np.sum(df['TotalReviews'].astype('int'))
    prodname = df['ProductName']

    metrics = np.array([prodname, mean_rating, mean_salepx, totalreview
s])

    return metrics
```

We then call the function above to compute aggregated key metrics for each product (brand/model) group:

```
    # get aggregated product metrics- mean averagerating, mean salepric
e, total reviews per product group

    # get aggregated product metrics for apple ipad air products
    apple_ipad_air_metrics = get_metrics(apple_ipadair_prods)
    # pprint(apple_ipad_air_metrics)

    # get aggregated product metrics for apple ipad mini products
    apple_ipadmini_metrics = get_metrics(apple_ipadmini_prods)
    # pprint(apple_ipadmini_metrics)

    # get aggregated product metrics for samsung galaxy tab pro products
    samsung_galaxytabpro_metrics = get_metrics(samsung_galaxy_tab_pro_pr
ods)
    # pprint(samsung_galaxytabpro_metrics)

    # get aggregated product metrics for samsung galaxy tab 3 products
    samsung_galaxytab3_metrics = get_metrics(samsung_galaxy_tab_3_prods)
    # pprint(samsung_galaxytab3_metrics)

    # get aggregated product metrics for samsung galaxy tab S products
    samsung_galaxytabS_metrics = get_metrics(samsung_galaxy_tab_S_prods)
    # pprint(samsung_galaxytabS_metrics)

    # get aggregated product metrics for samsung galaxy tab 4 products
    samsung_galaxytab4_metrics = get_metrics(samsung_galaxy_tab_4_prods)
    # pprint(samsung_galaxytab4_metrics)

    # get aggregated product metrics for acer iconia products
    acer_iconia_metrics = get_metrics(acer_iconia_prods)
    # pprint(acer_iconia_metrics)

    # get aggregated product metrics for acer iconia one products
    acer_iconia_one_metrics = get_metrics(acer_ICONIA_One_prods)
    # pprint(acer_iconia_one_metrics)

    # get aggregated product metrics for asus memo pad products
    asus_memo_pad_metrics = get_metrics(asus_memo_pad_prods)
    # pprint(asus_memo_pad_metrics)
```

```
    # get aggregated product metrics for asus memo pad products
    asus_memo_pad_metrics = get_metrics(asus_memo_pad_prods)
    # pprint(asus_memo_pad_metrics)

    # get aggregated product metrics for asus transformer pad products
    asus_transformer_pad_metrics = get_metrics(asus_transformer_pad_prod
s)
    # pprint(asus_transformer_pad_metrics)

    # get aggregated product metrics for asus t100 tab products
```

```
    asus_t100tab1gr_pad_metrics = get_metrics(asus_t100tab1gr_prods)
    # pprint(asus_t100tab1gr_pad_metrics)

    # get aggregated product metrics for asus 8 tab products
    asus_8_metrics = get_metrics(asus_8_prods)
    # pprint(asus_8_metrics)

    # get aggregated product metrics for asus vivotab products
    asus_vivotab_metrics = get_metrics(asus_vivotab_prods)
    # pprint(asus_vivotab_metrics)

    # get aggregated product metrics for toshiba encore products
    toshiba_encore_metrics = get_metrics(toshiba_encore_prods)
    # pprint(toshiba_encore_metrics)

    # get aggregated product metrics for toshiba satellite products
    toshiba_satellite_metrics = get_metrics(toshiba_satellite_prods)
    # pprint(toshiba_satellite_metrics)

    # get aggregated product metrics for hp slate products
    hp_slate_metrics = get_metrics(hp_slate_prods)
    # pprint(hp_slate_metrics)
```

Now we have to concat the aggregated metrics into a single data frame that we'll call tablet data, convert the grped names array to a data frame and create a new data frame called tablet data 1.

```
    # concat aggregated metrics into a single df
    tablet_data = pd.DataFrame([apple_ipad_air_metrics, apple_ipadmini_m
etrics, samsung_galaxytabpro_metrics, samsung_galaxytab3_metrics, samsun
g_galaxytabS_metrics, samsung_galaxytab4_metrics, acer_iconia_metrics, a
cer_iconia_one_metrics, asus_memo_pad_metrics, asus_transformer_pad_metr
ics, asus_t100tab1gr_pad_metrics, asus_8_metrics, asus_vivotab_metrics,
toshiba_encore_metrics, toshiba_satellite_metrics, hp_slate_metrics, al
l_winner_cortex_metrics])

    # convert grped_names array into a pandas data frame
    grped_names_df = pd.DataFrame(grped_names)

    # concat tablet_data dataframe that contains aggregated product grou
p metrics with product group names df grped_names
    tablet_data_1 = pd.concat([grped_names_df, tablet_data],axis = 1)

    # rename tablet_data_1 columns
    tablet_data_1.columns = ['Product Group Name', 'Model List', 'Mean A
vg Rating for Product Group', 'Mean Sale Price for Product Group', 'Tota
l Reviews for Product Group']
```

After renaming the columns we see we have 5 variables. The second variable is Model List which is an array of items for string match and we don't need that. As such we'll drop that column:

```
    # drop the model list column that contains an array of models for ea
ch group.
    tablet_data_1_col_dropped = tablet_data_1.drop('Model List', axis =
1)
```

Finally, we have a data frame that is ready to be loaded to html for rendering via javascript, googlevisualization api and jquery:

```
    # write teh final dataframe to csv and load to html page for visual
rendering
    tablet_data_1_col_dropped.to_csv("/Users/MicrostrRes/Desktop/table
t_data.csv", sep = ',')
```

Here is the resulting wmttablet_data.csv output. (Some small changes were made in Excel such as removing the index column- before loading to index.html via jquery).

wmttablet_data.csv

```
ProductName,MeanAvgRating,MeanSalePrice,TotalReviews
AppleiPadAir,4.83,374,81
AppleiPadmini,4.73,218.5,1717
SamsungGalaxyTabPro,2.31,379.99,142
SamsungGalaxyTab3,5,99.99,2
SamsungGalaxyTabS,4.033333333,348.6566667,252
SamsungGalaxyTab4,4.61,279.99,450
AcerIconia,0,146.52,0
AcerICONIAONE,5,104.15,1
AsusMeMOPad,0,133.97,0
ASUSTransformerPad,5,249.99,1
AsusT100,0,356.71,0
ASUS8,4.8,160.13,5
AsusVivoTab,0,199.99,0
ToshibaEncore,2,240.935,8
ToshibaExcite,0,461.18,0
ToshibaSatellite,4.225,412.685,40
HPSlate,1.375,107.425,4
```

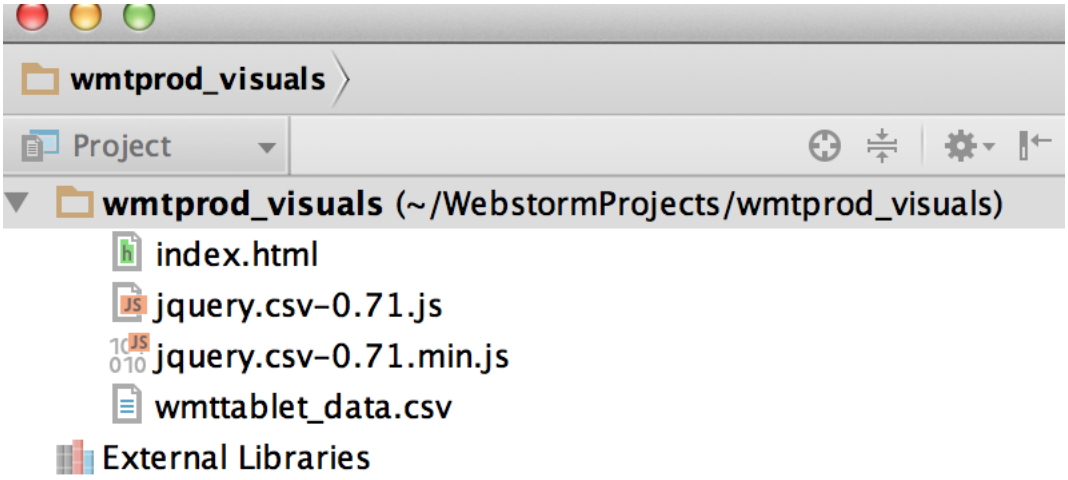
```
'**********FINAL DATA FOR RENDERING**********'
```

	Product Group Name	Mean Avg Rating for Product Group	\
0	Apple iPad Air	4.830000	
1	Apple iPad mini	4.730000	
2	Samsung Galaxy Tab Pro	4.620000	
3	Samsung Galaxy Tab 3	5.000000	
4	Samsung Galaxy Tab S	4.033333	
5	Samsung Galaxy Tab 4	4.610000	

6	Acer Iconia	1.096667
7	Acer ICONIA ONE	3.000000
8	Asus MeMO Pad	0.000000
9	ASUS Transformer Pad	1.666667
10	Asus T100TA-B1-GR	NaN
11	ASUS 8	4.800000
12	Asus VivoTab	NaN
13	Toshiba Encore	2.000000
14	Toshiba Excite	0.000000
15	Toshiba Satellite	4.450000
16	HP Slate	1.375000

	Mean Sale Price for Product Group	Total Reviews for Product Group
0	374.000000	81
1	218.500000	1718
2	479.990000	142
3	99.990000	2
4	348.656667	252
5	279.990000	450
6	154.936667	17
7	111.730000	2
8	133.970000	0
9	228.190000	1
10	NaN	0
11	160.130000	5
12	NaN	0
13	240.935000	8
14	461.180000	0
15	370.990000	22
16	107.425000	4

Next we move to Webstorm IDE where we create a project with the following directory:



Inside of index.html, we set the title, import the jsapi, latest jquery lib and the jquery csv lib that we'll use to load our csv file called "wmttablet_data.csv". We then load the google viz library and set a listener.

index.html ×

html head script

```
1   <!DOCTYPE html>
2   <html>
3   <head lang="en">
4       <title>Tablet Ratings, Prices at Walmart</title>
5       <script src="https://www.google.com/jsapi"></script>
6       <script src="http://code.jquery.com/jquery-latest.js"></script>
7       <script src="jquery.csv-0.71.js"></script>
8       <script>
9           //load the google viz library, set a listener
10          google.load("visualization", "1", {packages:["corechart"]});
11          google.setOnLoadCallback(drawChart);
12
```

Next, we create a callback function for the drawChart referenced in the google api listener. We also convert the loaded csv file into a two dimensional array and then create that two dimensional array into a data table that google viz can use to map data to our Bubble Chart. We also set columns that we want represented via the DataView object.

```
//create the callback function for drawChart reference
function drawChart(){
    //load csv data
    $.get("wmttablet_data.csv", function(csvString) {

        //convert the csv string into a two-dimensional array
        var arrayData = $.csv.toArrays(csvString, {onParseValue: $.csv.hooks.castToScalar});

        //convert the two-dimensional array into a data table product
        var data = new google.visualization.arrayToDataTable(arrayData);

        //create a view and set columns (Grouped Product Name, Mean AvgRating (per product group)
        // Mean Sale Price (per product group), TotalReviews (per product group)
        var view = new google.visualization.DataView(data);
        view.setColumns([0, 1, 2, 3]);
```

Next, we set options for our chart. We are creating a bubble chart that has horizontal and vertical axes. The horizontal axis represents mean avg rating per product group and the vertical axis represents mean sale price per product group. Each Bubble represents the 'value to consumer' of each product group.

Darker colors correspond to more user reviews while lighter colors correspond to less user reviews. Finally, hovering over Bubbles will reveal values of each key metric- Product Group Name, Mean Avg Rating, Avg Sale Price and Number of Reviews.

We set the configuration options for our Bubble Chart here:

```
//set options for bubble chart as well as horizontal and vertical axes
var options = {
    title: "Which Tablets Sold at Walmart Offer the Best Value for the Money?",
    'height':750,
    'width':1200,
    bubble: {textStyle: {fontSize: 8}},
    hAxis: {
        title: data.getColumnLabel(1),
        minValue: data.getColumnRange(1).min-2,
        maxValue: data.getColumnRange(1).max+2
    },
    vAxis: {
        title: data.getColumnLabel(2),
        minValue: data.getColumnRange(2).min,
        maxValue: data.getColumnRange(2).max+100

    },
```

Here we bind the buble chart to div and draw the specified View with the designated Options:

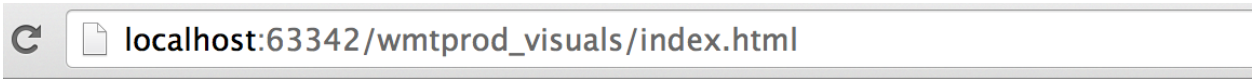
```
//bind bubble chart to <div>. Draw the current View with the selected options

var chart = new google.visualization.BubbleChart(document.getElementById('chart'));

chart.draw(view, options);
```

Here is the div:

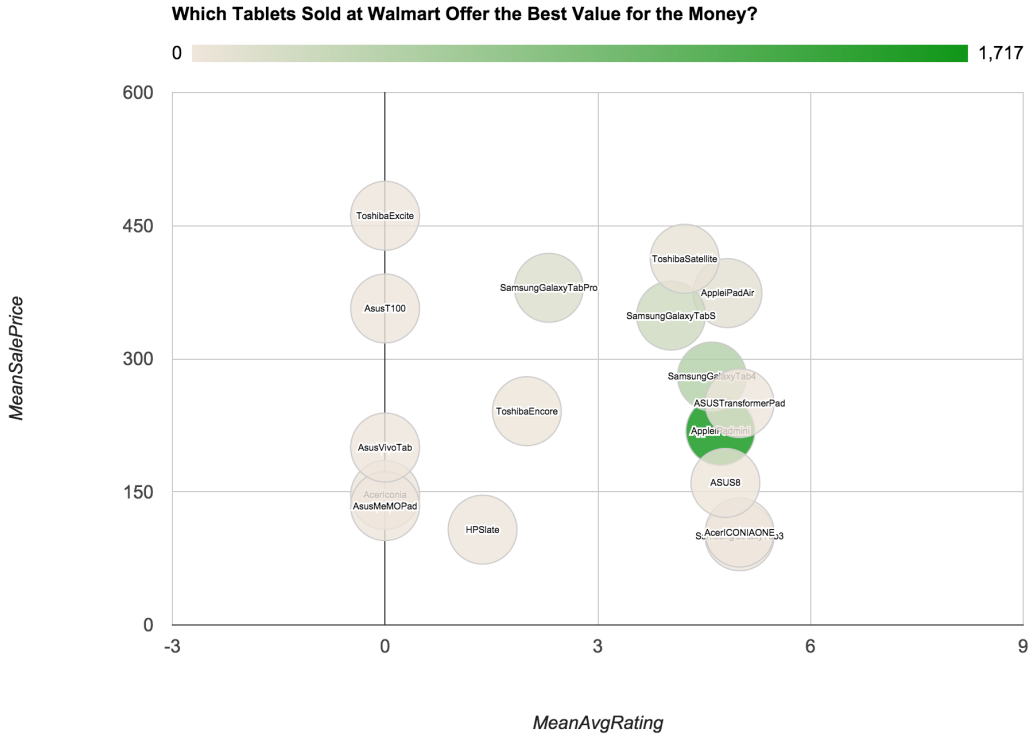
```
</head>
<body>
<div id = "chart"></div>

</body>
</html>
```

FINALLY,

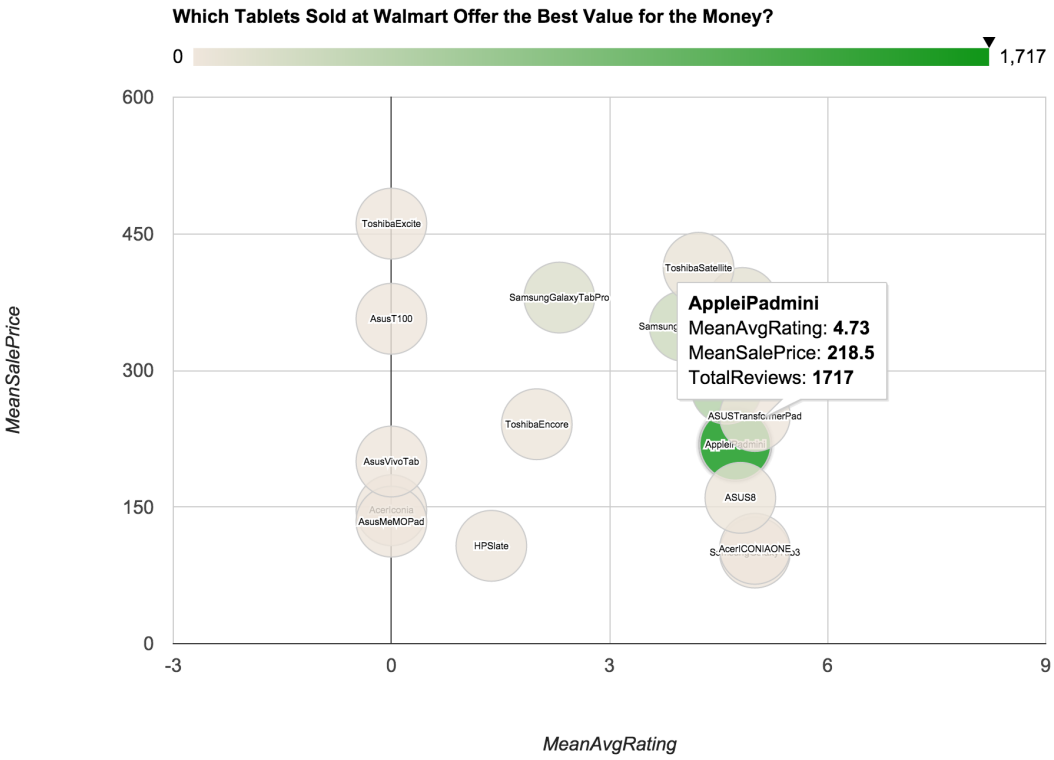
We run index.html on a local server as follows below:



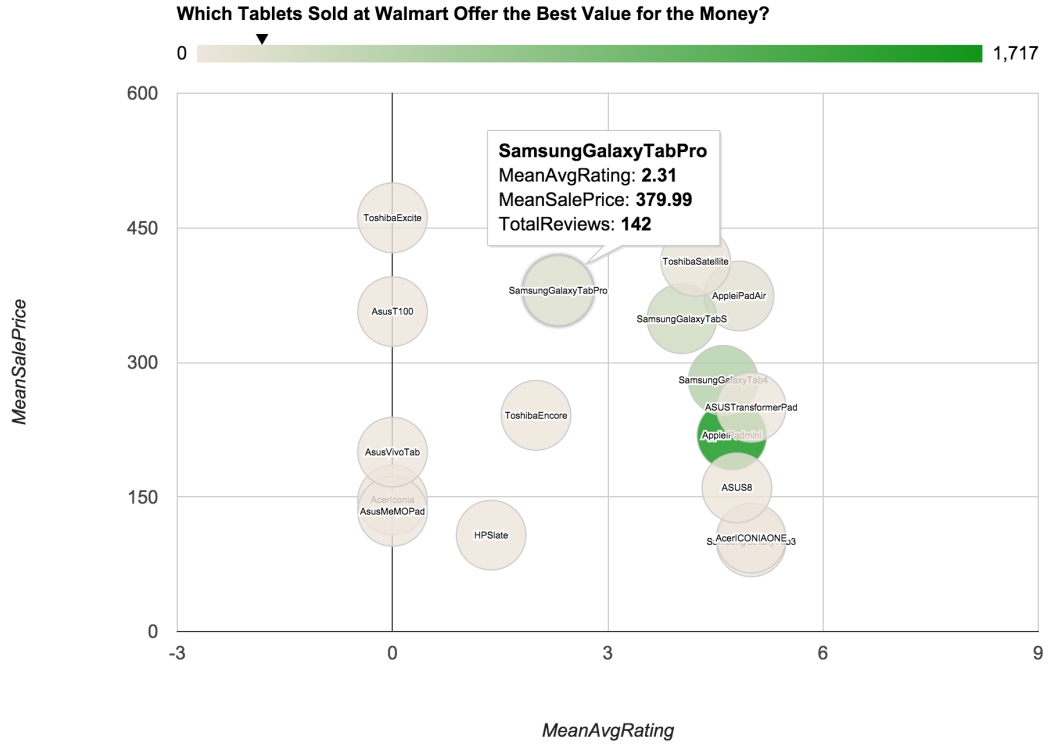
and we get:



If we hover over a bubble we see key metrics of each product group:



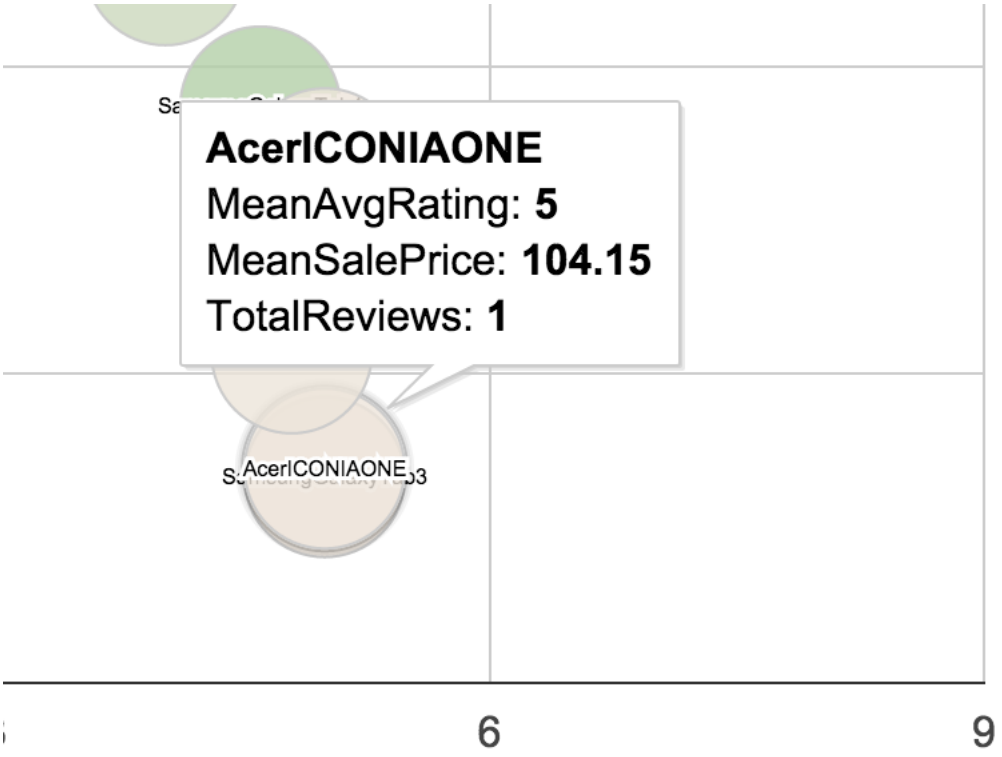
Above, we see that the Apple Ipad Mini is a solidly great value as compared to the Apple IPad Air, Galaxy Tab S and Galaxy Tab 4. The Apple Ipad Mini has 1,717 reviews that give an average rating of 4.73 across all product configurations and an average sale price of just $218.50.



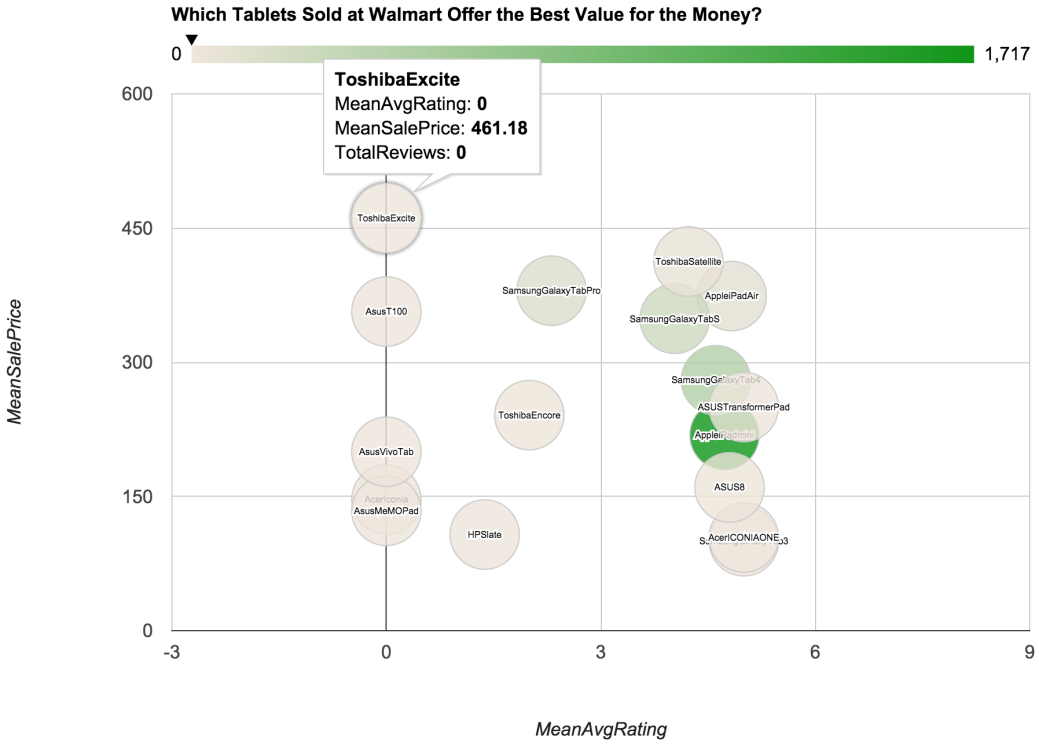
On the hand, the Samsung Galaxy Tab Pro has 142 reviews that give it a poor 2.31 average rating across all

configurations with a sale price that is $379.99.

The Acer ICONIA ONE below is pretty cheap at $104.15 with a high rating of 5; however, it only has 1 review so I'd be careful not to get carried away with this one.



There are several tablets that for whatever reason aren't selling at all despite having been posted for sale by Walmart such as the Toshiba Excite:



CONCLUSION

This visualization is interesting because it makes sense of a consumer process that is very confusing. The visualization illicits many different questions. For example, is there a correlation between the number of reviews of a product group and the popularity of that product group? For instance, the Apple Ipad Mini has a lot of reviews- can we also infer that the Apple Ipad Mini is the best selling tablet at Walmart?

The Apple Ipad Air and the Ipad Mini seem to have similar quality in terms of average consumer ratings; however, there are 5x the number of reviews for the Ipad Mini and the Ipad Mini costs nearly half of what the Ipad Air costs. The primary product difference is that the Ipad Air is thinner than the Ipad Mini. As such, perhaps consumers don't really care that much about how much thinner the Ipad Air is than the Ipad Mini and that's where the value to the consumer is!

As a consumer I'd stay away from the Galaxy Tab Pro, Toshiba Encore and any of the tablets that have zero reviews. I'd focus my purchase decision on the Galaxy Tab 4, Ipad Mini and I'd take a look at some of the very cheap Acer ICONIA One and Galaxy Tab 3 tablets that have higher ratings than even the Ipad Mini but only a few reviews. My logic there would be that maybe a cheap price is a deterrent for the average Walmart consumer that might mistaken price for quality.

In conclusion, this interactive visualization can help the Walmart consumer in our original example purchase the tablet that offers her the greatest possible value in the most efficient manner.