This repository Search Pull requests Issues Marketplace Gist

johnsblevins / MSDS7331-GroupProject

Watch 2 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights

Branch: master MSDS7331-GroupProject / OnlineNewsPopularityLab1.ipynb Find file Copy path

johnsblevins Updated Master 0f36b2b 3 hours ago

1 contributor

3333 lines (3332 sloc) 730 KB Raw Blame History

Analysis of Online News Popularity

Created by Phillip Efthimion, Scott Payne, Gino Varghese and John Blevins

MSDS 7331 Data Mining - Section 403 - Project 1

In this notebook a publicly available data set which is related to the popularities of online news sites will be analyzed. As part of the analysis the data will be introduced with a description of the source data items, collection strategies, experimental units and populations of interest. Python data analysis tools will then be used to verify data quality and tidy the data where needed. Descriptive statistics will be gathered and used to generate visualizations which will help in determining any interesting or meaningful relationships. Finally, a basic Principle Components Analysis (PCA) will be performed to assist with a possible dimension reduction strategy before attempting further analysis methods.

Business Understanding

The abundance of online news sources provides consumers with a diverse set of content choices. Content providers are constantly struggling to differentiate themselves from others in an effort to stand out and attract visitors to their site. It is of significant importance for authors and advertisers of these media outlets to understand what makes content relevant and interesting to the end user thereby driving traffic to their site. The data set utilized in this analysis includes online news popularity data collected by Mashable between 2013 and 2015. The dataset can provide interesting insights into the popularity of an article based on article metadata such as word patterns, media content, message sentiment, day of publication and more. In this analysis, the ability to predict popularity is examined based on these attributes. The goal of such an analysis would yield useful relationships in the data and lead to the development of a model which can predict the popularity rating of an article. This would be very helpful for social media contributors as they develop and publish content.

Data Meaning

The online news popularity data set utilized in this analysis is publicly accessible from the UCI machine learning repository at https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity (https://archive.ics.uci.edu/ml/datasets/Online+News+Popularity). The data was originally collected by Mashable based on articles published through their website between 2013 and 2015. The data set contains 39,644 data points with 61 attributes including 58 predictive attributes, 2 non-predictive attributes and 1 target attribute.

The 2 non-predictive attributes are as follows:

- URL of the news article
- Number of days between article publication and dataset acquisition

The explanatory attributes can be split into several different groupings as follows:

- Word Structure and Frequency
- Hyperlink References
- Digital Media References (Images and Videos)
- Channel Categorization and Keywords (content type such as lifestyle, entertainment, etc...)
- Publication Time
- Sentiment and Subjectivity Levels

The target attribute is the number of shares a site receives which indicates the popularity of the site. The complete list of attributes is shown in the following table:

Attribute	Data Type	Description
url	Object	URL of the article (non-predictive)
timedelta	Float	Days between the article publication and the dataset acquisition (non-predictive)
n_tokens_title	Float	Number of words in the title
n_tokens_content	Float	Number of words in the content
n_unique_tokens	Float	Rate of unique words in the content
n_non_stop_words	Float	Rate of non-stop words in the content
n_non_stop_unique_tokens	Float	Rate of unique non-stop words in the content
num_hrefs	Float	Number of links
num_self_hrefs	Float	Number of links to other articles published by Mashable
num_imgs	Float	Number of images

num_videos	Float	Number of videos
average_token_length	Float	Average length of the words in the content
num_keywords	Float	Number of keywords in the metadata
data_channel_is_lifestyle	Float	Is data channel 'Lifestyle'?
data_channel_is_entertainment	Float	Is data channel 'Entertainment'?
data_channel_is_bus	Float	Is data channel 'Business'?
data_channel_is_socmed	Float	Is data channel 'Social Media'?
data_channel_is_tech	Float	Is data channel 'Tech'?
data_channel_is_world	Float	Is data channel 'World'?
kw_min_min	Float	Worst keyword (min)
kw_max_min	Float	Worst keyword (max)
kw_avg_min	Float	Worst keyword (avg)
kw_min_max	Float	Best keyword (min)
kw_max_max	Float	Best keyword (max)
kw_avg_max	Float	Best keyword (avg)
kw_min_avg	Float	Avg keyword (min)
kw_max_avg	Float	Avg keyword (max)
kw_avg_avg	Float	Avg keyword (avg)
self_reference_min_shares	Float	Min of referenced articles in Mashable
self_reference_max_shares	Float	Max of referenced articles in Mashable
self_reference_avg_sharess	Float	Avg of referenced articles in Mashable
weekday_is_monday	Float	Was the article published on a Monday?
weekday_is_tuesday	Float	Was the article published on a Tuesday?
weekday_is_wednesday	Float	Was the article published on a Wednesday?
weekday_is_thursday	Float	Was the article published on a Thursday?
weekday_is_friday	Float	Was the article published on a Friday?
weekday_is_saturday	Float	Was the article published on a Saturday?
weekday_is_sunday	Float	Was the article published on a Sunday?
is_weekend	Float	Was the article published on the weekend?
LDA_00	Float	Closeness to LDA topic 0
LDA_01	Float	Closeness to LDA topic 1
LDA_02	Float	Closeness to LDA topic 2
LDA_03	Float	Closeness to LDA topic 3
LDA_04	Float	Closeness to LDA topic 4
global_subjectivity	Float	Text subjectivity
global_sentiment_polarity	Float	Text sentiment polarity
global_rate_positive_words	Float	Rate of positive words in the content
global_rate_negative_words	Float	Rate of negative words in the content
rate_positive_words	Float	Rate of positive words among non-neutral tokens
rate_negative_words	Float	Rate of negative words among non-neutral tokens
avg_positive_polarity	Float	Avg polarity of positive words
min_positive_polarity	Float	Min polarity of positive words
max_positive_polarity	Float	Max polarity of positive words
avg_negative_polarity	Float	Avg polarity of positive words
min_negative_polarity	Float	Min polarity of positive words
max_negative_polarity	Float	Max polarity of positive words
title_subjectivity	Float	Title subjectivity

title_subjectivity	Float	Title subjectivity
title_sentiment_polarity	Float	Title polarity
abs_title_subjectivity	Float	Absolute subjectivity level
abs_title_sentiment_polarity	Float	Absolute polarity level
Number of shares (target)	Integer	Number of Article Shares (tweets, shares, etc...)

Verify Data Quality

The data set is not a raw dataset and has already been cleaned prior to its upload on the UCI Machine Learning Repository. As a result, there are no missing values that need to be reconciled. All of the attributes are non-null floats, except for the attributes ‘url' and ‘shares'. ‘Url' is a non-null object data type and ‘shares’ is a non-null integer data type.

There are some records where the attribute ‘share’ makes the data point an outliers. This is due to some posts going viral and receiving a great deal of shares. These are not considered mistakes and will be included in the analysis.

To examine the data quality in more detail and perform subsequent analysis operations the following modules are imported:

- Pandas
- Numpy
- Matplotlib
- Seaborn
- Warnings

The Online New Popularity data set, which is in CSV format, is read into a dataframe using the read_csv() function. The first 5 rows of data can be shown using the head() function.

In [8]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
import warnings
warnings.simplefilter('ignore', DeprecationWarning)

df = pd.read_csv('data/OnlineNewsPopularity.csv')

df.head()
```

Out[8]:

	url	timedelta	n_tokens_title	n_tokens_content	n_unique_tokens	n_non_stop_wo
0	http://mashable.com/2013/01/07/amazon-instant-...	731.0	12.0	219.0	0.663594	1.0
1	http://mashable.com/2013/01/07/ap-samsung-spon...	731.0	9.0	255.0	0.604743	1.0
2	http://mashable.com/2013/01/07/apple-40-billio...	731.0	9.0	211.0	0.575130	1.0
3	http://mashable.com/2013/01/07/astronaut-notre...	731.0	9.0	531.0	0.503788	1.0
4	http://mashable.com/2013/01/07/att-u-verse-apps/	731.0	13.0	1072.0	0.415646	1.0

5 rows × 61 columns

Descriptive statistics for each attribute can be obtained using the describe() function.

In [9]:
```
df.describe()
```

Out[9]:

	timedelta	n_tokens_title	n_tokens_content	n_unique_tokens	n_non_stop_words	n_non_stop_unique_tokens	r
count	39644.000000	39644.000000	39644.000000	39644.000000	39644.000000	39644.000000	3
mean	354.530471	10.398749	546.514731	0.548216	0.996469	0.689175	1
std	214.163767	2.114037	471.107508	3.520708	5.231231	3.264816	1
min	8.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0
25%	164.000000	9.000000	246.000000	0.470870	1.000000	0.625739	4

50%	339.000000	10.000000	409.000000	0.539226	1.000000	0.690476	8
75%	542.000000	12.000000	716.000000	0.608696	1.000000	0.754630	1
max	731.000000	23.000000	8474.000000	701.000000	1042.000000	650.000000	3

8 rows × 60 columns

Data type information for each attribute can be verified using the info() function.

In [10]:
```
print (df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39644 entries, 0 to 39643
Data columns (total 61 columns):
url                            39644 non-null object
 timedelta                     39644 non-null float64
 n_tokens_title                39644 non-null float64
 n_tokens_content              39644 non-null float64
 n_unique_tokens               39644 non-null float64
 n_non_stop_words              39644 non-null float64
 n_non_stop_unique_tokens      39644 non-null float64
 num_hrefs                     39644 non-null float64
 num_self_hrefs                39644 non-null float64
 num_imgs                      39644 non-null float64
 num_videos                    39644 non-null float64
 average_token_length          39644 non-null float64
 num_keywords                  39644 non-null float64
 data_channel_is_lifestyle     39644 non-null float64
 data_channel_is_entertainment 39644 non-null float64
 data_channel_is_bus           39644 non-null float64
 data_channel_is_socmed        39644 non-null float64
 data_channel_is_tech          39644 non-null float64
 data_channel_is_world         39644 non-null float64
 kw_min_min                    39644 non-null float64
 kw_max_min                    39644 non-null float64
 kw_avg_min                    39644 non-null float64
 kw_min_max                    39644 non-null float64
 kw_max_max                    39644 non-null float64
 kw_avg_max                    39644 non-null float64
 kw_min_avg                    39644 non-null float64
 kw_max_avg                    39644 non-null float64
 kw_avg_avg                    39644 non-null float64
 self_reference_min_shares     39644 non-null float64
 self_reference_max_shares     39644 non-null float64
 self_reference_avg_sharess    39644 non-null float64
 weekday_is_monday             39644 non-null float64
 weekday_is_tuesday            39644 non-null float64
 weekday_is_wednesday          39644 non-null float64
 weekday_is_thursday           39644 non-null float64
 weekday_is_friday             39644 non-null float64
 weekday_is_saturday           39644 non-null float64
 weekday_is_sunday             39644 non-null float64
 is_weekend                    39644 non-null float64
 LDA_00                        39644 non-null float64
 LDA_01                        39644 non-null float64
 LDA_02                        39644 non-null float64
 LDA_03                        39644 non-null float64
 LDA_04                        39644 non-null float64
 global_subjectivity           39644 non-null float64
 global_sentiment_polarity     39644 non-null float64
 global_rate_positive_words    39644 non-null float64
 global_rate_negative_words    39644 non-null float64
 rate_positive_words           39644 non-null float64
 rate_negative_words           39644 non-null float64
 avg_positive_polarity         39644 non-null float64
 min_positive_polarity         39644 non-null float64
 max_positive_polarity         39644 non-null float64
 avg_negative_polarity         39644 non-null float64
 min_negative_polarity         39644 non-null float64
 max_negative_polarity         39644 non-null float64
 title_subjectivity            39644 non-null float64
 title_sentiment_polarity      39644 non-null float64
 abs_title_subjectivity        39644 non-null float64
 abs_title_sentiment_polarity  39644 non-null float64
 shares                        39644 non-null int64
dtypes: float64(59), int64(1), object(1)
memory usage: 18.5+ MB
```

```
None
```

Although no data is missing there are spaces at the beginning of each column which can cause problems. These spaces must be removed.

In [11]:
```
df.columns = df.columns.str.replace(' ', '')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39644 entries, 0 to 39643
Data columns (total 61 columns):
url                            39644 non-null object
timedelta                      39644 non-null float64
n_tokens_title                 39644 non-null float64
n_tokens_content               39644 non-null float64
n_unique_tokens                39644 non-null float64
n_non_stop_words               39644 non-null float64
n_non_stop_unique_tokens       39644 non-null float64
num_hrefs                      39644 non-null float64
num_self_hrefs                 39644 non-null float64
num_imgs                       39644 non-null float64
num_videos                     39644 non-null float64
average_token_length           39644 non-null float64
num_keywords                   39644 non-null float64
data_channel_is_lifestyle      39644 non-null float64
data_channel_is_entertainment  39644 non-null float64
data_channel_is_bus            39644 non-null float64
data_channel_is_socmed         39644 non-null float64
data_channel_is_tech           39644 non-null float64
data_channel_is_world          39644 non-null float64
kw_min_min                     39644 non-null float64
kw_max_min                     39644 non-null float64
kw_avg_min                     39644 non-null float64
kw_min_max                     39644 non-null float64
kw_max_max                     39644 non-null float64
kw_avg_max                     39644 non-null float64
kw_min_avg                     39644 non-null float64
kw_max_avg                     39644 non-null float64
kw_avg_avg                     39644 non-null float64
self_reference_min_shares      39644 non-null float64
self_reference_max_shares      39644 non-null float64
self_reference_avg_sharess     39644 non-null float64
weekday_is_monday              39644 non-null float64
weekday_is_tuesday             39644 non-null float64
weekday_is_wednesday           39644 non-null float64
weekday_is_thursday            39644 non-null float64
weekday_is_friday              39644 non-null float64
weekday_is_saturday            39644 non-null float64
weekday_is_sunday              39644 non-null float64
is_weekend                     39644 non-null float64
LDA_00                         39644 non-null float64
LDA_01                         39644 non-null float64
LDA_02                         39644 non-null float64
LDA_03                         39644 non-null float64
LDA_04                         39644 non-null float64
global_subjectivity            39644 non-null float64
global_sentiment_polarity      39644 non-null float64
global_rate_positive_words     39644 non-null float64
global_rate_negative_words     39644 non-null float64
rate_positive_words            39644 non-null float64
rate_negative_words            39644 non-null float64
avg_positive_polarity          39644 non-null float64
min_positive_polarity          39644 non-null float64
max_positive_polarity          39644 non-null float64
avg_negative_polarity          39644 non-null float64
min_negative_polarity          39644 non-null float64
max_negative_polarity          39644 non-null float64
title_subjectivity             39644 non-null float64
title_sentiment_polarity       39644 non-null float64
abs_title_subjectivity         39644 non-null float64
abs_title_sentiment_polarity   39644 non-null float64
shares                         39644 non-null int64
dtypes: float64(59), int64(1), object(1)
memory usage: 18.5+ MB
```

The binary fields for days of week and channel categories are replaced with shorter versions for easier consumption in later visualizations. After renaming the data columns we can view the updated names by running info() function.

In [12]:
```
df = df.rename(columns={'weekday_is_monday': 'monday', 'weekday_is_tuesday': 'tuesday', 'weekday_i
```

```
s_wednesday': 'wednesday', 'weekday_is_thursday': 'thursday', 'weekday_is_friday': 'friday', 'week
day_is_saturday': 'saturday', 'weekday_is_sunday': 'sunday', 'is_weekend': 'weekend'})
df = df.rename(columns={'data_channel_is_lifestyle':'lifestyle',
'data_channel_is_entertainment':'entertainment', 'data_channel_is_bus':'business', 'data_channel_i
s_socmed':'social_media', 'data_channel_is_tech':'technology', 'data_channel_is_world':'world'})
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39644 entries, 0 to 39643
Data columns (total 61 columns):
url                            39644 non-null object
timedelta                      39644 non-null float64
n_tokens_title                 39644 non-null float64
n_tokens_content               39644 non-null float64
n_unique_tokens                39644 non-null float64
n_non_stop_words               39644 non-null float64
n_non_stop_unique_tokens       39644 non-null float64
num_hrefs                      39644 non-null float64
num_self_hrefs                 39644 non-null float64
num_imgs                       39644 non-null float64
num_videos                     39644 non-null float64
average_token_length           39644 non-null float64
num_keywords                   39644 non-null float64
lifestyle                      39644 non-null float64
entertainment                  39644 non-null float64
business                       39644 non-null float64
social_media                   39644 non-null float64
technology                     39644 non-null float64
world                          39644 non-null float64
kw_min_min                     39644 non-null float64
kw_max_min                     39644 non-null float64
kw_avg_min                     39644 non-null float64
kw_min_max                     39644 non-null float64
kw_max_max                     39644 non-null float64
kw_avg_max                     39644 non-null float64
kw_min_avg                     39644 non-null float64
kw_max_avg                     39644 non-null float64
kw_avg_avg                     39644 non-null float64
self_reference_min_shares      39644 non-null float64
self_reference_max_shares      39644 non-null float64
self_reference_avg_sharess     39644 non-null float64
monday                         39644 non-null float64
tuesday                        39644 non-null float64
wednesday                      39644 non-null float64
thursday                       39644 non-null float64
friday                         39644 non-null float64
saturday                       39644 non-null float64
sunday                         39644 non-null float64
weekend                        39644 non-null float64
LDA_00                         39644 non-null float64
LDA_01                         39644 non-null float64
LDA_02                         39644 non-null float64
LDA_03                         39644 non-null float64
LDA_04                         39644 non-null float64
global_subjectivity            39644 non-null float64
global_sentiment_polarity      39644 non-null float64
global_rate_positive_words     39644 non-null float64
global_rate_negative_words     39644 non-null float64
rate_positive_words            39644 non-null float64
rate_negative_words            39644 non-null float64
avg_positive_polarity          39644 non-null float64
min_positive_polarity          39644 non-null float64
max_positive_polarity          39644 non-null float64
avg_negative_polarity          39644 non-null float64
min_negative_polarity          39644 non-null float64
max_negative_polarity          39644 non-null float64
title_subjectivity             39644 non-null float64
title_sentiment_polarity       39644 non-null float64
abs_title_subjectivity         39644 non-null float64
abs_title_sentiment_polarity   39644 non-null float64
shares                         39644 non-null int64
dtypes: float64(59), int64(1), object(1)
memory usage: 18.5+ MB
```

Descriptive Statistics

For the analysis and modeling of the online new data we will be using Python data analysis tools. After loading the dataset into Python built-in

Pandas' functions will be used for displaying simple statistics and matplotlib will be used to construct basic visualizations showing the relationships between explanatory and response variables.

Subsets of the existing week, word and channels dataframe can be created for basic analysis of each column.

In [13]:
```
# get a subset of the data for days of the week

# df_day = df[['weekday_is_monday', 'weekday_is_tuesday', 'weekday_is_wednesday', 'weekday_is_thur
sday', 'weekday_is_friday', 'weekday_is_saturday', 'weekday_is_sunday']]
df_day = df[['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday']]


df_tokens = df[['n_tokens_title', 'n_tokens_content', 'n_unique_tokens']]

df_channels = df[['lifestyle', 'entertainment', 'business', 'social_media', 'technology',
'world']]

df_day.describe()
#df_tokens.describe()
```

Out[13]:

	monday	tuesday	wednesday	thursday	friday	saturday	sunday
count	39644.000000	39644.000000	39644.000000	39644.000000	39644.000000	39644.000000	39644.000000
mean	0.168020	0.186409	0.187544	0.183306	0.143805	0.061876	0.069039
std	0.373889	0.389441	0.390353	0.386922	0.350896	0.240933	0.253524
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

The percentage of posts about social media can be determined.

In [14]:
```
# percentage of posts about social media
len(df[df.social_media==1])/len(df)*100.0
```

Out[14]:
```
5.85965089294723
```

The percentage of posts about technology can also be determined.

In [15]:
```
# percentage of posts about technology
len(df[df.technology==1])/len(df)*100.0
```

Out[15]:
```
18.52991625466653
```

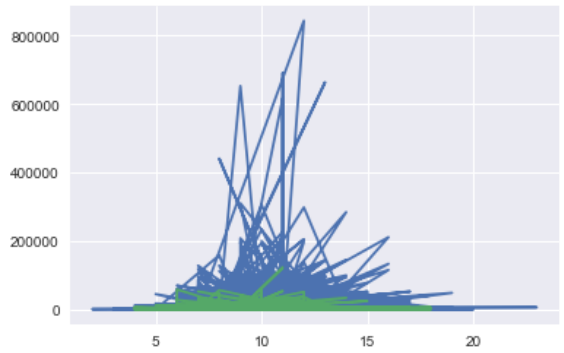
Using matplotlib visualization functions the plots for amount of words in title vs. shares and amount of words in title vs. shares for social media can be generated.

In [16]:
```
# amount of words in title vs. shares

plt.plot(df.n_tokens_title, df.shares)

# amount of words in title vs. shares for social media
plt.plot(df.n_tokens_title[df.social_media==1], df.shares[df.social_media==1])
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x2bbb76f0438>]
```



Descriptive statistitcs for tokens data frame can be shown using the describe() function.

In [17]:
```
df_tokens.describe()
```

Out[17]:

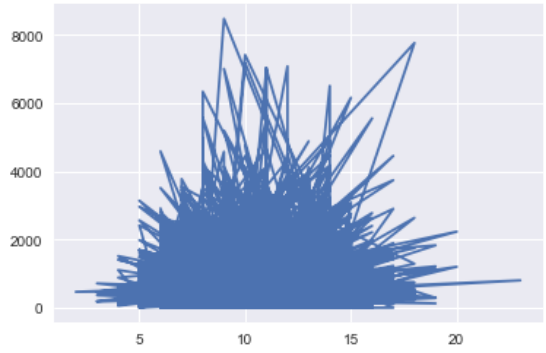
	n_tokens_title	n_tokens_content	n_unique_tokens
count	39644.000000	39644.000000	39644.000000
mean	10.398749	546.514731	0.548216
std	2.114037	471.107508	3.520708
min	2.000000	0.000000	0.000000
25%	9.000000	246.000000	0.470870
50%	10.000000	409.000000	0.539226
75%	12.000000	716.000000	0.608696
max	23.000000	8474.000000	701.000000

Using matplotlib visualization functions the plot amount of words in the title vs. number of shares. The plot shows us that there is no real correlation between the number of words in the title of articles and how many shares they get.

In [18]:
```
# The amount of words in the title vs. number of shares
plt.plot(df_tokens.n_tokens_title, df_tokens.n_tokens_content)

# The plot shows us that there is no real correlation between the number of words in the title of
 articles and how many shares they get.
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x2bbb71ec9e8>]
```



To determine whether social media or technology articles get more likes the average number of shares for social media articles can be determined.

In [19]:
```
np.mean(df[df.social_media==1].shares)
```

Out[19]:
```
3629.383125269049
```

In [20]:
```
np.mean(df[df.technology==1].shares)
```

Out[20]:
```
3072.283283419548
```

Therefore, social media articles are shared more times than technology articles, on average.

The words to images ratio can also be calculated.

In [21]:
```
np.mean(df.n_tokens_content / (df.num_imgs + 1)) # 1 added to denominator to prevent division by
 0.
```

Out[21]:
```
201.86719365551602
```

On average, the ratio of the number of words in an article per image is 201.87.

The words to images ratio for social media articles can be calculated.

In [22]:
```
np.mean(df[df.social_media==1].n_tokens_content / (df[df.social_media==1].num_imgs + 1)) # 1 added
 to denominator to prevent division by 0.
```

```
Out[22]: 217.17720937193357
```

On average, the ratio of the number of words in an article per image is 217.177.

The words to images ratio for technology articles can be calculated.

```
In [23]: np.mean(df[df.technology==1].n_tokens_content / (df[df.technology==1].num_imgs + 1)) # 1 added to
          denominator to prevent division by 0.
```

```
Out[23]: 185.32547223323883
```

On average, the ratio of the number of words in an article per image is 185.325.

The median number of total links in each article can be calculated.

```
In [24]: np.median(df.num_hrefs)
```

```
Out[24]: 8.0
```

The median number of links to other mashable articles in the each article can be calculated.

```
In [25]: np.median(df.num_self_hrefs)
```

```
Out[25]: 3.0
```

The range of the number of words in an article can be calculated.

```
In [26]: np.max(df.n_tokens_content) - np.min(df.n_tokens_content)
```

```
Out[26]: 8474.0
```

The variance in the number of shares for social media versus technology articles can be calculated.

```
In [27]: # Variance of shares for social media articles
         np.var(df[df.social_media==1].shares)
```

```
Out[27]: 30503285.451148584
```

```
In [28]: # Variance of shares for technology articles
         np.var(df[df.technology==1].shares)
```

```
Out[28]: 81427694.93241084
```

We can see that there is more variance in the number of shares for technology articles compared to social media articles by a factor of roughly 2.5.

The mean number of shares of social media articles can be calculated.

```
In [29]: np.mean(df[df.social_media==1].shares)
```

```
Out[29]: 3629.383125269049
```

The mean number of shares of technology articles can be calculated.

```
In [30]: np.mean(df[df.technology==1].shares)
```

```
Out[30]: 3072.283283419548
```

On average, articles about social media get shared more than articles about technology. As shown above, articles about social media also are less variable. This means that articles about social media are more likely to be shared than articles about technology. However, articles about technology could potentially become more viral (or shared much less) than articles about social media.

Data Visualization and Attribute Relationships

For the visualizaiton analysis a function will be created to consolidate all the weekday*is*[monday....sunday] columns to one single colum "weekday".

```
In [33]: plt.rcParams['figure.figsize']=(15,10)
```

```
def label_weekday (row):
   if row['monday'] == 1 :
      return "Monday"
   if row['tuesday'] == 1 :
      return "Tuesday"
   if row['wednesday'] == 1 :
      return "Wednesday"
   if row['thursday'] == 1 :
      return "Thursday"
   if row['friday'] == 1 :
      return "Friday"
   if row['saturday'] == 1 :
      return "Saturday"
   if row['sunday'] == 1 :
      return "Sunday"
   return 'Other'
```

The function will be applied to a new weekday column.

In [34]:
```
#adding a new weekday column
df['weekday'] = df.apply (lambda row: label_weekday (row),axis=1)
```

The new column will be reviewed to verify the data type.

In [35]:
```
df[['weekday']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39644 entries, 0 to 39643
Data columns (total 1 columns):
weekday    39644 non-null object
dtypes: object(1)
memory usage: 309.8+ KB
```

The unique data values in the column weekday can be reviewed.

In [36]:
```
df.weekday.unique()
```

Out[36]:
```
array(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday',
       'Sunday'], dtype=object)
```

A new column called "popularity" is created, with the shares data point split into either Low, Medium or High equal-sized buckets based on rank or based on sample quantiles.

In [37]:
```
#adding a new popularity column
df['popularity'] = pd.qcut(df['shares'].values, 3, labels=["Low","Medium","High"])
```

In [38]:
```
#review if the newly added column, had unique data from Low to High
df.popularity.unique()
```

Out[38]:
```
[Low, Medium, High]
Categories (3, object): [Low < Medium < High]
```

A subset of data was created to further the research on the data sets. The subset of data was called df_subset, and was limited to the 'num_imgs','num_videos','num_keywords','popularity','weekday' and 'shares' columns so that it would be easier to perform some ad-hoc analysis/visualization on these data sets.

In [39]:
```
# limiting to columns that are needed for visualization
df_subset = df[['num_imgs','num_videos','num_keywords','popularity','weekday','shares']].copy()
df_subset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39644 entries, 0 to 39643
Data columns (total 6 columns):
num_imgs        39644 non-null float64
num_videos      39644 non-null float64
num_keywords    39644 non-null float64
popularity      39644 non-null category
weekday         39644 non-null object
shares          39644 non-null int64
dtypes: category(1), float64(3), int64(1), object(1)
memory usage: 1.6+ MB
```

In [40]:
```
# limiting to columns that are needed for visualization
df_subset = df[['num_imgs','num_videos','num_keywords','popularity','weekday','shares']].copy()
```

```
df_subset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39644 entries, 0 to 39643
Data columns (total 6 columns):
num_imgs        39644 non-null float64
num_videos      39644 non-null float64
num_keywords    39644 non-null float64
popularity      39644 non-null category
weekday         39644 non-null object
shares          39644 non-null int64
dtypes: category(1), float64(3), int64(1), object(1)
memory usage: 1.6+ MB
```

In [41]:
```
# creating different cross tabs to perform further analysis
df_popularity = pd.crosstab([df['popularity']], df.shares.count())
df_week = pd.crosstab([df['weekday']], df.shares.count())
df_corr = pd.crosstab([df['popularity'], df['weekday']], df.shares.count())
```

In [42]:
```
print (df_popularity)
```

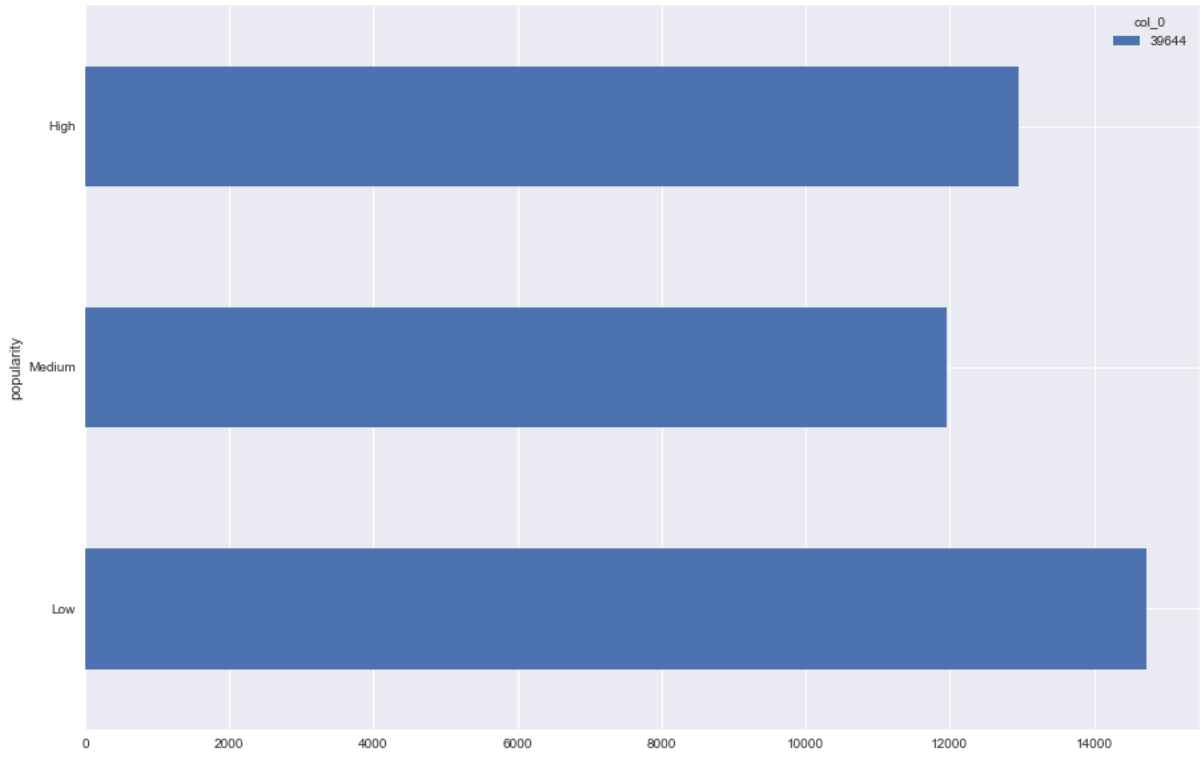
```
col_0       39644
popularity
Low         14732
Medium      11957
High        12955
```

In [43]:
```
print (df_week)
```

```
col_0      39644
weekday
Friday      5701
Monday      6661
Saturday    2453
Sunday      2737
Thursday    7267
Tuesday     7390
Wednesday   7435
```

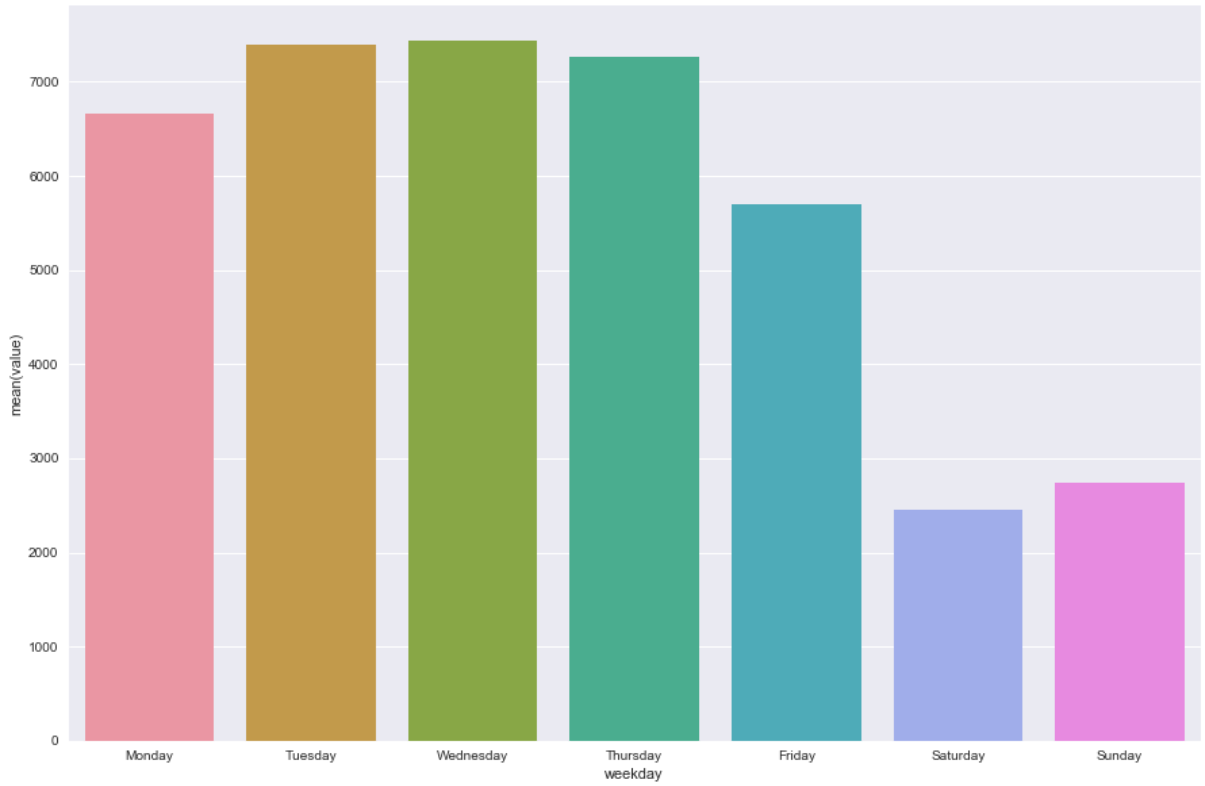
In [190]:
```
df_popularity.plot(kind='barh')
```

Out[190]: <matplotlib.axes._subplots.AxesSubplot at 0x29e10ffdcf8>



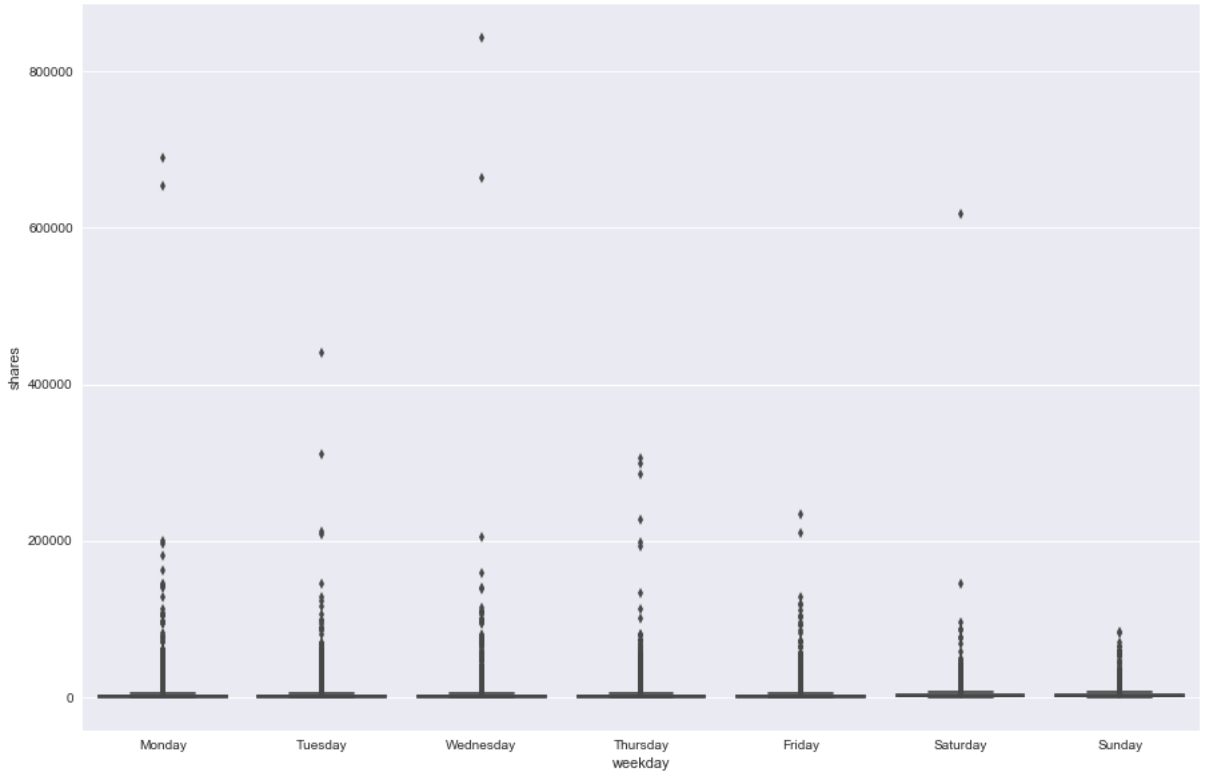
In this plot we split the column shares into equal-sized buckets based on rank or based on sample quantiles and labelled them as low, medium, high. [Low < Medium < High]. In this bar chart we are graphing the count of tweets against the number of times the tweet was shared. This is tracked in the popularity column as High, Medium and Low. We can see that the Low popularity had more counts because most of the articles fall under that criteria, due to the fact that they don't get shared as much.

In [191]:
```
df_weekstk = df_week.stack().reset_index().rename(columns={0:'value'})
ax = sns.barplot(x=df_weekstk.weekday, y=df_weekstk.value,order=["Monday", "Tuesday","Wednesday",
 "Thursday","Friday", "Saturday","Sunday"])
```



In this bar chart, we are graphing the count of tweets against the day the article was published (day of the week). This is tracked in the weekday column. We can see that articles published on Tuesday, Wednesday and Thursday get shared more than any other day of the week.

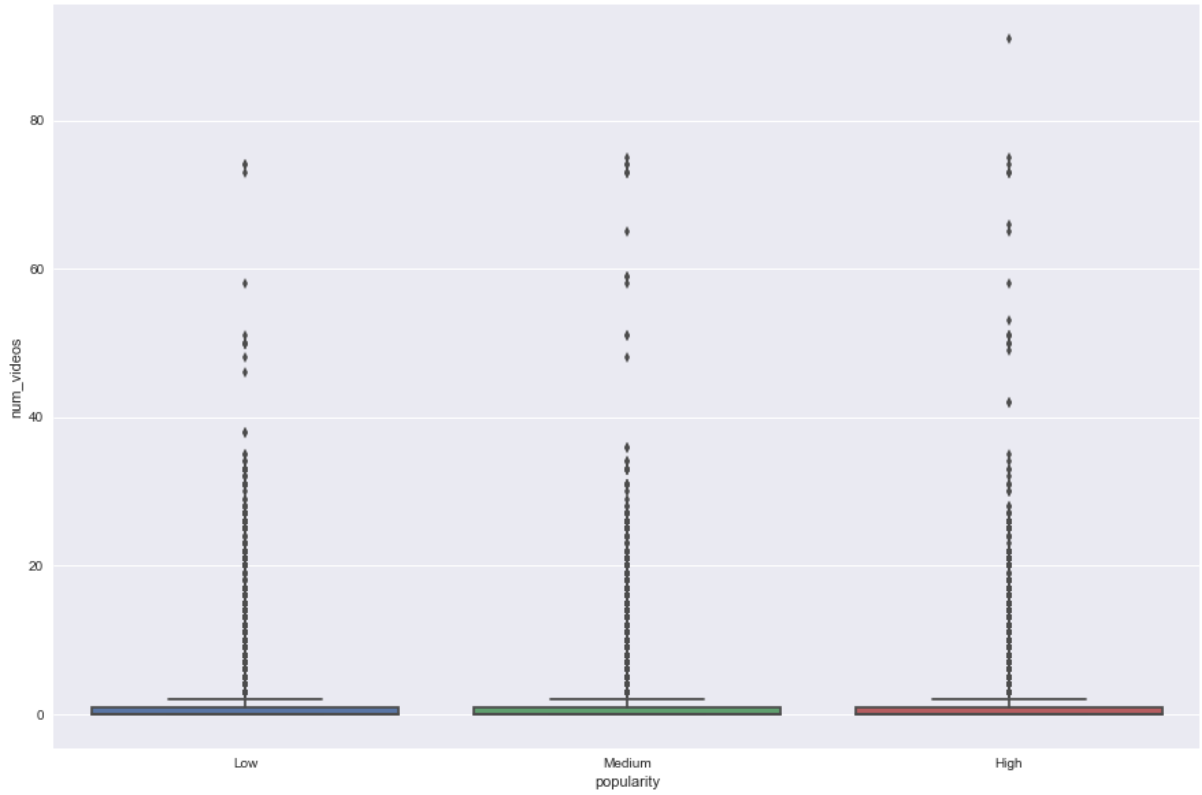
In [192]:
```
ax = sns.boxplot(x="weekday", y="shares",data=df_subset)
```



From the above box plot, we wanted to identifying any outliers and perform some comparisons by weekday. As we can see there are some extreme outliers present in the data: among them the data set for Monday, Tuesday, Wednesday and Saturday have extreme spread. From this we

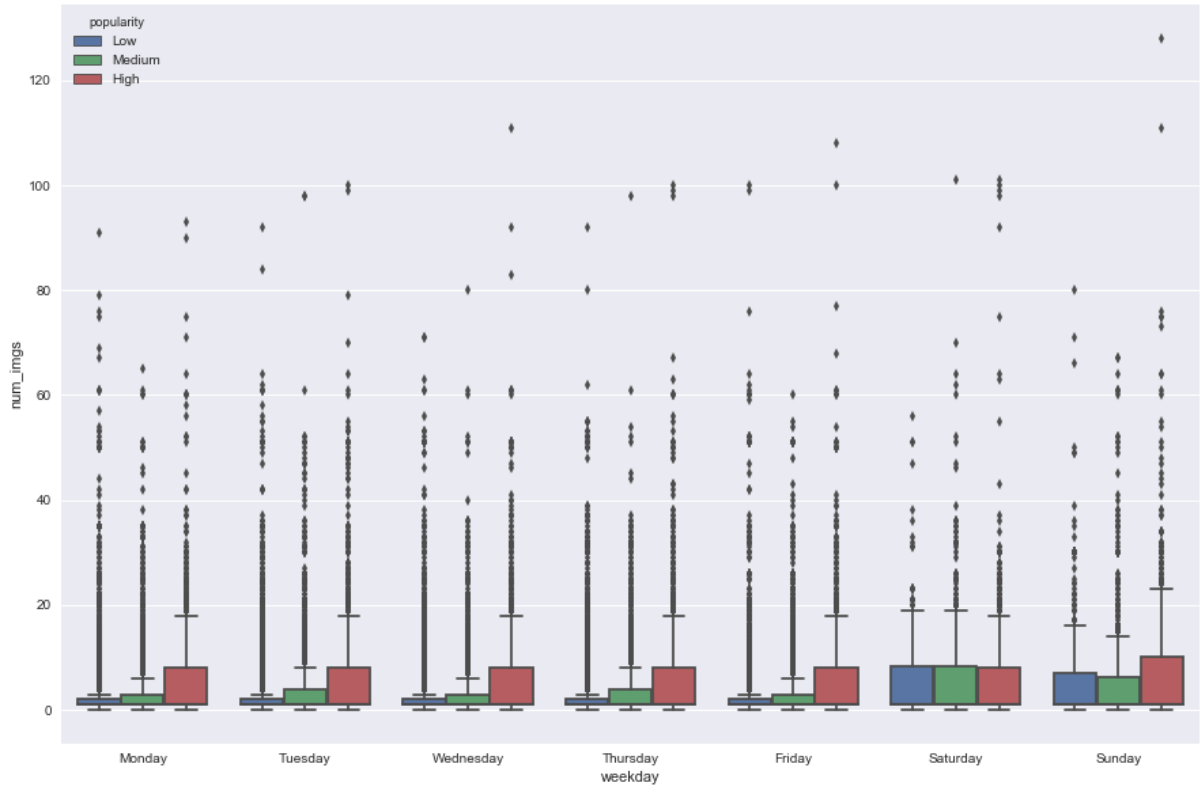
can speculate that there is no guarantee that an article will be shared more depending on the weekday it was published.

In [193]:
```
ax = sns.boxplot(x="popularity", y="num_videos",data=df_subset)
```



Similarly, to the previous box plot, outliers can be identified when comparing by popularity, as is shown by the three outcomes which had extreme outliers. Among these "high" had an extreme spread.

In [194]:
```
ax = sns.boxplot(x="weekday", y="num_imgs",hue="popularity",data=df_subset)
```

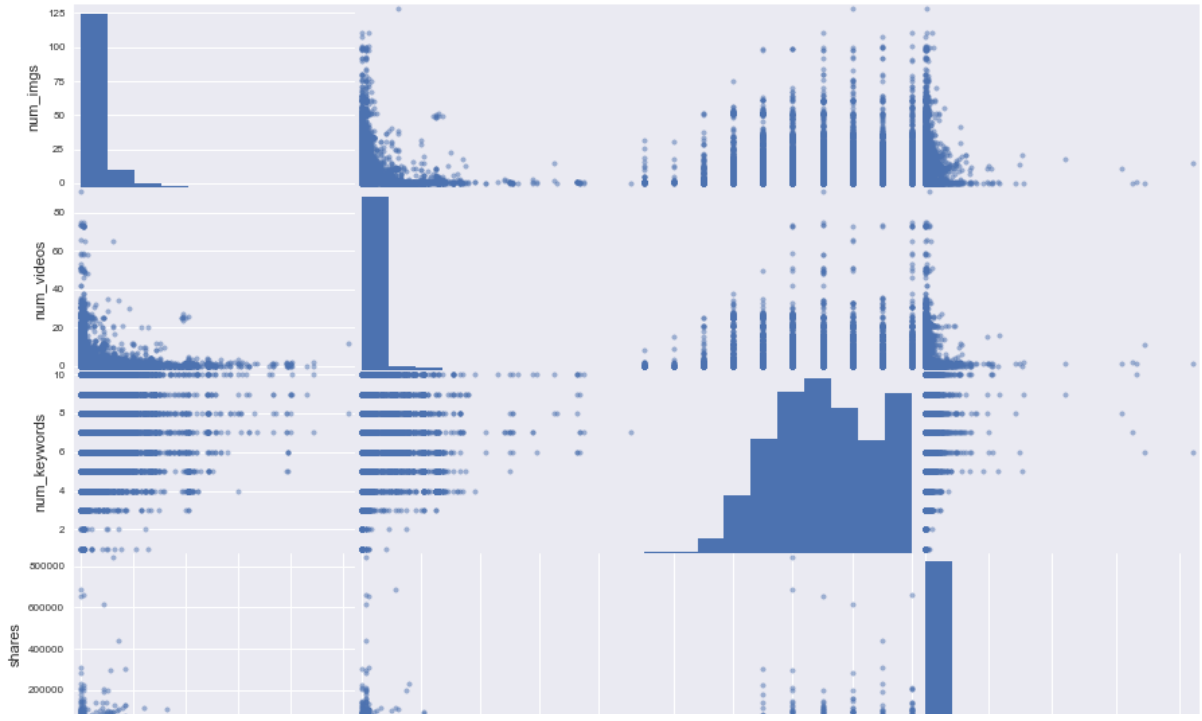


Similarly, to the previous box plot, we wanted to identify any outliers when comparing articles with number of images by weekday, as we can see all the outcomes had extreme outliers.

In [195]:
```
sns.pairplot(df_subset,hue='popularity')
```

Out[195]: <seaborn.axisgrid.PairGrid at 0x29e4fef7cf8>

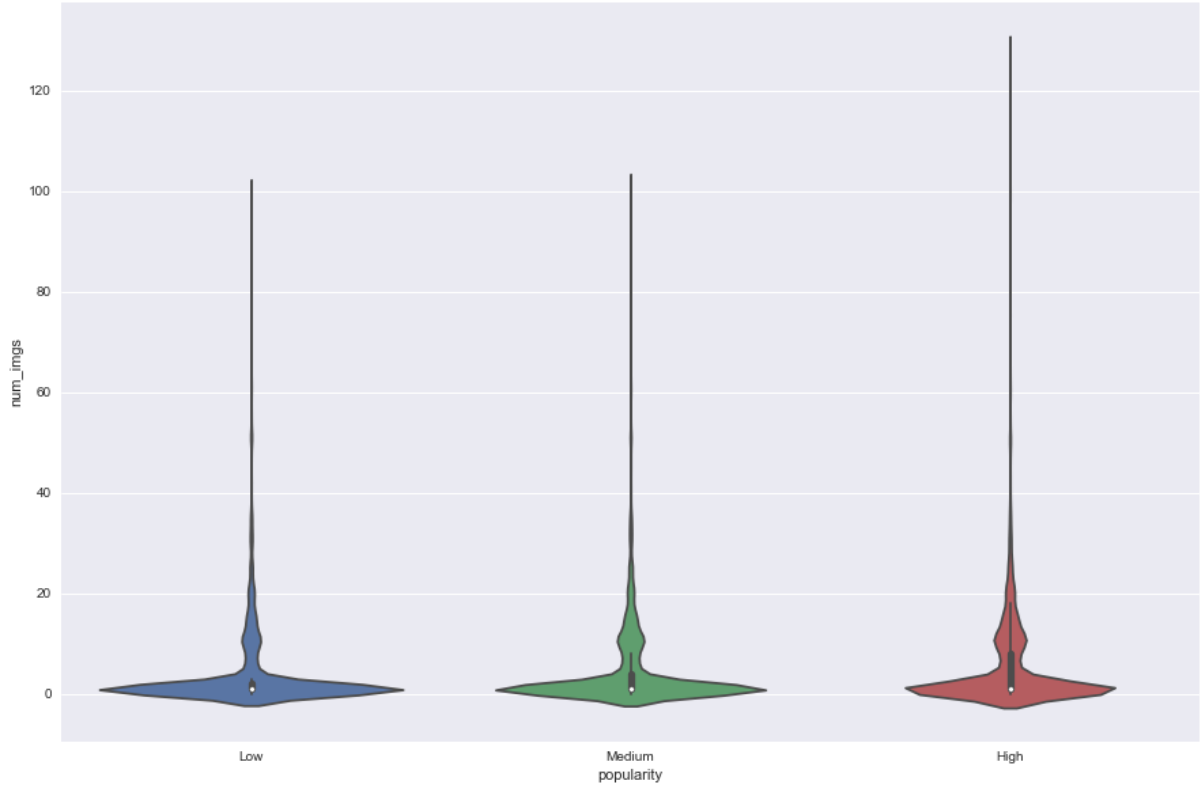
In [196]:
```
from pandas.tools.plotting import scatter_matrix
ax = scatter_matrix(df_subset, figsize=(15,10))
```



The Scatterplot matrix and the pairplot matrix show similar information. Data is not spread out in a linear fashion when looking at these plots, as at most the data points are clumped to the left, bottom corner of the graph. This shows that there is no valid correlation between the number of images increasing the popularity to "High" etc.

In [197]:
```
sns.violinplot(x="popularity", y="num_imgs",data=df_subset)
```

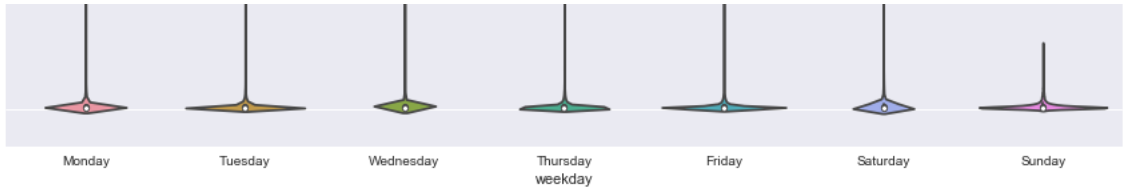
Out[197]: <matplotlib.axes._subplots.AxesSubplot at 0x29e22311048>



In this plot, we are graphing the number of images present in an article to it popularity. Similar to box plot, violin plots are used to represent comparison of a variable distribution (or sample distribution) across different "categories". From the above plot we can see how the density of data decreases as the popularity increases.

In [198]:
```
sns.violinplot(x="weekday", y="shares",data=df_subset)
```

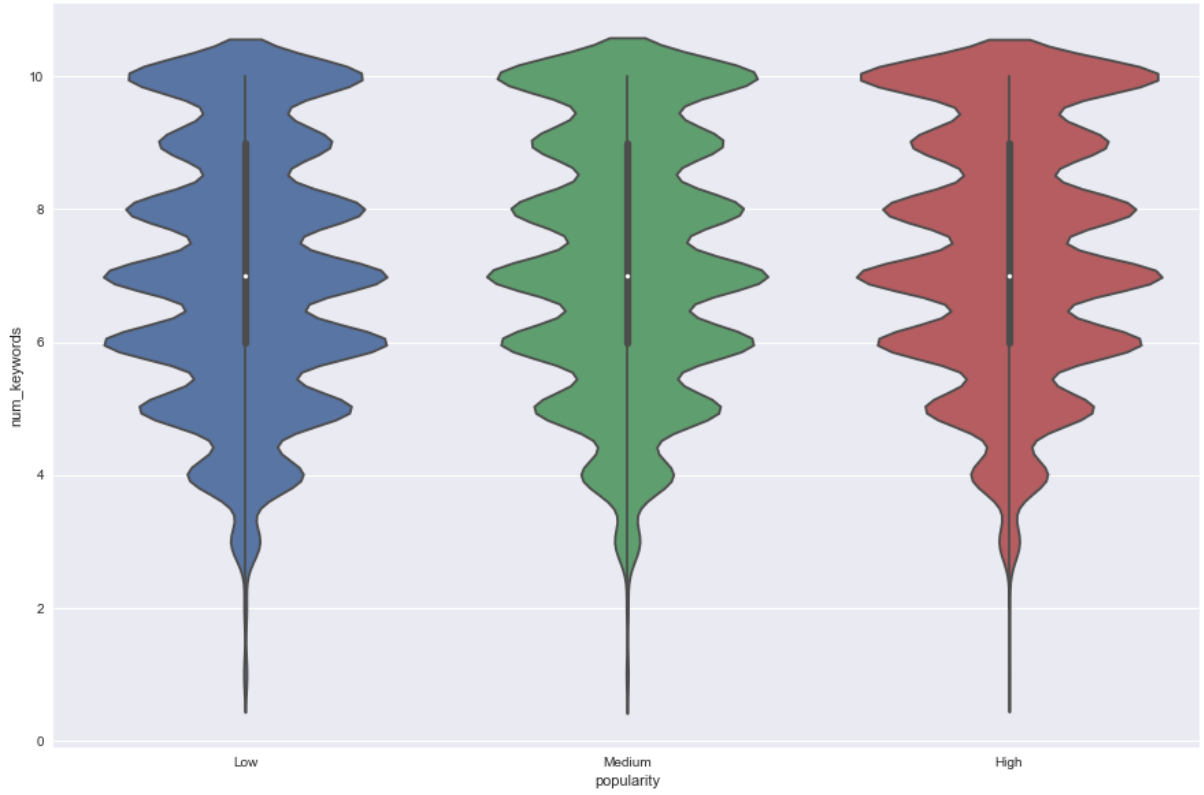
Out[198]: <matplotlib.axes._subplots.AxesSubplot at 0x29e1ebdf0f0>



In this plot, we are graphing the count of articles shared to the day it was published. From the violin plot we can see more data density for Tuesday, Friday and Sunday.

In [199]:
```
sns.violinplot(x="popularity", y="num_keywords",data=df_subset)
```

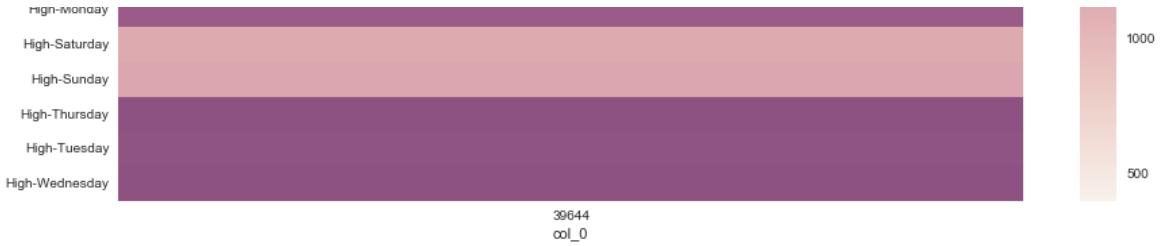
Out[199]: <matplotlib.axes._subplots.AxesSubplot at 0x29e17460a90>



In this plot, we are graphing number of keywords present in the article to its popularity. From the violin plot we can see similar data density for each of the popularity criteria.

In [200]:
```
sns.heatmap(df_corr)
```

Out[200]: <matplotlib.axes._subplots.AxesSubplot at 0x29e17703dd8>



From the heatmap we can see that the Low popularity should be a darker shade, because there are more articles that fall under that criteria, which means some articles were only shared a couple of times, and from the plot we can see that Wednesdays are when articles get read and shared; however, this does not mean the article is popular, it only means the article was read and it appealed to certain crowd.

New Features

Data Transformations

Before proceeding with further analysis it should be considered whether the raw data should be transformed. From the simple statistic plots we can see that some of the predictor variables are not normally distributed and are in fact are right-skewed. This violates the assumption of normality for many regression techniques and must be fixed if possible before proceeding. If the log of these values are taken and the procedure re-ran it may correct the distributions such that they become normally distributed. Before taking the log a small value of .001 should be added to each observation sample to ensure the log of any zero values aren't taken. If a log transform doesn't produce the desired output then a square root or other transformation may be required.

Population of Interest

The focus of this analysis is on online news articles only and does not include print or other forms of communication outlets. In addition the article data is collected from the Mashable website. As a consequence of this design decision we can only draw statistical inference with regard to these particular news articles from Mashable. It may be of interest, given adequate computing resources, to consider an analysis which includes more news outlets to draw inferences against the larger population.

URL Transformation

It may be desireable to transform the URL attribute for the purposes of classification between different media outlets. This may be used categorically to determine user preference based on media outlet.

PCA Analysis

A basic PCA and LDA analysis will be done on the raw data to help with dimension reduction strategy for future experimentation.

Data Preparation for PCA

First remove non-predictive variables ('url', ' timedelta')

In [133]:
```
df_drop=df.drop(labels=['url',' timedelta'],axis=1)
df_drop.head()
```

Out[133]:

	n_tokens_title	n_tokens_content	n_unique_tokens	n_non_stop_words	n_non_stop_unique_tokens	num_hrefs	num_se
0	12.0	219.0	0.663594	1.0	0.815385	4.0	2.0
1	9.0	255.0	0.604743	1.0	0.791946	3.0	1.0
2	9.0	211.0	0.575130	1.0	0.663866	3.0	1.0
3	9.0	531.0	0.503788	1.0	0.665635	9.0	0.0
4	13.0	1072.0	0.415646	1.0	0.540890	19.0	19.0

5 rows × 59 columns

Generate a list of columns in the updated dataframe.

In [134]:
```
cols = df_drop.columns.tolist()
cols
```

Out[134]:
```
[' n_tokens_title',
```

```
 ' n_tokens_content',
 ' n_unique_tokens',
 ' n_non_stop_words',
 ' n_non_stop_unique_tokens',
 ' num_hrefs',
 ' num_self_hrefs',
 ' num_imgs',
 ' num_videos',
 ' average_token_length',
 ' num_keywords',
 ' data_channel_is_lifestyle',
 ' data_channel_is_entertainment',
 ' data_channel_is_bus',
 ' data_channel_is_socmed',
 ' data_channel_is_tech',
 ' data_channel_is_world',
 ' kw_min_min',
 ' kw_max_min',
 ' kw_avg_min',
 ' kw_min_max',
 ' kw_max_max',
 ' kw_avg_max',
 ' kw_min_avg',
 ' kw_max_avg',
 ' kw_avg_avg',
 ' self_reference_min_shares',
 ' self_reference_max_shares',
 ' self_reference_avg_sharess',
 ' weekday_is_monday',
 ' weekday_is_tuesday',
 ' weekday_is_wednesday',
 ' weekday_is_thursday',
 ' weekday_is_friday',
 ' weekday_is_saturday',
 ' weekday_is_sunday',
 ' is_weekend',
 ' LDA_00',
 ' LDA_01',
 ' LDA_02',
 ' LDA_03',
 ' LDA_04',
 ' global_subjectivity',
 ' global_sentiment_polarity',
 ' global_rate_positive_words',
 ' global_rate_negative_words',
 ' rate_positive_words',
 ' rate_negative_words',
 ' avg_positive_polarity',
 ' min_positive_polarity',
 ' max_positive_polarity',
 ' avg_negative_polarity',
 ' min_negative_polarity',
 ' max_negative_polarity',
 ' title_subjectivity',
 ' title_sentiment_polarity',
 ' abs_title_subjectivity',
 ' abs_title_sentiment_polarity',
 ' shares']
```

Moves our response variable ' shares' to the first position(index 0)

In [135]:
```
cols.insert(0, cols.pop(cols.index(' shares')))
cols
```

Out[135]:
```
[' shares',
 ' n_tokens_title',
 ' n_tokens_content',
 ' n_unique_tokens',
 ' n_non_stop_words',
 ' n_non_stop_unique_tokens',
 ' num_hrefs',
 ' num_self_hrefs',
 ' num_imgs',
 ' num_videos',
 ' average_token_length',
 ' num_keywords',
 ' data_channel_is_lifestyle',
 ' data_channel_is_entertainment',
 ' data_channel_is_bus',
```

```
 ' data_channel_is_socmed',
 ' data_channel_is_tech',
 ' data_channel_is_world',
 ' kw_min_min',
 ' kw_max_min',
 ' kw_avg_min',
 ' kw_min_max',
 ' kw_max_max',
 ' kw_avg_max',
 ' kw_min_avg',
 ' kw_max_avg',
 ' kw_avg_avg',
 ' self_reference_min_shares',
 ' self_reference_max_shares',
 ' self_reference_avg_sharess',
 ' weekday_is_monday',
 ' weekday_is_tuesday',
 ' weekday_is_wednesday',
 ' weekday_is_thursday',
 ' weekday_is_friday',
 ' weekday_is_saturday',
 ' weekday_is_sunday',
 ' is_weekend',
 ' LDA_00',
 ' LDA_01',
 ' LDA_02',
 ' LDA_03',
 ' LDA_04',
 ' global_subjectivity',
 ' global_sentiment_polarity',
 ' global_rate_positive_words',
 ' global_rate_negative_words',
 ' rate_positive_words',
 ' rate_negative_words',
 ' avg_positive_polarity',
 ' min_positive_polarity',
 ' max_positive_polarity',
 ' avg_negative_polarity',
 ' min_negative_polarity',
 ' max_negative_polarity',
 ' title_subjectivity',
 ' title_sentiment_polarity',
 ' abs_title_subjectivity',
 ' abs_title_sentiment_polarity']
```

Re-index dataframe on new columns.

In [136]:
```
df_drop = df_drop.reindex(columns= cols)
df_drop.head()
```

Out[136]:

	shares	n_tokens_title	n_tokens_content	n_unique_tokens	n_non_stop_words	n_non_stop_unique_tokens	num_hrefs
0	593	12.0	219.0	0.663594	1.0	0.815385	4.0
1	711	9.0	255.0	0.604743	1.0	0.791946	3.0
2	1500	9.0	211.0	0.575130	1.0	0.663866	3.0
3	1200	9.0	531.0	0.503788	1.0	0.665635	9.0
4	505	13.0	1072.0	0.415646	1.0	0.540890	19.0

5 rows × 59 columns

Assign Explanatory variables and Response variables to x and y. Print the new dataframe dimension data.

In [139]:
```
X = df_drop.iloc[:,1:59].values
y = df_drop.iloc[:,0].values
X
np.shape(X)
```

Out[139]: (39644, 58)

Print dimension data for y variable.

In [140]:
```
y
np.shape(y)
```

```
Out[140]: (39644,)
```

PCA on all the variables

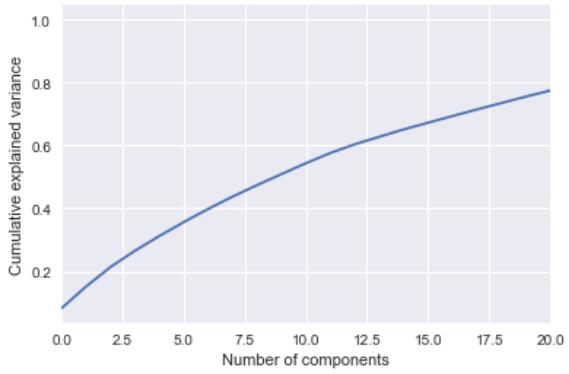
Uses the StandarScaler in sklearn to transform the data in preparation for PCA and then ran PCA to determine a suitable number of components to explain the variance.

```
In [141]: from sklearn.preprocessing import StandardScaler
          X_std = StandardScaler().fit_transform(X)
```

A sample plot shows cumulative explained variance versus number of components.

```
In [142]: from sklearn.decomposition import PCA
          pca = PCA().fit(X_std)
          plt.plot(np.cumsum(pca.explained_variance_ratio_))
          plt.xlim(0,20,1)
          plt.xlabel('Number of components')
          plt.ylabel('Cumulative explained variance')
```

```
Out[142]: <matplotlib.text.Text at 0x29e0bdefeb8>
```



Analysis

Initially I only ran PCA and graphed the first 10 Principal Components, this resulted in explaining less than 60% of the variance. I ran it a few more times and decided that around 20 Components is where the returns in explained covariance begins to diminish. With close to 80% of the variance explained using only 20 principal components, we are able to reduce the dimensionality of the data from 58 variables down to 20. This will help us when deciding what type of predictive analysis to run because we are able to explore a smaller number of features. While exploring the data it is worth noting that some of the variables in the original data are LDA components that were generated from the other features of the data set. I am not sure if these should be included in the PCA but I decided to leave them in because I could not find evidence that they should be removed. With more time devoted to PCA I think that the number of components necessary to explain a large portion of the variance could be reduced further. Another data concern is that the dummy variables that are included may not be suitable for PCA. It was difficult to find an answer on if binary variables(0,1 dummies) were appropriate for PCA. I would like to further explore PCA on different subsets of this data to see if the dimensions can be reduced while explaining as much variance as possible (preferably > 90%).

LDA Features Visualized

In a similar fashion LDA analysis plot can be generated.

```
In [146]: correlation = df[[' LDA_00', ' LDA_01', ' LDA_02', ' LDA_03', ' LDA_04',' shares']].corr()
          plt.figure(figsize=(10,10))
          sns.heatmap(correlation, vmax=1, square=True,annot=True,cmap='cubehelix')
```

© 2017 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub API Training Shop Blog About