Hacker News ¶

Hacker News is a site started by the startup incubator Y Combinator, where user-submitted stories (known as "posts") receive votes and comments, similar to reddit. Hacker News is extremely popular in technology and startup circles, and posts that make it to the top of the Hacker News listings can get hundreds of thousands of visitors as a result.

Data

The source data for this study can be found here (https://www.kaggle.com/hacker-news/hacker-news-posts). It contains almost 300,000 rows, each row representing a post. The data is of 2016. However, for this study we make use of a version that been reduced to approximately 20,000 rows by removing all submissions that did not receive any comments, and then randomly sampling from the remaining submissions. This file was prepared by Dataquest and can be downloaded from here (https://app.dataquest.io/m/356/guided-project%3A-exploring-hacker-news-posts/1/introduction).

Let us start with reading in the data, and displaying the header row and a small sample.

- id: the unique identifier from Hacker News for the post
- title: the title of the post
- url: the URL that the posts links to, if the post has a URL
- num_points: the number of points the post acquired, calculated as - - the total number of upvotes minus the total number of downvotes
- num_comments: the number of comments on the post
- author: the username of the person who submitted the post
- created_at: the date and time of the post's submission

- Do Ask HN or Show HN receive more comments on average?
- Do posts created at a certain time receive more comments on average?

Opening the Data Set

In [1]:
```
from csv import reader                                  # importing CSV module to
opened_file=open('hacker_news.csv', encoding='UTF-8')
read_file=reader(opened_file)
hacker_news_full=list(read_file)                        # The whole data set in Li

header=hacker_news_full[0]                            # Getting the header of the a
hn_full=hacker_news_full[1:]                                # The whole data set
print("The number of rows in data set: ", len(hn_full) )
```

```
The number of rows in data set:  20100
```

Function to find Index along with Column name

In [58]:
```
def indexer(dataset_header, variable_name):         # Input the dataset header
    index=0
    print("Index of {}".format(variable_name))        # printing the variable n
    for column_name in dataset_header:              # finding each column name
        print(index, ' : ', column_name)
        index+=1
    print('\n')

indexer(header,"Hacker New Data")
```

```
Index of Hacker New Data
0  :  id
1  :  title
2  :  url
3  :  num_points
4  :  num_comments
5  :  author
6  :  created_at
```

"Okay, so there is 'num-points' at index 3 and 'num_comments' at index 4. Now I am going to find out how many rows of these data set satify the conditions of having comments and points."

Removing rows with no comments and points

In [3]:
```
hn=[]                                          # Creating an empty list to sto

for row in hn_full:                            # Iterating through each rows o
    comments=int(row[4])
    points=int(row[3])
    if comments >0 and points>0:              # Applying the condition to creat
        hn.append(row)

print("Length of new data set is ", len(hn))
```

```
Length of new data set is  20100
```

In [4]:
```
def explore_data(dataset, start, end, rows_and_columns=False):
    dataset_slice = dataset[start:end]
    for row in dataset_slice:
        print(row)
        print('\n')

    if rows_and_columns:
        print('Number of rows:', len(dataset))
        print('Number of columns:', len(dataset[0]))
```

In [5]:
```
print(header)
```

```
['id', 'title', 'url', 'num_points', 'num_comments', 'author', 'created_at']
```

Creating Seperate list for Ask, Show and Other posts

The startswith() function will be used to find if the string starts with given slice of string. Here in this case, if it starts with 'ask hn' or 'show hn'. To standardise the string, it will be converted into smaller case by using lower function.

In [6]:
```
ask_posts = []
show_posts = []
other_posts = []

for row in hn:
    title = row[1].lower()
    if title.startswith('ask hn'):
        ask_posts.append(row)
    elif title.startswith('show hn'):
        show_posts.append(row)
    else:
        other_posts.append(row)
```

In [7]:
```
explore_data(ask_posts,0,2,True)
```

```
['12296411', 'Ask HN: How to improve my personal website?', '', '2', '6', 'ah
medbaracat', '8/16/2016 9:55']


['10610020', 'Ask HN: Am I the only one outraged by Twitter shutting down sha
re counts?', '', '28', '29', 'tkfx', '11/22/2015 13:43']


Number of rows: 1744
Number of columns: 7
```

In [8]:
```
print("No. of Ask posts", len(ask_posts))                    # printing the num
print("No. of Show posts", len(show_posts))                   # printing the nu
print("No. of Other posts", len(other_posts))                  # printing the n
print("Total length of Hacker News", len(hn))                  # checking the
print("Total",len(ask_posts)+len(show_posts)+len(other_posts))  # comparing wi
```

```
No. of Ask posts 1744
No. of Show posts 1162
No. of Other posts 17194
Total length of Hacker News 20100
Total 20100
```

Finding the Average Number of Comments and Points

In [18]:
```
total_ask_comments = 0
for row in ask_posts:
    num_comments = int(row[4])
    total_ask_comments += num_comments
avg_ask_comments = total_ask_comments / len(ask_posts)


print("Total Comments for Ask Posts: ", total_ask_comments)
print('Average comment count for Ask posts: {:.4f}'.format(avg_ask_comments))
```

```
Total Comments for Ask Posts:  24483
Average comment count for Ask posts: 14.0384
```

In [19]:
```
total_show_comments = 0
for row in show_posts:
    num_comments = int(row[4])
    total_show_comments += num_comments
avg_show_comments = total_show_comments / len(show_posts)

print("Total Comments for Show Posts: ", total_show_comments)
print('Average comment count for Show posts: {:.4f}'.format(avg_show_comments))
```

```
Total Comments for Show Posts:  11988
Average comment count for Show posts: 10.3167
```

It appears that 'ask' posts receive more comments on average than 'show' posts.

To analyze whether particular times of the day attact more comments, we will continue with these "ask" posts. We can check what time is the best time to post? For example if there are more users online at a specific time, there will be more upvotes in that time. So if we can find out the best time to post, that will be great.

Finding the Number of Ask Posts and Comments by Hour Created

In [13]:
```
import datetime as dt          # importing datetime module to work with date &
# Create a list that contains the creation times and number of comments (ask-p
result_list = []               #a list of lists
for row in ask_posts:
    created_at = row[6]
    num_comments = int(row[4])
    result_list.append([created_at, num_comments])

print(result_list[0:3])
```

```
[['8/16/2016 9:55', 6], ['11/22/2015 13:43', 29], ['5/2/2016 10:14', 1]]
```

In [14]:
```
# Build frequency tables for the number of posts and for the number of comment
counts_by_hour ={}              #the number of posts per hour of the day
comments_by_hour = {}           #the number of comments per hour of the day
for row in result_list:
    created_at = dt.datetime.strptime(row[0], '%m/%d/%Y %H:%M')
    created_at_hour = created_at.strftime('%H')
    if created_at_hour in counts_by_hour:
        counts_by_hour[created_at_hour] += 1
        comments_by_hour[created_at_hour] += row[1]
    else:
        counts_by_hour[created_at_hour] = 1
        comments_by_hour[created_at_hour] = row[1]

print("Here is the Number of posts in each hour \n \n", counts_by_hour)
print("\n Here is the Number of comments in each hour \n \n", comments_by_hour
```

```
Here is the Number of posts in each hour

 {'09': 45, '13': 85, '10': 59, '14': 107, '16': 108, '23': 68, '12': 73, '1
7': 100, '15': 116, '21': 109, '20': 80, '02': 58, '18': 109, '03': 54, '05':
46, '19': 110, '01': 60, '22': 71, '08': 48, '04': 47, '00': 55, '06': 44, '0
7': 34, '11': 58}

 Here is the Number of comments in each hour

 {'09': 251, '13': 1253, '10': 793, '14': 1416, '16': 1814, '23': 543, '12':
687, '17': 1146, '15': 4477, '21': 1745, '20': 1722, '02': 1381, '18': 1439,
'03': 421, '05': 464, '19': 1188, '01': 683, '22': 479, '08': 492, '04': 337,
'00': 447, '06': 397, '07': 267, '11': 641}
```

Second method of above task:

In [21]:
```
# Creating a dictionary with Hours as Key and Number of Comments as value.


import datetime as dt          # importing datetime module to work with date &



hourly_comment={}               # Empty dictionary to store number of comment
hourly_post= {}                  # Empty dictionary to store number of posts

for row in ask_posts:              # Iterating through each row of ask post li
    time_string = row[6]            # assigning row[6], "created at" to time g
    comments_number = int(row[4])    # Assigning the integer value of number o


        # Converting time given in String to datetime object. Sample time is g
        # This is in the format of month/date/4 digit Year Hour: Minute --> %m


    converted_time = dt.datetime.strptime(time_string,"%m/%d/%Y %H:%M")
    hour_posted = converted_time.strftime("%H")   # Assigning datetime object


        # Creating a frequency table using Dictionary

    if hour_posted in hourly_comment:                    # if hour_posted is
        hourly_comment[hour_posted] += comments_number     # Add up the commen
        hourly_post[hour_posted] += 1                       # Add 1 to number

    else:                                                   # if it is not pres
        hourly_comment[hour_posted] = comments_number     # Assign the value
        hourly_post[hour_posted] = 1                     # Assign the value as


print("Here is the Number of posts in each hour \n \n", hourly_post)
print("\n Here is the Number of comments in each hour \n \n", hourly_comment)
```

```
Here is the Number of posts in each hour

 {'09': 45, '13': 85, '10': 59, '14': 107, '16': 108, '23': 68, '12': 73, '1
7': 100, '15': 116, '21': 109, '20': 80, '02': 58, '18': 109, '03': 54, '05':
46, '19': 110, '01': 60, '22': 71, '08': 48, '04': 47, '00': 55, '06': 44, '0
7': 34, '11': 58}

 Here is the Number of comments in each hour

 {'09': 251, '13': 1253, '10': 793, '14': 1416, '16': 1814, '23': 543, '12':
687, '17': 1146, '15': 4477, '21': 1745, '20': 1722, '02': 1381, '18': 1439,
'03': 421, '05': 464, '19': 1188, '01': 683, '22': 479, '08': 492, '04': 337,
'00': 447, '06': 397, '07': 267, '11': 641}
```

Now that I have two dictionaries with number of comments and number of posts against each hours, I can straight away find the average number of comments posted in each hours.

Finding the Average Comments per post in Each Hour

A list will be created to list down the hour and average number of post in each hour. For the ease of sorting we will keep the first column as the average number of posts per hour.

In [69]:
```
# Create a table that contains the hours of day and the average number of comm
avg_comments_by_hour = []
avg_by_hour_rev = []

for hour, counts in counts_by_hour.items():
    avg_comments = comments_by_hour[hour] / counts
    avg_comments_by_hour.append([hour, avg_comments])
    avg_by_hour_rev.append([avg_comments, hour])

# Sort the list (on its first element, being the hour of day)
avg_comments_by_hour.sort()

# Print the result
output = "For hour {} the average number of comments per post is {:.2f}"
for row in avg_comments_by_hour:
    print (output.format(row[0], row[1]))
```

```
For hour 00 the average number of comments per post is 8.13
For hour 01 the average number of comments per post is 11.38
For hour 02 the average number of comments per post is 23.81
For hour 03 the average number of comments per post is 7.80
For hour 04 the average number of comments per post is 7.17
For hour 05 the average number of comments per post is 10.09
For hour 06 the average number of comments per post is 9.02
For hour 07 the average number of comments per post is 7.85
For hour 08 the average number of comments per post is 10.25
For hour 09 the average number of comments per post is 5.58
For hour 10 the average number of comments per post is 13.44
For hour 11 the average number of comments per post is 11.05
For hour 12 the average number of comments per post is 9.41
For hour 13 the average number of comments per post is 14.74
For hour 14 the average number of comments per post is 13.23
For hour 15 the average number of comments per post is 38.59
For hour 16 the average number of comments per post is 16.80
For hour 17 the average number of comments per post is 11.46
For hour 18 the average number of comments per post is 13.20
```

Sorting the Average comments per post per hour

Now that I have a list of average comments per post per each hour, I can sort it to see at what hour most number of comments were made per post. To sort the list, now I am going to use sorted() function in descending order.

In [70]:
```
sorted_avg = sorted(avg_by_hour_rev, reverse=True)
print ("The sorted averages per post per hour is here \n")
sorted_avg
```

```
The sorted averages per post per hour is here
```

Out[70]:
```
[[38.5948275862069, '15'],
 [23.810344827586206, '02'],
 [21.525, '20'],
 [16.796296296296298, '16'],
 [16.009174311926607, '21'],
 [14.741176470588234, '13'],
 [13.440677966101696, '10'],
 [13.233644859813085, '14'],
 [13.20183486238532, '18'],
 [11.46, '17'],
 [11.383333333333333, '01'],
 [11.051724137931034, '11'],
 [10.8, '19'],
 [10.25, '08'],
 [10.08695652173913, '05'],
 [9.41095890410959, '12'],
 [9.022727272727273, '06']
```

Making a short Report

From this list I can say that 3PM, 2AM, 8PM are the top 3 hours to post to generate most number of comments according to this data. Next, we will write some code to generate a report kind of thing. So I am going to use some formatting techinques.

In [72]:
```
# Create a list that is sorted on the average number of comments instead
swap_avg_by_hour = []
for row in avg_comments_by_hour:
    swap_avg_by_hour.append([row[1], row[0]])
print(swap_avg_by_hour)

# Created a sorted version of this list
sorted_swap = sorted(swap_avg_by_hour, reverse = True)

# Display the results
print('\n Top 5 Hours for Ask Posts Comments', '\n')
output = "{}: {:.2f} average comments per post"
for row in sorted_swap[:5]:
    thetime = dt.datetime.strptime(str(row[1]), '%H')
    thetime = thetime.strftime('%H:%M')
    print ( output.format(thetime,row[0] ))
```

```
[[8.127272727272727, '00'], [11.383333333333333, '01'], [23.810344827586206,
'02'], [7.796296296296297, '03'], [7.170212765957447, '04'], [10.086956521739
13, '05'], [9.022727272727273, '06'], [7.852941176470588, '07'], [10.25, '0
8'], [5.5777777777777775, '09'], [13.440677966101696, '10'], [11.051724137931
034, '11'], [9.41095890410959, '12'], [14.741176470588234, '13'], [13.2336448
59813085, '14'], [38.5948275862069, '15'], [16.796296296296298, '16'], [11.4
6, '17'], [13.20183486238532, '18'], [10.8, '19'], [21.525, '20'], [16.009174
311926607, '21'], [6.746478873239437, '22'], [7.985294117647059, '23']]

 Top 5 Hours for Ask Posts Comments

15:00: 38.59 average comments per post
02:00: 23.81 average comments per post
20:00: 21.52 average comments per post
16:00: 16.80 average comments per post
21:00: 16.01 average comments per post
```

second method:

In [23]:
```
for row in sorted_avg[:5]:                                          # iterating

    time = dt.datetime.strptime(row[1],"%H").strftime("%I %p")      # convertin
        # %I shows Hour in 12 hour format, %p shows AM or PM

    avg = row[0]                                                    # Assigning

        # Printing using .format method. Argument this_time is stored at time
        # this_avg is formatted to have 2 decimal points

    print(" If you post at {this_time}, you have a chance of getting {this_avg
          .format(this_time=time,this_avg=avg))


    # Printing our conclusion

print("So the best time to post to get good traction is {} EST".format(
    dt.datetime.strptime(sorted_avg[0][1],"%H").strftime("%I %p")))
```

```
 If you post at 03 PM, you have a chance of getting 38.59

 If you post at 02 AM, you have a chance of getting 23.81

 If you post at 08 PM, you have a chance of getting 21.52

 If you post at 04 PM, you have a chance of getting 16.80

 If you post at 09 PM, you have a chance of getting 16.01

So the best time to post to get good traction is 03 PM EST
```

Mini Conclusion

I think we have come to conclusion regarding what time to post. From the data we analysed, which is the data collected during 2016, we collected a subset of the data which has comments and points.

From that data set I can tell you that posting under Ask HN category can create a better engangement which leads to more number of comments.

But if you want more comments instantly, I have listed down some better time to post. 3PM, 2AM, 8PM, 4PM and 9PM are those times. Out of this 3PM is the best time according to our analysis with 38.59 as the average comments per post per hour.

So in short if you want to become popular on Hacker News, you need to find posts that you can submit on Ask HN Category and post them at 3PM EST

We will be working on this data a bit more and try to find if there is any connection with the authors, if someone is doing better than others

---

Relation between Authors and number of comments

In the mean time I got really interested in it and started digging deep to find some kind of correlation between the authors and number of comments. So I created a dictionary to find the comment distribution among the authors.

In [36]:
```
# Creating a Author - Comment distribution

authors={}                          # Creating an empty dictionary to store the
for row in ask_posts:                # iterating through ask_posts
    name=row[5]                       # assigning the name of the authors
    comment=int(row[4])                # assigning the number of comments
    if name in authors:                 # Checking if authors name is present i
        authors[name] += comment         # if it is present add the number of c
    else:                                 # if not
        authors[name] = comment            # assign the first number of comment


author_list=[]                                  # Creating a list with author n

for name in authors:                              # Iterating through author di
    author_list.append([authors[name], name])      # append the empty list with


sorted_list=sorted(author_list, reverse=True)        # Sort the list in descen
print(sorted_list[:10])                                # Print the first 10 au
```

```
[[3046, 'whoishiring'], [868, 'mod50ack'], [691, 'boren_ave11'], [531, 'schap
pim'], [520, 'sama'], [489, 'prmph'], [383, 'Apocryphon'], [309, 'curiousga
l'], [284, 'mikemajzoub'], [258, 'philippnagel']]
```

Finding Titles and corresponding comments

Now that we have a list of authors with highest number of comments, let me go through each author and find at least **10 titles of their posts and corresponding comments** they received. First 5 author's will be analysed to see if anything can be deduced from it.

In [65]:
```
# Printing the details of the author with highest comments

count=0

for row in ask_posts:                                                    # Iterati
    if (row[5]=='whoishiring' and count<11):                             # finding
        count+=1
        print( "Title :",row[1], "\n", "No. of comments: ", row[4])  #Printing
```

```
Title : Ask HN: Who wants to be hired? (June 2016)
 No. of comments:  250
Title : Ask HN: Freelancer? Seeking freelancer? (December 2015)
 No. of comments:  93
Title : Ask HN: Who is hiring? (September 2016)
 No. of comments:  910
Title : Ask HN: Who wants to be hired? (August 2016)
 No. of comments:  118
Title : Ask HN: Freelancer? Seeking freelancer? (September 2016)
 No. of comments:  85
Title : Ask HN: Who is hiring? (August 2016)
 No. of comments:  947
Title : Ask HN: Who wants to be hired? (April 2016)
 No. of comments:  283
Title : Ask HN: Freelancer? Seeking freelancer? (November 2015)
 No. of comments:  158
Title : Ask HN: Who wants to be hired? (March 2016)
 No. of comments:  202
```

In [59]:
```
# Printing author with 2nd highest comments

count=0

for row in ask_posts:
    if (row[5]=='mod50ack'):
        count+=1
        print( "Title :",row[1], "\n", "No. of comments: ", row[4])
```

```
Title : Ask HN: What's the best tool you used to use that doesn't exist anymo
re?
 No. of comments:  868
```

In [45]:
```
# Printing author with 3rd highest comments

count=0
for row in ask_posts:

    if (row[5]=='boren_ave11' and count<11):
        count+=1
        print( "Title :",row[1], "\n", "No. of comments: ", row[4])
```

```
Title : Ask HN: How much do you make at Amazon? Here is how much I make at Am
azon
 No. of comments:  691
```

In [63]:
```
# Printing author with 4th highest comments

count=0
for row in ask_posts:

    if (row[5]=='schappim' and count<11):
        count+=1
        print( "Title :",row[1], "\n", "No. of comments: ", row[4])
```

```
Title : Ask HN: What book have you given as a gift?
 No. of comments:  514
Title : Ask HN: Is it feasible to port Apple's Swift to the ESP8266?
 No. of comments:  17
```

In [47]:
```
# Printing author with 5th highest comments

count=0
for row in ask_posts:

    if (row[5]=='sama' and count<11):
        count+=1
        print( "Title :",row[1], "\n", "No. of comments: ", row[4])
```

```
Title : Ask HN: What should we fund at YC Research?
 No. of comments:  520
```

More Conclusions

From this analysis it came to my understanding that by consistently posting about Hiring related questions, the author 'whoishiring' received the top number of comments. So talking about Hiring process can be a good idea. But after going through Hacker News portal, I understood that this is a periodic post created by the team behind Hacker News to help with the recruitment process. So maybe this is not where my friend should focus.

Second highest commented post asks about Tools, third one about making profits from Amazon, fourth one about given books and the fifth one funding at YC Research. So there are a few things I could possibly infer from these topics.

- Ask a simple question that everyone can relate to, and people will respond, for eg about an obsolete tool. Maybe talk about MS Paint and some people might get all nostalgic while

others talk about it being completely obsolete.
- Ask about something that can make people curious, such as some one's online earning from Amazon or blog etc
- Ask about things that everyone has some kind of opinion on, for eg what would you tell your book given as a gift.
- Ask about any open topic in funding, might create an engaging discussion.

If one can find simple, genuine yet curious, at times controversial topics that can create emotions in people, which also have connection with the life experience, I think they can create some popular posts on Hacker News channel. That is what I am able to infer from this data set.

Now I have to write this all in an email and send it to my friend. What a Sunday it was! Such a fun day! All thanks to Hacker News data set!

In [ ]: