Basics

June 25, 2019

1 Exploring Hacker News Posts

Hacker News is a social news website geared towards computer science and entrepreneurship. It is run by Paul Graham's investment fund and startup incubator called Y Combinator. On the site, users can submit posts that are voted and commented on (like Reddit).

The data that we will be using is a subset of this dataset. Our subset contains about 20,000 rows.

Column Name	Description
`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 it the post has one
`num_points`	The number of points the post acquired (number of upvotes - number of downvotes)
`num_comments`	The number of comments that were made on the post
`author`	The username of the person who submitted the post
`created_at`	The date and time at which the post was submitted

For our analysis, we will be looking at titles that start with either `Ask HN` or `Show HN`. `Ask HN` posts allow a user to ask the Hacker News Community a question. `Show HN` allows users to show the Hacker News community a project, product, etc.

We want to look at these two types of posts to determine: * Do `Ask HN` or `Show HN` recieve more comments on average? * Do posts created at a certain time receive more comments on average?

1.1 Introduction

Let's start by loading in the Hacker News dataset.

```
In [1]: # Read in the file
        from csv import reader
        opened_file = open('hacker_news.csv')
        read_file = reader(opened_file)
```

```
        hn = list(read_file)

        #Print first 5 rows
        print(hn[:5])
```

```
[['id', 'title', 'url', 'num_points', 'num_comments', 'author', 'created_at'], ['12224879', 'In
```

1.2 Removing the Column Headers

In order to start analyzing our data, we must remove the header row.

```
In [2]: # Remove the Header
        headers = hn[0]
        hn = hn[1:]
        print(headers)
        print("\n")
        print(hn[0:5])
```

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


[['12224879', 'Interactive Dynamic Video', 'http://www.interactivedynamicvideo.com/', '386', '5
```

Each row contains the title of the post, the url in the post, the number of points the post received, the number of comments on the post, the author of the post, and the time the post was created.

1.3 Filter Data for Ask HN and Show HN Posts

First, we will isolate the posts that begin with either `Ask HN` or `Show HN` and separate those two types of posts into separate lists.

```
In [3]: ask_posts = []
        show_posts = []
        other_posts = []

        for post in hn:
            title = post[1]
            title_lower = title.lower()
            if title_lower.startswith("ask hn"):
                ask_posts.append(post)
            elif title_lower.startswith("show hn"):
                show_posts.append(post)
            else:
                other_posts.append(post)

        print("Number of Ask Posts: " + str(len(ask_posts)))
```

```
        print("Number of Show Posts: " + str(len(show_posts)))

        print("Number of Other Posts: " + str(len(other_posts)))
```

```
Number of Ask Posts: 1744
Number of Show Posts: 1162
Number of Other Posts: 17194
```

1.4 Calculating the Average Number of Comments on Ask HN and Show HN Posts

Now that we have separated the lists, we can do some analysis. Here, we will find the average number of comments for `Ask HN` and `Show HN` posts.

```
In [4]: print(ask_posts[0:4])
```

```
[['12296411', 'Ask HN: How to improve my personal website?', '', '2', '6', 'ahmedbaracat', '8/
```

```
In [5]: total_ask_comments = 0
        for post in ask_posts:
            total_ask_comments += int(post[4])

        avg_ask_comments = total_ask_comments/len(ask_posts)
        print("Average Number of Comments on Ask Posts: " + str(avg_ask_comments))
```

```
Average Number of Comments on Ask Posts: 14.038417431192661
```

```
In [6]: total_show_comments = 0
        for post in show_posts:
            total_show_comments += int(post[4])

        avg_show_comments = total_show_comments/len(show_posts)
        print("Average Number of Comments on Show Posts: " + str(avg_show_comments))
```

```
Average Number of Comments on Show Posts: 10.31669535283993
```

On average, `Ask FN` posts tend to have more comments than `Show FN` posts, with `Ask FN` and `Show FN` posts getting 14 and 10 comments respectvely. This makes sense because people are more likely to make a comment when a question is presented. Since the ask posts are more popular, we will direct our attention there.

1.5 Calculating the Average Number of Points on Ask HN and Show HN Posts

```
In [7]: # Ask Posts Avrg Points
        total_ask_points = 0
        for post in ask_posts:
            total_ask_points += int(post[3])

        avg_ask_points = total_ask_points/len(ask_posts)
        print("Average Number of Points on Ask Posts: " + str(avg_ask_points))

        # Show Posts Avrg Points
        total_show_points = 0
        for post in show_posts:
            total_show_points += int(post[3])

        avg_show_points = total_show_points/len(show_posts)
        print("Average Number of Points on Show Posts: " + str(avg_show_points))
```

```
Average Number of Points on Ask Posts: 15.061926605504587
Average Number of Points on Show Posts: 27.555077452667813
```

Interesting. `Show FN` posts on average receive more points than `Ask FN` posts. This means that show posts tend to receive around 83% more points than ask posts.

1.6 Finding the Number of Ask Posts and Comments by the Hour Created

We want to test if time has an effect on the amount of comments a post receives. Specifically, is there a certain time that is more like to attract more comments.

```
In [8]: import datetime as dt

        result_list = []
        for post in ask_posts:
            date_created = post[6]
            num_comments = int(post[4])
            result_list.append([date_created, num_comments])

        counts_by_hour = {}
        comments_by_hour = {}

        for row in result_list:
            date = row[0]
            comment = row[1]
            date_dt = dt.datetime.strptime(date, "%m/%d/%Y %H:%M")
            hour = date_dt.strftime("%H")
            if hour not in counts_by_hour:
                counts_by_hour[hour] = 1
```

```
            comments_by_hour[hour] = comment
        else:
            counts_by_hour[hour] += 1
            comments_by_hour[hour] += comment
    comments_by_hour
```

```
Out[8]: {'00': 447,
 '01': 683,
 '02': 1381,
 '03': 421,
 '04': 337,
 '05': 464,
 '06': 397,
 '07': 267,
 '08': 492,
 '09': 251,
 '10': 793,
 '11': 641,
 '12': 687,
 '13': 1253,
 '14': 1416,
 '15': 4477,
 '16': 1814,
 '17': 1146,
 '18': 1439,
 '19': 1188,
 '20': 1722,
 '21': 1745,
 '22': 479,
 '23': 543}
```

1.7 Calculating Average Number of Comments on Ask HN Posts per Hour

Below we are going to use our two dictionaries that we created above to find the average number of comments for Ask HN Posts.

```
In [9]: avg_by_hour = []
        for hour in comments_by_hour:
            avg_comments = comments_by_hour[hour] / counts_by_hour[hour]
            avg_by_hour.append ([hour, avg_comments])
        avg_by_hour
```

```
Out[9]: [['06', 9.022727272727273],
 ['17', 11.46],
 ['16', 16.796296296296298],
 ['09', 5.5777777777777775],
 ['13', 14.741176470588234],
 ['14', 13.233644859813085],
 ['15', 38.5948275862069],
```

```
['20', 21.525],
['08', 10.25],
['02', 23.810344827586206],
['07', 7.852941176470588],
['19', 10.8],
['12', 9.41095890410959],
['00', 8.127272727272727],
['11', 11.051724137931034],
['23', 7.985294117647059],
['22', 6.746478873239437],
['21', 16.009174311926607],
['05', 10.08695652173913],
['01', 11.383333333333333],
['18', 13.20183486238532],
['03', 7.796296296296297],
['04', 7.170212765957447],
['10', 13.440677966101696]]
```

1.8 Sorting & Printing the Top 5 Values from a List of Lists

In [10]:
```
'''
Swap the ordering of the "Year, Avg_comments" pairs to be
"Avg_comments, Year"
'''
swap_avg_by_hour = []
for row in avg_by_hour:
    swap_avg_by_hour.append([row[1], row[0]])
print(swap_avg_by_hour)
print("\n")

# Sort Values by Average Number of Comments in desc order
sorted_swap = sorted(swap_avg_by_hour, reverse = True)
print("Top 5 Hours for Ask Posts Comments")

time_template = "{}: {:.2f} average comments per post"
for row in sorted_swap[:5]:
    hour_dt = dt.datetime.strptime(row[1], "%H")
    hour_dtf = hour_dt.strftime("%H:%M")
    avg_comments = row[0]
    print(time_template.format(hour_dtf, avg_comments))
```

```
[[9.022727272727273, '06'], [11.46, '17'], [16.796296296296298, '16'], [5.5777777777777775, '09
```

```
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
```

In order to have the highest chance of receiving comments, it is recommended to post around 15:00 because it is the hour with the highest average number of comments with 38.59. The chance of getting more comments increases by 62% between 15:00 and the second highest hour (02:00).

According to the documentation, the time zone is Eastern Standard Time, so 15:00 is equivalent to 3:00 p.m. EST.

1.9 Conclusion

In this project, we explored ask posts and show posts from Hacker News to determine which type of post and what time receive the most comments on average. We also looked at which type of post received the most points on average. From our calculations, in order to maximize the number of comments a post recieves, posts should be `Ask FN` posts and should be created between 3:00 p.m. - 4:00 p.m. EST. `Show FN` had a higher average number of points, so in order to get more points, `Show FN` is the suggested post type.

Note: The dataset we analyzed excluded any posts with 0 comments. With that being said, it is more true to say "Of the posts with comments, ask posts received more comments than show posts, and ask posts created between 3:00 - 4:00 p.m. EST received the most comments on average.