Silver and Reddit's Wallstreetbets

Overview

Research Question:

How are bots on r/WallStreetBets influencing posts about silver stock?

Context:

On February 1, 2021, silver had an 11% price increase, the largest surge in over a decade. Redditors on the subreddit r/WallStreetBets claim to be uninvolved, but it remains unclear if bot activity is influencing these posts. The subreddit r/WallStreetBets has over 9 million members.

Motivation:

We aim to identify the extent to which misinformation and bots affect r/WallStreetBets, as well as identifying other subreddits that have the potential to generate a GME-level stock upset.

Intial Data Load

In the first phase of our project, we scraped r/WallStreetBets for posts containing the keyword "silver". The following code was used to pull data from reddit using the API.

We began by loading data using pushshift.

In [23]:
```
#from google.colab import files
#uploaded = files.upload()
import pandas as pd
final_df = pd.read_csv('updated_final_df.csv')
```

In [24]:
```
from urllib.request import urlretrieve


urlretrieve('https://files.pushshift.io/reddit/comments/RC_2006-02.bz2', fil

import bz2, json
```

Print out some sample information from the reddit file

In [25]:
```
#We processed the data by using a push shift. <br>
#This allowed us to gather the exact data that we wanted which included the
#'url','author','title','subreddit','id','num_comments', 'score','upvote_rat
#All these features were collected from submissions from the subreddit with


counter =0
with bz2.BZ2File('./example_reddit_comments.bz2', "r") as fp:
    for line in fp:
        counter = counter +1
        if counter < 5:
            job = json.loads(line)
            print( "subreddit", job['subreddit'])
            print( "author", job['author'])
            print( "upvotes", job['ups'])
            print( "score", job['score'])
            print
        else:
            break
```

In [26]:
```
#!pip install psaw
```

In [27]:
```
from psaw import PushshiftAPI

api = PushshiftAPI()

# Define some keys for submission attribtues you care about -- these are sim
filter_keys = ['url','author','title','subreddit','id','num_comments',
               'score','upvote_ratio','domain','selftext', 'gilded']
```

Below we create the search and filter by the subreddit and the key term which we want to look at, in this case it is silver.

In [29]:
```
#https://pushshift.io/api-parameters/

start = int(datetime(2021, 1, 20).timestamp())

#can add a before if we want to
#before = int(datetime(2021, 1, 20).timestamp())


search = api.search_submissions(after=start,
                                  subreddit='wallstreetbets',
                                  filter=filter_keys,
                                  q = 'silver',
                                  sort='asc',
                                  #score > 100,
                                  limit=50000)
```

Create a list of the necessary information from pushshift

In [30]:
```
# Storage for the results
all_subs = []

# Loop through the search results to actually get data
for i,sub in enumerate(search):

    # Add each result's dictionary (the .d_ attribute) to the all_subs
    all_subs.append(sub.d_)

    # Print out status updates every 10,000 submissions
    if i % 10000 == 0:

        # The current time so you know how long in between updates
        time_now = datetime.now().time().replace(microsecond=0)

        # The date of the submission to give you an idea of how far along yo
        record_date = datetime.utcfromtimestamp(sub.d_['created']).date()

        # Print it out
        print("{0:,} for {1} received at {2}".format(i,record_date,time_now)
```

In [31]:
```
import pandas as pd
#all_subs = pd.read_csv('final_df.csv')
subs_df = pd.read_csv('updated_final_df.csv')
#final_df = subs_df
# len(subs_df)
```

The JSON of the pushshift response is a dictionary of every post

In [32]:
```
all_subs[0]

OUTPUT

{'author': 'slackrooster',
 'created': 1611130062.0,
 'created_utc': 1611130062,
 'domain': 'self.wallstreetbets',
 'id': 'l14tq9',
 'num_comments': 0,
 'score': 1,
 'selftext': '[removed]',
 'subreddit': 'wallstreetbets',
 'title': "Silver lining of Shitron's $GME BS for tomorrow's trading day",
 'upvote_ratio': 1.0,
 'url': 'https://www.reddit.com/r/wallstreetbets/comments/l14tq9/silver_lini
```

What we want to acheieve here is create a date column that is easier to read instead of a timestamp.

We are then saving our DF as a CSV. We have 8,586 total data entries from r/WallStreetBets. This way, we won't have to go through the time consuming process of pulling from Pushshift, we can just load the CSV that we already have saved.

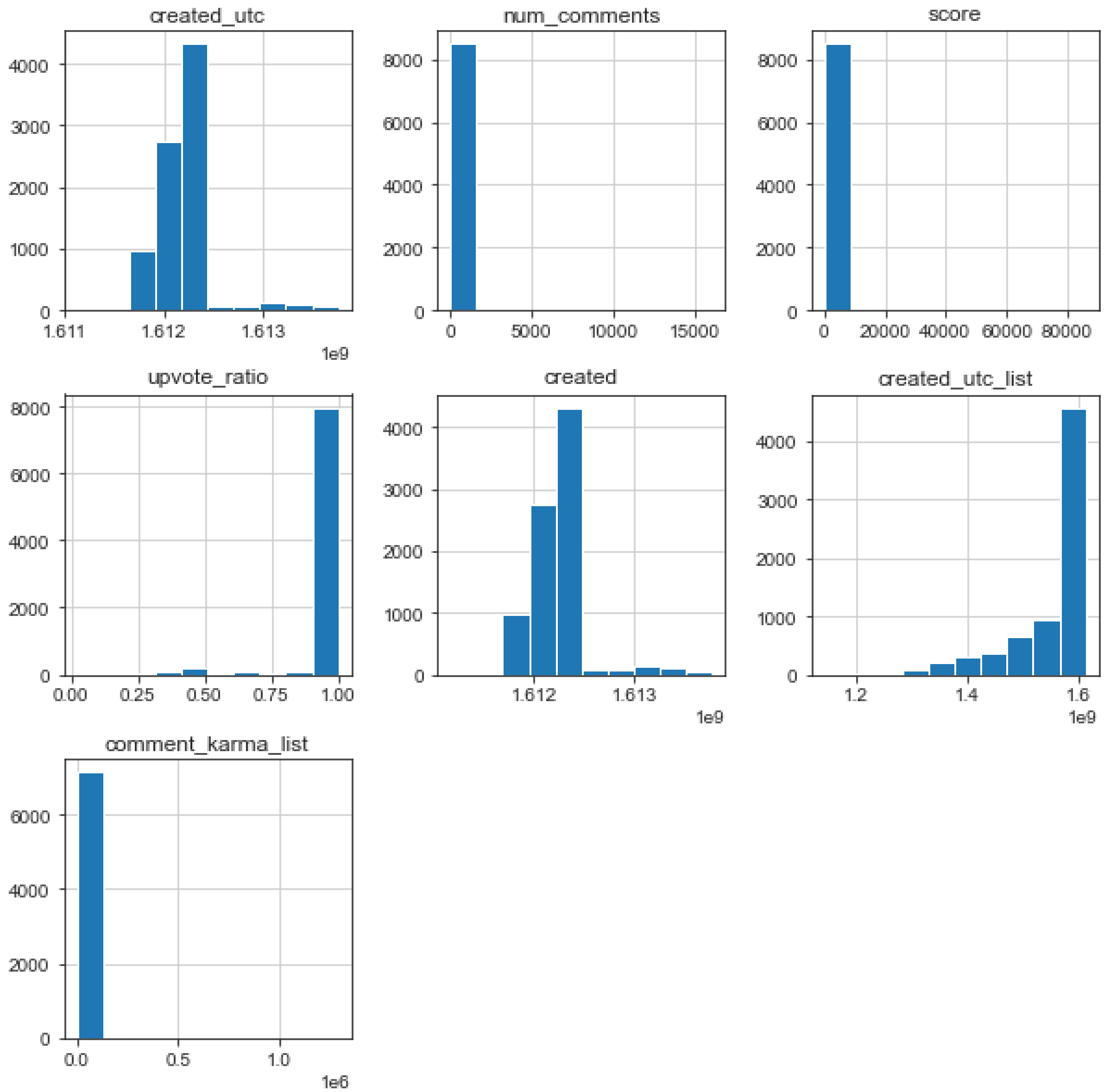
In [33]:
```
subs_df = pd.DataFrame(all_subs)
print('{:,}'.format(len(all_subs)))

subs_df['timestamp'] = subs_df['created'].apply(datetime.utcfromtimestamp)
subs_df['date'] = subs_df['timestamp'].apply(lambda x:x.date())
subs_df.to_csv('wallstreet_push_shift.csv',encoding='utf8',index=False)

subs_df = pd.read_csv('wallstreet_push_shift.csv')
from datetime import datetime
subs_df['timestamp'] = subs_df['created'].apply(datetime.utcfromtimestamp)
subs_df['date'] = subs_df['timestamp'].apply(lambda x:x.date())

subs_df.head()
```

Exploratory Analysis

Get a general idea of the shape of the data

In [34]:
```
final_df = pd.read_csv('final_df.csv')
final_df.hist(figsize=(10,10))

```

Out[34]:
```
array([[<AxesSubplot:title={'center':'created_utc'}>,
        <AxesSubplot:title={'center':'num_comments'}>,
        <AxesSubplot:title={'center':'score'}>],
       [<AxesSubplot:title={'center':'upvote_ratio'}>,
        <AxesSubplot:title={'center':'created'}>,
        <AxesSubplot:title={'center':'created_utc_list'}>],
       [<AxesSubplot:title={'center':'comment_karma_list'}>,
        <AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```

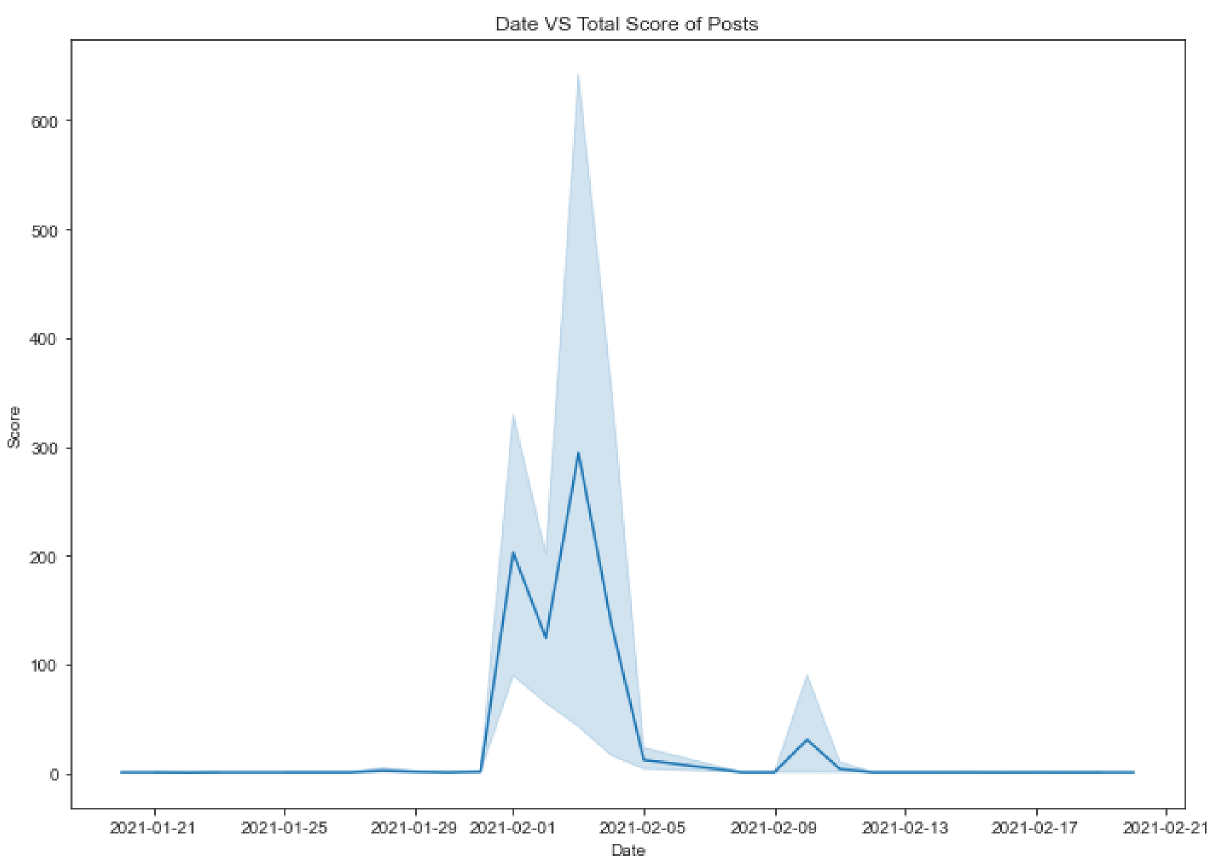


We created 8 graphs to begin with to initially visualize our data. This gave us a good starting point to illustrate activity on r/WallStreetBets during the last week of January 2021.

Next, we used the csv to create a visualization. Figure 1 shows us the date and scores of subreddit data over time which helps identify the uptick in activity.

In [36]:

```
import seaborn as sns

from datetime import datetime
import matplotlib.pyplot as plt

# csv_subs_df = pd.read_csv('wallstreet_push_shift.csv')
final_df['timestamp'] = final_df['created'].apply(datetime.utcfromtimestamp)
final_df['date'] = final_df['timestamp'].apply(lambda x:x.date())

# #for stretching out the plot
# #https://stackoverflow.com/questions/31594549/how-do-i-change-the-figure-s

sns.set_style('ticks')
fig, ax = plt.subplots()
# the size of A4 paper
fig.set_size_inches(11.7, 8.27)

sns.lineplot(x='date',y='score',data=final_df)
plt.xlabel("Date")
plt.ylabel("Score")
plt.title("Date VS Total Score of Posts")
plt.show()
```

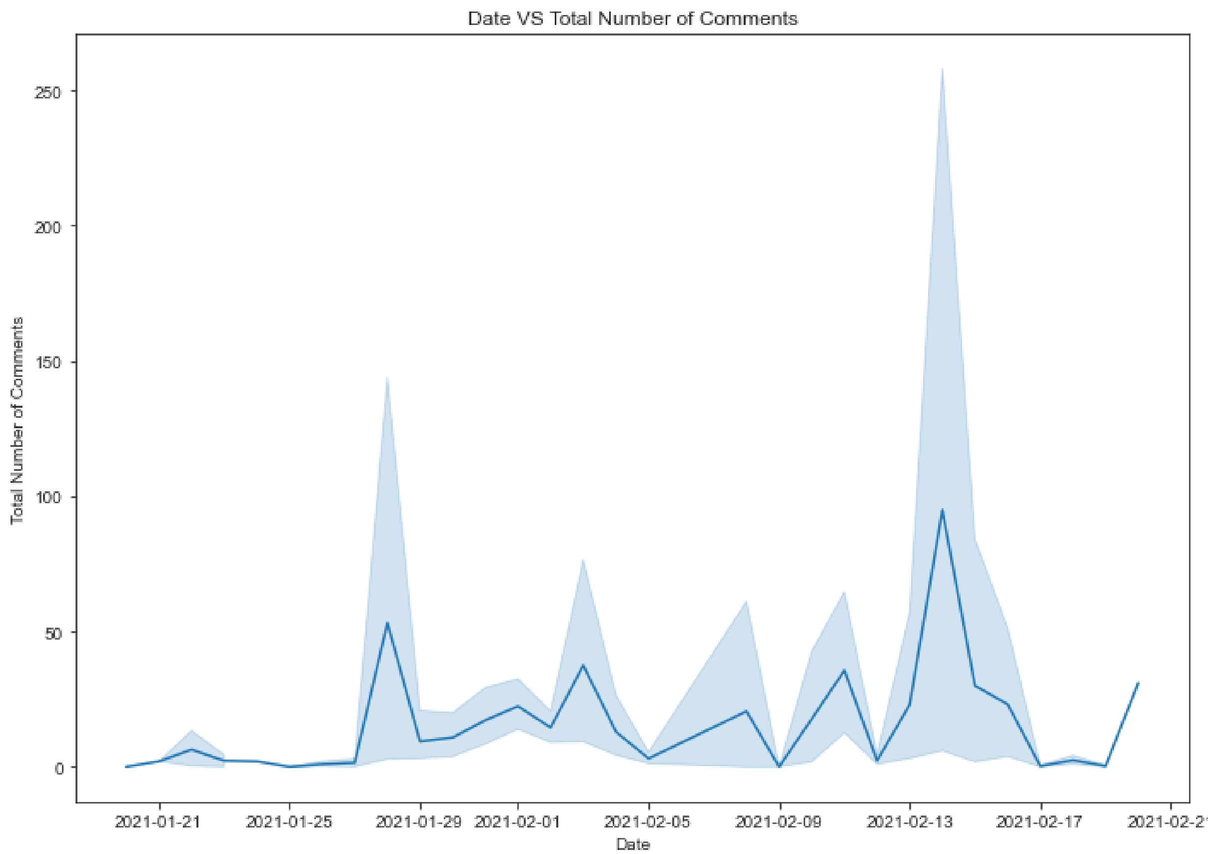


Based on the visualization, February 4th had the highest scoring posts, with another small peak on February 1st. This gives us some information, but score only reflects the ratio of upvoted and downvoted posts. We also want to learn about which accounts are doing the majority of upvotes.

Next, we chose to illustrate the number of comments on r/WallStreetBets over time.

In another visualization what we did was look at the number of comments based on dates over time and we chose to look at the 3 week span from about the January 21st to February 17th. This showed us where there were lots of comments.

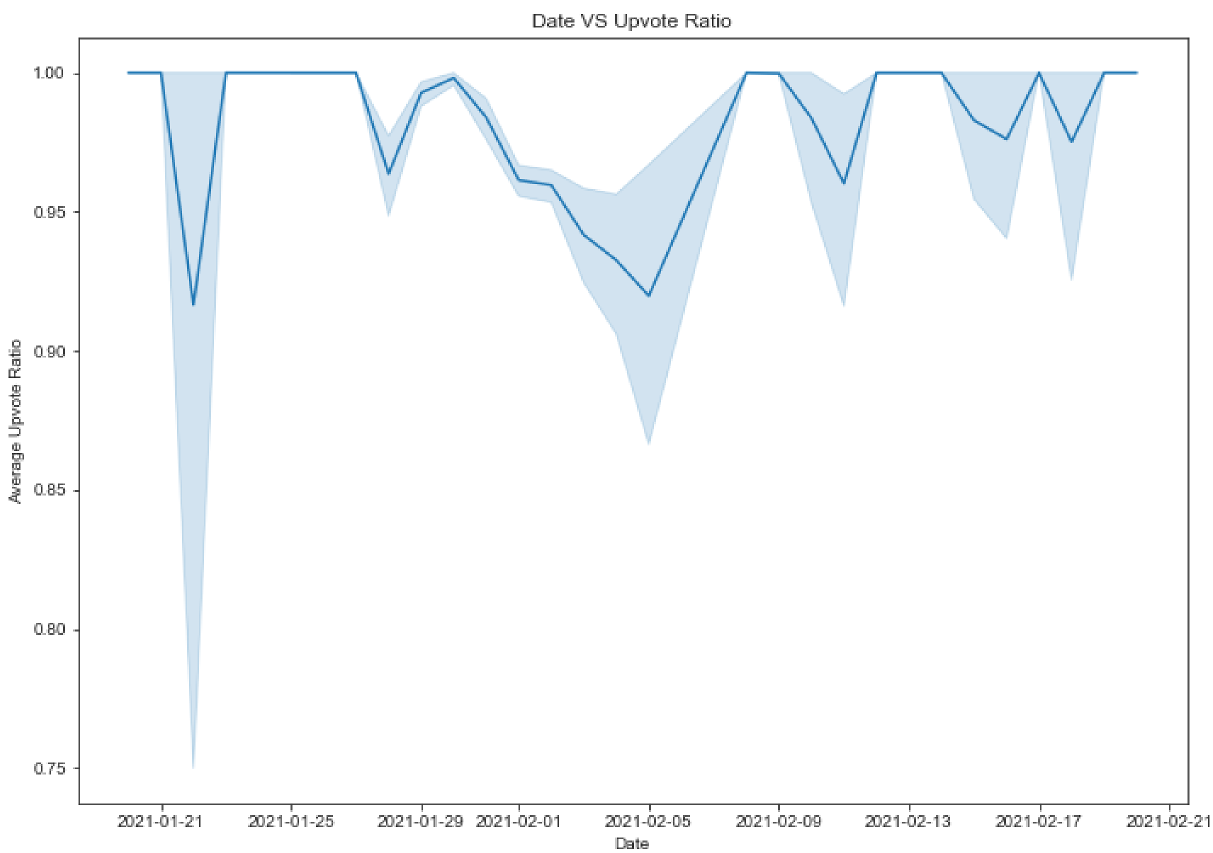
In [37]:
```
sns.set_style('ticks')
fig, ax = plt.subplots()
# the size of A4 paper
fig.set_size_inches(11.7, 8.27)

sns.lineplot(x='date',y='num_comments',data=final_df)
plt.xlabel("Date")
plt.ylabel("Total Number of Comments")
plt.title("Date VS Total Number of Comments")
plt.show()
```



The peaks on January 27 and February 14 were most notable in this visualization. Silver stock reached a relative low point on January 27, prompting many users to buy stock to force a short squeeze (Silver 2021).

Another visualization we have is one which shows us the upvote to downvote ratio over time.

In [38]:
```
import seaborn as sns

from datetime import datetime
import matplotlib.pyplot as plt

#for stretching out the plot
#https://stackoverflow.com/questions/31594549/how-do-i-change-the-figure-siz

sns.set_style('ticks')
fig, ax = plt.subplots()
# the size of A4 paper
fig.set_size_inches(11.7, 8.27)

sns.lineplot(x='date',y='upvote_ratio',data=final_df)

plt.xlabel("Date")
plt.ylabel("Average Upvote Ratio")
plt.title("Date VS Upvote Ratio")
plt.show()
```

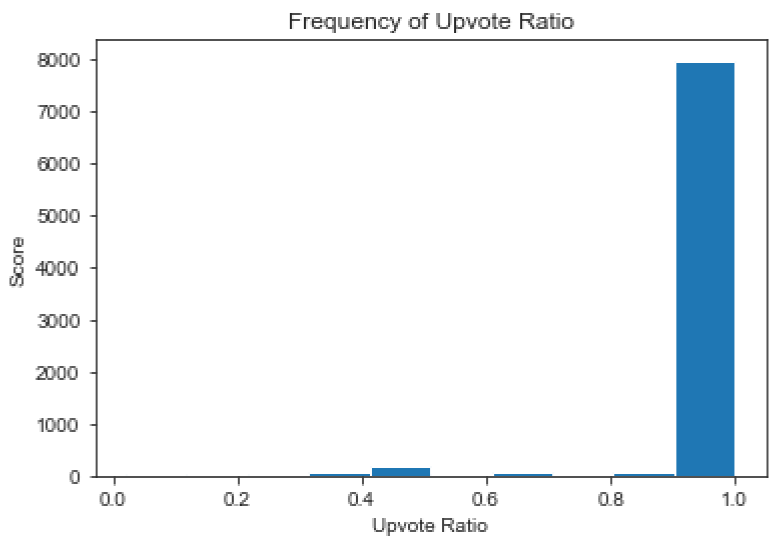


What we saw from the above graph showed us where there were some highly downvoted posts. However, the biggest peaks were on January 22 and February 6. This visualization did not give us as much information as we were hoping for, as there were no notable price changes in Silver stock in the 24 hours around these peaks.

We then chose to graph the distribution of posts with different scores. Our next visualiztion puts into perspective that there were lots of posts which had close to a 1.0 ratio of upvoted posts.

In [39]:

```
#upvote ratio distruibtion

subs_df.upvote_ratio.plot.hist()
plt.xlabel("Upvote Ratio")
plt.ylabel("Score")
plt.title("Frequency of Upvote Ratio")
plt.show()
```



Additional Data Cleaning and Sentiment Analysis

Add column showing if the selftext has been removed

In [40]:

```
is_self_text_removed = []

for post in final_df['selftext']:
    if post == "[removed]":
        is_self_text_removed.append('yes')
    else:
        is_self_text_removed.append('no')

final_df['is_self_text_removed'] = is_self_text_removed
final_df
```

Use textblob to add a column that calculates the polarity of the post title

In [41]:
```
from textblob import TextBlob

#blob = TextBlob('testing this is great')


blob_sent_polarity = []
for title in final_df['title']:
    blob = TextBlob(title)
    blob_sent_polarity.append(blob.polarity)

final_df['blob_sent_polarity'] = blob_sent_polarity

final_df.head()
```

Add a column tokenizing the title by word

In [42]:
```
import nltk
nltk.download('punkt')
from nltk.tokenize import sent_tokenize, word_tokenize#, #WordPunctTokenizer


final_df['word_tokenize_title'] = final_df['title'].apply(word_tokenize)
final_df
```

Import the empath library to run sentiment analysis on tokenized text.

In [43]:
```
#!pip install empath
from empath import Empath

lexicon = Empath()

empath_list = []
empath_list_title = []
for title in final_df['word_tokeinze_title']:
    categ = lexicon.analyze(title, normalize = True)
    for key, value in categ.items():
        if value != 0:
            empath_list_title.append(key)
    empath_list.append(empath_list_title)
    empath_list_title = []


final_df['empath_categories'] = empath_list

final_df
```

Remove stopwords

In [44]:
```
# import nltk
# #nltk.download('stopwords')
# #!python -m nltk.downloader stopwords
from nltk.corpus import stopwords


from nltk.tokenize import sent_tokenize, word_tokenize, WordPunctTokenizer,
from nltk.tokenize.treebank import TreebankWordDetokenizer

stopeng = set(stopwords.words('english'))

stop_list =[]
for word in final_df['title']:
    tokens = word_tokenize( word.lower() )
    tokens_nostop = [w for w in tokens if w not in stopeng]
    tokens_nostop_updated = TreebankWordDetokenizer().detokenize(tokens_nost
    stop_list.append(tokens_nostop_updated)

stop_list

final_df['title_tokenized'] = stop_list
```

Get the 20 most used words. Buy, squeeze and short seem to be some of the most popular words.

In [46]:
```
final_df = pd.read_csv('updated_final_df.csv')

from collections import Counter
from collections import defaultdict

def top20(thislist):
    # First make a string out of the entire list
    BIGstr = " ".join(thislist)
    wordlist = BIGstr.split(" ")
    wordcount = Counter(wordlist)
    return(wordcount.most_common(20))

print(top20(final_df['title_tokenized']))
```

```
[('silver', 7093), ('.', 2284), ('buy', 1169), ('’', 971), ('gme', 719), ('sque
eze', 697), ('short', 557), ('buying', 447), ('-', 425), ('physical', 406), ('s
ilver?', 380), ('silver,', 335), ('slv', 318), ('gold', 304), ('hold', 298),
('like', 296), ('amc', 296), ('news', 276), ('reddit', 275), ('silver!', 262)]
```

Look at the amount of posts with self text removed vs not removed

Visualizations and Analysis

Post Deletion

T test

Null Hypothesis: deleted post would have same upvote ratio as a undeleted post.

Alternitive Hypothesis: Deleted post and Undeleted post would have different upvote ratio.

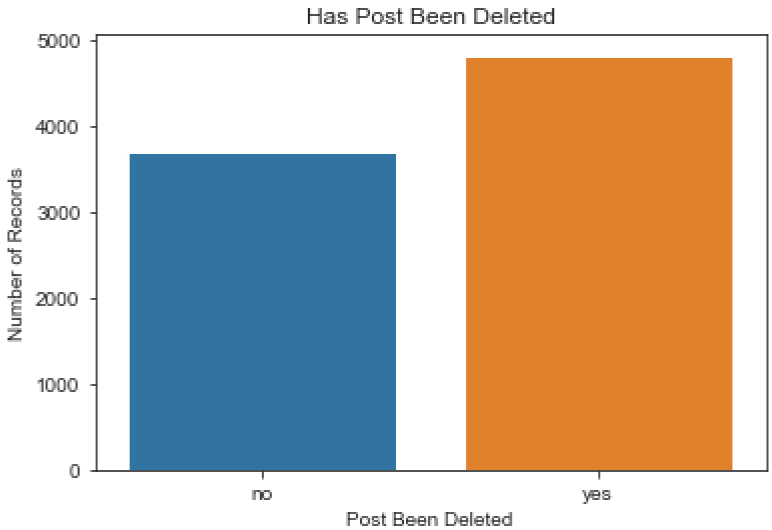
Based on the test, there was a signficant difference between the samples.

In [47]:
```
#Import dataset
df = pd.read_csv('updated_final_df.csv')

#Array of delete post
options = ['[removed]', '[deleted]']

#create deleted only dataset
deleted_df = df.loc[df['selftext'].isin(options)]

#create non-deleted only dataset
notdeleted_df = df.loc[~df['selftext'].isin(options)]

#Sample 1000 rows from each dataset
deleted_df_sample = deleted_df.sample(n = 1000)
notdeleted_df_sample = notdeleted_df.sample(n = 1000)

#T test
from scipy import stats as st
#to each upvote_ratio to array
a = deleted_df_sample[['upvote_ratio']].to_numpy()
b = notdeleted_df_sample[['upvote_ratio']].to_numpy()
st.ttest_ind(a=a, b=b, equal_var=True)
```

Out[47]:
```
Ttest_indResult(statistic=array([2.46741533]), pvalue=array([0.01369252]))
```

Of our dataset, over 60% of posts ended up being deleted. This shows that r/WallStreetBets moderators didn't want anyone to see most of the posts about silver, implying these posts contained misinformation.

In [48]:
```
#import seaborn as sns
final_df = pd.read_csv('updated_final_df.csv')

sns.countplot(x='is_self_text_removed', data = final_df)
plt.xlabel("Post Been Deleted")
plt.ylabel("Number of Records")
plt.title("Has Post Been Deleted")
plt.show()
```



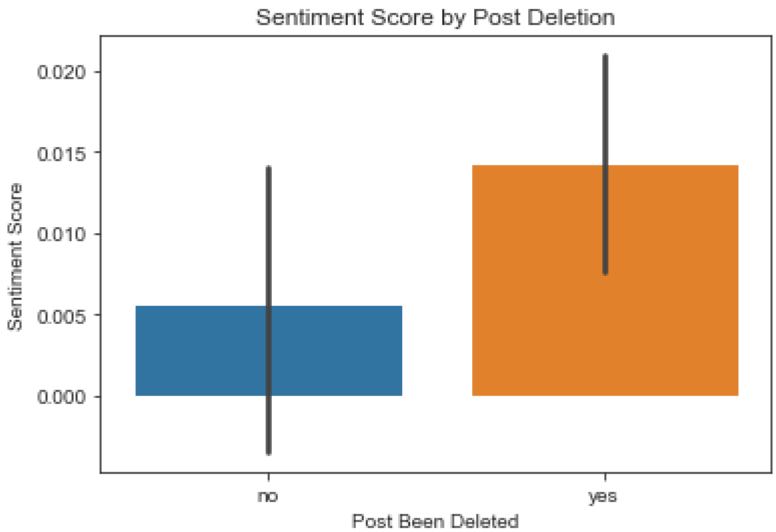
What the next visualization does is shows the comparison between the number of comments and the karma that the commments get on a given post.

In [49]:
```


final_df = pd.read_csv('updated_final_df.csv')
temporary_df = final_df.copy()
temporary_df.rename(columns= {'is_self_text_removed': 'Is Post Deleted'}, in

fig, ax = plt.subplots()
# the size of A4 paper

fig.set_size_inches(8.7, 8.27)

#sns.boxplot(x= "Is Post Deleted", y= "comment_karma_list", data= temporary_
sns.scatterplot(x='num_comments', y='comment_karma_list', data= temporary_df
plt.xlabel("Number of Comments")
plt.ylabel("Comment Karma")
plt.title("Number of Comments vs Karma")
plt.show()
```

From the results, we can see that there were several deleted posts with a lot of comment karma.

Look at the sentiment between removed posts and not removed posts. Posts which ended up being deleted had higher sentiment than posts which did not. This may be because people advocating for silver used more positive words to describe silver.

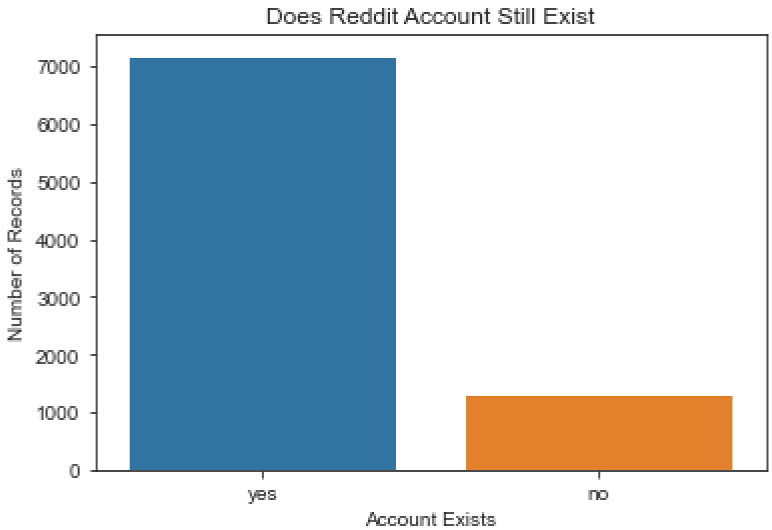
In [50]:
```
sns.barplot(x="is_self_text_removed", y="blob_sent_polarity", data=final_df)
plt.xlabel("Post Been Deleted")
plt.ylabel("Sentiment Score")
plt.title("Sentiment Score by Post Deletion")
plt.show()

```



Account Deletion

Look to see how many reddit accounts have been removed since the orginal post. Over 12% of accounts have been deleted since their orginal post relating to silver. Although we don't currently have a baseline to compare this to, this number seems very high.

In [51]:
```
import seaborn as sns

sns.countplot(x='account_exists', data = final_df)

plt.xlabel("Account Exists")
plt.ylabel("Number of Records")
plt.title("Does Reddit Account Still Exist")
plt.show()
```

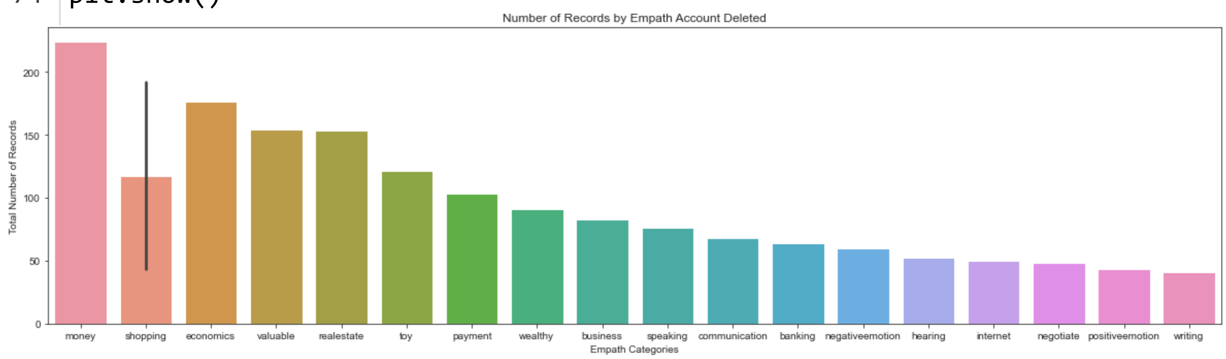


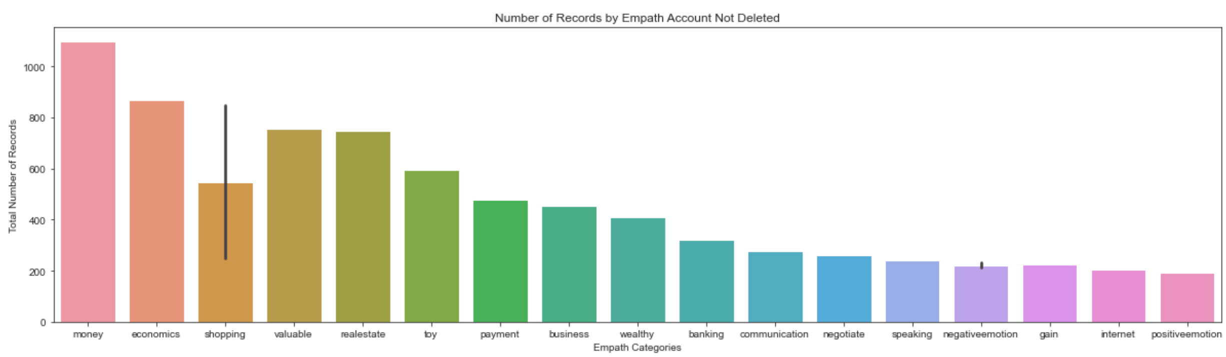
Posts which have been deleted have slighly lower average comment karma than posts which have not been deleted.

Look at the 20 most used words for empath categories for account deleted and not deleted. They had very similar results. The account deleted had slighly more occurences of economic related posts than the accounts which have not been deleted.

In [53]:
```
account_no_exist = final_df[final_df.account_exists == 'no']
account_yes_exist = final_df[final_df.account_exists == 'yes']

#split account extists vs account doesn't exist


word_cloud_list_key = []
word_cloud_list_value = []
word_cloud_thing = top20(account_no_exist['empath_categories'])
for thing in word_cloud_thing:
  the_key = thing[0]
  import re
  the_key = re.sub(r'[^A-Za-z]', '', the_key)
  word_cloud_list_key.append(the_key)
  the_value = thing[1]
  #the_value = re.sub(r'[^A-Za-z]', '', the_value)
  word_cloud_list_value.append(the_value)

word_cloud_list_key[0] = "blank"
word_cloud_list_key

from pandas import DataFrame

word_cloud_df = DataFrame (word_cloud_list_key,columns=['Key'])
word_cloud_df['value'] = word_cloud_list_value

word_cloud_df = word_cloud_df.drop(labels= 0, axis=0)

word_cloud_df

sns.set_style('ticks')
fig, ax = plt.subplots()
# the size of A4 paper
fig.set_size_inches(20.4, 5.27)
sns.barplot(x="Key", y = "value", data = word_cloud_df)
plt.xlabel("Empath Categories")
plt.ylabel("Total Number of Records")
plt.title("Number of Records by Empath Account Deleted")
plt.show()


word_cloud_list_key = []
word_cloud_list_value = []
word_cloud_thing = top20(account_yes_exist['empath_categories'])
for thing in word_cloud_thing:
  the_key = thing[0]
  import re
  the_key = re.sub(r'[^A-Za-z]', '', the_key)
  word_cloud_list_key.append(the_key)
  the_value = thing[1]
  #the_value = re.sub(r'[^A-Za-z]', '', the_value)
  word_cloud_list_value.append(the_value)

word_cloud_list_key[0] = "blank"
word_cloud_list_key

```

```
from pandas import DataFrame

word_cloud_df = DataFrame (word_cloud_list_key,columns=['Key'])
word_cloud_df['value'] = word_cloud_list_value

word_cloud_df = word_cloud_df.drop(labels= 0, axis=0)

word_cloud_df

sns.set_style('ticks')
fig, ax = plt.subplots()
# the size of A4 paper
fig.set_size_inches(20.4, 5.27)
sns.barplot(x="Key", y = "value", data = word_cloud_df)
plt.xlabel("Empath Categories")
plt.ylabel("Total Number of Records")
plt.title("Number of Records by Empath Account Not Deleted")
plt.show()
```





Account Creation

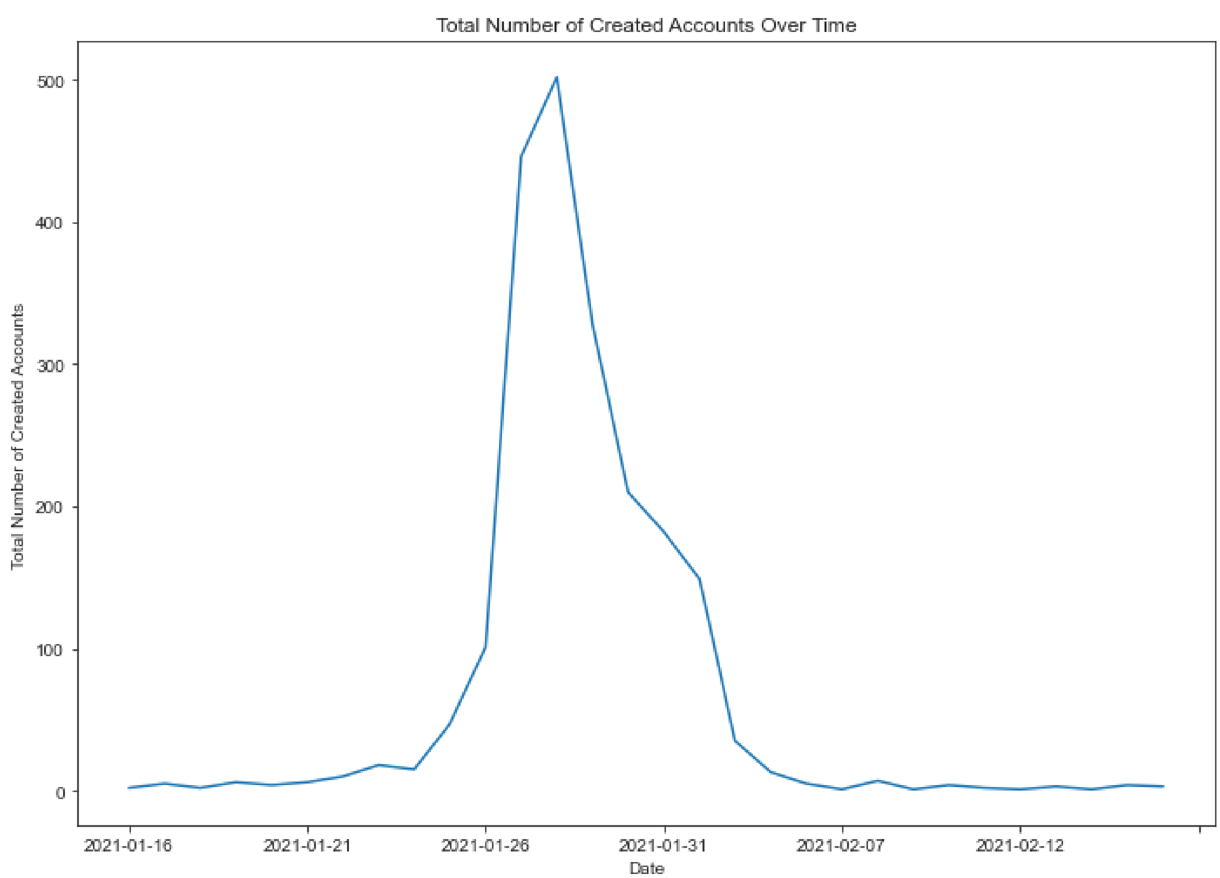
Look at account creation over time. The amount of created accounts showed a similar pattern to the total number of comments peaking around the end of January. Over 1,000 accounts were created and one of their first posts had to do with silver.

In [54]:

```
import datetime
import time

created_df = final_df.copy()
#created_df = final_df[['created_utc_list']]
created_df = created_df.dropna()
created_df['created_pst'] = created_df['created_utc_list'].apply(lambda t: t
filter_created_df = created_df[created_df.created_pst > "2021-01-15"]

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
s = filter_created_df['created_pst'].value_counts().sort_index()
#print(s)

#plt.tick_params(axis='x', which='major', labelsize=3)
fig, ax = plt.subplots()
# the size of A4 paper
fig.set_size_inches(11.7, 8.27)
plt.xlabel("Date")
plt.ylabel("Total Number of Created Accounts")
plt.title("Total Number of Created Accounts Over Time")
s.plot()
```

Out[54]:

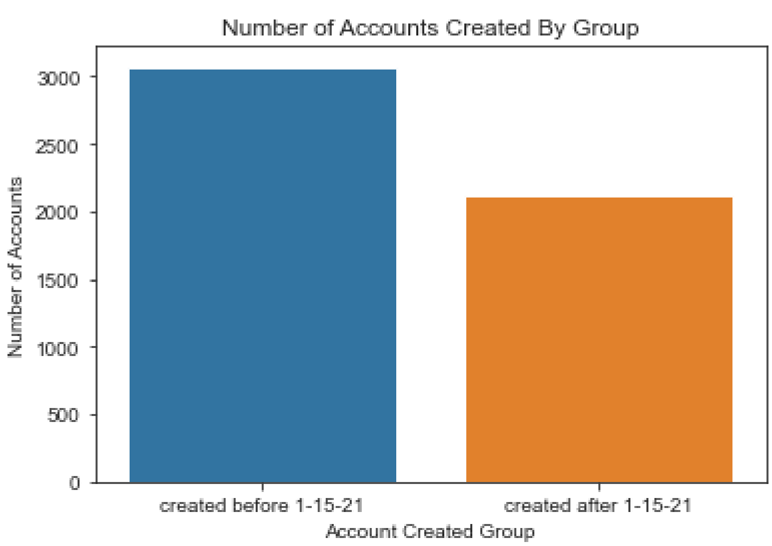
```
<AxesSubplot:title={'center':'Total Number of Created Accounts Over Time'}, xla
bel='Date', ylabel='Total Number of Created Accounts'>
```



Next, we split our data into accounts that were created before and after January 15th. We saw that nearly 40% of accounts which posted about silver were posted after January 15, 2021.

In [55]:

```
#downvoted posts

#subs_df.dtypes

#as_int(sub_df['upvote_ratio'])

created_df_list = []

for x in created_df['account_exists']:
  created_df_list.append('created before 1-15-21')

created_df['date_filter'] = created_df_list


created_after_1_15_21 = (created_df.created_pst > "2021-01-15")

created_df.loc[created_after_1_15_21,'date_filter'] = 'created after 1-15-21

sns.countplot(x='date_filter', data= created_df)
plt.xlabel("Account Created Group")
plt.ylabel("Number of Accounts")
plt.title("Number of Accounts Created By Group")
plt.show()
```



Summary of Key Findings

1. Many posts have been deleted by moderators (60%)

2. A high percentage of reddit accounts have been deleted
3. There are some notable differences in sentiment of deleted posts and non deleted posts
4. There was a spike in activity on January 27-28, concurrently with the relative drop in price in silver stock
    - This activity included mass downvoting of posts, surges in creating and deleting accounts, and comments

Overall Summary

Overall when we were looking at Reddit data on r/WallStreetBets we started with the raw data and transformed that into data which we would be able to analyze. We layed out our definitions for bots this allowed us to look at specific characteristics in which we could determine if there was significant bot activity which could influence the prices of silver by influencing users to pursue it the same way that they bought up Gamestop stocks.

What our visualizations have showed us is that there has been a significant amount of accounts which were created just before the price of silver went up and also deleted days later, to us this proves that there were bots influencing users of this reddit and causing the price of silver to increase more than it had in many years.