Reddit Posts Analysis

Introduction

Exploratory data analysis using Pandas for a given public sample of random Reddit posts.

Analysis:

What are the most popular reddits? Which topics are viral? Which posts have been removed and why? What % removed reddits are deleted by moderatos? Who are the most popular authors? Who are the biggest spammers at Reddit platform?

In [1]:
```
#Getting all the packages we need:

import numpy as np # linear algebra
import pandas as pd # data processing

import seaborn as sns #statist graph package
import matplotlib.pyplot as plt #plot package
import pandasql as ps #sql package
import wordcloud #will use for the word cloud plot
from wordcloud import WordCloud, STOPWORDS # optional to filter out the stopword

#Optional helpful plot stypes:
plt.style.use('bmh') #setting up 'bmh' as "Bayesian Methods for Hackers" style s
#plt.style.use('ggplot') #R ggplot stype
#print(plt.style.available) #pick another style
```

Accessing Reddit dataset:

In [2]:
```
df = pd.read_csv('r_dataisbeautiful_posts.csv', low_memory=False)
```

In [3]:
```
df
```

Out[3]:

	id	title	score	author	author_flair_text	remove
0	ll1p9h	Wordcloud of trending video titles on YouTube ...	1	OmarZiada	OC: 1	
1	ll1o4h	Wordcloud of trending videos on YouTube in the...	1	OmarZiada	OC: 1	mode
2	ll15gx	Immunization in India. Source: https://niti.go...	1	Professional_Napper_	NaN	mode
3	ll0iup	How to quickly estimate the impact of players ...	1	Viziball	NaN	automod_fil
4	ll0g9a	How to quickly estimate the impact of players ...	1	Viziball	NaN	mode
...	...	...	...	...	...	
190848	pqbdl	Infosthetics seems like it belongs here.	15	magiclamp	NaN	
190849	pqav2	Time lapse of every nuclear detonation from 19...	9	th3sousa	NaN	
190850	pq922	Wavii.	13	ddshroom	NaN	
190851	ppx09	An interactive representation of Pres. Obamas ...	21	zanycaswell	NaN	
190852	ppvl7	A map showing the geographical distribution of...	45	zanycaswell	NaN	

190853 rows × 11 columns

Getting a feel of the dataset

In [4]:
```
#Let's run basic dataframe exploratory commands
df.info()
```

```
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 190853 entries, 0 to 190852
Data columns (total 11 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   id                     190853 non-null  object 
 1   title                  190852 non-null  object 
 2   score                  190853 non-null  int64  
 3   author                 190853 non-null  object 
 4   author_flair_text      28845 non-null   object 
 5   removed_by             20744 non-null   object 
 6   total_awards_received  65146 non-null   float64
 7   created_utc            190853 non-null  int64  
 8   full_link              190853 non-null  object 
 9   num_comments           190853 non-null  int64  
 10  over_18                190853 non-null  bool   
dtypes: bool(1), float64(1), int64(3), object(6)
memory usage: 14.7+ MB
```

Out[4]:

	score	total_awards_received	created_utc	num_comments
count	190853.000000	65146.000000	1.908530e+05	190853.000000
mean	176.016159	0.013109	1.512494e+09	27.604732
std	1951.936524	0.589425	6.822624e+07	213.236378
min	0.000000	0.000000	1.329263e+09	0.000000
25%	1.000000	0.000000	1.463862e+09	1.000000
50%	1.000000	0.000000	1.518662e+09	2.000000
75%	4.000000	0.000000	1.576576e+09	5.000000
max	116226.000000	93.000000	1.613474e+09	18801.000000

In [5]:
```
#is null values:
df.isnull().sum().sort_values(ascending = False)
```

Out[5]:
```
removed_by               170109
author_flair_text        162008
total_awards_received    125707
title                         1
id                            0
score                         0
author                        0
created_utc                   0
full_link                     0
num_comments                  0
over_18                       0
dtype: int64
```

Removed reddits deep dive

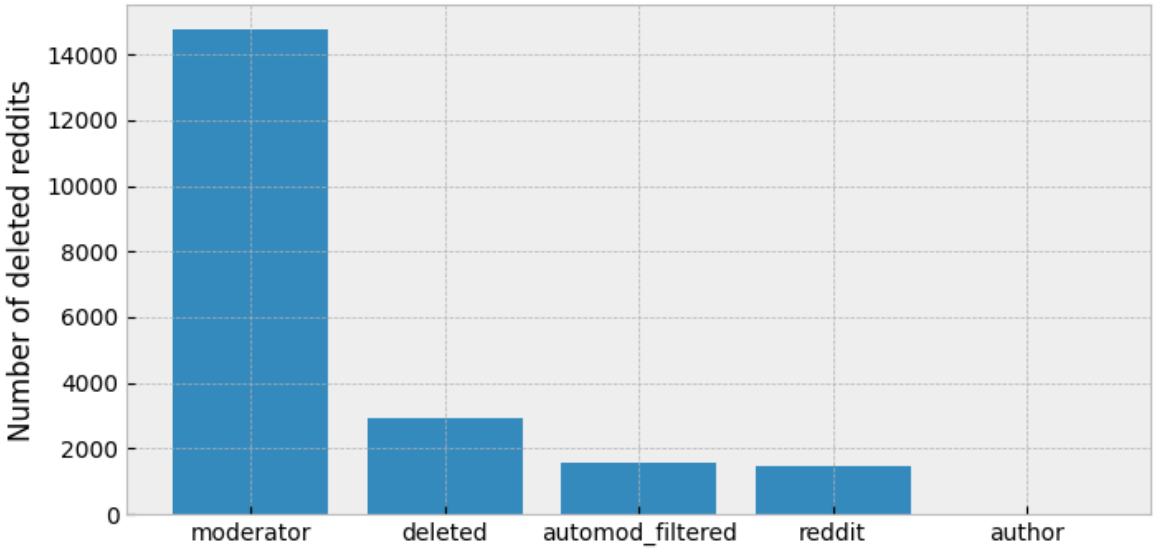
In [6]:
```
#Let's see who and why removes posts:
q1 = """SELECT removed_by, count(distinct id)as number_of_removed_posts
FROM df
```

```
where removed_by is not null
group by removed_by
order by 2 desc """

grouped_df = ps.sqldf(q1, locals())
grouped_df
```

Out[6]:

	removed_by	number_of_removed_posts
0	moderator	14789
1	deleted	2948
2	automod_filtered	1553
3	reddit	1453
4	author	1

In [7]:
```
#Visualizing bar chart based of SQL output:

removed_by = grouped_df['removed_by'].tolist()
number_of_removed_posts = grouped_df['number_of_removed_posts'].tolist()

plt.figure(figsize=(8,4))
plt.ylabel("Number of deleted reddits")
plt.bar(removed_by, number_of_removed_posts)

plt.show()
```



Who are the top 3 users who had the most their posts removed by moderator?

In [8]:
```
q2 = """SELECT author, count(id) as number_of_removed_posts
FROM df
where removed_by = 'moderator'
group by author
order by 2 desc
limit 3"""
print(ps.sqldf(q2, locals()))
```

```
           author  number_of_removed_posts
0    hornedviper9                       71
1  peter_mladenov                       35
2     ad55mul1994                       20
```

Let's find out how many posts with "virus" keyword are removed by moderator.

In [9]:
```
#Step 1: Getting proportion of all removed posts / removed "virus" posts
q3 = """
with Virus as (
SELECT id
FROM df
where removed_by = 'moderator'
and title like '%virus%'
)

SELECT count(v.id) as virus_removed, count(d.id) as all_removed
FROM df d
left join virus v on v.id = d.id
where d.removed_by = 'moderator';"""

removed_moderator_df = ps.sqldf(q3, locals())

#print(type(removed_moderator_df))
print(removed_moderator_df.values)
print(removed_moderator_df.values[0])
```

```
[[ 1056 14789]]
[ 1056 14789]
```

In [10]:
```
#Step 2: getting % virus reddits from all removed posts:

virus_removed_id = removed_moderator_df.values[0][0]
all_removed_id = removed_moderator_df.values[0][1]


print(virus_removed_id/all_removed_id)
```

```
0.07140442220569342
```

From all removed reddits by moderator, 7 % posts contain the "virus" keyword.

The most popular reddits

In [11]:
```
#Top 10 reddits with the most number of comments:

q4 = """SELECT title, num_comments as number_of_comments
FROM df
where title != 'data_irl'
order by 2 desc
limit 10"""
print(ps.sqldf(q4, locals()))
```

	title	number_of_comments
0	The best (& worst) countries for raising a...	12508
1	Police killing rates in G7 members [OC]	10425
2	[oc] How taboo and popular are sexual fetishes...	10396
3	Worst Episode Ever? The Most Commonly Rated Sh...	9217
4	The environmental impact of Beyond Meat and a ...	9120
5	What's getting cut in Trump's budget	8768
6	[OC] Mississippi, the Poorest State in the US	8703
7	[OC] Trump voters are less likely to have a co...	8354
8	[OC] Trending Google Searches by State Between...	8270
9	Almost all men are stronger than almost all wo...	7538

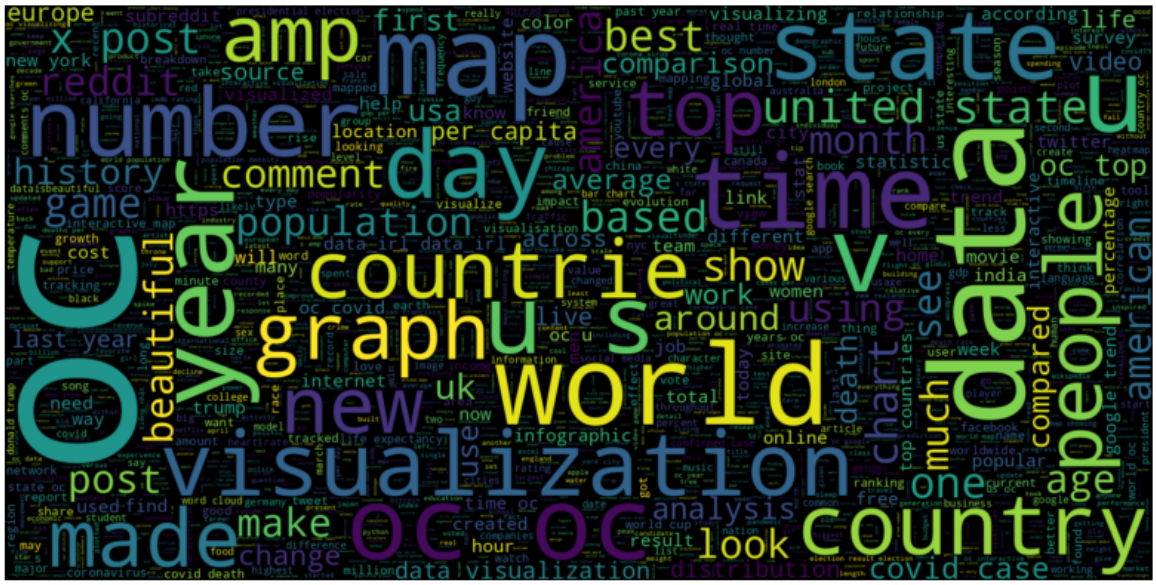
The most common words in reddits:

In [12]:
```
#Let's see the word map of the most commonly used words from reddit titles:
#To build a wordcloud, we have to remove NULL values first:
df["title"] = df["title"].fillna(value="")
```

In [13]:
```
#Now let's add a string value instead to make our Series clean:
word_string=" ".join(df['title'].str.lower())

#word_string
```

In [14]:
```
#And - plotting:

plt.figure(figsize=(10,15))
wc = WordCloud(background_color="black", stopwords = STOPWORDS, max_words=2000,
wc.generate(word_string)

plt.imshow(wc.recolor( colormap= 'viridis' , random_state=17), interpolation="bi
plt.axis('off')
```

Out[14]: (-0.5, 1599.5, 799.5, -0.5)

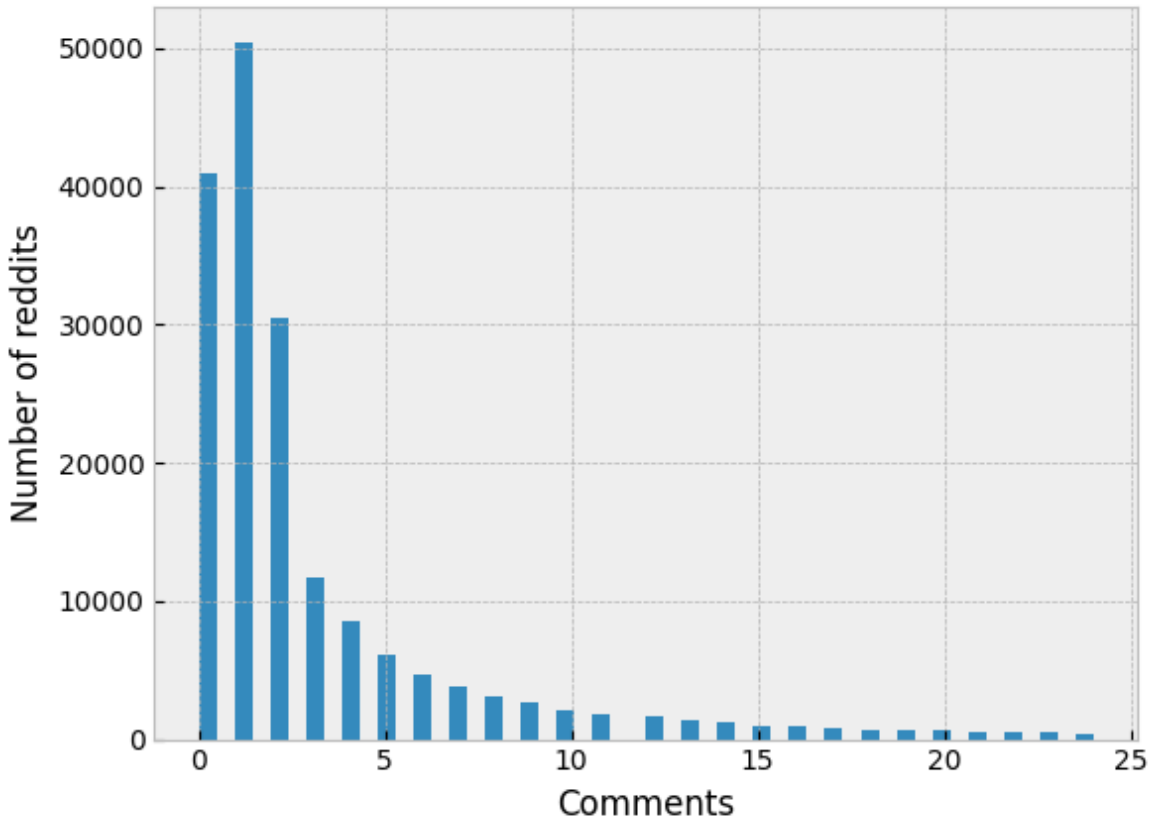


Comments distribution

The average reddit has less than 25 comments. Let's see the comment distribution for those reddits who have <25 comments

In [15]:
```
#Comments distribution plot:

fig, ax = plt.subplots()
_ = sns.distplot(df[df["num_comments"] < 25]["num_comments"], kde=False, rug=Fal
_ = ax.set(xlabel="num_comments", ylabel="id")

plt.ylabel("Number of reddits")
plt.xlabel("Comments")

plt.show()
```

```
C:\Users\Sethu\AppData\Local\Temp\ipykernel_19052\2455252396.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  _ = sns.distplot(df[df["num_comments"] < 25]["num_comments"], kde=False, rug=Fa
lse, hist_kws={'alpha': 1}, ax=ax)
```



As we can see, the most reddits have less than 5 comments.

Correlation between dataset variables

Now let's see how the dataset variables are correlated with each other:

How score and comments are correlated? Do they increase and decrease together (positive correlation)? Does one of them increase when the other decrease and vice versa (negative correlation)? Or are they not correlated? Correlation is represented as a value between -1 and +1 where +1 denotes the highest positive correlation, -1 denotes the highest negative correlation, and 0 denotes that there is no correlation.

Let's see the correlation table between our dataset variables (numerical and boolean variables only)

In [16]:
```
df = df.drop('id', axis=1)
```

In [18]:
```
import pandas as pd

# Step 1: Identify non-numeric columns
non_numeric_columns = []
for column in df.columns:
    if df[column].dtype not in ['int64', 'float64']:
        non_numeric_columns.append(column)

# Step 2: Drop non-numeric columns or convert them to numeric
df = df.drop(non_numeric_columns, axis=1)
df[non_numeric_columns] = df[non_numeric_columns].apply(pd.to_numeric, errors='c

# Step 3: Calculate the correlation
correlation_matrix = df.corr()
```

In [19]:
```
df.corr()
```

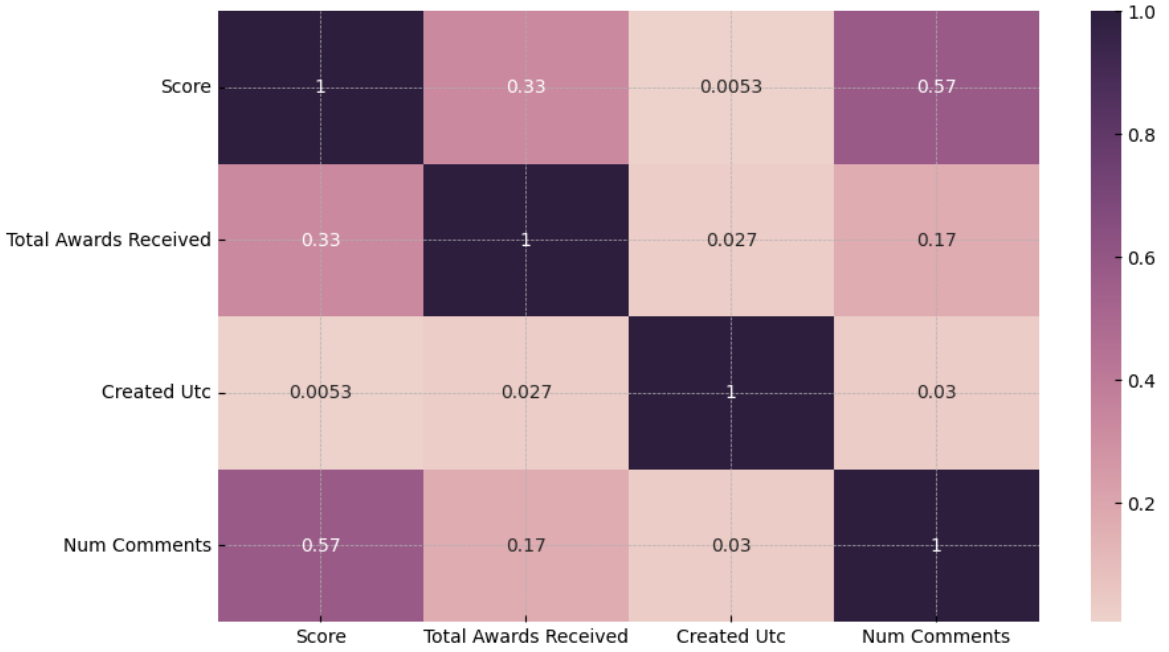
Out[19]:

	score	total_awards_received	created_utc	num_comments
score	1.000000	0.330506	0.005262	0.574893
total_awards_received	0.330506	1.000000	0.027446	0.168387
created_utc	0.005262	0.027446	1.000000	0.029795
num_comments	0.574893	0.168387	0.029795	1.000000

We see that score and number of comments are highly positively correlated with a correlation value of 0.57

Now let's visualize the correlation table above using a heatmap

In [20]:
```
h_labels = [x.replace('_', ' ').title() for x in
            list(df.select_dtypes(include=['number', 'bool']).columns.values)]

fig, ax = plt.subplots(figsize=(10,6))
_ = sns.heatmap(df.corr(), annot=True, xticklabels=h_labels, yticklabels=h_label
```



Score distribution

In [21]:
```
df.score.describe()
```

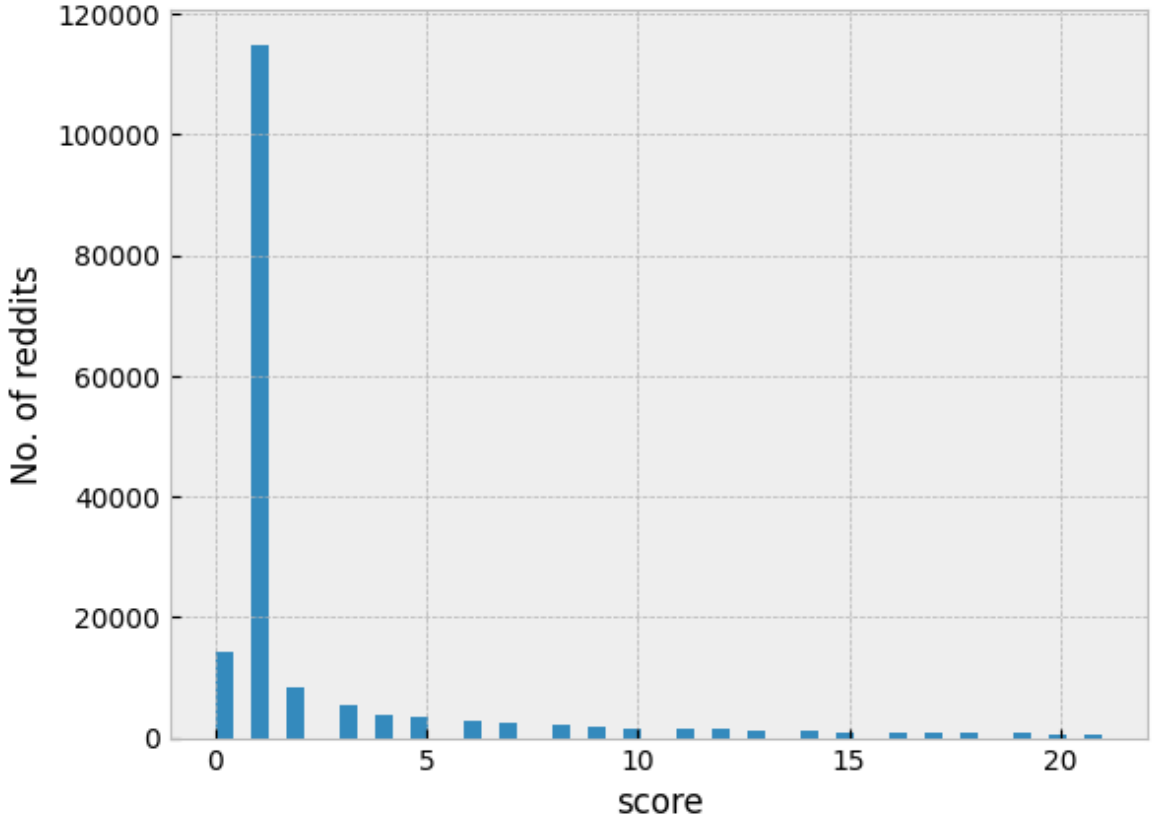
Out[21]:
```
count    190853.000000
mean        176.016159
std        1951.936524
min           0.000000
25%           1.000000
50%           1.000000
75%           4.000000
max      116226.000000
Name: score, dtype: float64
```

In [22]:
```
df.score.median()
```

Out[22]:
```
1.0
```

In [23]:
```
#Score distribution:

fig, ax = plt.subplots()
_ = sns.distplot(df[df["score"] < 22]["score"], kde=False, hist_kws={'alpha': 1}
_ = ax.set(xlabel="score", ylabel="No. of reddits")
```

```
C:\Users\Sethu\AppData\Local\Temp\ipykernel_19052\3022113631.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  _ = sns.distplot(df[df["score"] < 22]["score"], kde=False, hist_kws={'alpha':
1}, ax=ax)
```


120000
100000
80000
60000
40000
20000
0
No. of reddits
0
5
10
15
20
score