Scuola universitaria professionale
della Svizzera italiana
SUPSI

CAll for Regulation support In Social MediA

Exploratory Data Analysis

Amirhossein Shirzad

amir.shirzad@gmail.com

The dataset contains tweets that mention virus and the virusShell vulnerability, involving arbitrary code execution. The data has been gathered through the Twitter API and contains tweets from December 9, 2021 through December 24, 2021, and is available at the following link: https://filesender.switch.ch/filesender2/?s=download&token=e0d419b5-8641-443a-8637-ca45eed8fbcb Its purpose was to study the evolution of the interest in the virus vulnerability.

Dataset Overview

Import libraries

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set_style("whitegrid")
```

Read from CSV and import as dataframe using pandas:

In [2]:
```
df = pd.read_csv('dataset.csv')
```

Check the DataFrame:

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 81034 entries, 0 to 81033
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   status_id            81034 non-null  int64 
 1   status_date          81034 non-null  object
 2   text                 81034 non-null  object
 3   favourite_count      81034 non-null  int64 
 4   retweet_count        81034 non-null  int64 
 5   user_name            81032 non-null  object
 6   screen_name          81034 non-null  object
 7   user_follower_count  81034 non-null  int64 
 8   user_friends_count   81034 non-null  int64 
 9   user_created_date    81034 non-null  object
 10  user_location        63494 non-null  object
 11  source               81034 non-null  object
dtypes: int64(5), object(7)
memory usage: 7.4+ MB
```

Remove status_id column because we already have unique screen_name:

In [4]:
```
df = df.drop(['status_id'], axis=1)
```

Some tweets don't have a location, we can change the user_location column to boolean value to see if a row has a location or not:

In [5]:
```
df['user_location'] = df['user_location'].where(df['user_location'].isnull(), 1).fillna(0).astype(int)
```

In [6]:
```
df['user_location'].value_counts()
```

Out[6]:
```
1    63494
0    17540
Name: user_location, dtype: int64
```

DataFrame.describe summarize the central tendency, dispersion and shape of a dataset's distribution.

In [7]:
```
df.describe()
```

Out[7]:

	favourite_count	retweet_count	user_follower_count	user_friends_count	user_location
count	81034.000000	81034.000000	8.103400e+04	81034.000000	81034.000000
mean	10.163327	2.949083	1.977634e+04	1360.054841	0.783548
std	116.718514	31.370396	4.544552e+05	4092.717109	0.411829
min	0.000000	0.000000	0.000000e+00	0.000000	0.000000
25%	0.000000	0.000000	1.670000e+02	184.000000	1.000000
50%	0.000000	0.000000	6.720000e+02	537.000000	1.000000
75%	2.000000	1.000000	2.883000e+03	1316.000000	1.000000
max	13605.000000	2434.000000	5.568216e+07	280270.000000	1.000000

In [8]:
```
df.head(1)
```

Out[8]:

	status_date	text	favourite_count	retweet_count	user_name	screen_name	user_follower_count	user_friends_count	user_created
0	2021-12-24 05:03:38	ZHNR — Chinese regulator pauses partnership wi...	0	0	Stigmabase | UN	PairsonnalitesU	2511	2409	2010-1 15:

Text length

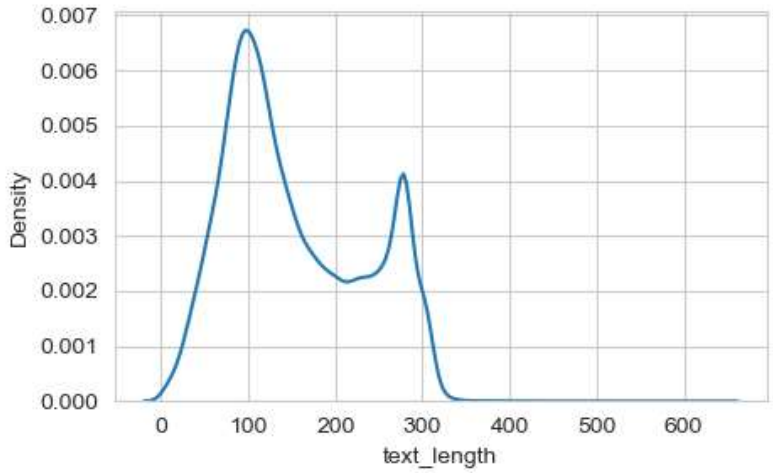
We can also add text length column for tweet texts(Without Cleaning):

In [9]:
```
# Add text length column for tweet texts
df['text_length'] = df['text'].str.len()
# Show Dataframe
df['text_length'].describe()
```

Out[9]:
```
count    81034.000000
mean       153.273182
std         78.932446
min          5.000000
25%         91.000000
50%        132.000000
75%        220.000000
max        637.000000
Name: text_length, dtype: float64
```

In [10]:
```
plt.figure(figsize=(5, 3))
sns.kdeplot(df['text_length'])
```

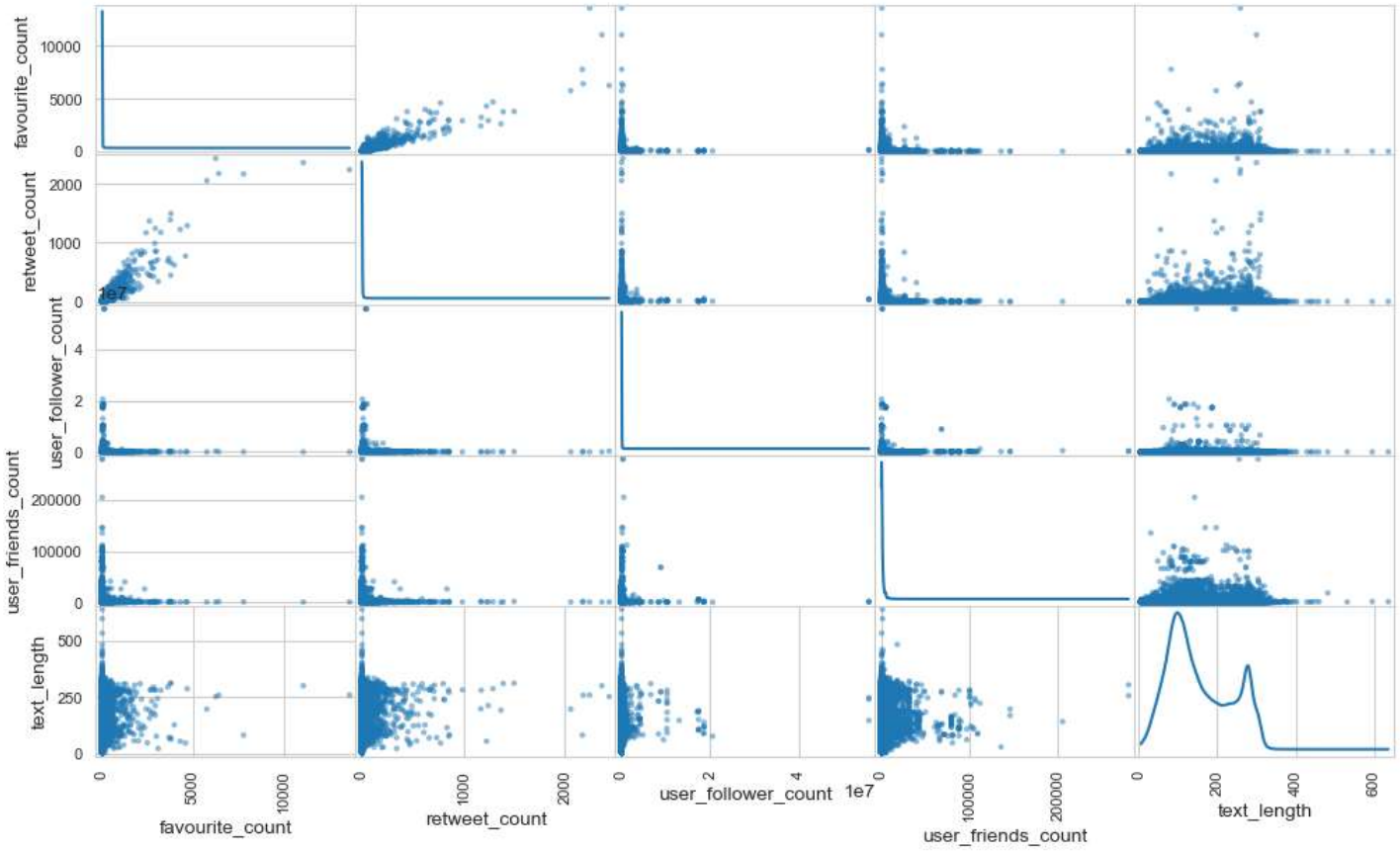
Out[10]:
```
<AxesSubplot:xlabel='text_length', ylabel='Density'>
```



Most tweets have text length between 50 to 300

Count Plots

In [11]:
```
scatter = pd.plotting.scatter_matrix(df.drop(['user_location'],axis=1), diagonal="kde", figsize=(12, 7))
```



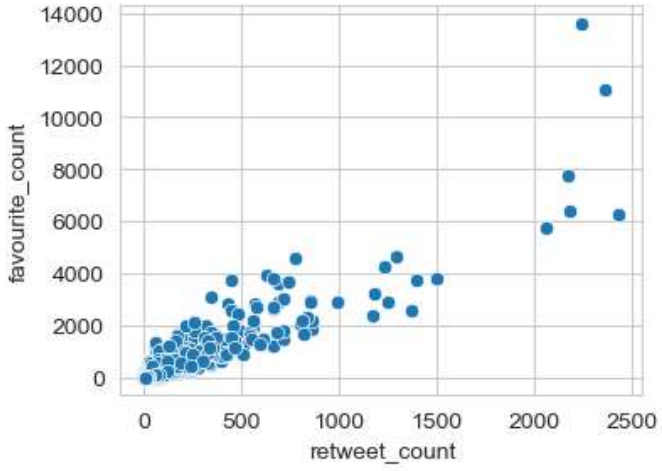
We can observe a relationship between the number of retweets and favorites.

Let's take a closer look at this correlation:

Columns Correlation

In [12]:
```
plt.figure(figsize=(4, 3))
sns.scatterplot(x='retweet_count',y='favourite_count',data=df)
```

Out[12]:
```
<AxesSubplot:xlabel='retweet_count', ylabel='favourite_count'>
```



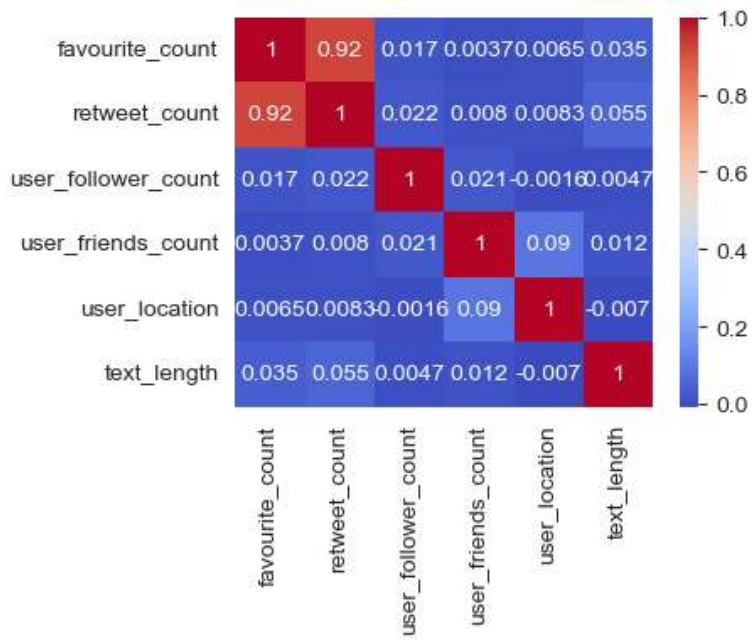
- These columns are directly connected
- The more retweets, the more likes(favorites) we have
- Since with higher retweets, we have higher reachability, having a higher number of likes seem logical, and it doesn't necessarily relate to this specific dataset.

Correlation heatmaps:

We also see a slight relation between users' friend count and whether they have a location or not.

In [13]:
```
plt.figure(figsize=(4, 3))
sns.heatmap(df.corr(),cmap='coolwarm', annot= True)
```

Out[13]: <AxesSubplot:>



Sharing Activities

Can we observe different clusters of users based on their characteristics and/or sharing activities?

Users:

Let's take a look at the users in the dataset:

In [14]:
```
df['screen_name'].value_counts()
```

Out[14]:
```
CyberIQs_          433
IT_securitynews    291
ZDNet              220
ohhara_shiojiri    181
MrsYisWhy          178
                  ...
luisfrik             1
mccart_richard       1
Wirefloss            1
jwjody               1
0x__0                1
Name: screen_name, Length: 36636, dtype: int64
```

There are 36636 unique users

Now we can find how many times each user tweeted (Total tweets for each user):

In [15]:
```
df['total_tweets'] = df['screen_name'].map(df['screen_name'].value_counts())
```

In [16]:
```
df_by_name = df.groupby(["screen_name",'total_tweets'], as_index= False)[['favourite_count','retweet_count']]
```

In [17]:
```
temp = df_by_name.sum().sort_values('total_tweets', ascending=False)
```

In [18]:
```
temp.head()
# scatter = pd.plotting.scatter_matrix(temp, diagonal="kde", figsize=(10, 10))
```

Out[18]:

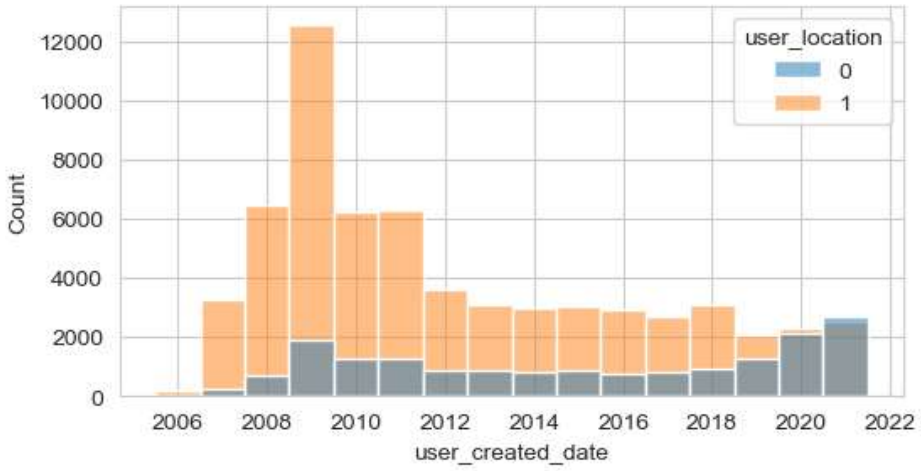
	screen_name	total_tweets	favourite_count	retweet_count
3497	CyberIQs_	433	303	1169
6502	IT_securitynews	291	16	30
15449	ZDNet	220	1051	827
29463	ohhara_shiojiri	181	28	5
9819	MrsYisWhy	178	63	102

Here we can find out when these users created:

We only extract the year that the account was created:

In [19]:
```
df['user_created_date'] = df['user_created_date'].apply(lambda x: int(x[:4]))
```

In [20]:
```
plt.figure(figsize=(6, 3))
sns.histplot(df, x = 'user_created_date', discrete = True, hue='user_location');
```



In [21]:
```
# df['user_created_date'].value_counts()
```

- Most of the accounts created in 2008-2011
- Most of the accounts created in 2021 has no location

In [22]:
```
df[df['user_created_date']==2021]['user_location'].value_counts()
```

Out[22]:
```
0    2729
1    2535
Name: user_location, dtype: int64
```

Status Date:

Here we analyze the day and time of the day for each tweet.

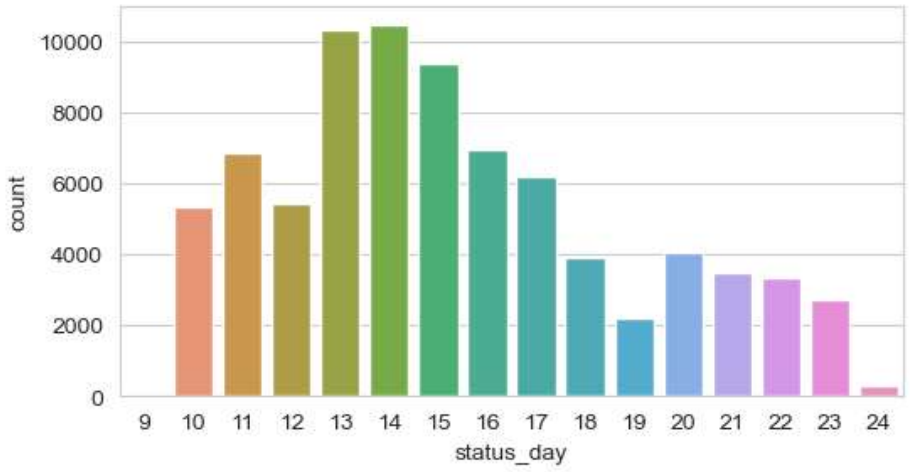
Since all data were taken from December 9, 2021, through December 24, 2021, we only need the day number and hour:

In [23]:
```
df['status_day'] = df['status_date'].apply(lambda x: int(x[8:10])) # To get day number from status_date
df['status_time'] = df['status_date'].apply(lambda x: int(x[11:13])) # To get hour from status_date
```

In [24]:
```
df = df.drop(['status_date'], axis=1)
```

In [25]:
```
plt.figure(figsize=(6, 3))
sns.countplot(x='status_day',data=df)
```

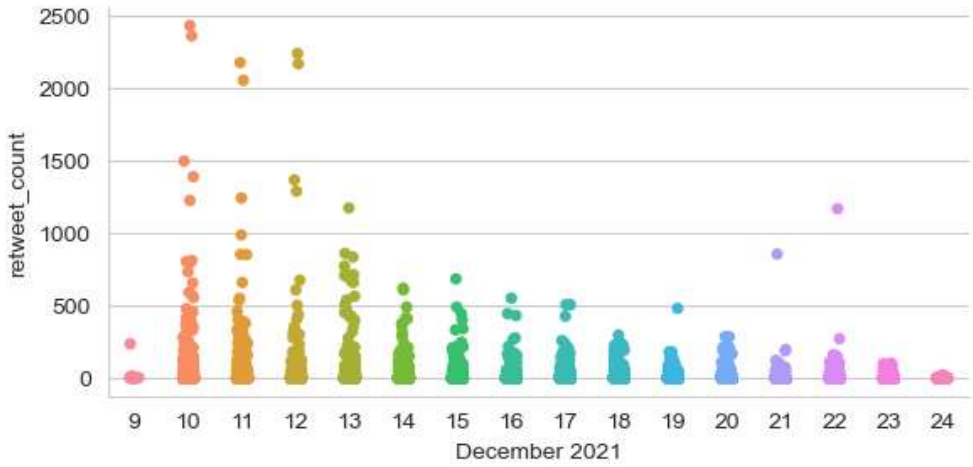
Out[25]:
```
<AxesSubplot:xlabel='status_day', ylabel='count'>
```



There was a high number of tweets between the 13th and 15th of December 2021

In [26]:
```
fig = sns.catplot(x='status_day',y="retweet_count",data=df,height=3, aspect=2)
plt.xlabel("December 2021")
```

Out[26]:
```
Text(0.5, 9.444444444444445, 'December 2021')
```



There was a high number of retweets between the 10th and 13th of December 2021

Text Analysis

In this section, we analyze the texts column to find if they have the following:

- mentions @
- URLs
- hashtags #

Then we perform some cleaning on text column.

In [27]:
```
df['text'].apply(lambda x: '@' in x).value_counts()
```

Out[27]:
```
False    58232
True     22802
Name: text, dtype: int64
```

In [28]:
```
df['text'].apply(lambda x: '#' in x).value_counts()
```

Out[28]:
```
False    47980
True     33054
Name: text, dtype: int64
```

In [29]:
```
df['text'].apply(lambda x: 'https://' in x).value_counts()
```

Out[29]:
```
True     57042
False    23992
Name: text, dtype: int64
```

In [30]:
```
# Create new columns:
df['has_mention'] = df['text'].apply(lambda x: '@' in x)
df['has_hashtag'] = df['text'].apply(lambda x: '#' in x)
df['has_url'] = df['text'].apply(lambda x: 'https://' in x)
```

Cleaning data:

In [31]:
```
import nltk
import re
import numpy as np
```

In [32]:
```
# from: https://www.natasshaselvaraj.com/twitter-sentiment-analysis-with-python/

def cleaner(tweet):
    tweet = re.sub("@[A-Za-z0-9]+","",tweet) #Remove @ sign
    tweet = re.sub(r"(?:\@|http?\://|https?\://|www)\S+", "", tweet) #Remove http links
    tweet = " ".join(tweet.split())
    tweet = tweet.replace("#", "").replace("_", " ") #Remove hashtag sign but keep the text
    # tweet = " ".join(w for w in nltk.wordpunct_tokenize(tweet)
    #      if w.lower() in words or not w.isalpha())
    return tweet


df['text_clean'] = df['text'].apply(cleaner)
```

In [33]:
```
df['text'][3] # sample text
```

Out[33]:
```
'Maintaining IT security is important, especially in the face of the global VIRUS vulnerability and other exploitation
s. Check out ControlCase’s free template packet today. \r\n\r\nhttps://t.co/NO8eCxQS8w \r\n\r\n#ControlCase #VIRUS #Cy
berSecurity #Java #CyberAttack'
```

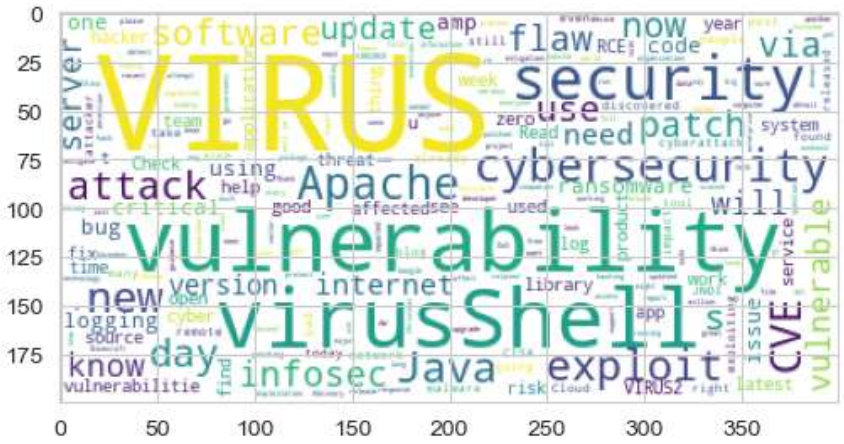
In [34]:
```
df['text_clean'][3] # cleaned text
```

Out[34]:
```
'Maintaining IT security is important, especially in the face of the global VIRUS vulnerability and other exploitation
s. Check out ControlCase’s free template packet today. ControlCase VIRUS CyberSecurity Java CyberAttack'
```

Word Cloud

In [35]:
```
from wordcloud import WordCloud
long_string = ','.join(list(df['text_clean'].values))
wordcloud = WordCloud(background_color="white", max_words=5000, contour_width=3,
contour_color='steelblue',collocations=False)# collocations= False to remove repetitive words
wordcloud.generate(long_string)# Visualize the word cloud
plt.figure( figsize=(6,3))
```

```
plt.imshow(wordcloud)
plt.show()
```



Sentiment Analysis

Now we have clean data, we can add another column to this dataframe for sentiment score, we do this using Vader/NLTK:

The SID module takes in a string and returns a score in each of these four categories - positive, negative, neutral, and compound.

The compound score is calculated by normalizing the positive, negative, and neutral scores. If the compound score is closer to 1, then the Tweet can be classified as positive. If it is closer to -1, then the Tweet can be classified as negative.

In [ ]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
sid = SentimentIntensityAnalyzer()
```

In [37]:
```
sid.polarity_scores(df['text_clean'][3])
```

Out[37]:
```
{'neg': 0.055, 'neu': 0.727, 'pos': 0.218, 'compound': 0.6808}
```

In [38]:
```
df['polarity_scores'] =  df['text_clean'].apply(sid.polarity_scores).apply(lambda x : x['compound'])
```

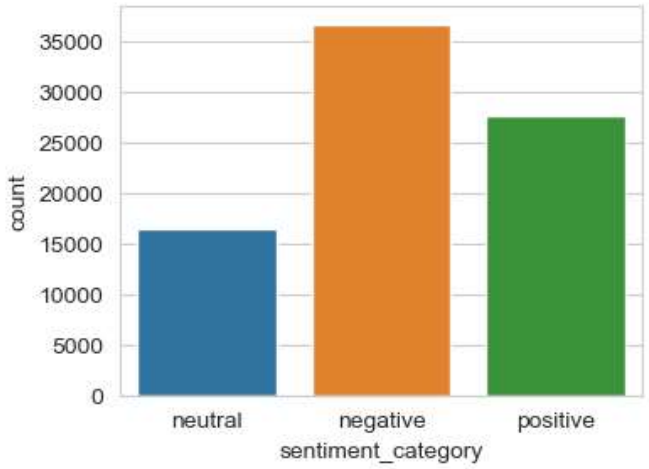
Converting the compound scores into categories - 'positive', 'negative', and 'neutral.'

In [39]:
```
def sentiment_category(polarity):
    label = ''
    if(polarity>0):
        label = 'positive'
    elif(polarity == 0):
        label = 'neutral'
    else:
        label = 'negative'
    return(label)

df['sentiment_category'] = df['polarity_scores'].apply(sentiment_category)
```

In [40]:
```
plt.figure(figsize=(4, 3))
sns.countplot(data=df, x="sentiment_category")
```

Out[40]:
```
<AxesSubplot:xlabel='sentiment_category', ylabel='count'>
```



Retweet Prediction

The goal is to create a Machine Learning model that makes predictions of the number of retweets, based on other columns like the number of friends, favorites, location, status date, etc.

Feature extraction:

In previous sections, we already extracted some features like sentiment, location, text length, status day, and status time.

Predicting the number of retweets can be treated as a classification problem. But for this, we need to use the output as discrete values.

We can experiment to find suitable ranges for our features:

In [41]:
```
df.describe()
```

Out[41]:

	favourite_count	retweet_count	user_follower_count	user_friends_count	user_created_date	user_location	text_length	total_tweets
count	81034.000000	81034.000000	8.103400e+04	81034.000000	81034.000000	81034.000000	81034.000000	81034.000000
mean	10.163327	2.949083	1.977634e+04	1360.054841	2012.899696	0.783548	153.273182	18.663820
std	116.718514	31.370396	4.544552e+05	4092.717109	4.353454	0.411829	78.932446	47.527507
min	0.000000	0.000000	0.000000e+00	0.000000	2006.000000	0.000000	5.000000	1.000000
25%	0.000000	0.000000	1.670000e+02	184.000000	2009.000000	1.000000	91.000000	1.000000
50%	0.000000	0.000000	6.720000e+02	537.000000	2012.000000	1.000000	132.000000	3.000000
75%	2.000000	1.000000	2.883000e+03	1316.000000	2017.000000	1.000000	220.000000	12.000000
max	13605.000000	2434.000000	5.568216e+07	280270.000000	2021.000000	1.000000	637.000000	433.000000

In [42]:
```
df['favourite_cat'] = pd.cut(df['favourite_count'], bins=[0,10,30,np.inf], labels=['low','med','high'],include_lowest=
df['favourite_cat'].value_counts()
```

Out[42]:
```
low     73919
med      4055
high     3060
Name: favourite_cat, dtype: int64
```

In [43]:
```
df['retweet_cat'] = pd.cut(df['retweet_count'], bins=[0,1,15,40,np.inf], labels=[0,1,2,3],include_lowest=True)
df['retweet_cat'].value_counts()
```

Out[43]:
```
0    66765
1    12235
2     1158
3      876
Name: retweet_cat, dtype: int64
```

In [44]:
```
df['user_follower_cat'] = pd.cut(df['user_follower_count'], bins=[0,200,1000,np.inf], labels=['low','med','high'],incl
df['user_follower_cat'].value_counts()
```

Out[44]:
```
high    34262
med     24124
low     22648
Name: user_follower_cat, dtype: int64
```

In [45]:
```
df['user_friends_cat'] = pd.cut(df['user_friends_count'], bins=[0,200,1000,np.inf], labels=['low','med','high'],includ
df['user_friends_cat'].value_counts()
```

Out[45]:
```
med     34064
high    25471
low     21499
Name: user_friends_cat, dtype: int64
```

The ranges for these columns are:

- favourite_cat: 0–10, 10–30, 30+
- retweet_cat: 0–1, 1–15, 40+
- user_follower_cat: 0–200, 200–1000, 1000+
- user_friends_cat: 0–200, 200–1000, 1000+

One hot encoding on tweets features:

For the columns status_day, status_time, user_created_date encoding done using get_dummies() from the pandas library:

In [46]:
```
favourite_cat = pd.get_dummies(df['favourite_cat'],prefix = 'favourite')
# retweet_cat = pd.get_dummies(df['retweet_cat'],prefix = 'retweet')
user_follower_cat = pd.get_dummies(df['user_follower_cat'],prefix = 'follower')
user_friends_cat = pd.get_dummies(df['user_friends_cat'],prefix = 'friends')
status_day = pd.get_dummies(df['status_day'],prefix = 'day')
status_time = pd.get_dummies(df['status_time'],prefix = 'time')
user_created_cat = pd.get_dummies(df['user_created_date'],prefix = 'created')
```

In [47]:
```
columns_drop = [
                "text",
                "retweet_count",
                "screen_name",
                "user_name",
                "source",
                "text_clean",
                "sentiment_category",
                "favourite_cat",
                "user_follower_cat",
                "user_friends_cat",
                "status_day",
                "status_time",
                "user_created_date"
               ]

dfs = [favourite_cat, user_follower_cat, user_friends_cat, status_day, status_time,user_created_cat]

df_final = df.drop(columns_drop, axis=1).join(dfs)

df_final['retweet_cat'] = df['retweet_cat']
df_final.columns
```

Out[47]:
```
Index(['favourite_count', 'user_follower_count', 'user_friends_count',
       'user_location', 'text_length', 'total_tweets', 'has_mention',
       'has_hashtag', 'has_url', 'polarity_scores', 'retweet_cat',
       'favourite_low', 'favourite_med', 'favourite_high', 'follower_low',
       'follower_med', 'follower_high', 'friends_low', 'friends_med',
       'friends_high', 'day_9', 'day_10', 'day_11', 'day_12', 'day_13',
       'day_14', 'day_15', 'day_16', 'day_17', 'day_18', 'day_19', 'day_20',
       'day_21', 'day_22', 'day_23', 'day_24', 'time_0', 'time_1', 'time_2',
       'time_3', 'time_4', 'time_5', 'time_6', 'time_7', 'time_8', 'time_9',
       'time_10', 'time_11', 'time_12', 'time_13', 'time_14', 'time_15',
       'time_16', 'time_17', 'time_18', 'time_19', 'time_20', 'time_21',
       'time_22', 'time_23', 'created_2006', 'created_2007', 'created_2008',
       'created_2009', 'created_2010', 'created_2011', 'created_2012',
       'created_2013', 'created_2014', 'created_2015', 'created_2016',
       'created_2017', 'created_2018', 'created_2019', 'created_2020',
       'created_2021'],
      dtype='object')
```

Using Machine Learning

In [48]:
```
# Load scikit's random forest classifier library
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [49]:
```
features = df_final.drop(['retweet_cat'],axis=1)
target = df_final['retweet_cat'] # 0,1,2,3
```

In [50]:
```
# Create a random forest Classifier.
clf = RandomForestClassifier(n_jobs=2, random_state=0)

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3, random_state=1)

# Fit the model
#
clf.fit(X_train, y_train)
```

Out[50]:
```
RandomForestClassifier(n_jobs=2, random_state=0)
```

In [51]:
```
y_pred = clf.predict(X_test)
```

In [52]:
```
print('Accuracy: %.3f' % accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.887
              precision    recall  f1-score   support

           0       0.91      0.97      0.94     20060
           1       0.71      0.47      0.57      3669
           2       0.60      0.40      0.48       324
           3       0.78      0.83      0.80       258

    accuracy                           0.89     24311
   macro avg       0.75      0.67      0.70     24311
weighted avg       0.88      0.89      0.88     24311

[[19504   553     2     1]
 [ 1883  1725    50    11]
 [    2   141   131    50]
 [    1     7    36   214]]
```

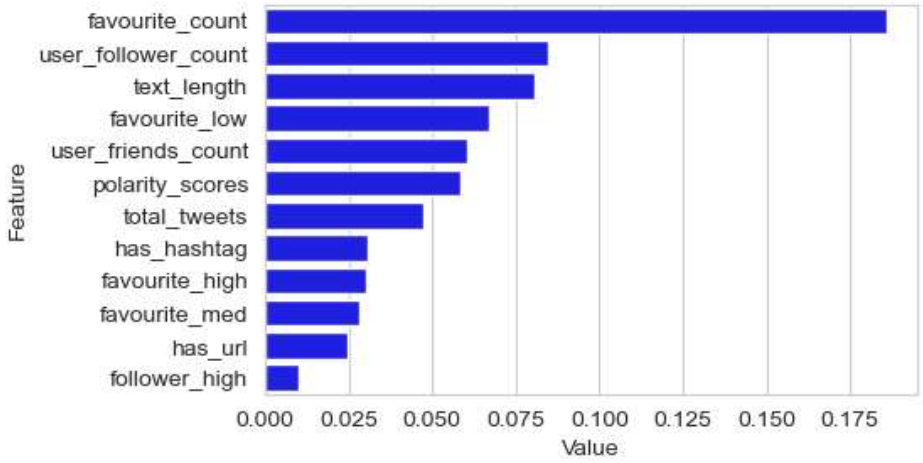
Our model performed well detecting lowest and highest retweet rate categories.

We can also plot the features that were important in our model:

In [53]:
```
list(zip(X_train, clf.feature_importances_))[:10]

feature_imp = pd.DataFrame(sorted(zip(clf.feature_importances_,X_train.columns)), columns=['Value','Feature'])

plt.figure(figsize=(5, 3))
sns.barplot(x="Value", y="Feature", data=feature_imp.sort_values(by="Value", ascending=False)[:12], color='blue');
```



As shown above, we were able to extract some important features like sentiment polarity, text length, and if text contains url or hashtag!

Further improvements

We can improve this model by performing better data preprocessing and employing better Feature Engineering.

For example, in this work, we didn't include the user's name, and we used English as the main language. But it's obvious that different languages for tweets have different audiences.

References

1. **Seaborn: statistical data visualization**
   - https://seaborn.pydata.org/
2. **Random Forest Classifier Example**
   - https://chrisalbon.com/code/machine_learning/trees_and_forests/random_forest_classifier_example/#
3. **Using Data Science to Predict Viral Tweets**
   - https://towardsdatascience.com/using-data-science-to-predict-viral-tweets-615b0acc2e1e
4. **Twitter Sentiment Analysis with Python**
   - https://www.natasshaselvaraj.com/twitter-sentiment-analysis-with-python/
5. **How to predict likes and shares based on your article's title using Machine Learning**
   - https://medium.com/free-code-camp/how-to-predict-likes-and-shares-based-on-your-articles-title-using-machine-learning-47f98f0612ea