Twitter Final Project

November 18, 2022

0.1 Twitter

Big Data & AI - Final Project

Team: - Raúl Cárdenas - Luisa Toro - Thomas Werner - Aman Kumar

0.1.1 I. Project Overview

Problem Statement: What makes a tweet viral? Is it possible to forecast if a tweet will become viral or not? How can this information be leveraged to unlock new ways of monetization for users and increase the value proposition of Twitter Blue?

0.1.2 II. Data Collection

[2]:
```
# import pandas and load df
import pandas as pd
tweets = pd.read_json("random_tweets.json", lines=True)
```

0.1.3 III. Exploratory Analysis

[ ]:
```
# Explore df
print(tweets.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11099 entries, 0 to 11098
Data columns (total 31 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   created_at                 11099 non-null  datetime64[ns, UTC]
 1   id                         11099 non-null  int64
 2   id_str                     11099 non-null  int64
 3   text                       11099 non-null  object
 4   truncated                  11099 non-null  bool
 5   entities                   11099 non-null  object
 6   metadata                   11099 non-null  object
 7   source                     11099 non-null  object
 8   in_reply_to_status_id      1402 non-null   float64
 9   in_reply_to_status_id_str  1402 non-null   float64
 10  in_reply_to_user_id        1503 non-null   float64
```

```
 11  in_reply_to_user_id_str    1503 non-null   float64
 12  in_reply_to_screen_name    1503 non-null   object
 13  user                       11099 non-null  object
 14  geo                        17 non-null     object
 15  coordinates                17 non-null     object
 16  place                      156 non-null    object
 17  contributors               0 non-null      float64
 18  retweeted_status           7372 non-null   object
 19  is_quote_status            11099 non-null  bool
 20  retweet_count              11099 non-null  int64
 21  favorite_count             11099 non-null  int64
 22  favorited                  11099 non-null  bool
 23  retweeted                  11099 non-null  bool
 24  lang                       11099 non-null  object
 25  possibly_sensitive         3192 non-null   float64
 26  quoted_status_id           1154 non-null   float64
 27  quoted_status_id_str       1154 non-null   float64
 28  extended_entities          1199 non-null   object
 29  quoted_status              327 non-null    object
 30  withheld_in_countries      2 non-null      object
dtypes: bool(4), datetime64[ns, UTC](1), float64(8), int64(4), object(14)
memory usage: 2.3+ MB
None
```

[ ]:
```
# Explore df
print(tweets.iloc[0])
```

```
created_at                                        2018-07-31 13:34:40+00:00
id                                                    1024287229525598210
id_str                                                1024287229525598208
text                        RT @KWWLStormTrack7: We are more than a month …
truncated                                                              False
entities                    {'hashtags': [], 'symbols': [], 'user_mentions…
metadata                    {'iso_language_code': 'en', 'result_type': 're…
source                      <a href="http://twitter.com/download/android" …
in_reply_to_status_id                                                    NaN
in_reply_to_status_id_str                                                NaN
in_reply_to_user_id                                                      NaN
in_reply_to_user_id_str                                                  NaN
in_reply_to_screen_name                                                 None
user                        {'id': 145388018, 'id_str': '145388018', 'name…
geo                                                                     None
coordinates                                                             None
place                                                                   None
contributors                                                             NaN
retweeted_status            {'created_at': 'Mon Jul 30 16:49:41 +0000 2018…
is_quote_status                                                        False
retweet_count                                                              3
```

```
favorite_count                                                      0
favorited                                                       False
retweeted                                                       False
lang                                                               en
possibly_sensitive                                                NaN
quoted_status_id                                                  NaN
quoted_status_id_str                                              NaN
extended_entities                                                 NaN
quoted_status                                                     NaN
withheld_in_countries                                             NaN
Name: 0, dtype: object
```

[ ]:
```
# Explore df
print(tweets.columns)
```

```
Index(['created_at', 'id', 'id_str', 'text', 'truncated', 'entities',
       'metadata', 'source', 'in_reply_to_status_id',
       'in_reply_to_status_id_str', 'in_reply_to_user_id',
       'in_reply_to_user_id_str', 'in_reply_to_screen_name', 'user', 'geo',
       'coordinates', 'place', 'contributors', 'retweeted_status',
       'is_quote_status', 'retweet_count', 'favorite_count', 'favorited',
       'retweeted', 'lang', 'possibly_sensitive', 'quoted_status_id',
       'quoted_status_id_str', 'extended_entities', 'quoted_status',
       'withheld_in_countries'],
      dtype='object')
```

[ ]:
```
# Explore text column
print(tweets["text"].iloc[40])
```

```
LinkFest ~ Best Reads on Writing, Screenwriting &amp;amp; Self-Publishing: The Door
In The Floor #selfpublishing #writing  https://t.co/MoKWmzSw0r
```

[ ]:
```
# Explore user column
print(tweets.user.iloc[0])
```

```
{'id': 145388018, 'id_str': '145388018', 'name': 'Derek Wolkenhauer',
'screen_name': 'derekw221', 'location': 'Waterloo, Iowa', 'description': '',
'url': None, 'entities': {'description': {'urls': []}}, 'protected': False,
'followers_count': 215, 'friends_count': 335, 'listed_count': 2, 'created_at':
'Tue May 18 21:30:10 +0000 2010', 'favourites_count': 3419, 'utc_offset': None,
'time_zone': None, 'geo_enabled': True, 'verified': False, 'statuses_count':
4475, 'lang': 'en', 'contributors_enabled': False, 'is_translator': False,
'is_translation_enabled': False, 'profile_background_color': '022330',
'profile_background_image_url':
'http://abs.twimg.com/images/themes/theme15/bg.png',
'profile_background_image_url_https':
'https://abs.twimg.com/images/themes/theme15/bg.png', 'profile_background_tile':
False, 'profile_image_url':
```

```
'http://pbs.twimg.com/profile_images/995790590276243456/cgxRVviN_normal.jpg',
'profile_image_url_https':
'https://pbs.twimg.com/profile_images/995790590276243456/cgxRVviN_normal.jpg',
'profile_banner_url':
'https://pbs.twimg.com/profile_banners/145388018/1494937921',
'profile_link_color': '0084B4', 'profile_sidebar_border_color': 'A8C7F7',
'profile_sidebar_fill_color': 'C0DFEC', 'profile_text_color': '333333',
'profile_use_background_image': True, 'has_extended_profile': True,
'default_profile': False, 'default_profile_image': False, 'following': False,
'follow_request_sent': False, 'notifications': False, 'translator_type': 'none'}
```

[ ]:
```
# Explore entities column
print(tweets.entities.iloc[11096])
```

```
{'hashtags': [], 'symbols': [], 'user_mentions': [{'screen_name':
'ChrisDanicic', 'name': 'Chris Danicic', 'id': 774299495790108672, 'id_str':
'774299495790108672', 'indices': [0, 13]}, {'screen_name': 'Mediaite', 'name':
'Mediaite', 'id': 29465136, 'id_str': '29465136', 'indices': [14, 23]},
{'screen_name': 'benshapiro', 'name': 'Ben Shapiro', 'id': 17995040, 'id_str':
'17995040', 'indices': [24, 35]}], 'urls': [{'url': 'https://t.co/PAByGjQsvw',
'expanded_url': 'https://twitter.com/i/web/status/1024287114459074560',
'display_url': 'twitter.com/i/web/status/1…', 'indices': [117, 140]}]}
```

[ ]:
```
# Explore metadata column
print(tweets.metadata.iloc[0])
tweets["metadata"].apply(lambda tweet: tweet["iso_language_code"]).unique()
```

```
{'iso_language_code': 'en', 'result_type': 'recent'}
```

[ ]:
```
array(['en', 'tl', 'ja', 'fi', 'ko', 'und', 'nl', 'es', 'pt', 'in', 'ur',
       'vi', 'it', 'th', 'ca', 'fr', 'ru', 'el', 'ar', 'pl', 'ro', 'tr',
       'sl', 'de', 'zh', 'fa', 'sv', 'et', 'hi', 'ht'], dtype=object)
```

[ ]:
```
# Explore rewteeted status column
print(tweets.retweeted_status.iloc[0])
```

```
{'created_at': 'Mon Jul 30 16:49:41 +0000 2018', 'id': 1023973918959382528,
'id_str': '1023973918959382528', 'text': 'We are more than a month into summer
but the days are getting shorter. The sunrise is about 25 minutes later on Jul…
https://t.co/fEfTJIfrA7', 'truncated': True, 'entities': {'hashtags': [],
'symbols': [], 'user_mentions': [], 'urls': [{'url': 'https://t.co/fEfTJIfrA7',
'expanded_url': 'https://twitter.com/i/web/status/1023973918959382528',
'display_url': 'twitter.com/i/web/status/1…', 'indices': [117, 140]}]},
'metadata': {'iso_language_code': 'en', 'result_type': 'recent'}, 'source': '<a
href="http://www.socialnewsdesk.com" rel="nofollow">SocialNewsDesk</a>',
'in_reply_to_status_id': None, 'in_reply_to_status_id_str': None,
'in_reply_to_user_id': None, 'in_reply_to_user_id_str': None,
'in_reply_to_screen_name': None, 'user': {'id': 131864835, 'id_str':
```

```
'131864835', 'name': 'KWWL Storm Track 7', 'screen_name': 'KWWLStormTrack7',
'location': 'Waterloo, IA', 'description': 'The latest weather information from
the StormTrack 7 team at KWWL.', 'url': 'http://t.co/W2x2myBw0j', 'entities':
{'url': {'urls': [{'url': 'http://t.co/W2x2myBw0j', 'expanded_url':
'http://www.kwwl.com/weather', 'display_url': 'kwwl.com/weather', 'indices': [0,
22]}]}, 'description': {'urls': []}}, 'protected': False, 'followers_count':
7225, 'friends_count': 42, 'listed_count': 134, 'created_at': 'Sun Apr 11
15:36:02 +0000 2010', 'favourites_count': 379, 'utc_offset': None, 'time_zone':
None, 'geo_enabled': False, 'verified': False, 'statuses_count': 60957, 'lang':
'en', 'contributors_enabled': False, 'is_translator': False,
'is_translation_enabled': False, 'profile_background_color': 'C0DEED',
'profile_background_image_url':
'http://abs.twimg.com/images/themes/theme1/bg.png',
'profile_background_image_url_https':
'https://abs.twimg.com/images/themes/theme1/bg.png', 'profile_background_tile':
True, 'profile_image_url':
'http://pbs.twimg.com/profile_images/884503227110223873/ocabR3F0_normal.jpg',
'profile_image_url_https':
'https://pbs.twimg.com/profile_images/884503227110223873/ocabR3F0_normal.jpg',
'profile_banner_url':
'https://pbs.twimg.com/profile_banners/131864835/1477690280',
'profile_link_color': '0084B4', 'profile_sidebar_border_color': 'C0DEED',
'profile_sidebar_fill_color': 'DDEEF6', 'profile_text_color': '333333',
'profile_use_background_image': True, 'has_extended_profile': False,
'default_profile': False, 'default_profile_image': False, 'following': False,
'follow_request_sent': False, 'notifications': False, 'translator_type':
'none'}, 'geo': None, 'coordinates': None, 'place': None, 'contributors': None,
'is_quote_status': False, 'retweet_count': 3, 'favorite_count': 0, 'favorited':
False, 'retweeted': False, 'possibly_sensitive': False, 'lang': 'en'}
```

[ ]:
```
# Check for missing data
tweets.isnull().sum()
```

[ ]:
```
created_at                    0
id                            0
id_str                        0
text                          0
truncated                     0
entities                      0
metadata                      0
source                        0
in_reply_to_status_id      9697
in_reply_to_status_id_str  9697
in_reply_to_user_id        9596
in_reply_to_user_id_str    9596
in_reply_to_screen_name    9596
user                          0
```

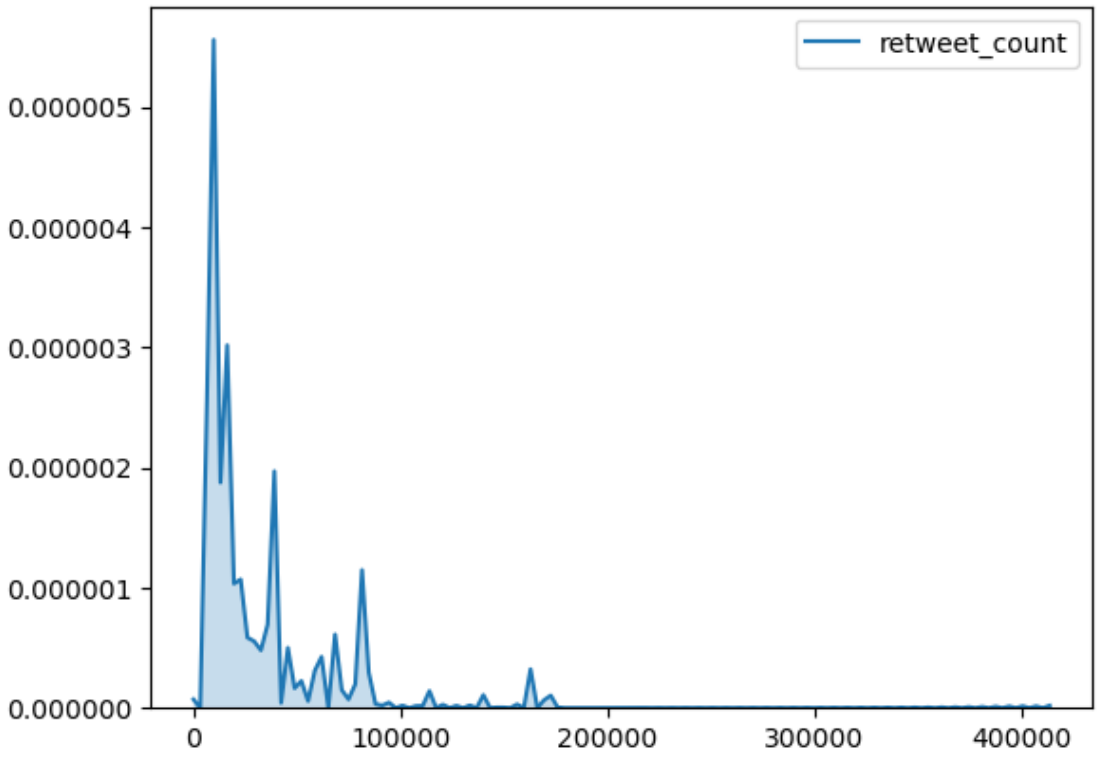
```
geo                          11082
coordinates                  11082
place                        10943
contributors                 11099
retweeted_status              3727
is_quote_status                  0
retweet_count                    0
favorite_count                   0
favorited                        0
retweeted                        0
lang                             0
possibly_sensitive            7907
quoted_status_id              9945
quoted_status_id_str          9945
extended_entities             9900
quoted_status                10772
withheld_in_countries        11097
dtype: int64
```

0.1.4 IV. Feature Engineering

[3]:
```
# Import libraries
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from transformers import pipeline
```

[ ]:
```
# What makes a tweet viral? Median retweets, mean retweets, retweets over time, ␣
 ↪etc.?
print(tweets["retweet_count"].describe())
ax = plt.subplot()
sns.kdeplot(tweets["retweet_count"], shade=True)
plt.show()
```

```
count     11099.000000
mean       2777.956392
std       12180.169923
min           0.000000
25%           0.000000
50%          13.000000
75%         428.500000
max      413719.000000
Name: retweet_count, dtype: float64
```



[ ]:
```
# Create target column to classify tweets as viral or non-viral based on the␣
 ↪number of retweets
# Median
median_retweets = tweets["retweet_count"].median()
print("Median Retweets: {}".format(median_retweets))
tweets["is_viral"] = np.where(tweets["retweet_count"] >= median_retweets, 1, 0)
print(tweets["is_viral"].value_counts())
```

```
Median Retweets: 13.0
1    5591
0    5508
Name: is_viral, dtype: int64
```

[ ]:
```
# Create target column to classify tweets as viral or non-viral based on the␣
 ↪number of retweets (method 2)
# Everything above Q3
retweet_threshold = 428
tweets["is_viral"] = np.where(tweets["retweet_count"] >= retweet_threshold, 1,␣
 ↪0)
print(tweets["is_viral"].value_counts())
```

```
Median Retweets: 13.0
0    8322
```

```
1    2777
Name: is_viral, dtype: int64
```

[ ]:
```
# Explore datetime metadata
# Extract dates from created_at
tweets["created_at"] = tweets.apply(lambda tweet: tweet["user"]["created_at"],␣
 ↪axis=1)
print(tweets.created_at.iloc[0])
# Convert to datetime
from datetime import datetime
tweets["created_at"] = tweets.apply(lambda tweet: datetime.
 ↪strptime(tweet["created_at"], "%a %b %d %H:%M:%S %z %Y"), axis=1)
# Extract year, month, day, time, and day of week
tweets["year"] = tweets.apply(lambda tweet: tweet["created_at"].year, axis=1)
tweets["month"] = tweets.apply(lambda tweet: tweet["created_at"].month, axis=1)
tweets["day"] = tweets.apply(lambda tweet: tweet["created_at"].day, axis=1)
tweets["time"] = tweets.apply(lambda tweet: tweet["created_at"].time(), axis=1)
tweets["day_of_week"] = tweets.apply(lambda tweet: tweet["created_at"].
 ↪weekday(), axis=1)
print(tweets[["created_at", "year", "month", "day", "time", "day_of_week"]].
 ↪iloc[0])
```

```
Tue May 18 21:30:10 +0000 2010
created_at     2010-05-18 21:30:10+00:00
year                                2010
month                                  5
day                                   18
time                            21:30:10
day_of_week                            1
Name: 0, dtype: object
```

[ ]:
```
# How many years are covered in the dataset?
print(np.sort(tweets.year.unique()))
print("\ntotal number of years: {}\n".format(len(tweets.year.unique())))
# How many tweets per year?
print("Numer of tweets per year:")
tweets_per_year = tweets.groupby("year").size()
print(tweets_per_year)
# Retweets over time
tweets.groupby(["created_at"]).retweet_count.sum()
```

```
[2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018]

total number of years: 13

Numer of tweets per year:
year
2006       2
```

```
2007      34
2008     227
2009    1158
2010     864
2011    1182
2012    1128
2013     972
2014     888
2015     867
2016    1068
2017    1404
2018    1305
dtype: int64
```

[ ]:
```
created_at
2006-11-21 18:15:42+00:00      0
2006-11-29 01:51:08+00:00      1
2007-01-01 08:15:11+00:00     99
2007-01-10 01:35:12+00:00      0
2007-01-31 23:14:07+00:00     14
                              ..
2018-07-31 13:26:52+00:00     75
2018-07-31 13:27:33+00:00    238
2018-07-31 13:28:43+00:00    177
2018-07-31 13:29:34+00:00    131
2018-07-31 13:29:46+00:00      2
Name: retweet_count, Length: 10400, dtype: int64
```

[ ]:
```
print(tweets.user.iloc[0])
```

```
{'id': 145388018, 'id_str': '145388018', 'name': 'Derek Wolkenhauer',
'screen_name': 'derekw221', 'location': 'Waterloo, Iowa', 'description': '',
'url': None, 'entities': {'description': {'urls': []}}, 'protected': False,
'followers_count': 215, 'friends_count': 335, 'listed_count': 2, 'created_at':
'Tue May 18 21:30:10 +0000 2010', 'favourites_count': 3419, 'utc_offset': None,
'time_zone': None, 'geo_enabled': True, 'verified': False, 'statuses_count':
4475, 'lang': 'en', 'contributors_enabled': False, 'is_translator': False,
'is_translation_enabled': False, 'profile_background_color': '022330',
'profile_background_image_url':
'http://abs.twimg.com/images/themes/theme15/bg.png',
'profile_background_image_url_https':
'https://abs.twimg.com/images/themes/theme15/bg.png', 'profile_background_tile':
False, 'profile_image_url':
'http://pbs.twimg.com/profile_images/995790590276243456/cgxRVviN_normal.jpg',
'profile_image_url_https':
'https://pbs.twimg.com/profile_images/995790590276243456/cgxRVviN_normal.jpg',
'profile_banner_url':
'https://pbs.twimg.com/profile_banners/145388018/1494937921',
```

```
'profile_link_color': '0084B4', 'profile_sidebar_border_color': 'A8C7F7',
'profile_sidebar_fill_color': 'C0DFEC', 'profile_text_color': '333333',
'profile_use_background_image': True, 'has_extended_profile': True,
'default_profile': False, 'default_profile_image': False, 'following': False,
'follow_request_sent': False, 'notifications': False, 'translator_type': 'none'}
```

[ ]:
```
# Create new feature columns that might help us determine whether a tweet is␣
 ↪viral or not
# Length of tweet
tweets["tweet_length"] = tweets.apply(lambda tweet: len(tweet["text"]), axis=1)
# Extract followers count from user column
tweets["followers_count"] = tweets.apply(lambda tweet:␣
 ↪tweet["user"]["followers_count"], axis=1)
# Extract friends count from user column
tweets["friends_count"] = tweets.apply(lambda tweet:␣
 ↪tweet["user"]["friends_count"], axis=1)
# Extract favourite count from user column
tweets["favourites_count"] = tweets.apply(lambda tweet:␣
 ↪tweet["user"]["favourites_count"], axis=1)
# Extract verified status from user column
tweets["verified"] = tweets.apply(lambda tweet: tweet["user"]["verified"],␣
 ↪axis=1)
# Extract language from metadata column
tweets["language"] = tweets.apply(lambda tweet:␣
 ↪tweet["metadata"]["iso_language_code"], axis=1)
# Extract number of hashtags from text column
tweets["hashtags_count"] = tweets.apply(lambda tweet: tweet["text"].count("#"),␣
 ↪axis=1)
# Extract number of words in text column
tweets["words_count"] = tweets.apply(lambda tweet: len(tweet["text"].strip().
 ↪split(" ")), axis=1)
```

[ ]:
```
# Perform sentiment analysis on tweets
sentiment_pipeline = pipeline("sentiment-analysis")
tweets = (
    tweets
    .assign(sentiment = lambda x: x["text"].apply(lambda tweet:␣
 ↪sentiment_pipeline(tweet)))
    .assign(
         label = lambda x: x["sentiment"].apply(lambda s: (s[0]["label"])),
         score = lambda x: x["sentiment"].apply(lambda s: (s[0]["score"]))
    )
)
print(tweets[["text", "sentiment", "label", "score"]].iloc[0:5])
```

```
No model was supplied, defaulted to distilbert-base-uncased-finetuned-
sst-2-english and revision af0f99b (https://huggingface.co/distilbert-base-
```

```
uncased-finetuned-sst-2-english).
Using a pipeline without specifying a model name and revision in production is
not recommended.
```

```
                                                text  \
0  RT @KWWLStormTrack7: We are more than a month …
1  @hail_ee23 Thanks love its just the feeling of…
2  RT @TransMediaWatch: Pink News has more on the…
3  RT @realDonaldTrump: One of the reasons we nee…
4  RT @First5App: This hearing of His Word doesn'…

                                           sentiment     label     score
0  [{'label': 'NEGATIVE', 'score': 0.974782645702…  NEGATIVE  0.974783
1  [{'label': 'POSITIVE', 'score': 0.998418807983…  POSITIVE  0.998419
2  [{'label': 'NEGATIVE', 'score': 0.997975289821…  NEGATIVE  0.997975
3  [{'label': 'NEGATIVE', 'score': 0.943356394767…  NEGATIVE  0.943356
4  [{'label': 'NEGATIVE', 'score': 0.992612719535…  NEGATIVE  0.992613
```

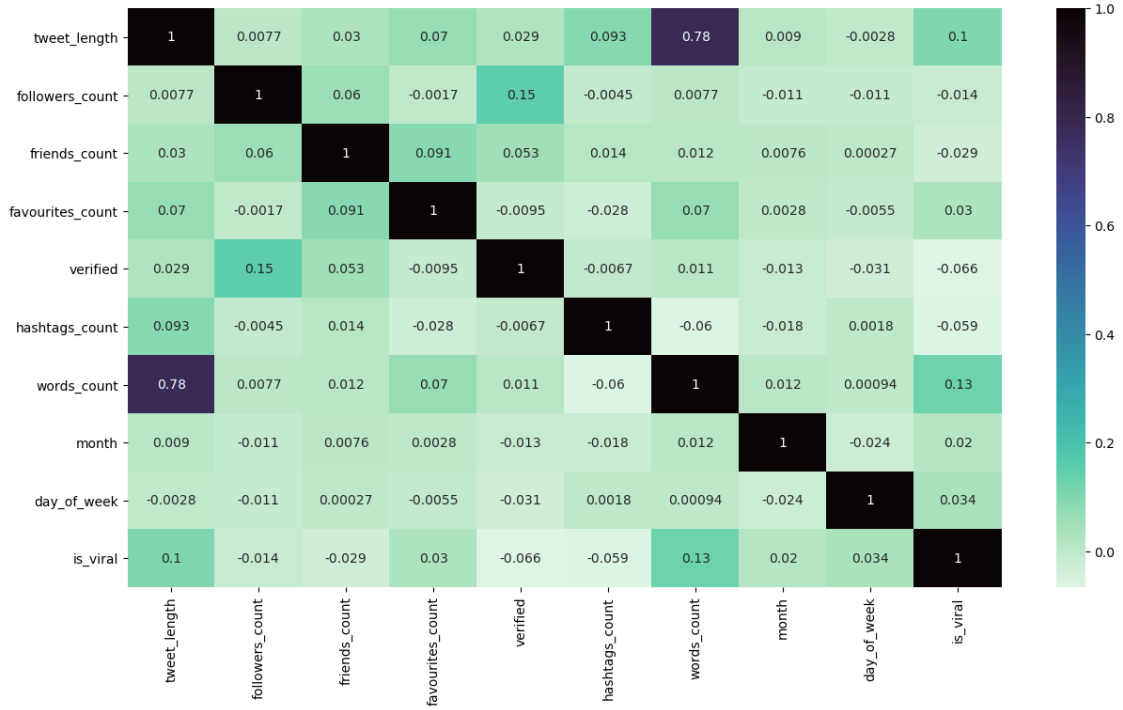
[ ]:
```
# Save dataframe to json file
tweets.to_json("random_tweets_fe.json")
```

[4]:
```
# Import dataframe from checkpoint
tweets = pd.read_json("random_tweets_fe.json")
```

[5]:
```
# Create label and feature columns
labels = tweets["is_viral"]
features = tweets[["tweet_length","followers_count","friends_count",␣
 ↪"favourites_count", "verified", "language", "hashtags_count", "words_count",␣
 ↪"month", "day_of_week", "label"]]
```

[6]:
```
# Create correlation matrix with new df
temp_df = tweets[["tweet_length","followers_count","friends_count",␣
 ↪"favourites_count", "verified", "language", "hashtags_count", "words_count",␣
 ↪"month", "day_of_week", "label", "is_viral"]]
corr = temp_df.corr(method="pearson")
plt.figure(figsize=(15,8))
sns.heatmap(corr, annot=True, cmap="mako_r")
```

[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa56fb56210>
```



[7]:
```
# Dummy encode categorical features
features = pd.get_dummies(features, columns=["verified", "language", "month",␣
 ↪"day_of_week", "label"], drop_first=True)
```

0.1.5 V. Modelling

[8]:
```
# import libraries
from sklearn.model_selection import train_test_split
from sklearn.model_selection import learning_curve
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.inspection import permutation_importance
from sklearn.model_selection import GridSearchCV
from imblearn.over_sampling import RandomOverSampler
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.pipeline import make_pipeline
from collections import Counter
import matplotlib.pyplot as plt
import seaborn as sns
```

[9]:
```
# Split training and test sets
x_train, x_test, y_train, y_test = train_test_split(features, labels, test_size
 ↪= 0.20, random_state = 1)
```

[10]:
```
# Oversampling to correct data imbalance
ovs = RandomOverSampler(random_state=42)
x_res, y_res = ovs.fit_resample(x_train, y_train)
```

[11]:
```
# Print before and after oversampling
print("Original dataset shape {}".format(Counter(labels)))
print("Resampled dataset shape {}".format(Counter(y_res)))
```

```
Original dataset shape Counter({0: 8322, 1: 2777})
Resampled dataset shape Counter({0: 6646, 1: 6646})
```

Random Forests

[12]:
```
# Set parameters for parameter optimization
param_grid = {"n_estimators": [100,200,300,500], "max_depth" : [4,7,10,15]}
```

[13]:
```
# Run optimizer
cv_grid = GridSearchCV(estimator=RandomForestClassifier(),
 ↪param_grid=param_grid, cv=5)
cv_grid.fit(x_res, y_res)
```

[13]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [4, 7, 10, 15],
                         'n_estimators': [100, 200, 300, 500]})
```

[14]:
```
# print best parameters
cv_grid.best_params_
```

[14]:
```
{'max_depth': 15, 'n_estimators': 300}
```

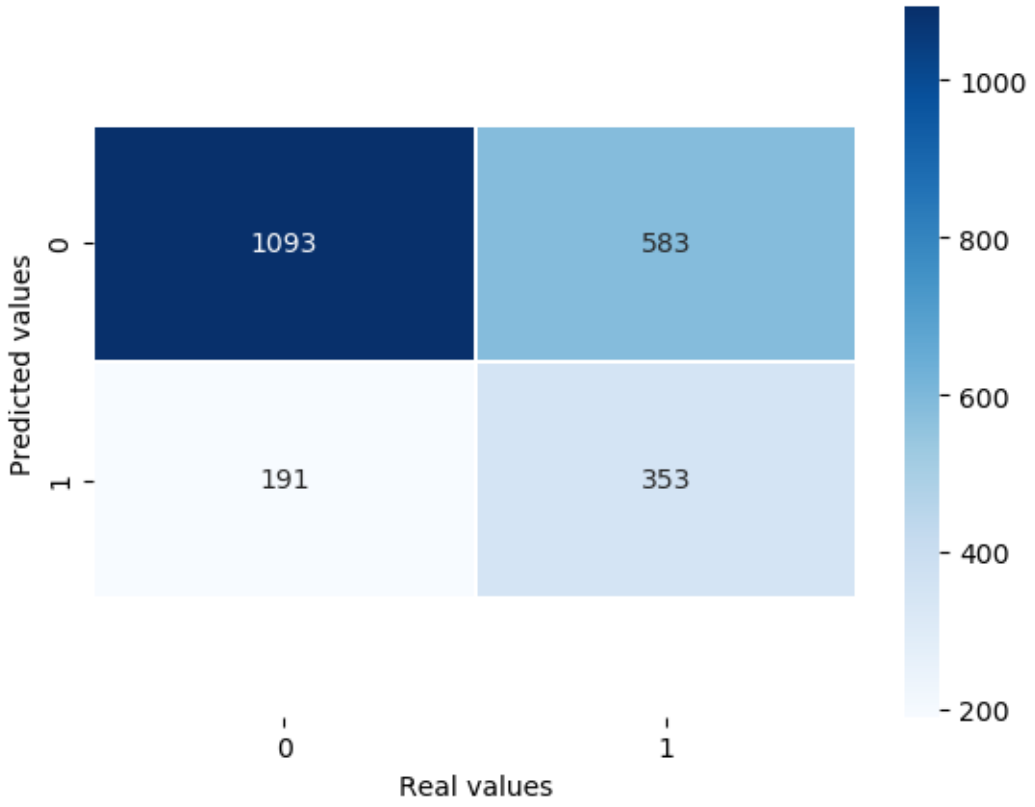
[15]:
```
rf = cv_grid.best_estimator_
print(rf)
```

```
RandomForestClassifier(max_depth=15, n_estimators=300)
```

[16]:
```
# Predictions
rf_predictions = rf.predict(x_test)
```

[17]:
```
# Confusion Matrix
cm = confusion_matrix(y_test, rf_predictions)
ax = sns.heatmap(cm, linewidth = 0.5, annot = True, cmap = "Blues", fmt = "g")
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
plt.ylabel("Predicted values")
```

```
plt.xlabel("Real values")
plt.show()
```



[18]:
```
# Model score
print("Train Score: ", rf.score(x_res, y_res))
print("Test Score: ", rf.score(x_test, y_test))
```

```
Train Score:  0.857508275654529
Test Score:  0.6513513513513514
```

[19]:
```
# Classification Report
print(classification_report(y_test, rf_predictions))
# Precision: Out of all tweets that were predicted to be viral, how many were␣
 ↪actually viral?
# Recall: Out of all tweets that are viral, how many were predicted to be viral?
# Accuracy: Total number of correct predictions out of all predictions
```

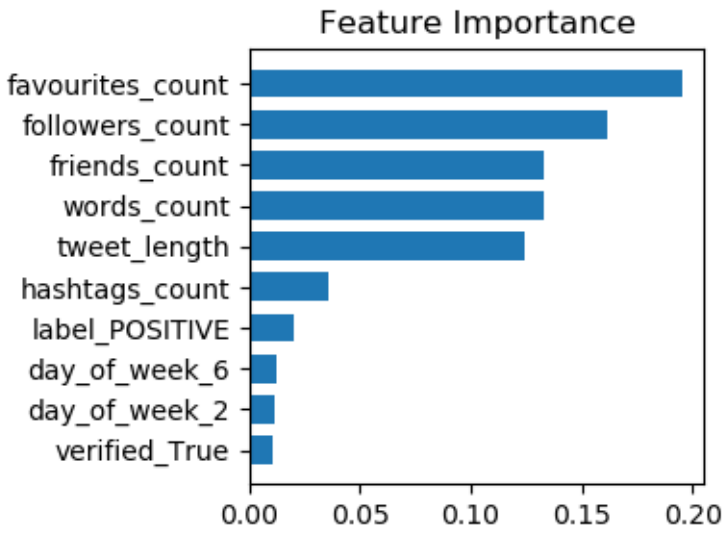
```
              precision    recall  f1-score   support

           0       0.85      0.65      0.74      1676
           1       0.38      0.65      0.48       544
```

```
    accuracy                           0.65      2220
   macro avg       0.61      0.65      0.61      2220
weighted avg       0.74      0.65      0.67      2220
```

[20]:
```
# Permutation importance
"""
result = permutation_importance(rf, x_train, y_train, n_repeats=10,␣
 ↪random_state=0)
perm_sorted_idx = result.importances_mean.argsort()[-10:]
"""
# Feature importance
tree_importance_sorted_idx = np.argsort(rf.feature_importances_)[-10:][::-1]
tree_indices = (np.arange(0, len(rf.feature_importances_)) + 0.5)[-10:][::-1]

# Graph of feature importance
fig, ax1 = plt.subplots(1, 1, figsize=(4, 3))
ax1.barh(tree_indices, rf.feature_importances_[tree_importance_sorted_idx],␣
 ↪height=0.7)
ax1.set_yticks(tree_indices)
ax1.set_yticklabels(features.columns[tree_importance_sorted_idx])
ax1.set_title("Feature Importance")
#ax1.set_ylim((0, len(rf.feature_importances_)))
"""
ax2.boxplot(
    result.importances[perm_sorted_idx].T,
    vert=False,
    labels=features.columns[perm_sorted_idx],
)
"""
fig.tight_layout()
plt.show()
```



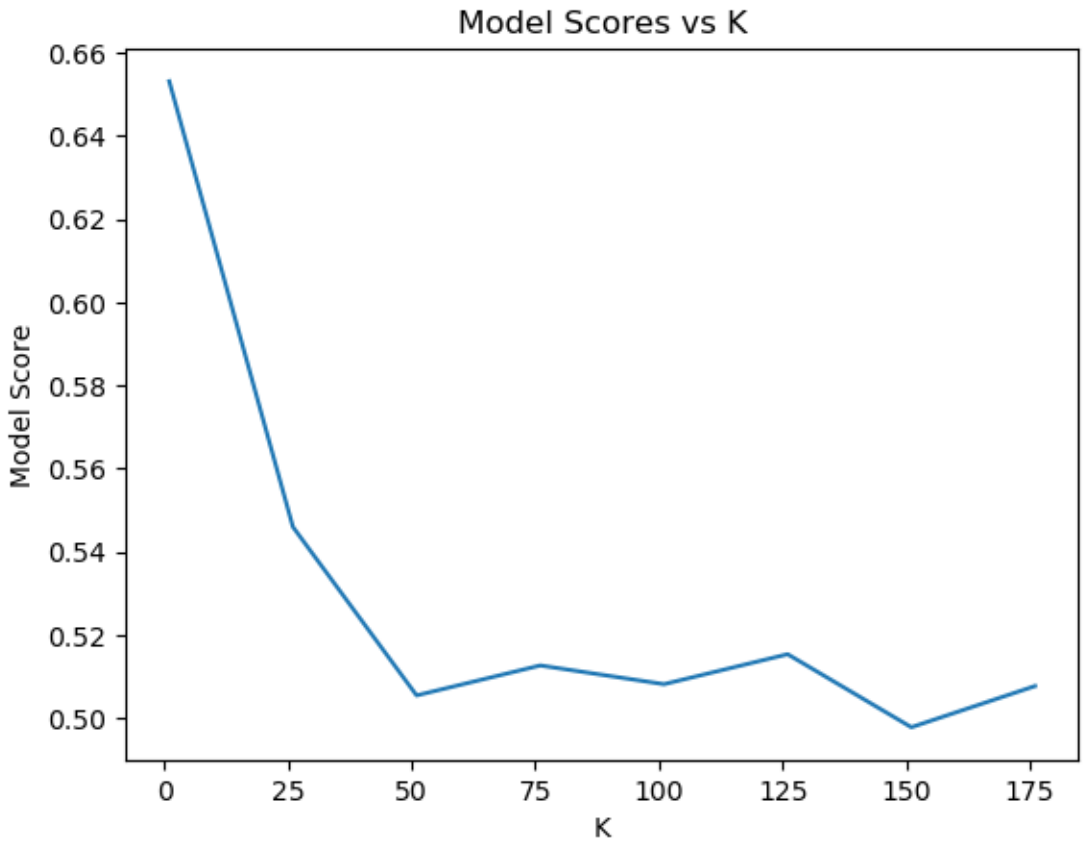
K-Neighbors

[21]:
```
# Determine the optimal number of neighbors
scores = []
for k in range(1,201,25):
    knn_pipe = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors␣
 ↪= k))
    knn_pipe.fit(x_res, y_res)
    scores.append(knn_pipe.score(x_test, y_test))
```

[22]:
```
max(scores)
scores.index(max(scores))+1
```

[22]: 1

[23]:
```
# Plot scores when k varies
ax = plt.subplot()
plt.plot(range(1,201,25), scores)
ax.set_title("Model Scores vs K")
plt.ylabel("Model Score")
plt.xlabel("K")
# wrap text in x axis
plt.show()
```

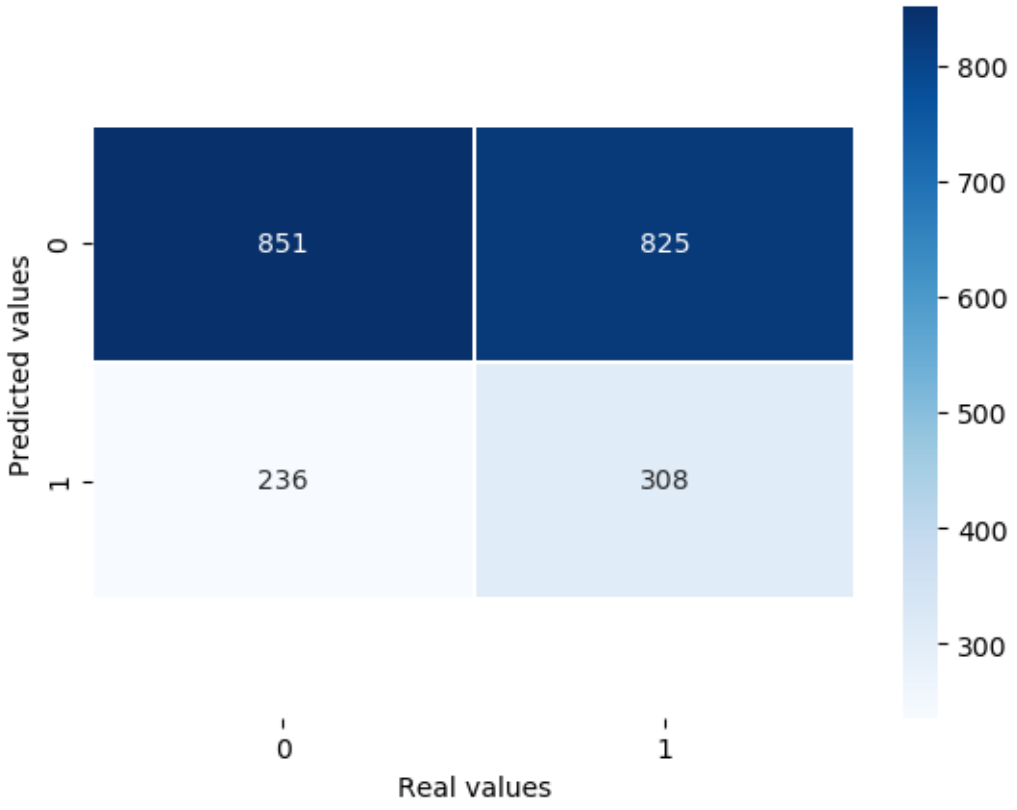


[24]:
```
# Create and train the model with pipeline
knn_pipe = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors =␣
↪100))
knn_pipe.fit(x_res, y_res)
```

[24]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('kneighborsclassifier',
                 KNeighborsClassifier(n_neighbors=100))])
```

[25]:
```
# Predictions
knn_predictions = knn_pipe.predict(x_test)
```

[26]:
```
# Confusion Matrix
cm = confusion_matrix(y_test, knn_predictions)
ax = sns.heatmap(cm, linewidth = 0.5, annot = True, cmap = "Blues", fmt = "g")
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
plt.ylabel("Predicted values")
plt.xlabel("Real values")
plt.show()
```



[27]:
```
# Model score
print("Train Score: ", knn_pipe.score(x_res, y_res))
print("Test Score: ", knn_pipe.score(x_test, y_test))
```

```
Train Score:  0.5868943725549203
Test Score:  0.5220720720720721
```

[28]:
```
# Classification Report
print(classification_report(y_test, knn_predictions))
```

```
              precision    recall  f1-score   support

           0       0.78      0.51      0.62      1676
           1       0.27      0.57      0.37       544

    accuracy                           0.52      2220
   macro avg       0.53      0.54      0.49      2220
weighted avg       0.66      0.52      0.56      2220
```

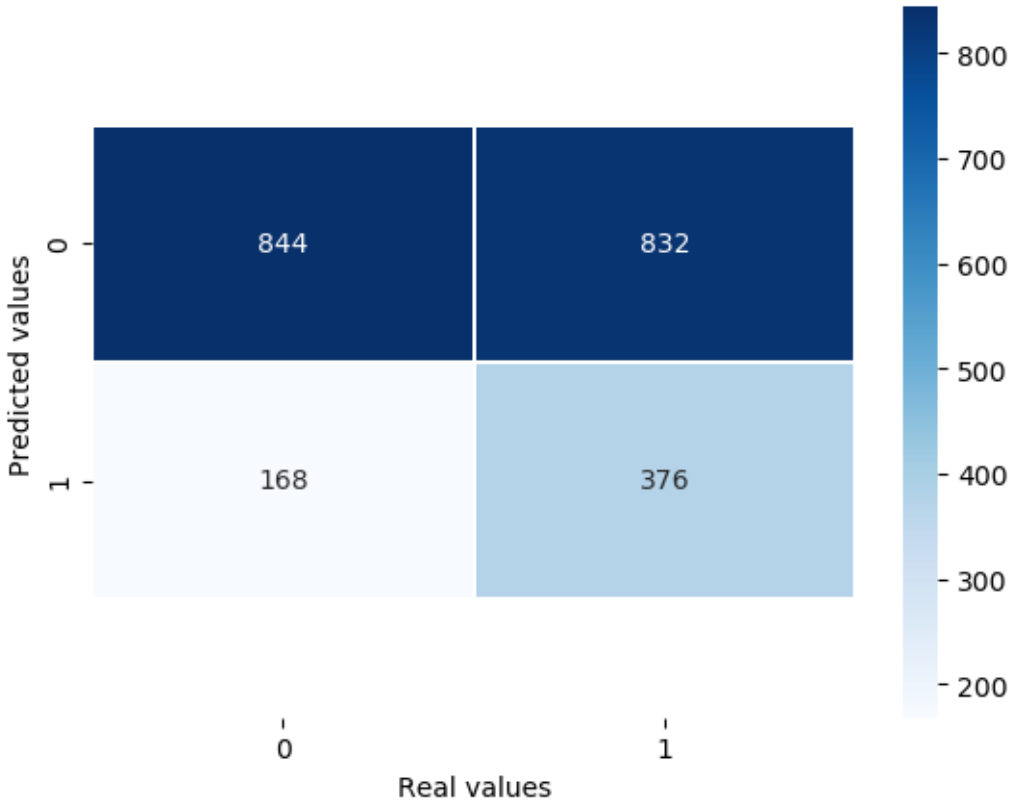
Logistic Regression

[29]:
```
# Create and train the model with pipeline
lr_pipe = make_pipeline(StandardScaler(), LogisticRegression())
lr_pipe.fit(x_res, y_res)
```

[29]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('logisticregression', LogisticRegression())])
```

[30]:
```
# Predictions
lr_predictions = lr_pipe.predict(x_test)
```

[31]:
```
# Confusion Matrix
cm = confusion_matrix(y_test, lr_predictions)
ax = sns.heatmap(cm, linewidth = 0.5, annot = True, cmap = "Blues", fmt = "g")
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
plt.ylabel("Predicted values")
plt.xlabel("Real values")
plt.show()
```



[32]:
```
# Model score
print("Train Score: ", lr_pipe.score(x_res, y_res))
```

```
print("Test Score: ", lr_pipe.score(x_test, y_test))
```

```
Train Score:  0.5941919951850737
Test Score:  0.5495495495495496
```

[33]:
```
# Classification Report
print(classification_report(y_test, lr_predictions))
```

```
              precision    recall  f1-score   support

           0       0.83      0.50      0.63      1676
           1       0.31      0.69      0.43       544

    accuracy                           0.55      2220
   macro avg       0.57      0.60      0.53      2220
weighted avg       0.71      0.55      0.58      2220
```