

پیاده سازی روش های پردازش زبان  
بر روی داده های توییتر

## مقدمه

در اینجا، سه مسئله علوم داده با استفاده از داده های توییتر مطرح شده است که هدف از آن ها پیاده سازی پیش پردازش متون و داده های عددی، مدل سازی داده ها با استفاده از روش های یادگیری ماشین و بکارگیری روش های مناسب برای بهبود عملکرد آن است.

## مسائل

### 1- دسته بندی متون

#### 1-1- داده ها

در جدول زیر، تعداد و نوع داده ها را مشاهده میکنیم. داده ها شامل متن توییت، URL های استفاده شده در آنها، تعداد یک سری ویژگی های مربوط به توییت مانند لایک ها و ریپلای ها، و همچنین متون نقل قول شده است. متغیر هدف ستون با نام relevant است که مرتبط یا نامرتب بودن توییت را مشخص میکند.

```
RangeIndex: 7591 entries, 0 to 7590
Data columns (total 13 columns):
#   Column                      Non-Null Count  Dtype
---  -
0   in_reply_to_user_id         3542 non-null   float64
1   is_quote                    7591 non-null   bool
2   is_retweet                  7591 non-null   bool
3   like_count                  7591 non-null   int64
4   quote_count                 7591 non-null   int64
5   quoted_text                 575 non-null    object
6   reply_count                 7591 non-null   int64
7   retweet_count               7591 non-null   int64
8   retweet_text                0 non-null      float64
9   text                        7591 non-null   object
10  urls_expanded_url           3747 non-null   object
11  urls_url                    3747 non-null   object
12  relevant                    7591 non-null   object
dtypes: bool(2), float64(2), int64(4), object(5)
```

#### 1-2- هدف

دسته بندی توییت ها به دو دسته مرتبط و نامرتب با استفاده از روش های یادگیری ماشین.

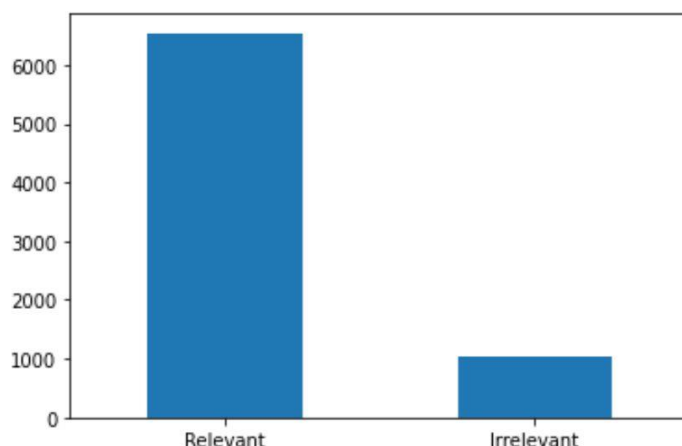
#### 1-3- روش

ابتدا داده ها را نمایش میدهیم و مشاهده میکنیم ستون retweet\_text هیچ داده ای در خود نگه نمیدارد، پس این ستون را حذف میکنیم. ستون is\_reply\_to\_user\_id شامل اطلاعات کاربری است که این توییت یک ریپلای به آن است. از این ویژگی، یک ویژگی جدید با نوع داده boolean با عنوان is\_reply استخراج میکنیم. ستون is\_quote اگر true باشد به

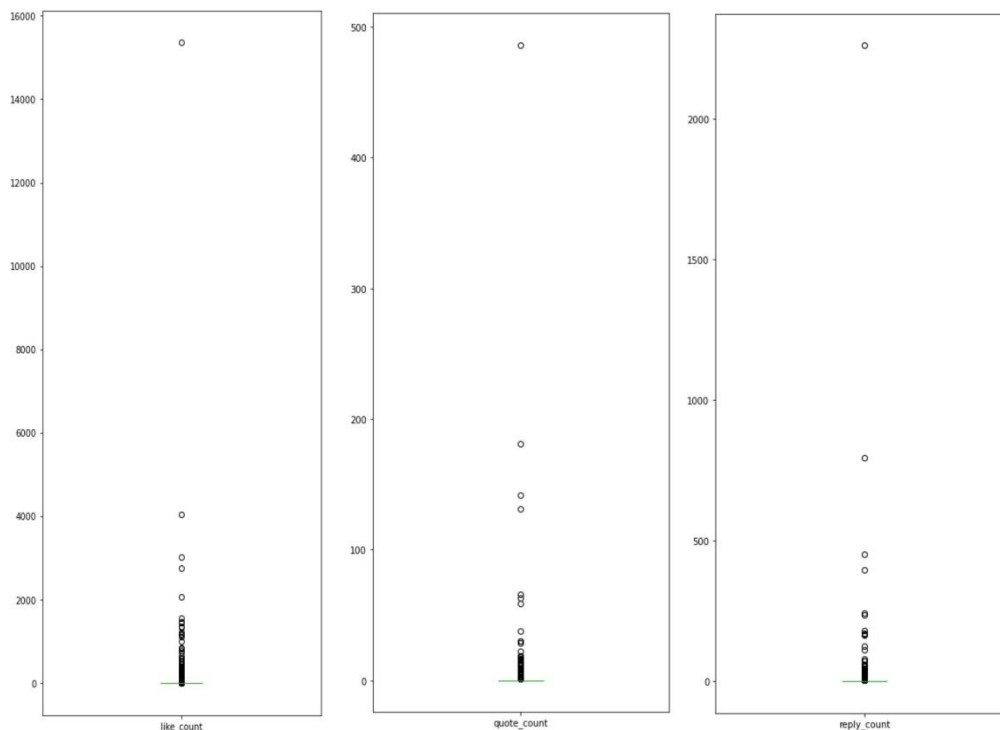
معنای این است که یک متن نقل قول شده در تویییت استفاده شده و این متن در ستون `quoted_text` آمده، از طرفی این متن نقل قول شده در متن تویییت اصلی نیز آمده، پس ستون `quoted_text` را حذف میکنیم زیرا که اطلاعات آن در دو ستون `is_quote` و `text` آمده است. دو ستون `urls_url` و `urls_exapnded_url` آدرس هایی که در متن تویییت آمده است را بصورت `short` و `expanded`، به ترتیب، نگه میدارند. ما در اینجا آدرس ها را پردازش نمیکنیم و از این اطلاعات ویژگی جدید به نام `url_count` استخراج میکنیم که تعداد آدرس های بکار برده شده در هر تویییت را نگه میدارد. در جدول زیر، اطلاعات آماری مربوط به داده های عددی آمده است. با مشاهده سه چارک اول در داده های مربوط به `count`، متوجه میشویم در خود داده های `outlier` دارند زیرا که چارک سوم مقدار صفر یا یک دارد در حالی که مقدار ماکزیمم این متغیرها به چند هزار میرسد.

	<code>in_reply_to_user_id</code>	<code>like_count</code>	<code>quote_count</code>	<code>reply_count</code>	<code>retweet_count</code>
<b>count</b>	3.542000e+03	7591.000000	7591.000000	7591.000000	7591.000000
<b>mean</b>	3.194062e+17	11.169938	0.318667	1.375972	2.106705
<b>std</b>	5.055444e+17	198.442037	6.581126	29.109233	29.621685
<b>min</b>	1.200000e+01	0.000000	0.000000	0.000000	0.000000
<b>25%</b>	2.581237e+07	0.000000	0.000000	0.000000	0.000000
<b>50%</b>	4.328953e+08	0.000000	0.000000	0.000000	0.000000
<b>75%</b>	8.300799e+17	1.000000	0.000000	1.000000	0.000000
<b>max</b>	1.357398e+18	15352.000000	486.000000	2261.000000	2275.000000

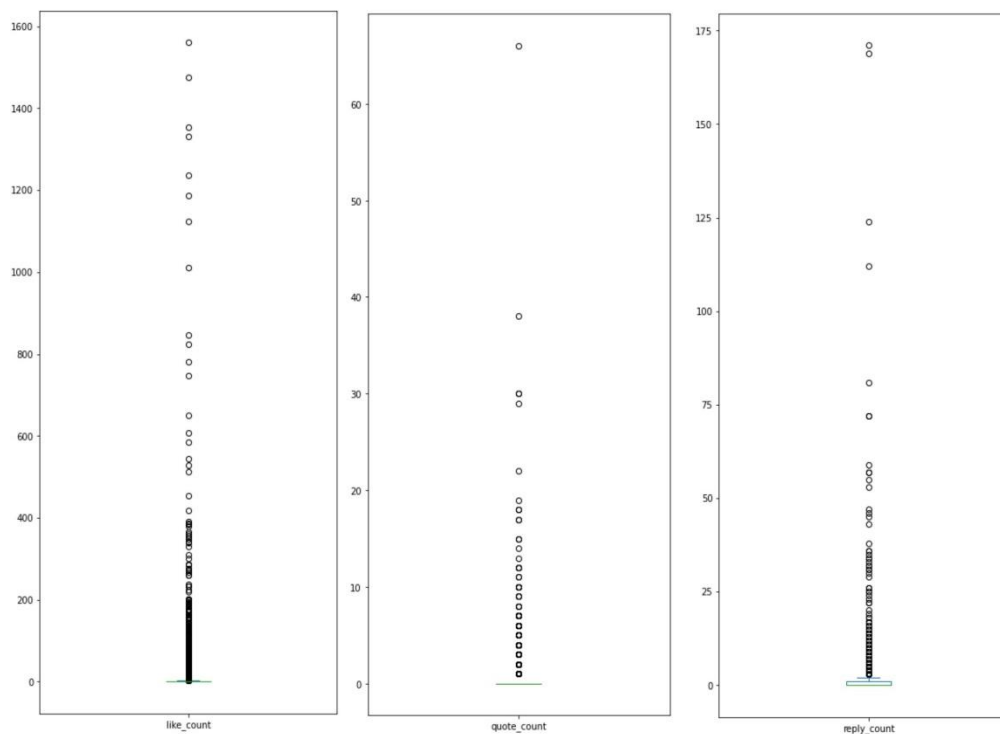
سپس به سراغ متغیر هدف میرویم. مقادیر دو کلاس را مشاهده میکنیم و متوجه میشویم که داده ها بالانس نیستند و کلاس مثبت تقریباً 6 برابر کلاس منفی است. به این دلیل از متریک های `precision`، `recall` و `f1-score` برای ارزیابی عملکرد مدل استفاده میکنیم تا مشاهده کنیم که آیا مدل میتواند کلاس کوچکتر را تشخیص دهد.



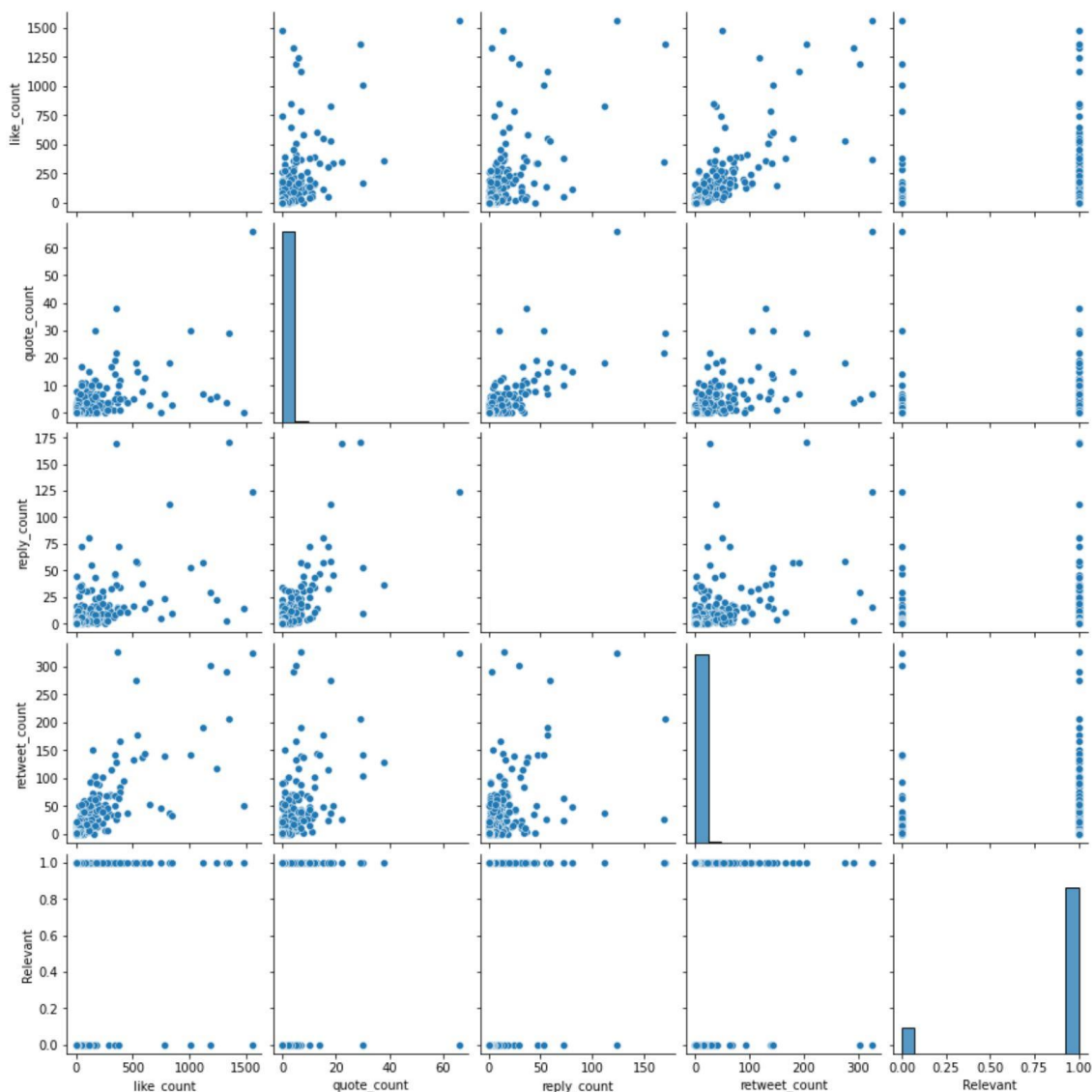
حال از داده های عددی **boxplot** رسم میکنیم تا با استفاده از آن بتوانیم محدوده ای برای **outlier** ها انتخاب کنیم.



در نمودار های بالا، تمرکز داده ها و نقاط دور افتاده را مشاهده میکنیم. برای هر کدام محدوده ای تعیین میکنیم و مقادیر خارج از آن محدوده را حذف میکنیم که در نهایت تعداد 9 داده از داده ها حذف میشود. با رسم دوباره **boxplot** مشاهده میکنیم پراکندگی نقاط کمتر است.



در مرحله بعد، برای مشاهده پراکندگی داده های عددی نسبت به یکدیگر و به متغیر هدف، از این داده ها pairplot رسم میکنیم.



مشاهده میکنیم هیچکدام از جفت ویژگی ای با یکدیگر ارتباط خطی ندارند. از این نمودار همچنین میتوانیم اینکه اگر پراکندگی یک ویژگی نسبت به ویژگی های دیگر ثابت باشد را استخراج کنیم که در اینصورت این ویژگی را حذف میکنیم زیرا اطلاعات اضافه ای به ما نمیدهد. در اینجا چنین توزیعی مشاهده نمیکنیم.

به علت اینکه توزیع این ویژگی ها نرمال نیست، از آنها لگاریتم طبیعی میگیریم تا توزیع آنها به توزیع نرمال نزدیک شود ولی مشاهده کردیم تغییر محسوسی ایجاد نشد. حال به سراغ پیش پردازش متون میرویم و 'punctuation'، 'URL'، اعداد و 'stopwords' را از آنها حذف میکنیم، سپس 'stemming' را بر روی کلمات پیاده میکنیم.

```

StopWords = stopwords.words('english')
def clean_tweet(tweet):
    if type(tweet) == float:
        return ""
    # Lowercase all the letters
    temp = tweet.lower()
    # remove mentions and hashtags
    temp = re.sub(r"@(\w)+", "", temp)
    temp = re.sub(r"#(\w)+", "", temp)
    # remove links
    temp = re.sub(r"http\S+", "", temp)
    temp = re.sub(r"www.\S+", "", temp)
    # remove non-alphanumeric characters
    temp = re.sub(r"(\d|\W)+", " ", temp)
    # tokenization
    temp = word_tokenize(temp)
    # remove stopwords
    temp = [word for word in temp if word not in StopWords]
    # stem words
    ps = PorterStemmer()
    temp = [ps.stem(word) for word in temp]
    # join words back together
    temp = " ".join(temp)
    return temp

```

در اینجا، دو مرحله از پیش پردازش شامل normalization برای داده های عددی و bag of words برای متون باقی مانده که ابتدا نیاز است داده های train و test را جدا کنیم تا data leakage اتفاق نیفتد.

ابتدا داده ها را جدا میکنیم و برای این کار 33 درصد داده ها را به test اختصاص میدهم.

سپس از MinMaxScaler برای normalization و از CountVectorizer برای bag of words استفاده میکنیم. این الگوریتم ها را ابتدا روی داده های train مدل میکنیم و روی هر دو داده پیاده میکنیم.

در قدم بعدی، داده ها را مدل میکنیم. ابتدا از الگوریتم Random Forest استفاده کردیم و بعد از مدلسازی، دیتای تست را به آن دادیم تا پیش بینی کند. مشاهده میکنیم دقت خیلی کمی بر روی کلاس منفی دارد.

	precision	recall	f1-score	support
0	0.31	0.04	0.07	346
1	0.86	0.99	0.92	2157
accuracy			0.86	2503
macro avg	0.59	0.51	0.49	2503
weighted avg	0.79	0.86	0.80	2503

سپس از الگوریتم XGBoost استفاده میکنیم و مشاهده میکنیم دقت آن بر روی کلاس منفی از Random Forest کمتر است.

	precision	recall	f1-score	support
0	0.14	0.01	0.03	346
1	0.86	0.99	0.92	2157
accuracy			0.85	2503
macro avg	0.50	0.50	0.47	2503
weighted avg	0.76	0.85	0.80	2503