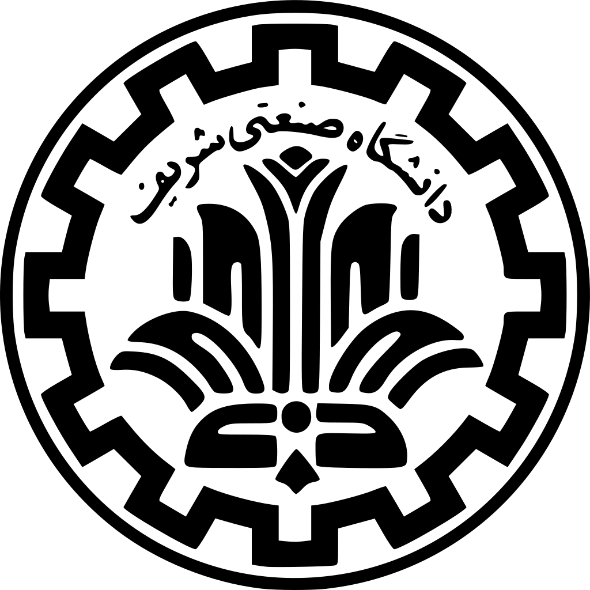
هوالحق



دانشگاه صنعتی شریف-دانشکده مهندسی صنایع

**گزارش پروژه‌ی «پیش‌بینی وایرال‌شدن توییت»**

استاد درس:

**دکتر مجید خدمتی**

نگارندگان:

|  |  |
| --- | --- |
| محمدعلی اله‌دادی | **95103786** |
| محمدحسین اله‌دادیان | **95103797** |

تابستان 99

فهرست مطالب

[قسمت اول: مقدمه و مرور ادبیات 1](#_Toc45031043)

[مقدمه 1](#_Toc45031044)

[منابع مطالعه‌شده 2](#_Toc45031045)

[قسمت دوم: ارائه مدل پیش‌بینی 3](#_Toc45031046)

[مرحله اول: شناسایی و استخراج داده‌های موردنظر جهت داده‌کاوی (Feature Engineering) 3](#_Toc45031047)

[مرحله‌ی دوم: تشریح داده‌ها 5](#_Toc45031048)

[مرحله‌ی سوم: پیش‌پردازش داده‌ها 6](#_Toc45031049)

[تبدیل داده‌ها به مقادیر قابل پردازش 6](#_Toc45031050)

[حذف داده‌های پرت 7](#_Toc45031051)

[مرحله‌ی چهارم: ارائه‌ی مدل جهت پیش‌بینی و ارزیابی آن 8](#_Toc45031052)

[مرحله‌ی پنجم: نتیجه‌گیری و تحلیل 9](#_Toc45031053)

# قسمت اول: مقدمه و مرور ادبیات

## مقدمه

هدف ما از انجام این پروژه، پیش‌بینی وایرال‌[[1]](#footnote-1)شدن یا نشدن یک توییت[[2]](#footnote-2) در شبکه‌ی اجتماعی توییتر[[3]](#footnote-3) است. پست وایرالِ (هم‌خانواده‌ی کلمه‌ی ویروس) محتوایی است که مانند ویروس رشد نمایی شدیدی دارد و در مدتی کوتاه، توسط افراد بسیار زیادی دیده می‌شود. شاهد این مسأله در شبکه‌ی توییتر می‌تواند تعداد لایک[[4]](#footnote-4) و ریتوییت[[5]](#footnote-5) بالا در مدت‌زمان کوتاه باشد.



تصویر 1 - نمونه‌ی یک توییت وایرال

به عنوان مثال، توییت بالا بعد از 16 ساعت، حدود 100 هزار لایک و 18 هزار ریتوییت دریافت کرده است. کاربر پست‌کننده هم حدود 12 هزار فالوور[[6]](#footnote-6) دارد. یعنی عمده‌ی افرادی که این پست را پسندیده‌اند، در جایگاه اول حتی او را دنبال هم نکرده‌اند و به واسطه‌ی ریتوییت کاربران و الگوریتم‌های پیشنهادی توییتر، این توییت در فید[[7]](#footnote-7) آن‌ها آمده است.

با شناخت و تجربه‌ی مختصری که از این فضا داریم، در جایگاه اول می‌توان حدس‌هایی در مورد متغیرهای تاثیرگذار در وایرال‌شدن زد. متغیرهایی مثل تعداد لایک و ریتوییت، عمر توییت، تعداد فالوورها، کلیدواژه‌های استفاده‌شده در توییت و...

## منابع مطالعه‌شده

در کنار حدس‌های اولیه، با جستجو در اینترنت در مورد «پیش‌بینی توییت‌های وایرال» به منابع زیر رسیدیم که خلاصه‌ی مباحث مطرح‌شده در هر یک را به اختصار در زیر می‌آوریم. مواردی که از این منابع به دست آمده‌اند، کمابیش در اجرای پروژه نیز استفاده شده‌اند:

* [تاثیر جمله‌بندی در تعداد ریتوییت‌ها](https://chenhaot.com/pages/wording-for-propagation.html). نویسنده‌ی مقاله ادعا می‌کند که با مدل‌هایی که در اختیار دارد، می‌تواند صرفاً با بررسی جمله‌بندی دو توییت، پیش‌بینی کند که کدام‌یک در شرایط برابر، تعداد ریتوییت بیشتری دریافت خواهد کرد. یک نسخه‌ی دمو از این مدل در [اینجا](https://chenhaot.com/retweetedmore/) برای مشاهده موجود است.
* [پیش‌بینی توییت‌های وایرال با استفاده از دیتای توییتر](https://www.smartdatacollective.com/20223/). در این گزارش، عوامل موثر بررسی و در نهایت مدلی پیش‌بینی‌کننده آماده شده است. ستون‌های داده‌ی پروژه‌ی درس مشابهت زیادی با پارامترهای استفاده‌شده در گزارش بالا دارند. ستون‌هایی مثل منشن‌کردن[[8]](#footnote-8) کاربر دیگر، استفاده‌کردن از هشتگ[[9]](#footnote-9) و... در پروژه استفاده شده‌اند و ستون‌های پیچیده‌تری مثل موضوع توییت، درخواست ریتوییت‌کردن (Please retweet, etc)، ربط توییت به موضوعات روز و... استفاده نشده‌اند.

# قسمت دوم: ارائه مدل پیش‌بینی

## مرحله اول: شناسایی و استخراج داده‌های موردنظر جهت داده‌کاوی (Feature Engineering)

در زیر تمامی ستون‌های موردنظر برای انجام فرآیند داده‌کاوی به همراه توضیح مختصر در مورد چرایی استفاده از آن‌ها آمده است:

|  |  |  |  |
| --- | --- | --- | --- |
| چرایی استفاده از ستون | معنی ستون | نام ستون |  |
| ممکن است بیش از اندازه کوتاه باشد و نامفهوم. | تعداد کارکترها | len | Tweet-related Information |
| ممکن است به دلیل طولانی بودن توییت، محتوای آن کامل داده نشود. | ر.ک. متادیتا[[10]](#footnote-10) | truncated |
| دیده‌شدن بیشتر توییت در صورت داشتن برچسب | تعداد هشتگ‌ها | hashtags |
| جنجال = دیده‌شدن | تعداد منشن‌ها | user\_mentions |
| جنجال = دیده‌شدن | ر.ک. متادیتا | is\_quote\_status |
| زبان توییت، تعداد کاربرانی که آن را می‌فهمند را تعیین می‌کند | ر.ک. متادیتا | lang |
| عمر بیشتر کاربر، می‌تواند نمایان‌گر سابقه‌ی بیشتر او باشد | زمانِ ساختن حساب کاربری | created\_at | User-related Information |
| کسی که هنوز عکسی برای خودش در نظر نگرفته است، احتمالاً تجربه‌ی چندانی ندارد. | آیا کاربر از عکس پیش‌فرض توییتر استفاده می‌کند؟ | default\_profile\_image |
| در صورتی که اینترکشن کاربر بالا باشد، اینترکشن اطرافیانش هم نسبت به او بالا می‌رود. | ر.ک. متادیتا | favourites\_count |
| تعداد فالوور بالا یعنی افراد بیشتری توییت را می‌بینند. | تعداد فالوورها | followers\_count |
| زبان توییت، تعداد کاربرانی که آن را می‌فهمند را تعیین می‌کند | ر.ک. متادیتا | lang |
| تعداد لیست بالا یعنی افراد بیشتری توییت را می‌بینند. | تعداد لیست‌هایی که کاربر در آن‌ها آمده است | listed\_count |
| استاتوس‌های بیشتر کاربر، می‌تواند نمایان‌گر سابقه‌ی بیشتر او باشد | تعداد توییت‌های کاربر | statuses\_count |
| پذیرش حرف کسی که توییتر او را تایید کرده است، می‌تواند راحت‌تر باشد. | حساب کاربر توسط توییتر تایید شده است یا خیر؟ | verified |

این ستون‌ها، به دو دسته‌ی کلی تقسیم می‌شوند: اطلاعات توییت و اطلاعات کاربر. خود اطلاعات کاربر هم به دو دسته‌ی کاربر ریتوییت‌کننده (retweeter-user) و کاربر توییت‌کننده (original-user) تقسیم شده است. (در صورتی که ریتوییتی در کار نباشد، مقادیر کاربر ریتوییت‌کننده برابر 0 در نظر گرفته می‌شوند. یعنی انگار کسی که آن را ریتوییت کرده، شخص بی‌اثری بوده است)

این سه دسته از ستون‌ها در کنار هم قرار می‌گیرند و ستون‌های X ما را تشکیل می‌دهند. برای تشکیل ستون Y، مقدار retweet\_count را از هر خط می‌خوانیم و آن‌هایی را که از مدیان[[11]](#footnote-11)ستون بیشتر هستند را با 1 و بقیه را با 0 جایگذاری می‌کنیم.

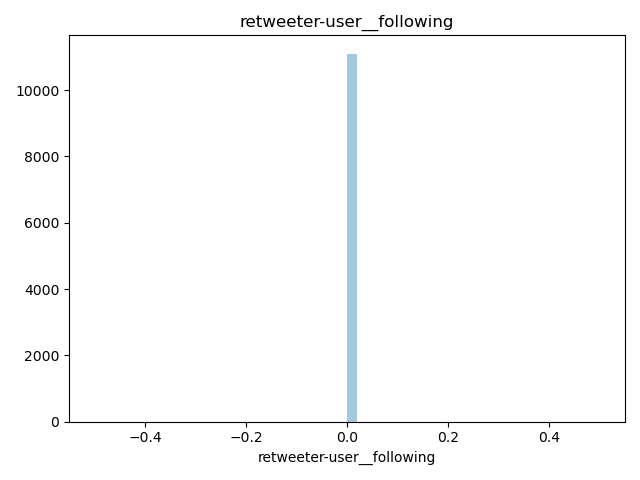
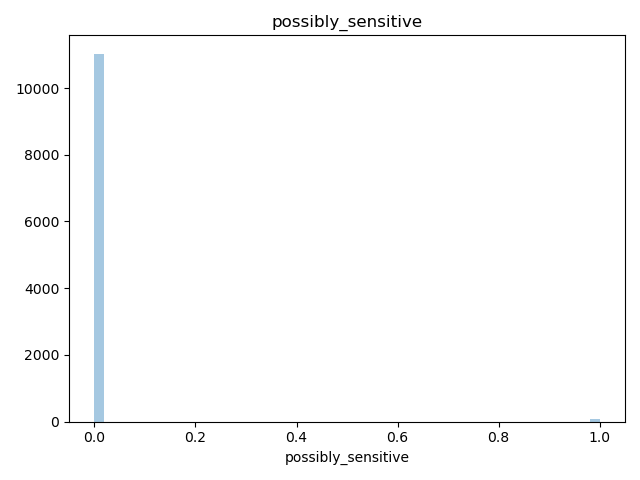
res\_x[**'viral'**] = json.retweeted\_status.agg(lambda x: None if type(x) is not dict else x[**"retweet\_count"**])  
res\_x[**'viral'**] = res\_x[**'viral'**].agg(lambda x: 1 if x >= res\_x[**'viral'**].median() else 0)

در کد بالا، مقدار res\_x[‘viral’] مشخص می‌کند که آیا توییت ما وایرال محسوب می‌شود یا خیر.

**یک نکته:** ستون‌هایی که در جدول بالا مشخص شده‌اند، تنها بخشی از ستون‌هایی هستند که پیش از نگارش کد و گزارش مشخص شده بودند. ستون‌هایی مثل protected (مشابه پروفایل[[12]](#footnote-12)‌های پرایوت در اینستاگرام[[13]](#footnote-13)) یا friends\_count یا following (تعداد کاربرهای دیگری که توسط کاربر دنبال می‌شوند) و... از ابتدا در بین ستون‌های مورد استفاده بودند. اما پس از مشاهده‌ی نمودار هیستوگرام[[14]](#footnote-14) و استفاده از تابع describe() مشخص شد که عده‌ای از آن‌ها همیشه مقدار خاصی دارند (مثل protected) یا مثل friends\_count، دقیقاً برابر مقدار فالوور های هر شخص هستند. به همین دلیل و برای جلوگیری از اطناب، این ستون‌های در جدول بالا نیامده‌اند.

## مرحله‌ی دوم: تشریح داده‌ها

همان‌طور که در نکته‌ی انتهایی صفحه‌ی قبل اشاره شد، تعدادی از ستون‌ها که در ابتدا موثر به نظر می‌رسیدند، پس از رسم هیستوگرام و استفاده از تابع describe()، حذف شدند. در زیر، دو نمونه از ستون‌هایی که به کمک نمودار و تابع یاد‌شده حذف شده‌اند، مشاهده می‌شود.



نمودار 1 - هیستوگرام ستون following

نمودار 2 - هیستوگرام ستون possibly\_sensitive

ستون نمودار اول به دلیل ناکافی‌بودن مقادیر possibly\_sensitive=1 و ستون نمودار دوم به دلیل صفربودن تمام مقادیر حذف شده‌اند. در متن کد، ستون‌های حذف‌شده به همراه مختصری از دلیل حذف آن‌ها به شکل کامنت[[15]](#footnote-15) آمده‌اند:

# All the values are 0  
# res\_x[str + "\_\_protected"] = user.agg(filledReader("protected", False))

## مرحله‌ی سوم: پیش‌پردازش داده‌ها

### تبدیل داده‌ها به مقادیر قابل پردازش

در این مرحله، قصد داریم تا داده‌های غیرعددی را به روش‌های مختلفی به داده‌های عددی تبدیل کنیم و داده‌های عددی موجود را هم به گونه‌ی دسته‌بندی کنیم که مدل‌سازی ما زمان کم‌تری برای پردازش نیاز داشته باشد.

به طور خلاصه، ستون‌های Boolean را به ستون‌های 0 و 1 تبدیل می‌کنیم، ستون‌های Float را به Int تبدیل می‌کنیم و ستون‌های Int که واریانس بسیار بالایی دارند را دسته‌بندی می‌کنیم. برای تبدیل ستون lang به مقادیر مناسب، از دستور get\_dummies() استفاده می‌کنیم و در نهایت، ستونِ created\_at را تبدیل به عمر اکانت (به سال) می‌کنیم.

در جدول زیر، از موارد ساده‌تر (مثل تبدیل Boolean به 0 و 1 یا تبدیل Float به Int) صرف نظر می‌کنیم و صرفاً مواردی را که نیاز به توضیح بیشتر دارند شرح می‌دهیم.

|  |  |  |  |
| --- | --- | --- | --- |
| اقدام پیش‌پردازشی | جنس ستون | نام ستون |  |
| دسته‌بندی بر حسب طول توییت (کوتاه=1، متوسط=2، بلند=3) (البته پس از حذف داده‌های غلط) | int | len | Tweet-related Information |
| تبدیل به سال | timestamp | created\_at | User-related Information |
| چارک‌بندی داده‌ها (البته پس از حذف داده‌های پرت) | int | favourites\_count |
| followers\_count |
| listed\_count |
| statuses\_count |

دسته‌بندی طول توییت، در خط زیر صورت می‌گیرد. تعداد کارکتر‌های مورد نیاز برای حضور در هر دسته هم در خط کامنت آن مشخص شده است.

# 1->Short Tweet(1-50) / 2->Medium Tweet(50-100) / 3->Long Tweet(100-140)  
res\_x[**'len'**] = res\_x[**'len'**].agg(lambda x: 1 if x < 51 else (2 if x < 101 else 3))

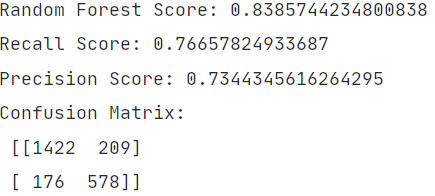
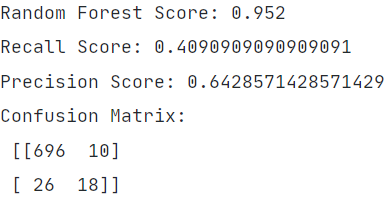
### حذف داده‌های پرت

فرآیند حذف داده‌های پرت دیتاست[[16]](#footnote-16) از چند روش مختلف انجام می‌شود که در زیر به ترتیب آمده‌اند:

1. حذف توییت‌هایی با طول بیشتر از 140 (حد مجاز طول توییت در زمان جمع‌آوری داده‌ها)
2. حذف ستون‌هایی که مقادیر int آن‌ها منفی است. (این گام به دلیل نبود مقادیر منفی در ستون‌ها تبدیل به کامنت شده است)
3. استفاده از روش 6-سیگما[[17]](#footnote-17) (برای ستون‌های favourites\_count، followers\_count، listed\_count و statuses\_count)

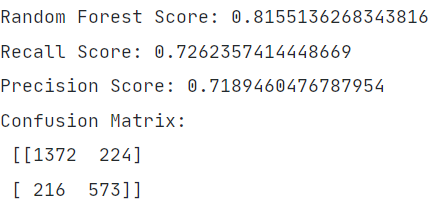
# 6-sigma method  
fence\_low = df\_in[column].mean() - df\_in[column].std() \* 3  
fence\_high = df\_in[column].mean() + df\_in[column].std() \* 3  
return df\_in[(df\_in[column] < fence\_high) & (df\_in[column] > fence\_low)]

کد بالا -که بخشی از تابع حذف داده‌های پرت است- دو سر بازه‌ی قابل قبول را مشخص و دیتافریمی[[18]](#footnote-18) شامل مقادیر بین این دو سر باز می‌گرداند. برای حذف داده‌های پرت از IQR هم استفاده کردیم اما به دلیل حذف بیش از حدِ داده‌ها، داده‌ی بسیاری باقی نمی‌ماند و مدل ما اُوِرفیت[[19]](#footnote-19) می‌شود. لذا در مدل نهایی از همان روش 6-سیگما استفاده کرده‌ایم.



تصویر 2 - دقت مدل RF با حذف داده‌ی پرت به روش 6-سیگما

تصویر 3 - دقت مدل RF با حذف داده‌ی پرت به روش IQR

پس از حذف داده‌های پرت، نوبت به Binning آن‌ها می‌رسد. برای این بخش، از روش ساده‌ی چارک‌بندی استفاده کرده‌ایم. داده‌های ستون‌های چهارگانه‌ی بالا که در چارک i-ام قرار دارند، با مقدار i جایگزین می‌شوند. استفاده از Binning زمان محاسبه‌ی ما را به شدت کاهش می‌دهد؛ اما از طرف دیگر باعث کاهش دقت مدل ما می‌شود. (به عنوان مثال، دقت مدل RF در صورت انجام Binning از 84% به 82% کاهش پیدا می‌کند.)

تصویر 4 - دقت مدل RF در صورت اعمال عملیات Binning

## مرحله‌ی چهارم: ارائه‌ی مدل جهت پیش‌بینی و ارزیابی آن

مراحل طی‌شده در این بخش، با تقریب بسیار زیادی مشابه تمرین‌های قبلی است. مدل‌های انتخابی ما برای این کار KNN، Bagging و Random Forest هستند. برای برآورد بهترین پارامترهای این مدل‌ها، از روش جستجو استفاده می‌کنیم. (به دلیل پیچیدگی بالای دیتاست) به عنوان مثال، برای برآورد پارامتر n\_estimator در مدل Random Forest، در ابتدا از 10 تا 210 با گام‌های 50تایی جلو می‌رویم و دقت‌ها را مقایسه می‌کنیم. اگر فرضاً بالاترین دقت متعلق به 160 شد، از 110 تا 210 با گام‌های 25 تایی جلو می‌رویم و این فرآیند را مدام تکرار می‌کنیم تا بازه‌ی مناسبی برای n\_estimator بیابیم. در مدل ما، این مقدار عمدتاً بین 140 و 155 قرار می‌گیرد.

دقت هر یک از مدل‌ها به همراه Confusion Matrix و پارامترهای بهینه در زیر آمده است:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The Best Parameters | Confusion Matrix | Recall Score | Accuracy Score |  |
| n\_neigbors=1  p=1  weights=uniform |  | ۶۶.۰٪ | ۷۷.۶٪ | **KNN** |
| n\_estimator=5  max\_samples=18 |  | ۶۵.۹٪ | ۷۴.۲٪ | **Bagging** |
| n\_estimator=145  max\_features=auto |  | ۷۵.۲٪ | ۸۴.۴٪ | **Random Forest** |

مشاهده می‌شود که دقت مدل Random Forest از KNN و آن نیز از مدل Bagging بیشتر است. به همین خاطر از مدل Random Forest استفاده می‌کنیم و پس از پیش‌بینی داده‌های test، ستون viral\_prediction را به فایل test اضافه می‌کنیم.

## مرحله‌ی پنجم: نتیجه‌گیری و تحلیل

برای این‌که تحلیل دقیقی از داده‌ها داشته باشیم، می‌توانیم از قابلیت‌های جانبی مدل‌ساز استفاده کنیم. آرایه‌ی feature\_importances\_[[20]](#footnote-20) که در کلاس مدل قرار گرفته است، درصد اهمیت هر یک از ستون‌ها را در پیش‌بینی نمایش می‌دهد. مقادیر این آرایه را در کنار نام ستون مربوط به خودش گذاشته، به ترتیب اهمیت سورت[[21]](#footnote-21) و در فایل اکسلی ذخیره می‌کنیم.

نمودار درصد اهمیت ستون‌ها در مدل‌سازی برای 25 ستون برتر به شکل زیر است:

نمودار 3 - درصدهای اهمیت 25 ستونِ پراهمیت در مدل Random Forest

با توجه به نمودار بالا، می‌توان گفت که **تعداد فالوورها، تعداد استاتوس‌ها، تعداد لیست‌هایی که کاربر عضو آن است، عمر اکانت کاربر و تعداد پست‌های لایک‌شده توسط هر کاربر (هم ریتوییت‌کننده و هم نویسنده‌ی توییت)** از عوامل درجه یک در تعیین وایرال‌شدن یک توییت هستند.

در درجه‌ی دوم تعداد کاربران منشن‌شده، تعداد برچسب‌ها، وریفاید‌بودن یا نبودن کاربر، طول توییت، نقل‌قول‌کردن یا نکردن و زبان انگلیسی از عوامل تاثیرگذار در وایرال‌شدن یا نشدن یک توییت هستند.

در درجه‌های بعدی که بسیار کم تاثیر هستند، باقی زبان‌ها مشاهده می‌شوند که در بین آن‌ها، استفاده از زبان فرانسه بیشترین شانس را برای وایرال‌شدن توییت دارد.

نمودار 4 – درصد تاثیر ستون‌های باقی‌مانده بر وایرال‌شدن توییت

1. Viral [↑](#footnote-ref-1)
2. Tweet [↑](#footnote-ref-2)
3. Tweeter [↑](#footnote-ref-3)
4. Like [↑](#footnote-ref-4)
5. Retweet [↑](#footnote-ref-5)
6. Follower [↑](#footnote-ref-6)
7. Feed [↑](#footnote-ref-7)
8. Mention [↑](#footnote-ref-8)
9. Hashtag [↑](#footnote-ref-9)
10. Metadata [↑](#footnote-ref-10)
11. Median [↑](#footnote-ref-11)
12. Profile [↑](#footnote-ref-12)
13. Instagram [↑](#footnote-ref-13)
14. Histogram [↑](#footnote-ref-14)
15. Comment [↑](#footnote-ref-15)
16. Dataset [↑](#footnote-ref-16)
17. 6-Sigma [↑](#footnote-ref-17)
18. Data frame [↑](#footnote-ref-18)
19. Overfit [↑](#footnote-ref-19)
20. https://scikit-learn.org/stable/auto\_examples/ensemble/plot\_forest\_importances.html [↑](#footnote-ref-20)
21. Sort [↑](#footnote-ref-21)