Ferdowsi university of Mashhad

Faculty of Engineering

computer engineering group

Repo Report on the final project of the Artificial intelligence course

Title

Sentiment analysis

Writers

Mohammad Raee

Ali MoghadasZade

Elahe Mottaghin

Teacher

Dr. Ahad Harati

2021-January

چکیده

در تشخیص احساسات متن و دسته بندی کردن متن ها بر اساس احساساتشون روش های متعددی وجود دارد که سه تا از روش های شناخته شده در موضوع classification مدل های Random Forest، LineatSVC و SVM با کرنل rbf است؛ که برای توییت هایی با محتوا نژاد پرستانه و نفرت آمیز از آنان استفاده کردیم بر روی آن مقایسه ای انجام دادیم. پس با تمیز کردن داده و استفاده از مدل word2vec و TF-IDF به میانگین وزن داری از بردار هر توییت رسیدیم و با مقایسه دقت این سه مدل بر روی داده مورد نظر به این نتیجه رسیدیم که Random Forest از همه نتیجه بهتری دارد و از طرفی دو روش دیگر تقریبا تمام توییت ها را غیرنژاد پرستانه و نفرت آمیز که درصد بیشتری از داده ها را به خود اختصاص داده است، تشخیص می‌دهند و عملا یادگیری خوبی نداشتند.

۱- مقدمه

تحلیل احساسات یا نظر کاوی به معنی کشف و یا شناخت احساسات مثبت و منفی مردم راجع به یک مسئله یا محصول در متون است . افزایش اهمیت Sentiment analysis با رشد رسانه های اجتماعی مانند نظر سنجی ها ، forum discussion ها ، وبلاگ ها ،توییتر ها و شبکه های اجتماعی همزمان شده است . سیستم های Sentiment analysis تقریبا در همه زمینه های تجاری و اجتماعی مورد استفاده قرار میگیرند ، زیرا نظرات و عقاید برای همه فعالیت های انسانی مهم بوده و تاثیر کلیدی بر رفتار ما دارند .

پروژه تحلیل احساسات متن به کمک الگوریتم های مختلف یادگیری ماشین با زبان پایتون انجام شده است. در این پروژه فرایند تشخیص احساسات در یک متن روی مجموعه داده های متن گفتاری انجام شده است. تحلیل و بررسی در طی پروژه با ترکیب چند الگوریتم یادگیری ماشین از جمله random forest ، svm with linear kernel و svm with rbf انجام شده است.

این پروژه در چهار بخش انجام شده است . بخش اول شامل پیش پردازش روی داده ها ، بخش دوم تبدیل کلمات به بردار (word2vec) ، بخش سوم شامل کاهش ویژگی و کلاس بندی داده ها و بخش چهارم نیز ارزیابی میباشد .

بخش اول : در این بخش به کمک کتابخانه nltk روی داده ها پیش پردازش انجام شده است و داده ها آماده پردازش شده اند.

پیش پردازش انجام شده شامل تبدیل حروف بزرگ به حروف کوچک ، حذف اعداد ، حذف خط فاصله ، حذف ادرس ها ، حذف کلمات کوچک و جایگذاری ۳ عبارت پی در پی با ۲ عبارت میباشد .

بخش دوم : در این بخش با کمک کتابخانه genism، داده های پیش پردازش شده را به بردارهای نظیر آن ها تبدیل شده است. در الگوریتم بکار رفته برای این بخش یعنی word2vec سعی شده تا حد ممکن ویژگی ها و روابط کلمات شبیه سازی شود .

بخش سوم : در این بخش ابتدا میزان اهمیت هر کلمه در متن به وسیله مولفه tf_idf مشخص شد . پس از آن به کمک بردار های هر داده ، برداری به کل متن اختصاص یافت و سپس مجموعه ای از این بردار ها (بردار های متن ها) در ارایه ای ذخیره شد تا روی آن کلاس بندی انجام شود . در این مرحله توسط ۳ نوع الگوریتم رایج برای classification یعنی random forest ، svm with linear kernel و svm with rbf عملیات کلاس بندی روی داده ها انجام شد .

بخش چهارم : در این بخش به ارزیابی نتایج بدست آمده برداخته شد . در نهایت برای هر یک از ۳ الگوریتم مذکور در بخش ۳ ، confusion matrix رسم شد و میزان دقت به وسیله ی معیار های accurity ، precision ، recall و f1-score ارزیابی شد که در نهایت الگوریتم random forest از سایرین بهتر ارزیابی شد .

۲- جمع آوری و بررسی داده ها

در انجام این پروژه از داده های متن گفتاری استفاده شده است. داده ها از سایت Kaggle.com انتخاب شده اند و به صورت مجموعه ای از متن توییت های افراد هستند.

لینک دریافت داده ها:

https://www.kaggle.com/arkhoshghalb/twitter-sentiment-analysis-hatred-speech

داده ها را به کمک کتابخانه pandas خوانده شده است. به عنوان نمونه چند سطر اول داده ها در ادامه آورده شده اند.

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation
...	...	...	...
31957	31958	0	ate @user isz that youuu?Ã°ÂŸÂ˜Â□Ã°ÂŸÂ˜Â□Ã°ÂŸÂ...
31958	31959	0	to see nina turner on the airwaves trying to...
31959	31960	0	listening to sad songs on a monday morning otw...
31960	31961	1	@user #sikh #temple vandalised in in #calgary,...
31961	31962	0	thank you @user for you follow

برای درک بهتر و شناخت نسبت به داده ها نمودار نشان دهنده تکرار داده ها در مجموعه داده ها(word cloud) رسم شده است. این نمودار به صورت زیر است.

نمودار تکرار داده ها در مجموعه داده ها

به کمک نمودار فوق متوجه می شویم که داده ها شامل تعداد زیادی حروف غیر الفبا هستند. همچنین کلمه «user» نیز در داده ها پر تکرار بوده است. با این بررسی متوجه می شویم که لازم است روی داده ها پیش پردازش انجام شود تا داده ها به حالت استاندارد و قابل خواندن برای ماشین تبدیل شوند. در ادامه مراحل و روش‌های لازم برای پیش پردازش متن مورد بررسی قرار می‌گیرند.

۳- پیش پردازش متن

بعد از جمع‌آوری متن(Collecting Text Data)، نرمال‌سازی داده‌های متنی جمع‌آوری شده انجام می شود. در ادامه روش های نرمال سازی انجام شده را معرفی می کنیم.

- حذف user@ از داده ها. زیرا این در همه داده ها تکرار شده است و در نتیجه نیز تاثیری نمی گذارد.
- تبدیل کردن تمامی حروف موجود در داده‌های متنی به حروف کوچک (Lowercase letters)
- حذف کردن اعداد از داده‌ها
- حذف کردن علائم نگارشی
- پاک کردن فضاهای خالی«white space» از متن
- حذف آدرس ها و لینک ها (حذف الگو ها به صورت ...\:http)
- حذف کلمات کوتاه(کلماتی که طول آن ها کمتر از ۳ است. از جمله ... ,the ,a ,is)
- حذف حروف تکراری در کلمات که به علت شیوه گفتاری به وجود آمده اند.(به عنوان مثال اگر به جای کلمه you نوشته شده باشد youuuu آن را به حالت درست تبدیل می کنیم.)

با انجام نرمال سازی های فوق داده ها به صورت زیر تغییر می کنند.

	id	label	tweet	cleanTweet
0	1	0	@user when a father is dysfunctional and is s...	when father dysfunctional selfish drags kids i...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks lyft credit cant cause they dont offer ...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	model love take with time
4	5	0	factsguide: society now #motivation	factsguide society motivation

پس از نرمال سازی متن لازم است جداسازی واژگان(Tokenization) انجام شود. این کار را نیز به کمک کتابخانه nltk انجام می دهیم. داده ها پس از جداسازی واژگان به صورت زیر خواهند بود.

```
0     [when, father, dysfunctional, selfish, drags, ...
1     [thanks, lyft, credit, cant, cause, they, dont...
2                                [bihday, your, majesty]
3                      [model, love, take, with, time]
4                    [factsguide, society, motivation]
```

پس از جدا سازی واژگان، کلمات بی اثر (stop word) را از داده ها حذف می کنیم. کلمات بی اثر شایع‌ترین کلمات استفاده شده در یک زبان هستند؛ به عنوان نمونه، کلماتی نظیر The ، On، Is، All و a، کلمات بی اثر در زبان انگلیسی محسوب می‌شوند. از آنجایی که این کلمات بار معنایی خاصی ندارند و محتوای معنایی قابل توجهی را انتقال نمی‌دهند، از داده حذف شده اند.

در این مرحله لازم است کلمات به حالت ریشه کلمه برگردند به عنوان مثال کلمه reading به کلمه read و یا کلمه well به کلمه good تبدیل می شود. برای این کار نیز از کتابخانه nltk استفاده می کنیم. ابتدا به کمک الگوریتم Porter اجرا شده است که این الگوریتم بخش‌های عطفی (Inflectional) و مورفولوژیک (Morphological) را از انتهای کلمات حذف می‌کند؛ سپس به کمک الگوریتم Lemmatization ریشه یابی را انجام می دهیم که این الگوریتم از پایگاه‌های دانش لغوی (Lexical Knowledge Base) جهت پیدا کردن شکل ریشه‌ای صحیح کلمات استفاده می‌کند.

بعد از انجام تمام مراحل قبل واژگان جداسازی شده را دوباره به هم پیوسته می کنیم و به صورت جمله در می آوریم. نتیجه پردازش داده ها تا به اینجا به صورت زیر است.

	id	label	tweet	cleanTweet
0	1	0	@user when a father is dysfunctional and is s...	father dysfunct selfish drag kid dysfunct
1	2	0	@user @user thanks for #lyft credit i can't us...	thank lyft credit cant caus dont ofer whelchai...
2	3	0	bihday your majesty	bihday majesti

	id	label	tweet	cleanTweet
3	4	0	#model i love u take with u all the time in ...	model love take time

حال برای بررسی نتیجه کار تا به اینجا دوباره نمودار نشان دهنده تکرار داده ها در مجموعه داده ها رسم شده است. این نمودار به صورت زیر است.

نمودار تکرار داده ها در مجموعه داده های پردازش شده

به کمک نمودار فوق متوجه می شویم که دیگر در داده های پردازش شده حروف زائد و کلمات بی معنی نیست؛ همچنین متوجه می شویم که کلمات love، make، today، life و smile در داده ها بسیار پر تکرار اند.

۴- تبدیل کلمات به بردار(word2vec)

برای بسیاری از روش ها پردازش متن، نیاز به نمایش عددی کلمات و متون داریم تا بتوانیم از انواع روش های عددی حوزه یادگیری ماشین مانند اکثر الگوریتم های دسته بندی روی لغات و اسناد استفاده کنیم .
فرض کنید فرهنگ لغتی داریم با N کلمه و لغت که به ترتیب الفبایی مرتب شده اند و هر لغت یک مکان مشخص در این فرهنگ لغت دارد. حال برای نمایش هر کلمه، برداری در نظر می گیریم با طول N که هر خانه آن، متناظر با یک لغت در فرهنگ لغت ماست که برای راحتی کار فرض می کنیم شماره آن خانه بردار،

همان اندیس لغت مربوطه در این فرهنگ لغت خواهد بود. با این پیش فرض، برای هر لغت ما یک بردار به طول N داریم که همه خانه های آن بجز خانه متناظر با آن لغت صفر خواهد بود. در خود ستون متناظر با لغت عدد یک ذخیره خواهد شد(One-Hot encoding). با این رهیافت، هر متن یا سند را می توان با یک بردار نشان داد که به ازای هر کلمه و لغتی که در آن به کار رفته است، ستون مربوطه از این بردار برابر تعداد تکرار آن لغت خواهد بود و تمام ستون های دیگر که نمایانگر لغاتی از فرهنگ لغت هستند که در این متن به کار نرفته اند، برابر صفر خواهد بود.

به این روش نمایش متون، صندوقچه کلمات (bags of words) می گوییم که بیانگر این است که برای هر لغت در صندوقچه یا بردار ما، مکانی درنظر گرفته شده است.
این روش در واقع از نخستین ایده های تبدیل کلمات به بردار بود. اما پس از مدتی مشخص شد که این روش مشکل عمده ای دارد. مثلاً اگر فرهنگ لغت ما صد هزار لغت داشته باشد، به ازای هر متن ما باید برداری صد هزارتایی ذخیره کنیم که هم نیاز به فضای ذخیره سازی زیادی خواهیم داشت و هم پیچیدگی الگوریتم ها و زمان اجرای آنها را بسیار بالا می برد. مشکل دیگر در آن است که در چنین تبدیلی از متن به بردار، اطلاعات متن را نمی توان مدل سازی کرد. یعنی شما روابط میان کلمات را از این بردار ها نمی توانید استخراج کنید. برای مثال در دنیای واقعی می دانیم که رابطه پدر و پسر همانند رابطه مادر و دختر است پس در شبیه سازی ما به بردار نیز این انتظار می رود تا نسبت بردار پدر به بردار پسر همانند نسبت بردار مادر به بردار دختر باشد اما در این روش اینگونه نبود. بنابراین به سراغ روش دیگری از آموزش به نام word2vec میرویم که توسط گوگل در سال ۲۰۱۳ بیان شد و روشی بسیار موفق است. این روش با دو رویکرد اجرا می شود .

۱-۴- روش کیفیت لغات پیوسته

در روش کیفیت لغات پیوسته (CBOW)، ابتدا به ازای هر لغت یک بردار با طول مشخص و با اعداد تصادفی (بین صفر و یک) تولید می شود. سپس به ازای هر کلمه از یک سند یا متن، تعدادی مشخص از کلمات بعد و قبل آن را به شبکه عصبی می دهیم (به غیر از خود لغت فعلی) و با عملیات ساده ریاضی، بردار لغت فعلی را تولید می کنیم (یا به عبارتی از روی کلمات قبل و بعد یک لغت، آنرا حدس می زنیم) که این اعداد با مقادیر قبلی بردار لغت جایگزین می شوند. زمانی که این کار بر روی تمام لغات در تمام متون انجام گیرد، بردارهای نهایی لغات همان بردارهای مطلوب ما هستند.

در واقع این روش یک فرض اولیه از بردار هر کلمه دارد و با پیمایش متن، این فرض را رفته رفته بهبود می بخشد و این بهبود برای هر کلمه، از روی کلمات اطراف آن در متن است. در نهایت پس از پایان اجرای الگوریتم برداری که برای هر کلمه بدست آمده است رابطه ی تنگانگی با بردار های کلمات نزدیک به آن کلمه دارد. با این ایده میتوان روابط بن کلمات را نیز در قالب بردار مدل سازی کرد .

۴-۲- روش skip-gram

در این روش برعکس روش قبل کار می کند به این صورت که بر اساس یک لغت داده شده ، می خواهد چند لغت قبل و بعد آن را تشخیص دهد و با تغییر مداوم اعداد بردارهای لغات، نهایتا به یک وضعیت باثبات می رسد که همان بردارهای مورد بحث ماست.

در واقع در این روش از روی یک کلمه سعی می شود کلمات اطراف آن حدس زده شود و بردار های کلمات اطراف آن رفته رفته تکامل می یابند . بنابراین در فاز آموزش در نقطه مقابل روش کیفیت لغات پیوسته قرار دارد اما همانند روش قبل چون بردار هر کلمه از روی کلمات اطراف آن تکامل میابد ، بنابر این روش بسیار کار آمدی برای مدل سازی روابط کلمات در بردار هاست.

در هر دو این روش ها یک پنجره در نظر گرفته می شود که الگوریتم بر اساس سایز این پنجره روی متن حرکت می کند و هر بار به انداز سایز کلمات متوالی را خوانده و پردازش های گفته شده را روی بردار ها جهت تکامل آن انجام می دهد .

۴-۳- پیاده سازی word2vec در این پروژه

در این پروژه word2vec بر اساس الگوریتم اول یعنی روش کیفیت لغات پیوسته پیاده سازی شده است. برای این کار از کتابخانه gensim درپایتون استفاده می شود .

در اینجا ۴ مولفه برای ورودی به word2vec داده می شود :

- Num-feature: که در حقیقت ابعاد کلمات در تبدیل به بردار است .
- Min-word-cout: همان تعداد حداقل برای یک کلمه است برای اینکه بخواهد در مدل آموزش داده شود .
- Num-workers: همان تعداد thread های آموزش داده هاست .
- Context: همان سایز پنجره ای است که دربالا توضیح داده شد .

- Downsampling: یک نمونه برداری روی داده هاست که بر این اساس تعداد داده های انتخابی رو به پایین کاهش می دهیم .

نتایج بررسی مقادیر مختلف برای ورودی های مدل در پروژه به صورت زیر است:

- مقدار اول را مقادیر ۱۰۰، ۲۰۰، ۳۰۰ و ۴۰۰ دادیم و با توجه به اندازه مدل تغییر خاصی نداشت اما از لحاظ تاثیرش برای contex های مختلف ، مقدار ۴۰۰ از همه بهتر بود بنابراین مقدار مولفه اول را ۴۰۰ قرار می دهیم.
- برای مقدار دوم که با توجه به کوچیک بودن دیتا ست همان ۱ را قرار دادیم .
- برای مورد سوم هم به طور معمول بیش از نصف تعداد هسته های cpu قرار داده می شود که در اینجا ۶ را قرار دادیم .
- برای مورد چهارم هم مقادیر ۵ و ۸ و ۱۰ بررسی شد که برای هر کدام چند نمونه کوئری خاص را که در کد نیز قرار داده شده بررسی کردیم که بین آن ها نتایج ۱۰ از همه بهتر بود .
- مورد آخر هم که به طور معمول همان ۰.۰۰۱ را باتوجه به اندازه کورپس قرار دادیم .

پس از آن نوبت به آموزش مدل بر اساس داده های train است که مطابق آنچه در کد قرار داده شد در ورودی آن مدل ساخته شده به همراه مولفه epochs داده شد .

مولفه epochs در واقع تعداد دفعات تکرار را نشان می دهد. در حالت کلی دفعات تکرار را به صورت دستی وارد نمی کنند مگر در حالت های خاص که خارج از بحث ماست. بنابراین از این مولفه برای تکرار استفاده شد. در این ارزیابی بدان مقادیر بین مقادیر۳۰، ۵۰، ۶۰، ۷۰، ۹۰ داده شد که با توجه به نتایج حاصل از چند نمونه کوئری که در کد قرار داده شد، مقدار ۶۰ از بقیه بهتر بود .

در ادامه به کاهش ابعاد بردار های با PCA می پردازیم .

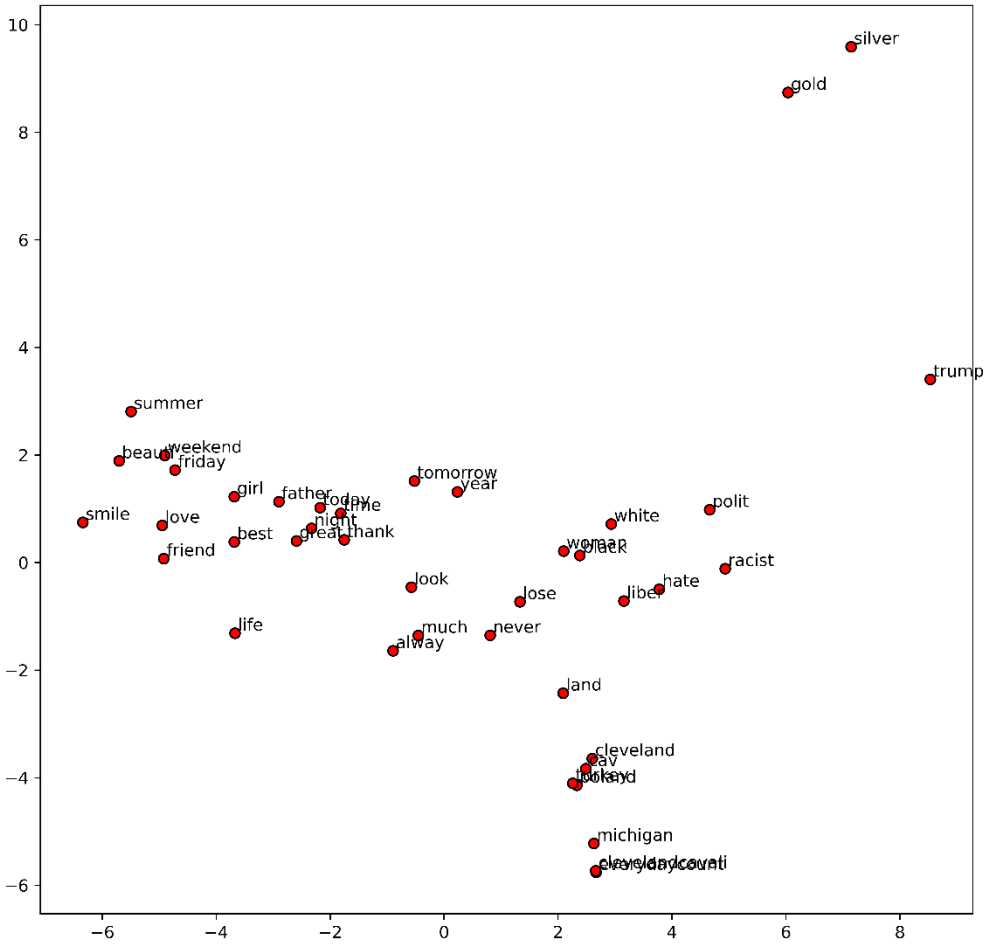
۴-۴- آنالیز مولفه اصلی (principal component analysis) یا همان pca چیست ؟

یکی از کاربردهای اصلی آنالیز مولفه اصلی در عملیات کاهش ویژگی (dimensionality reduction)است. pca همانطور که از نامش پیداست می تواند مولفه های اصلی را شناسایی کند و به ما کمک می کند تا به جای اینکه تمامی ویژگی ها را مورد بررسی قرار دهیم، یک سری ویژگی هایی را که ارزش بیشتری دارند ، تحلیل کنیم. در واقع pca آن ویژگی هایی را که ارزش بیشتری فراهم می کنند، برای ما استخراج می کند.

پیاده سازی آن در کد در قالب تابع display_pca_scatterplot صورت گرفته است. این تابع همانگونه که گفته شد، مدل آموزش داده شده را در وروی گرفته و بردار های کاهش یافته (از لحاظ ابعاد) برای هر کلمه را در خورجی می دهد .

۱-۴-۴- بصری سازی نتایج بخش دوم و ارزیابی ها

در ادامه نمونه ای از نتایج بدست آمده آورده شده است. در این نمونه سعی بر آن بوده است تا نمونه هایی انتخاب شوند که نمودار واضح تر و بصری سازی با توصیف بهتری همراه باشد.



با توجه به نمونه بالا مشاهده می شود که کشور ها گوشه پایین سمت راست قرار گرفته اند، زمان ها نزدیک هم هستند و الفاظ خوب و اشارات به خانواده که زمینه نژادپرستانه ندارند در گوشه چپ پایین قرار گرفته اند و الفاظ نژاد پرستانه به وسط و در محدوده کلمه ترامپ کشیده شده اند.کلمه hate در وسط افتاده که نقطه مقابل کلمه love است که در گوشه پایین سمت چپ قرار دارد.

۴-۵- term frequency-invers document

tf-idf به معنای فراوانی وزنی یک کلمه کلیدی است. این مولفه صرفا میزان تکرار یک کلمه یا عبارت را در متن نشان نمی دهد، بلکه هدف آن نشان دادن اهمیت کلمه است.

اما سوالی که مطرح است اینکه tf-idf چگونه اهمیت کلمه در متن را بدست می آورد؟ این معیار اهمیت کلمه مورد نظر را از طریق مقایسه تعداد تکرار کلمه در متن کوچک تر، با تعداد تکرار کلمه در مجموعه ای بزرگ تر بدست می آورد.

برای مثال فرض کنید دو کلمه سهام و سبد سهام در یک متن داریم. در اینجا اگر صرفا تعداد تکرار را در نظر بگیریم بدیهتا تعداد تکرار کلمه سهام بزرگتر یا حداقل مساوی با کلمه سبد سهام است. چون به ازای هر عبارت سبد سهام که می بینیم یک کلمه سهام هم دیده می شود. اما اگر tf idf را بررسی کینم خواهیم دید اهمیت عبارت سبد سهام بیشتر از سهام است، این یعنی اگر در متنی کلمه سبد سهام تکرار شود کلمه مهمی است در حالی که آستانه مهم در نظر گرفتن کلمه سهام بسیار بالاتر است.(برای انکه کلمه سهام را در متنی مهم در نظر بگیریم باید به تعداد بیشتری بار تکرار شود)

بنابر توضیحات داده شده واضح است که پیاده سازی این مولفه روی کلمات مدل و بدست آوردن اهمیت آن ها می تواند در بهبود ارزیابی ها و نمونه برداری ها کمک بسیاری کند .

پیاده سازی این الگوریتم در کد توسط تابع TfidfVectorizer انجام شده است. اما کاربرد این مولفه در چیست؟ در این مرحله نوبت به اختصاص یک بردار به کل توییت است. این کار توسط دو تابع انجا می شود. انجام می شود .

۱- make_feature_vec در این تابع میانگین وزن دار از بردار های کلمات درون متن گرفته می شود. در اینجا وزن همان tf-idf است که در مرحله قبل بدان اشاره شد .

۲- get_avg_feature_vec در این تابع نیز برای تمامی متن ها تابع اول صدا زده می شود. و نتایج در یک آرایه ذخیره میشود تا برای classification که در ادامه توضح می دهیم، قابل استفاده باشد.

۵- طبقه بندی (classification)

در این مرحله به سراغ طبقه بندی داده های پردازش شده می رویم. ابتدا تعریفی از طبقه بندی ارائه می کنیم. برای درک بهتر میتوانیم از یک مثال آغاز کنیم و کلاس بندی و مفاهیم آن را در قالب این مثال بیان می کنیم:

فرض کنید شما مدیریت یک بانک را بر عهده دارید و قرار است به ۱۰۰ هزار مشتری وام بدهید . طبیعتا به افرادی وام می دهید که شاخص هایی دارند که احتمال برگرداندن وام توسط آن ها بالا باشد . مثلا این که فرد خانه دارد یا نه؟ دارای اتومبیل هست یا خیر؟ حقوق دریافتی وی چقدر است؟ و ...

خب حال فرض کنید به ۱۰ هزار نفر وام داده اید که برخی از آنها بازگردانده اند و برخی خیر . بنابراین میتوان افراد را به دو کلاس طبقه بندی کرد . شما می توانید جدولی ایجاد کنید که هر سطر آن متناظر با یک فرد باشد که وام گرفته است و هر ستون های آن نیز متناظر با ویژگی های مذکور فرد (خانه ، اتومبیل ، حقوق و ...) باشد و ستون آخر هم بیانگر ان است که فرد وام خود را پس داده یا خیر . خب این جدول یک ماتریس است که در آن به هر سطر یک رکورد (sample یا tuple) گفته میشود . هر ستون نمایشگر یک ویژگی (feature) است . به این ویژگی ها بعد (dimention) گفته می شود و هر کدام از رکورد ها به یک دسته (class) تعلق دارند. (در اینجا دو کلاس داریم : وام را پس داده یا پس نداده است)

به طور کلی به مسئله هایی که ستون طبقه (class) داشته باشند، مسائل طبقه بندی (classification) گفته می شود. این دسته از مسائل به یادگیری با ناظر (supervised learning) نیز معروف است. چون در واقع یک ناظر وجود دارد که ستون آخر را برای ما بر چسب زنی کند.(مثلا در اینجا مدیر بانک تعداد مشخصی از مشتریان را برای ما بر چسب زنی کرده است)

الگوریتم های طبقه بندی مانند (random forest, svm with linear kernel, svm with rbf) در واقع ماتریسی مانند ماتریس بالا را دریافت می کنند و از این ماتریس و ویژگی های آن، الگوریم موجود در هر کلاس را یاد می گیرند . سپس اگر یک نمونه ی جدید – که طبقه ی آن را نمی دانیم – به الگوریتمی که یادگرفته است اضافه شود، این الگوریتم می تواند این نمونه ی جدید را به طبقه های احتمالا درست طبقه بندی کند .

حال به سراغ الگوریتم های طبقه بندی می رویم. در این پروژه ۳ الگوریتم برای طبقه بندی استفاده شده که در ادامه به معرفی و پیاده سازی هر یک می پردازیم .

۵-۱- الگوریتم جنگل تصادفی (Random Forest)

برای درک چگونگی عملکرد جنگل تصادفی، ابتدا باید الگوریتم «درخت تصمیم» (decision tree) که بلوک سازنده جنگل تصادفی است را آموخت. انسان‌ها همه روزه از درخت تصمیم برای تصمیم‌گیری‌ها و انتخاب‌های خود استفاده می‌کنند، حتی اگر ندانند آنچه که از آن بهره می‌برند نوعی الگوریتم یادگیری ماشین است. برای شفاف کردن مفهوم الگوریتم درخت تصمیم، همانند بخش قبل از یک مثال روزمره یعنی پیش‌بینی حداکثر درجه حرارت هوای شهر برای روز بعد (فردا) استفاده می‌شود.

فرض کنید می خواهیم درجه حرارت شهر سیاتل در که در ایالت واشینگتون واقع شده است را برای فردا پیش بینی کنیم . برای پاسخ دادن به پرسش ساده اینکه «درجه حرارت فردا چقدر است ؟» نیاز به کار کردن روی یک سری پرسش هاست . ابتدا با یک درجه حرارت پیشنهادی که بر اساس دانش اولیه (domain knowledge) انتخاب شده، پیش می رویم . فرض کنید پیشنهاد اولیه بازه بین ۳۰ تا ۷۰ درجه فارنهایت باشد. در ادامه از طریق یک مجموعه پرسش و پاسخ بازه موجود را به نحوی محدود می کنیم که بتوان یک پیش بینی به اندازه کافی مطمئن داشت. سوالی که مطرح می شود این است که چه پرسش و پاسخی مناسب تر است؟ مسلما پرسشی که مرتبط تر باشد، اطلاعات بیشتری به ما می دهد. در اینجا چون آب و هوا با زمان مرتبط است پرسش اینکه الان چه فصلی از سال است می تواند بسیار مفید باشد . از آنجا که در اینجا پاسخ فصل زمستان است پس می توان بازه خود را به ۳۰-۵۰ محدود کرد . سوال خوب بعدی که می توان پرسید این است که «میانگین حداکثر درجه حرارت در این روز بر اساس اطلاعات تاریخی چقدر بوده؟». برای سیاتل در ۷ دسامبر، پاسخ ۴۶ درجه است. این کار امکان محدود کردن طیف به ۴۰-۵۰ را فراهم می‌کند . اما همچنان دو سوال برای پیش بینی کافی نیست زیرا امسال امکان دارد نسبت به میانگین سال‌های پیشین گرم‌تر یا سردتر باشد. بنابراین، نگاهی به حداکثر درجه حرارت امروز انداخته می‌شود تا مشخص شود که آیا امسال هوا نسبت به سال‌های پیش سردتر است یا گرم‌تر. اگر درجه حرارت امروز ۴۳ درجه باشد، نسبت به سال گذشته اندکی سردتر بوده و این یعنی فردا نیز امکان دارد درجه حرارت اندکی از میانگین تاریخی کمتر باشد.

در این نقطه می‌توان با اطمینان کامل پیش‌بینی کرد که بیشینه درجه حرارت فردا ۴۴ درجه است . بنابراین، برای رسیدن به تخمین مناسب، از یک مجموعه پرسش استفاده می‌شود و هر پرسش دامنه مقادیر پاسخ ممکن را محدودتر می‌کند تا اطمینان لازم برای انجام پیش‌بینی فراهم شود. این فرآیند تصمیم‌گیری هر روز در زندگی انسان‌ها تکرار می‌شود و تنها پرسش‌ها هستند که متناسب با نوع مساله از فردی به فردی دیگر تغییر می‌کنند. همانطور که مشاهده شد، در یک درخت تصمیم، کار با یک حدس اولیه بر مبنای دانش فرد از جهان آغاز و با دریافت اطلاعات بیشتر، این حدس پالایش می‌شود. طی این فرایند و به تدریج ، به گردآوری داده‌ها پایان داده و تصمیمی اتخاذ می‌شود که در اینجا پیش‌بینی بیشینه درجه حرارت است.

درخت تصمیم مطابق آنچه گفته شد، بهترین پرسش‌هایی که باید برای رسیدن به صحیح‌ترین تخمین طرح کند را محاسبه می‌کند. هنگامی که از مدل خواسته می‌شود برای روز بعد پیش‌بینی کند، باید به آن داده‌های مشابهی (ویژگی‌ها) با آنچه در طول آموزش داده شده ارائه شود، تا مدل تخمین را بر پایه ساختاری که آموخته انجام دهد. همانطور که انسان‌ها با استفاده از مثال‌ها می‌آموزند، درخت تصمیم نیز از طریق تجربه کردن یاد می‌گیرد، با این تفاوت که هیچ دانش پیشینی برای استفاده در مساله ندارد. آنچه بیان شد مفهوم سطح بالای یک درخت تصمیم است. در واقع فلوچارتی از پرسش‌ها که منجر به انجام یک پیش‌بینی می‌شوند، درخت تصمیم را شکل می‌دهند.

خب حال که درخت تصمیم را شناختیم، اکنون، جهشی بزرگ از یک درخت تصمیم تنها به جنگل تصادفی به وقوع می‌پیوندد . حال می خواهیم به این پرسش پاسخ دهیم که جنگل تصادفی چیست ؟

جنگل تصادفی یک مدل یادگیری نظارت شده است و یاد میگیرد که در فاز آموزش (training) داده ها را (که برای مثال در اینجا درجه حرارت امروز ، میانگین تاریخی و ...) به خروجی ها (که در اینجا حداکثر درجه حرارت فردا) نگاشت کند . در طول آموزش دو نوع اطلاعات باید به مدل داده شود :

۱- داده‌هایی به مدل داده می‌شوند که مرتبط با دامنه مساله هستند (در مثال ما درجه حرارت روز قبل، فصل سال و میانگین تاریخی)
۲- مقدار صحیحی (که در این مثال بیشینه درجه حرارت فردا است) که مدل باید بیاموزد تا بتواند پیش‌بینی کند .

یکی از مزایای جنگل تصادفی قابل استفاده بودن آن ، هم برای مسائل دسته بندی و هم رگرسیون است که غالبا سیستم های یادگیری ماشین کنونی را تشکیل می دهند.

۱-۱-۵- چگونگی عملکرد جنگل تصادفی

جنگل تصادفی همان گونه که بیان شد یک الگوریتم یادگیری نظارت شده محسوب میشود. همانطور که از نام آن مشهود است این الگوریتم جنگلی را به طور تصادفی میسازد . در ابتدا باید به چند سوال پاسخ داد :

۱- منظور از جنگل در این الگوریتم چیست ؟ جنگل ساخته شده در واقع مجموعه ای از درخت های تصمیم (decision tree) است .
۲- روش مورد استفاده برای ساخت این جنگل از درخت های تصمیم چیست ؟ در واقع کار ساخت جنگل با استفاده از درخت ها اغلب به روش کیسه گذاری (bagging) انجام میشود .

۳- ایده اصلی روش کیسه گذاری چیست ؟ ایده اصلی آن است که ترکیبی زا مدل های یادگیری ، نتایج کلی را افزایش میدهد . به بیان ساده جنگل تصادفی در روش کیسه گذاری چندین درخت تصمیم ساخته و آن ها را با یکدیگر ترکیب میکند تا پیش بینی های صحیح و پایداری حاصل شود .

در اینجا عملکرد جنگل تصادفی برای انجام « دسته بندی » (classification) را تشریح میکنیم :

مجددا با مثالی این عملکرد را تشریح میکنیم . فرض کنید اندرو میخواهد تصمیم گیری کند که برای یک سفر تفریحی یکساله به کدام مکان‌ها سفر کند. او از مردمی که او را می‌شناسند درخواست می‌کند که پیشنهادات خود را بگویند. از این رو، ابتدا به سراغ دوست قدیمی خود می‌رود، دوست اندرو از او می‌پرسد که در گذشته به کجا سفر کرده و آیا آن مکان را دوست داشته یا خیر. در نهایت بر اساس این پرسش و پاسخ، چند مکان را به اندرو پیشنهاد می‌دهد. این رویکرد متداولی است که الگوریتم درخت تصمیم دنبال می‌کند .

دوست اندرو با استفاده از پاسخ‌های او، قوانینی را برای هدایت تصمیم خود مبنی بر اینکه چه چیزی را پیشنهاد کند می‌سازد. پس از آن، اندرو شروع به پرسیدن سوالات بیشتر و بیشتری از دوست خود برای دریافت پیشنهاد از او می‌کند، بنابراین دوست اندرو نیز پرسش‌های متفاوتی را می‌پرسد که می‌تواند بر اساس آن‌ها توصیه‌هایی را به او ارائه کند. در نهایت، اندرو مکان‌هایی که بیشتر به او توصیه شده‌اند را انتخاب می‌کند. این رویکرد کلی الگوریتم جنگل تصادفی است . در واقع دوست اندرو و هر کسی او از وی درخواست پیشنهاد میکند ، مانند درخت تصمیم عمل میکند و اندرو از نتایج این درخت های تصمیم برای نتیجه گیری خود استفاده میکند.

۲-۱-۵- اهمیت ویژگی ها

از مهم ترین خصوصیات جنگل تصادفی این است که اندازه گیری اهمیت نسبی هر ویژگی روی پیش بینی در آن آسان است. کتابخانه ای که در این پروژه برای این کار استفاده شده «sklean» است . این ابزار، اهمیت یک ویژگی را با نگاه کردن به تعداد گره‌های درخت که از آن ویژگی استفاده می‌کنند، اندازه‌گیری کرده و ناخالصی را در سرتاسر درخت‌های جنگل کاهش می‌دهد.

از طریق بررسی اهمیت ویژگی‌ها، کاربر می‌تواند تصمیم بگیرد که کدام ویژگی‌ها را ممکن است حذف کند، با توجه به اینکه به طور کلی و یا به اندازه کافی در فرآیند تصمیم‌گیری نقش ندارند. این مساله حائز اهمیت

است زیرا، یک قانون کلیدی در یادگیری ماشین آن است که هرچه ویژگی‌ها بیشتر باشند، احتمال آنکه مدل دچار بیش پردازش (overfitting) و یا کم پردازش (underfitting) شود وجود دارد.

۳-۱-۵- هایپر پارامترها

هایپر پارامتر ها در جنگل تصادفی برای افزایش قدرت پیش بینی مدل و یا سریع تر کردن آن مورد استفاده قرار می گیرند.

۴-۱-۵- پیاده سازی جنگل تصادفی در پروژه

این الگوریتم برای کلاس بندی مدل ها در پروژه پیاده سازی شده است. پیاده سازی آن به این صورت بوده است که ابتدا با تابع tarian_test_split داده ها به دو بخش آموزش و مجموعه صحت سنجی جدا سازی شدند؛ سپس از طریق تابع modelResult نتایج پیش بینی های مدل گزارش شده است. این تابع در ورودی خود مقدار پیش بینی شده برای هر داده آزمون را می گیرد. در این تابع دسته بندی به دو گروه مثبت و منفی انجام شده است و گروه هایی که برای ارزیابی مورد بررسی قرار می گیرند شامل چهار گروه tn, fp, fn, tp می باشد. پس از آن توسط تابع randomForest الگوریتم جنگل تصادفی اجرا می شود. سپس توسط تابع score نتیجه ارزیابی آن با دقت ۰/۹۴ گزارش می شود. توضیحات مربوط به هر یک از گروه های مذکور و گزاش ارزیابی این روش در بخش ارزیابی ها به تفضیل توضیح داده خواهد شد .

۵-۱-۵- معایب و مزایا این روش

در اینجا به نظر می رسد خوب است تا پس از بررسی نتایج حاصل از پروژه، اندکی از مزایا و معایبی که با آن برخورد کردیم بیان کنیم.

در مورد مزایا به جز اینکه میدانیم الگوریتم جنگل تصادفی هم برای رگرسیون و هم برای دسته بندی قابل استفاده است، دو مزیت دیگر که برای ما در این پروژه مشهود بود را بیان می کنیم. نخست آن که جنگل تصادفی الگوریتمی بسیار مفید و با استفاده آسان محسوب می‌شود، زیرا هایپر پارامترهای پیش‌فرض آن اغلب نتایج پیش‌بینی خوبی را تولید می‌کنند. همچنین، تعداد هایپرپارامترهای آن بالا نیست و درک آن‌ها آسان است و به عنوان مزیت دیگر می توان گفت که آموزش دادن آن بسیار سریع انجام می شود .

محدودیت اصلی جنگل تصادفی آن است که تعداد زیاد درخت ها می‌توانند الگوریتم را برای پیش‌بینی‌های جهان واقعی کند و غیر موثر کنند. به طور کلی، آموزش دادن این الگوریتم‌ها سریع انجام می‌شود، اما پیش‌بینی کردن پس از آنکه مدل آموزش دید، اندکی کند به وقوع می‌پیوندد. یک پیش‌بینی صحیح‌تر نیازمند درختان بیشتری است که منجر به کندتر شدن مدل نیز می‌شود.

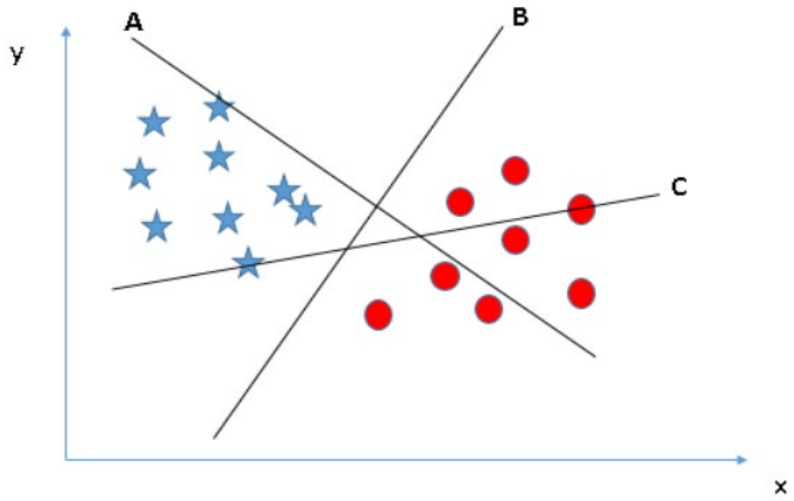
۲-۵- ماشین بردار پشتیبان (SVM)

ماشین بردار پشتیبان «یک الگوریتم نظارت‌شده یادگیری ماشین است که هم برای مسائل طبقه‌بندی و هم مسائل رگرسیون قابل استفاده است؛ با این حال از آن بیشتر در مسائل طبقه‌بندی استفاده می‌شود. در الگوریتم SVM ، هر نمونه داده را به عنوان یک نقطه در فضای n-بعدی روی نمودار پراکندگی داده ها ترسیم کرد (nتعداد ویژگی‌هایی است که یک نمونه داده دارد) و مقدار هر ویژگی مربوط به داده‌ها، یکی از مؤلفه‌های مختصات نقطه روی نمودار را مشخص می‌کند. سپس، با ترسیم یک خط راست، داده‌های مختلف و متمایز از یکدیگر را دسته‌بندی می‌کند . به بیان ساده، بردارهای پشتیبان در واقع مختصات یک مشاهده منفرد هستند. ماشین بردار پشتیبان مرزی است که به بهترین شکل دسته‌های داده‌ها را از یکدیگر جدا می‌کند.

۱-۲-۵- الگوریتم ماشین بردار پشتیبان چگونه کار می کند؟

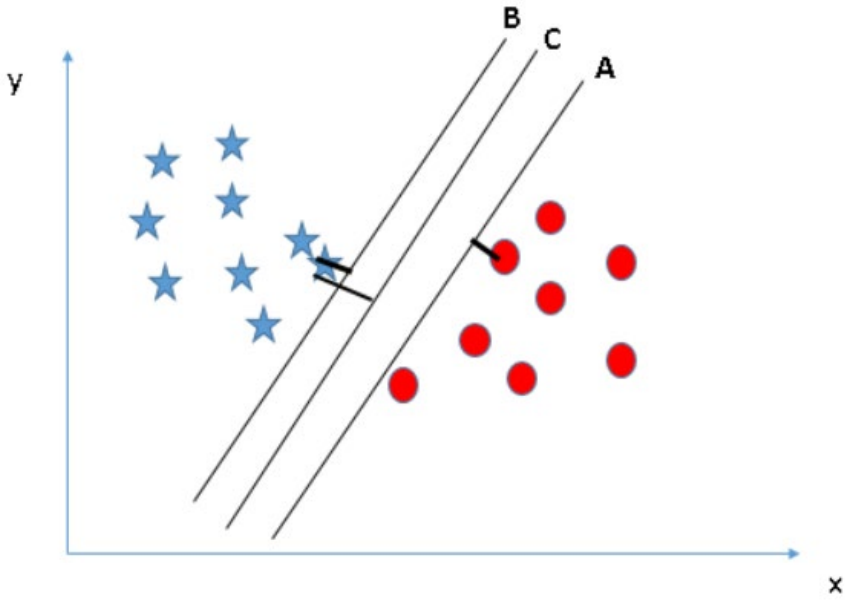
در اینجا باید به صورت مرحله به مرحله پیش رفت. ابتدا فرایند جداسازی دو دسته با یک خط راست را بررسی می کنیم. برای این کار چهار سناریو مطرح است :

سناریو اول : در شکل زیر سه خط راس B ، A و C وجود دارند. اکنون نیاز به تعیین خط راست صحیح برای دسته‌بندی ستاره‌های آبی و دایره‌های قرمز است.



در سناریو اول بیان میشود که یک قانون کلیدی وجود دارد که از آن استفاده می کنیم: « خط راستی که دو دسته را به طور بهتری از یکدیگر جدا کند، خط راست بهتری است که باید انتخاب شود.» در شکل بالا خط B به شکل بهتری دو دسته را از یکدیگر جدا کرده است .

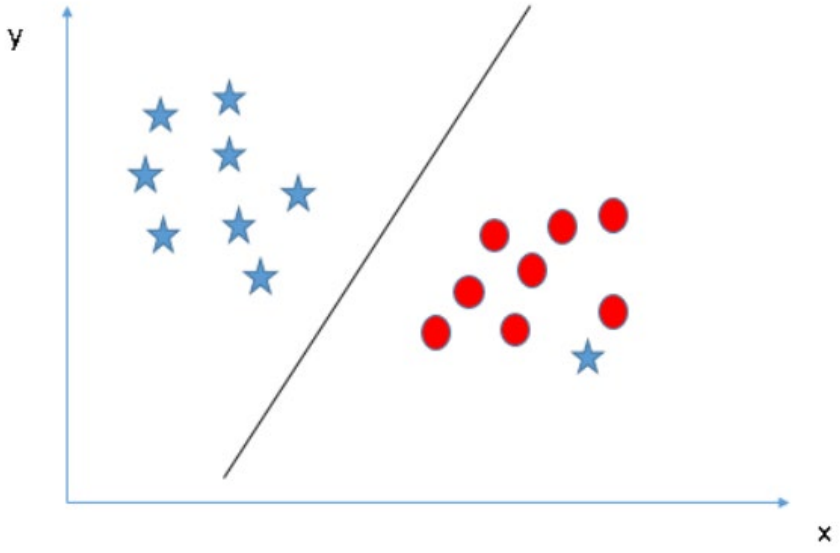
سناریو دوم : حال سوال این است که اگر چند خط همگی به طرز مناسبی دو دسته را از یکدیگر جدا کرده اند آنگاه باید کدام خط به عنوان خط راست صحیح انتخاب شود؟



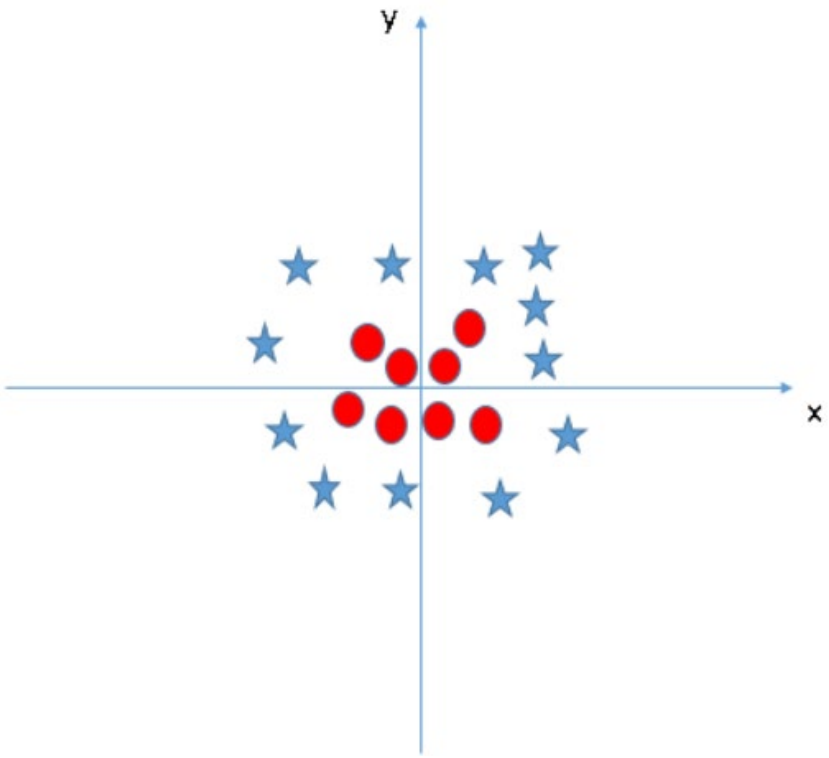
با توجه به شکل بالا، محاسبه فاصله نزدیک‌ترین نقطه داده (که از هر دسته‌ای می‌تواند باشد) از خط راست می‌تواند به انتخاب خط راست صحیح کمک کند. به این فاصله حاشیه گفته می‌شود. می‌توان مشاهده کرد که فاصله خط راست C در مقایسه با هر دو خط A و B از نزدیک‌ترین نقاط داده‌ای موجود در هر کلاس، بیشتر است. بنابراین، خط C را به عنوان خط راست صحیح برمی‌گزینیم. دلیل دیگر واضح برای انتخاب این خط استحکام بیشتر آن است. اگر خط راست حاشیه کمی داشته باشد، احتمال طبقه‌بندی نشدن برخی داده‌ها (miss-classification) وجود دارد.

نکته ی حائز اهمیت آن است که در اصول تعیین خط زاست در الگوریتم svm، خط راستی که دسته ها را به درستی تقسیم کند (صحت یا همان validation که در سناریو اول بررسی شد) بر خطی که حاشیه بیشتری دارد (سناریو دوم) ارجحیت دارد.

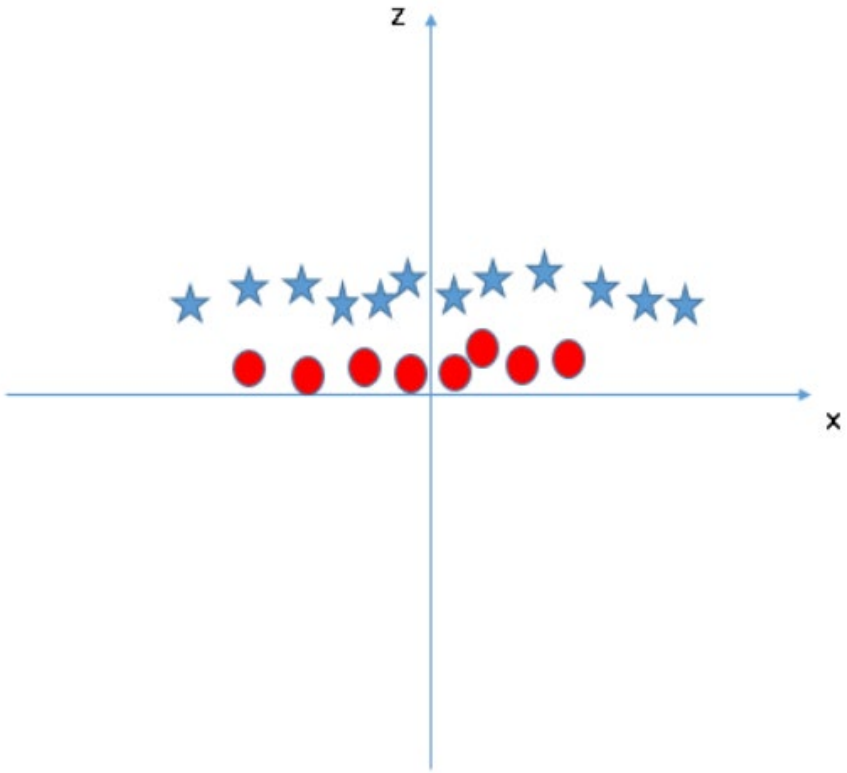
سناریو سوم : آیا میتوان داده های دارای دور افتادگی را دسته بندی کرد؟ در شکل زیر امکان طبقه بندی دو دسته با یک خط راست وجود ندارد. یکی از ویژگی های ماشین بردار پشتبان آن است که دور افتادگی ها را نادیده گرفته و تنها خط راستی را که بیشترین حاشیه را با نقاط داده دسته‌ها دارد انتخاب می‌کند. (مطابق شکل زیر)



سناریو چهارم : حل مسائل نیازمند خط غیر راست جهت جدا سازی دسته ها چگونه است؟ در شکل زیر نمیتوان یک خط راست بین دو کلاس داشت.



svm می تواند این مشکل را با افزودن یک ویزگی جدید حل کند. ویژگی جدید تبدیل $x^2 + y^2 = z$ است که باید بر روی داده‌ها اعمال شود. اکنون می‌توان داده‌ها را روی محور x و z ترسیم کرد. مطابق شکل زیر)



پس بنابراین سوالی که مطرح است آن است که این توابع چگونه انتخاب می شوند ؟

در الگوریتم ماشین بردار پشتیبان، داشتن یک خط راست بین این دو کلاس آسان است. اما، سؤال دیگری که در این مرحله مطرح می‌شود آن است که آیا لازم است این ویژگی به صورت دستی به خط راست اضافه شود؟ پاسخ منفی است، ماشین بردار پشتیبان از روشی که به آن ترفند هسته (کرنل) گفته می‌شود، استفاده می‌کند. در این روش در واقع توابعی وجود دارند که فضای ورودی بُعد پایین را دریافت کرده و آن را به فضای بُعد بالاتر تبدیل می‌کنند. این تبدیل، یک مسئله غیر قابل جداسازی را به مسئله قابل جداسازی مبدل می‌کند. به این توابع، تابع‌های هسته (کرنل) گفته می‌شود.

۲-۲-۵- پیاده سازی svm برای طبقه بندی پروژه

در پایتون، کتابخانه scikit-learn به طور گسترده برای پیاده‌سازی الگوریتم‌های یادگیری ماشین مورد استفاده قرار می‌گیرد و بنابراین الگوریتم svm نیز در این کتابخانه موجود است. تنظیم پارامترها برای الگوریتم svm به طور مؤثر باعث بهبود کارایی مدل می‌شود. در ادامه برخی از پارامترهای مهمی که تأثیر بیشتری بر کارایی مدل دارند، یعنی «کرنل»، «گاما» و «C» مورد بررسی قرار می‌گیرند.

Gamma : ضریب کرنل برای rbf، poly و sigmoid است. هرچه مقدار گاما بیشتر باشد، الگوریتم تلاش می‌کند برازش را دقیقاً بر اساس مجموعه داده‌های تمرینی انجام دهد و این امر موجب تعمیم یافتن خطا و وقوع مشکل بیش برازش (Over-Fitting) می‌شود.

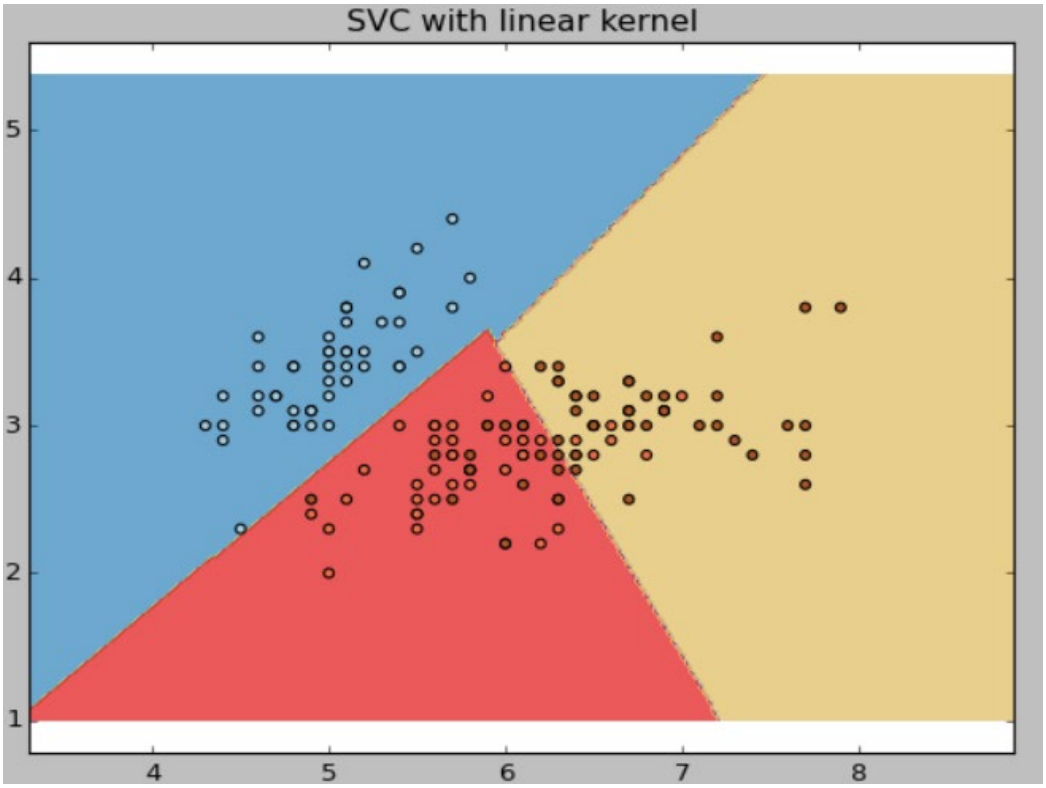
C : پارامتر جریمه**C** ، برای جمله خطا است. این پارامتر همچنین برقراری تعادل بین مرزهای تصمیم‌گیری هموار و طبقه‌بندی نقاط داده تمرینی را کنترل می‌کند .

۳-۲-۵- انواع کرنل ها

در این پروژه دو نوع کرنل استفاده شده که در ادامه به توضیح و پیاده سازی هر یک میپردازیم :

۱-۳-۲-۵- کرنل خطی (linear kernel)

همانگونه که در سناریو های بالا ۱ تا ۳ گفته شد برای جدا سازی دسته های از یکدیگر از خطوطی استفاده می شود. به این حالت اصطلاحا پیاده سازی svm با کرنل خطی گفته می شود. در شکل زیر یک نمونه کلاس بندی با کرنل خطی نشان داده شده است.

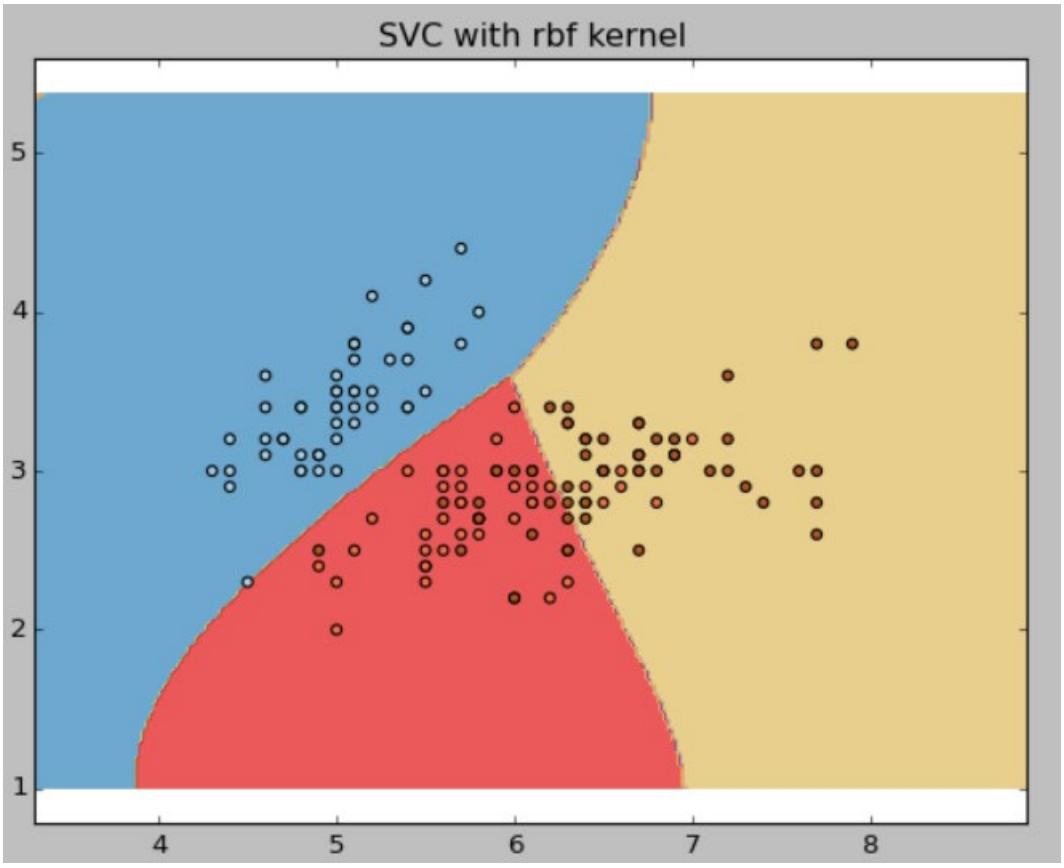


۲-۳-۲-۵- پیاده سازی در پروژه

با توجه به اینکه بردار های wrod2vec برای هر کلمه ۱۰۰ تا ویژگی پیدا کرده، یک فضا ۱۰۰ بعدی تشکیل می شود. به دلیل بزرگی فضا ابتدا توسط تابع pca به ۱۵ بعد می بریم. ۱۵ در اینجا ویژگی برتر است. برای این پارامتر مقادیر ۳۰ و ۶۰ و ۹۰ و ۱۰۰ هم تست شد. که تغییر جزیی در نتیجه داشت دقتش در حد ۰.۲ درصد فقط بهتر شد. در این قسمت مدل به تابع LinearSVC داده شد تا الگوریتم svm با کرنل خطی روی آن اجرا شود. در نهایت با تابع score، صحت ارزیابی روی این الگوریتم ۰/۹۳ گزارش شده است . گزارش بصری سازی شده این بخش در قسمت ارزیابی ها به تفضیل توضیح داده خواهد شد .

۳-۳-۲-۵- کرنل rbf

پارامتر کرنل به صورت پیش فرض روی rbf است . همنطور که در سناریو ۴ گفته شد ، در این نوع کلاس بندی ، دسته ها به وسیله خطوط که بیانگر توابع از درجات بالاتر (غیر خطی) هستند از یکدیگر جدا میشوند . در شکل زیر یک نمونه طبقه بندی الگوریتم svm با کرنل rbf نشان داده شده است .



۴-۳-۲-۵- مزایا و معایب روش svm

در اینجا از طریق تابع svc با کرنل rbf الگوریتم را پیاده سازی کردیم . این تابع در ورودی دو مقدار می گیرد. یکی gamma که مقدار آن را برابر 0/5 قرار دادیم، با تغییر گاما به مقادیر بالاتر و پایین تر نتایج ارزیابی تغییر بسیار جزیی در حد ۲/۰ درصد داشت که میتوان از آن چشم پوشی کرد و دیگری c که به ازای 0/1 ، الگوریتم پیاده سازی و ارزیابی شد. در نهایت با تابع score، صحت ارزیابی روی این الگوریتم با c=0/1 ، ۰/۹۳ گزارش شده است . گزارش بصری سازی شده این بخش در قسمت ارزیابی ها به تفضیل توضیح داده خواهد شد.

۵-۳-۲-۵- مزایا و معایب روش svm

در پایان خوب است به بررسی مزایا و معایب استفاده از طبقه بندی به روش svm که در این پروژه محسوس بود بپردازیم. ابتدا مزایا را نام می بریم.

- حاشیه جداسازی برای دسته های مختلف در visualise کاملا واضح است .
- در فضاهایی با ابعاد بالاتر کارایی بهتری دارد .
- در شرایطی که تعداد ابعاد بیش از تعداد نمونه ها باشد نیز کار میکند .
- یک زیر مجموعه از نقاط تمرینی را در تابع تصمیم گیری استفاده میکند (همان بردار های پشتیبان) ، بنابراین در مصرف حافظه نیز به صورت بهینه عمل می کند .

سپس به معرفی معایب این روش می پردازیم.

- هنگامی که مجموعه داده ها بسیار بزرگ باشد ، عملکرد خوبی ندارد، زیرا نیازمند زمان آموزش بسیار زیاد است.
- هنگامی که داده ها نویز زیادی داشته باشند، عملکرد خوبی ندارد و کلاس های هدف دچار همپوشانی می شوند.

۶- ارزیابی نتایج

یک مثل معروف وجود دارد که میگوید «چیزی را نتوانی ارزیابی کنی ، نمیتوانی بهبود دهی». در بحث الگوریتم های طبقه بندی هم برای اینکه متوجه شویم که آیا الگوریتم ما بر روی داده های مسئله خوب جواب داده است یا خیر، بایستی آن را ارزیابی کنیم.

فرض کنید می خواهید یک خودرو را ارزیابی کنید. حتما می دانید که برای ارزیابی یک خودرو یه صورت کامل، بایستی آن را از منظر های مختلف توسط افراد متخصص مختلف ارزیابی کرد. برای مثال شخصی موتور اتومبیل را ارزیابی کند و شخص دیگری رنگ اتومبیل را بررسی کند و در داده کاوی و یادگیری ماشین نیز روش های متعددی برای ارزیابی الگوریتم های طبقه بندی موجود است که هر کدام از زاویه ای خاص الگوریتم

را ارزیابی می کنند. در این پروژه الگوریتم ماتریش اغتشاش (confusion matrix) و دقت (accuracy) را بررسی می کنیم.

۱-۶- ماتریش اغتشاش (confusion matrix) و دقت (accuracy)

فرض کنید که مجموعه ای داده در اختیار داریم و از الگوریتم خود خواستیم تا یاد بگیرد که احساسات هر یک از داده ها چیست. در اینجا فرض کنید ما برای هر داده داشته باشیم که شاد هست یا نه. خب می دانیم که در چنین الگوریتم هایی همه داده ها را در اختیار الگوریتم قرار نمی دهیم بلکه تنها قسمتی از داده ها را در اختیار آن قرار داده و قسمتی دیگر را برای تست دقت الگوریتم نگه می داریم. (مانند دانش آموزش بعد از خواندن دروس و نمونه سوالات، در جلسه امتحان بایستی به سوالات که جواب آن ها در دست معلم است، پاسخ دهندو معلم، پاسخ های داده شده توسط دانش آموز را با پاسخ های درست موجود مقایسه می کند تا نمره دانشجو محاسبه شود.) خب در نگاه اول برای بررسی دقت چنین الگوریتمی ممکن است بگوییم دو حالت بیشتر وجود ندارد، یا الگوریتم درست تشخیص داده یا اشتباه. یعنی مثلا برای ۱۰۰ داده، احساسات ۱۴ تا را اشتباه تشخیص داده و مابقی یعنی ۸۶ تا را درست تشخیص داده. اما کار به این سادگی نیست. برای درک بهتر این مفهوم میتوان به جدول زیر مراجعه کرد.

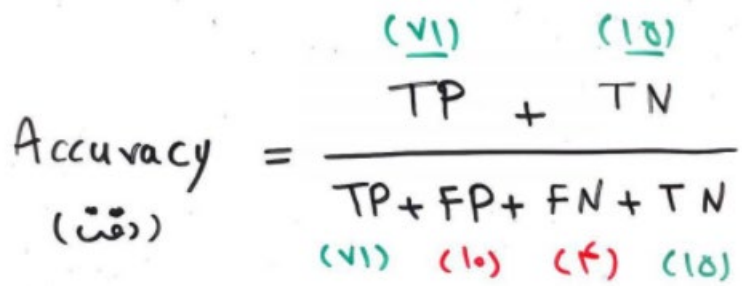
		پیش‌بینی توسط الگوریتم	
		بلی	خیر
برچسب واقعی	بلی	TP (۷۱)	FN (۴)
	خیر	FP (۱۰)	TN (۱۵)

برای درک بهتر شکل بالا فرض کنید الگوریتم طبقه بندی (فرضا در اینجا جنگل تصادفی را استفاده کردیم) بعد از یادگیری و ساخت مدل، مورد آزمون قرار گرفته است. در این آزمون ما ۱۰۰ داده را به الگوریتم می دهیم تا الگوریتم این داده ها را بر اساس ویژگی هایشان طبقه بندی کند دارای احساس شاد هستند یا نه. (پس دو دسته داریم) توجه کنید که ما در برچسب واقعی هر داده را از قبل می دانیم و در اینجا می خواهیم دقت (میزان درست گویی) الگوریتم خود را بررسی کنیم. در شکل بالا سطر ها، برچسب های واقعی ما هستند و ستون ها، پیش بینی های الگوریتم را تشکیل می دهند. بر این اساس چهار حالت زیر را داریم:

۱- حالتی که داده دارای احساسات شاد باشد و الگوریتم هم احساسات را به درستی، شاد، تشخیص داده است که به آن true positive یا همان tp می گویند.
۲- حالتی که برخی داده ها شاد باشند و الگوریتم به اشتباه آنها را شاد تشخیص نداده (غمگین تشخیص داده است) که به آن false negative یا همان fn می گویند.
۳- حالتی که برخی داده ها شاد نباشند اما الگوریتم به اشتباه آن ها را شاد تشخیص داده که بدان false positive یا همان fp میگویند .
۴- حالتی که برخی از داده ها شاد نباشند و الگوریتم هم به درستی آنها را شاد تشخیص نداده است (غمگین تشخیص داده است) که بدان true negetive یا همان tn می گویند.

همانطور که مشاهده میکنید فقط در حالت های tn , tp صحت برقرار است و در دو حالت دیگر خطا رخ داده است . مثلا در مثال بالا ، تعداد ۷۱ داده شاد بوده اند که الگورتیم به درستی آنها را شاد تشخیص داده است (tp=71) و یا تعداد ۴ داده شاد بوده اند که الگوریتم انها را به اشتباه شاد تشخیص نداده است (fn=4) همچنین تعداد ۱۰ داده شاد نبوده اند که الگوریتم به اشتباه آنها را شاد تشخیص داده است (fp=10) و در نهایت تعداد ۱۵ داده شاد نبوده اند و الگوریتم به درستی آنها را غمیگین تشخیص داده است (tn=15).

این جدول که بدان ماتریس اغتشاش یا confusion matrix می گویند، مبنای بسیاری از معیار های اندازه گیری کیفیت یک الگوریتم طبقه بندی است. یکی از معیارها ، معیار دقت (accuracy) است که به صورت زیر محاسبه می شود .



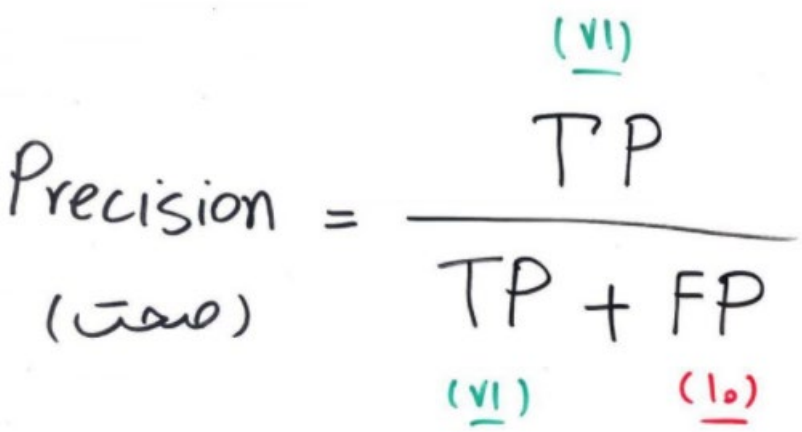
همانطور که ملاحظه میشود صورت کسر تعداد داده هایی است که درست تشخیص داده شدند و مخرج کسر تعداد کل داده هاست.

در مثال بالا فقط دو کلاس داشتیم (شاد هست یا نه). اگر تعداد کلاس ها بیشتر بود (multi class)، بایستی برای هر کدام زا کلاس ها (عصبانیت ، شادی ، ترس ...) یک ماتریس اغتشاش طراحی کرد و معیار دقت (accuracy) را برای هر کدام از این کلاس ها محاسبه می کنیم و سپس میانگین تمامی این دقت ها، دقت کل را تشکیل می دهد.

۲-۶- اشکال معیار ارزیابی دقت

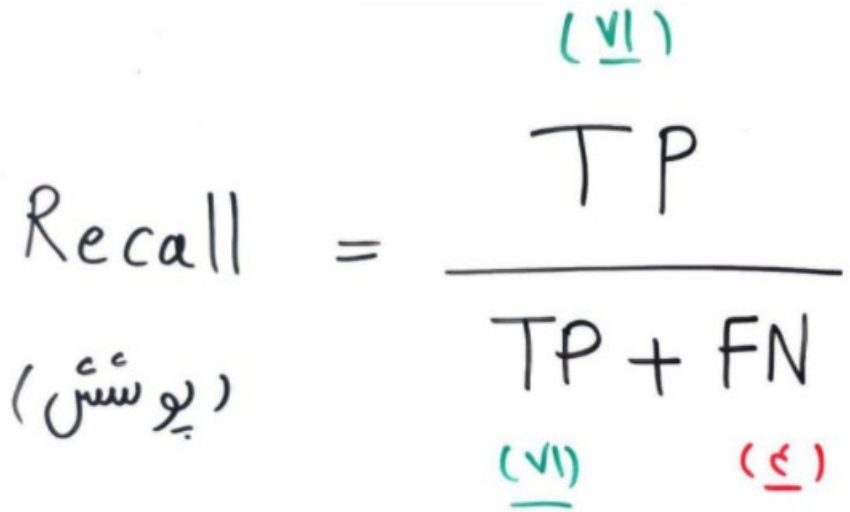
همانطور که از فرمول بالا بر می آید معیار دقت نمی تواند تفاوتی میان خطای ft و fp قائل شود. در واقع اگر در همین مثال بالا تعداد خطای fn برابر ۱۰ و تعداد خطای fp برابر ۴ می بود، الگوریتم باز هم دقت را ۸۶ درصد نشان میداد که تفاوتی با حالت قبل نداشت !

برای غلبه براین مشکل دو معیار معرفی شده است. یکی از آن ها معیار صحت (precision) بوده که فرمول آن به صورت زیر است :



با دقت در این فرمول می توان دریافت که تمرکز الی این معیار بر روی درستی تشخیص های «بلی» توسط الگوریتم است. در واقع معیار صحت معیاری است که به ما می گوید الگوریتم چند درصد «بلی» هایش درست بوده است.

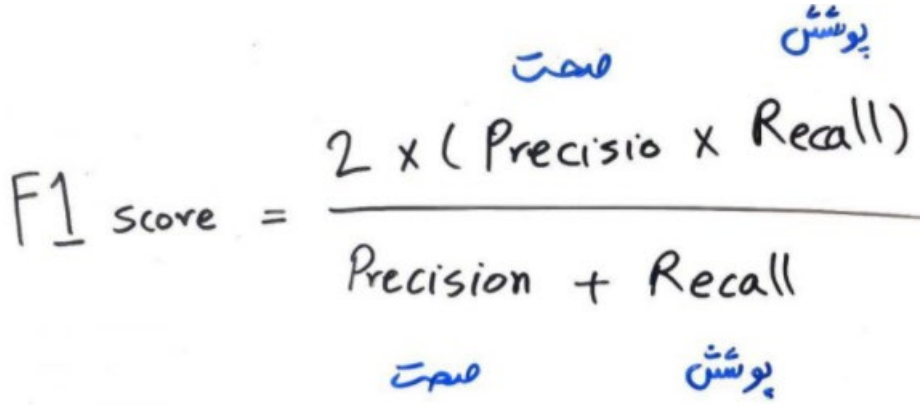
اما معیار دیگری نیز وجود به نام پوشش (recall) که همانطور که از نامش پیداست، به دنبال محاسبه پوشش بر روی کل داده هاست. فرمول اسیت معیار به صورت زیر است :



همانطور که مشاهده می کنید ، تمرکز اصلی معیار پوشش (recall) بر خلاف معیار صحت (precision) بر روی داده هایی است که واقعا «بلی» بوده اند.

۳-۶- معیار f1

معیار سومی به نام f1 نیز وجود دارد که در واقع ترکیبی از معیار های دقت و صحت است و به صورت زیر فرموله می شود :



معیار f1 در واقع ترکیب متعادلی بین معیار های دقت و صحت است و می تواند در مواردی که هزینه fp و fn متفاوت است به کار رود. اگر هزینه ی ft و fp برابر بود بدیهی است که همان روش accuracy که در ابتدا گفته شد مناسب است چون رد این حالت مشکل آن که در نظر نگرفتن fp و fn بود حل می شود.

۶-۴- نتایج ارزیابی ها در پروژه

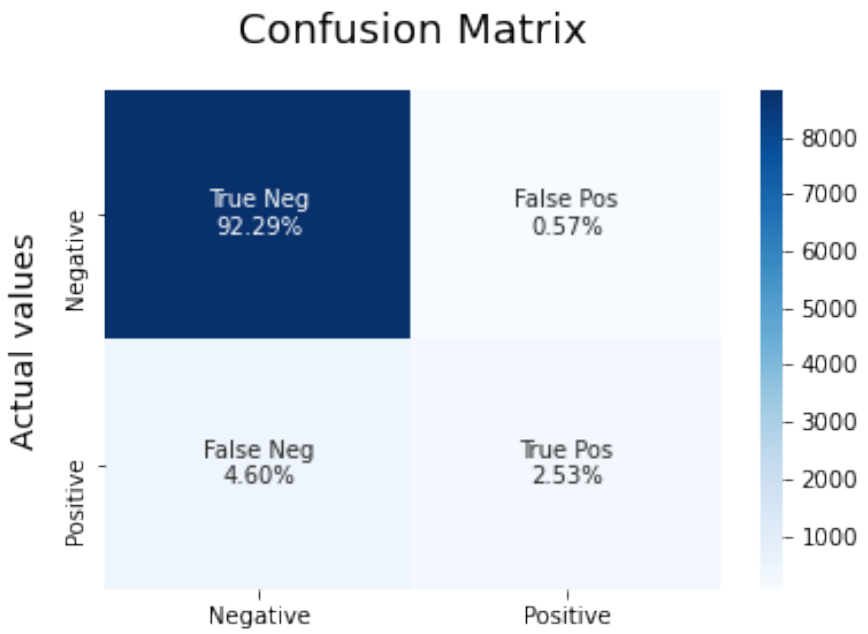
از طریق تابع modelResult نتایج پیش بینی های مدل گزارش شده است. این تابع در ورودی خود مقدار پیش بینی شده برای هر داده آزمون را می گیرد. در این تابع دسته بندی به دو گروه مثبت و منفی انجام شده است و گروه هایی که برای ارزیابی مورد بررسی قرار می گیرند شامل چهار گروه tn, fp, fn, tp می باشد.

۶-۴-۱- نتایج ارزیابی random forest

در این ارزیابی y های پیش بینی شده توسط الگوریتم random forest به تابع modelResult داده شده و نتایج زیر گزارش شده است.

	precision	recall	f1-score	support
0	0.95	0.99	0.97	8905
1	0.82	0.36	0.49	684
accuracy			0.95	9589
macro avg	0.88	0.67	0.73	9589
weighted avg	0.94	0.95	0.94	9589

هم چنین confusion matrix در این ارزیابی به صورت زیر گزارش شده است :

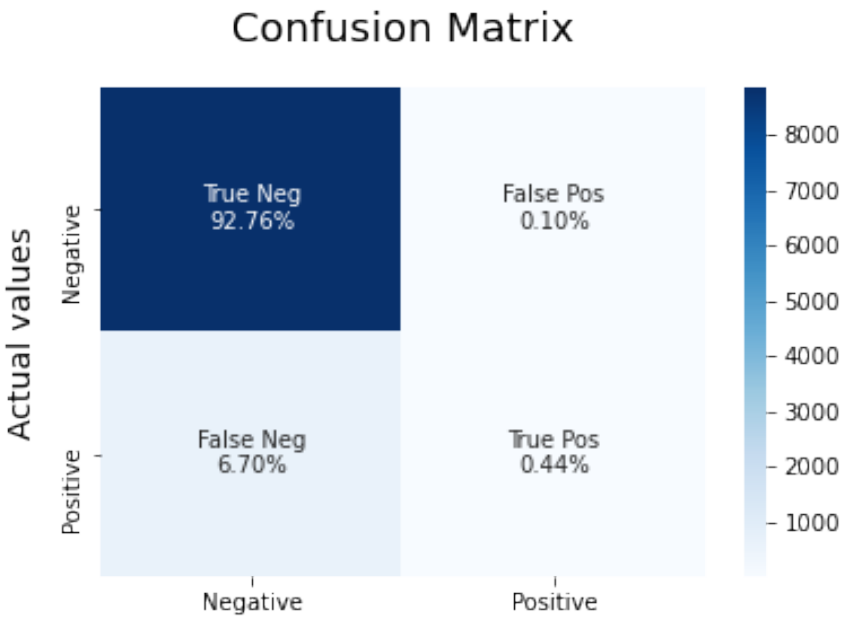


۲-۴-۶- نتایج ارزیابی svm with linear kernel

در این ارزیابی y های پیش بینی شده توسط الگوریتم random forest به تابع modelResult داده شده و نتایج زیر گزارش شده است.

	precision	recall	f1-score	support
0	0.93	1.00	0.96	8905
1	0.81	0.06	0.11	684
accuracy			0.93	9589
macro avg	0.87	0.53	0.54	9589
weighted avg	0.92	0.93	0.90	9589

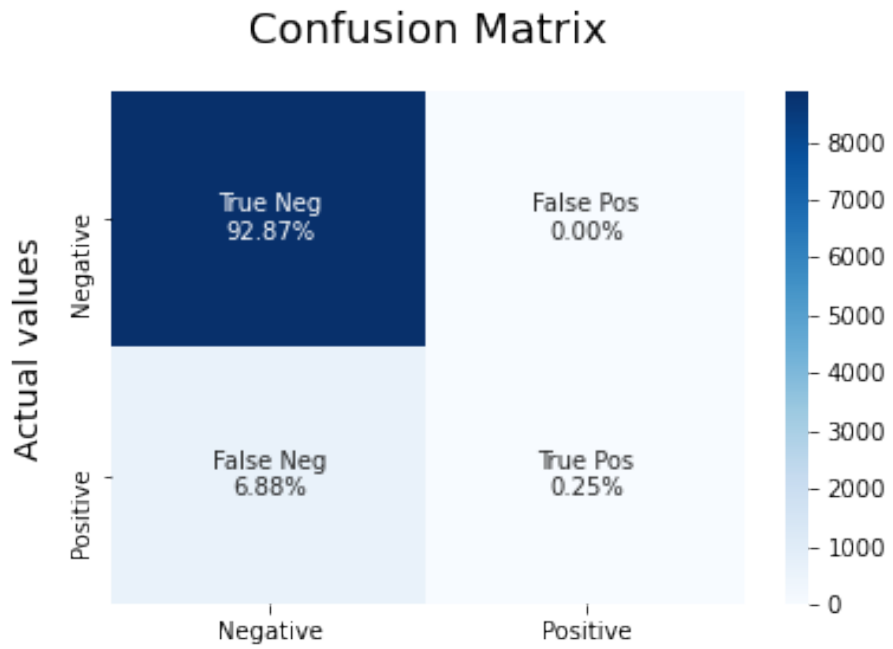
هم چنین confusion matrix در این ارزیابی به صورت زیر گزارش شده است :



۳-۴-۶- نتایج ارزیابی **svm with rbf and c=0/1**

	precision	recall	f1-score	support
0	0.93	1.00	0.96	8905
1	1.00	0.04	0.07	684
accuracy			0.93	9589
macro avg	0.97	0.52	0.52	9589
weighted avg	0.94	0.93	0.90	9589

هم چنین confusion matrix در این ارزیابی به صورت زیر گزارش شده است :



۷- نتیجه گیری

با توجه به نتایج بدست آمده، مشاهده می شودکه الگوریتم جنگل تصادفی با دقت تقریبا ۹۵ درصد و همچنین tp بیشتر از دیگر الگوریتم ها بهتر عمل کرده است.