دانشگاه آزاد اسلامی

واحد بندرعباس

دانشکده فنی و مهندسی

رشته مهندسی کامپیوتر –ارشد نرم افزار

عنوان:

تشخیص نویسنده‌ها از روی متن

استاد راهنما:

دکتر عباس عکاسی

نگارندگان:

کمیل آقابابایی، سحر ارزمانی، مریم یوسفی‌زاده، سمیه رجبی، امیر جعفری‌زاده ، پری دریاکش، هادی ناظری، مرتضی‌سهرابی، میثم سالاری، امید نیاستی

بهار ۱۳۹۸

تقدیم به :

او که آموخت مارا تا بیاموزیم

{استاد گرامی جناب آقای دکتر عکاسی}

تقدیم به آنان که وجودمان جز هدیه وجودشان نیست

{ پدران و مادران عزیزمان}

سپاسگزاری:

از استاد گرامی دکتر عباس عکاسی که با حسن خلق و فروتنی، ما را در این عرصه یاری نمودند کمال تشکر را داریم.

فهرست

فهرست اشکال

فهرست جداول

چکیده

چالش شناسایی نویسنده نیازمند کاربران است که بخش‌هایی از متن نوشته شده توسط نویسندگان را بازیابی کنند، یک مدل پیشگویانه داده‌شده با داده را آموزش دهند و قادر باشد تا متن‌های جدید را به نویسندگان درست دسته بندی کنند. این یک مساله یادگیری نظارت شده در حوزه پردازش زبان طبیعی است.

در این پژوهش از الگوریتم های متعددی برای دسته بندی ارائه شده‌اند که از آن دسته می‌توان؛ به شبکه‌های بیزین، الگوریتم دسته بندی کننده بردار پشتیبان خطی ، الگوریتم رگرسیون منطقی والگوریتم درخت های اضافی اشاره کرد که با استفاده از هر الگوریتم ۱۰۰ دسته بند را ساخته و با توجه به معیار های ارزیابی هر مدل را بر روی داده های اعتبارسنجی ارزیابی کرده و بهترین دسته بند را انتخاب می کنیم . و در نهایت از روش های ترکیب دسته بند ها استفاده کرده و بهترین روش را برروی داده های تست، ارزیابی می کنیم.

کلمات کلیدی : شناسایی نویسنده، الگوریتم شبکه بیزین، الگوریتم دسته بندی کننده بردار پشتیبان خطی، الگوریتم رگرسیون منطقی، الگوریتم درخت های اضافی، ترکیب دسته بندها

فصل اول:

کلیات تحقیق

۱-۱- مقدمه

افزایش استفاده از کامپیوترها در فعالیت‌های کسب و کار، منجر به رشد سریع پایگاه‌های اطلاعاتی و اجتماع داده‌ها توسط بیشتر سازمان‌ها شده است. روزانه حجم عظیمی از داده‌ها تولید شده و در پایگاه‌های مختلف داده ذخیره می‌شود. کشف دانش در پایگاه‌های اطلاعاتی (KDD[۱]) یک تحلیل خودکار ، اکتشافی و مدل‌سازی از مخازن داده بزرگ است . KDD یک فرآیند سازمان‌یافته شناسایی الگوهای معتبر ، جدید ، مفید ، و قابل‌فهم از مجموعه‌های داده بزرگ و پیچیده است. داده‌کاوی (DM[۲]) هسته اصلی این فرآیند است که به کشف داده‌ها ، توسعه مدل و کشف الگوهای ناشناخته پیشین می‌پردازد[1] . داده‌کاوی , فرآیند کشف الگوهایی است که برای ساخت مدل‌های پیش‌بینی‌کننده استفاده می‌شود، که در بسیاری از رشته‌های علوم مالی گرفته تا مراقبت‌های بهداشتی گسترده شده‌است [2].

در سال‌های اخیر، صحت اطلاعات (آنلاین) توجه زیادی را به خود جلب کرده‌است، به ویژه در زمینه " اخبار جعلی " در زمینه انتخابات ریاست جمهوری آمریکا . در حال حاضر تاکید زیادی بر روی منشا و صحت اطلاعات صورت‌گرفته است . در مورد اسناد نوشتاری ، یک جنبه مهم از این نوع انتقاد، به نویسندگی مربوط می‌شود . ارزیابی صحت اطلاعات اساسا مربوط به شناسایی نویسنده اصلی این اسناد می‌باشد . شناسایی نویسنده، تلاش می‌کند نویسندگان پشت متن را آشکار کند . این یک حوزه نوظهور از تحقیقات مرتبط با کاربرد در تحقیقات ادبی ، امنیت سایبری ، قانونی ، و تحلیل رسانه‌های اجتماعی است[3] .

۲-۱- بیان مسأله

رویکردهای کمّی به وظایفی مانند تخصیص دادن اسناد ، تایید، ثبت مشخصات و یا خوشه‌بندی نویسنده بر این فرض پایه تکیه دارند که سبک نوشتاری اسناد به گونه‌ای مشخص ، آموخته و برای ساختن مدل‌های پیش‌بینی استفاده می‌شود . هر سیستم شناسایی موفقِ نویسنده ، می‌تواند در یک تنظیمات اختصاصی یا در یک تنظیمات تایید(بازبینی) ، تعیین هویتِ قدرتمندی در متن در ژانرهای مختلف ، رفتار با موضوعات مختلف و یا داشتن مخاطبان هدف مختلف در ذهن ایجاد کند . مدل‌های سنتی برای تخصیص دادن اسناد در مواردی که چندین

---

[۱] KDD : Knoledge Discovery in Database

[۲] DataMining

نویسنده در یک سند مشارکت دارند، قابل‌اجرا نیستند . در نتیجه می توان با ابزار های داده کاوی این موضوع را حل کرد[3] .

در این پژوهش ابتدا با توجه به ماهیت مسئله با استفاده از کتابخانه پاندا داده های ورودی با سه ویژگی را دریافت می‌کنیم . روش دسته بندی یک روش یادگیری با نظارت است که داده‌های ورودی به سه بخش داده‌های آموزش و داده‌های آزمون و داده های اعتبارسنجی تقسیم می‌شوند . هر الگوریتم کاندید، ابتدا با استفاده از مجموعه داده آموزش یک مدل را که نشان دهنده الگوی حاکم بر داده‌ها می‌باشد را استخراج می‌کند و سپس با استفاده از مجموعه آزمون و اعتبارسنجی، دقت مدل ارائه شده برای دسته‌بندی را بررسی می‌کند.

الگوریتم‌های متعددی برای دسته بندی ارائه شده‌اند که از آن دسته می‌توان؛ به شبکه‌های بیزین، الگوریتم دسته بندی کننده بردار پشتیبان خطی ، الگوریتم رگرسیون منطقی والگوریتم درخت های اضافی اشاره کرد.

در این نوشتار ابتدا با استفاده از توابع کتابخانه پاندا داده های مربوط به نویسندگان تقسیم بندی می‌شوند تا به صورت داده های آموزش و آزمون و اعتبار سنجی درآیند . سپس بعد از پاکسازی داده و تهیه ورودی برای الگوریتم های پایه که در بالا ذکر شد، ۱۰۰ دسته بند را با استفاده از هر الگوریتم میسازیم . سپس با استفاده از معیار های ارزیابی بر روی داده های اعتبار سنجی، هر مدل را ارزیابی کرده و بهترین دسته بند را انتخاب میکنیم و در مرحله بعد با روش های ترکیبِ دسته بند ها از جمله حذف پس رونده و انتخاب پیش رونده و انتخاب تک دسته بندِ بهترین، بهترین روش را انتخاب کرده و بر روی داده های آزمون تست می‌کنیم . در نهایت سعی شده با در نظر گرفتن نقاط ضعف و قوت روش‌های مختلف داده کاوی یک الگوریتم ترکیبی برای تشخیص نویسنده ارائه شود.

۳-۱- اهداف تحقیق

هیچ راهنمای مشخصی برای آنکه پژوهشگران یا تحلیلگران بدانند چگونه یک الگوریتم داده کاوی را انتخاب کنند وجود ندارد. در نتیجه انتخاب یک الگوریتم مشخص امری بسیار پیچیده است، لذا در این پژوهش برای ارتقای نتایج داده کاوی از چندین الگوریتم استفاده شده و هر کدام را جداگانه توضیح و تشریح داده و پردازش‌ها را با الگوریتم‌های مختلف تکرار و سرانجام از تکنیک رای گیری اکثریت استفاده می‌شود. هدف از روش های ارائه شده افزایش دقت تشخیص نویسنده با استفاده ازکشف الگوها درمیان مجموعه داده ها می‌باشد.

۱-٤- نوآوری‌های تحقیق

- استفاده از ترکیب دسته بندها
- ساخت تعداد مدل های زیاد

۱-۵- تعریف واژگان

داده کاوی : استخراج داده فرآیند کشف الگوهای جالب و دانش از مقادیر زیاد داده‌ها است[4] .

دسته بندی[۳] : : جهت اختصاص دادن یک برچسب به مجموعه‌ای از داده‌ها که دسته‌بندی نشده‌اند، استفاده می‌شود. در حقیقت دسته بندی یک وظیفه آنالیز داده است ، به عبارتی فرآیند یافتن مدلی است که کلاس‌های داده و مفاهیم را توصیف و تمایز می‌دهد . در دسته بندی داده‌ها بر اساس ویژگی‌هایشان به دسته‌هایی که نام آن‌ها از قبل مشخص می‌باشد، تخصیص داده می‌شوند[5] .

یادگیری تجمعی[٤] : یادگیری تجمعی یک تکنیک یادگیری ماشین است که با استفاده از یک الگوریتم یادگیری متفاوت ، چندین یادگیرنده را با همان داده‌ها آموزش می‌دهد . یادگیری تجمعی اغلب از یک الگوریتم یادگیری واحد بهتر عمل می‌کند. این امر به خصوص زمانی صادق است که گروه شامل الگوریتم‌های متنوعی است که هر کدام یک رویکرد کاملا متفاوتی را اتخاذ می‌کنند . هر الگوریتم یادگیری به عنوان یک یادگیرنده پایه است . الگوریتم مورد استفاده برای ترکیب نتایج با نام گروه[۵] شناخته می‌شود .

زمانی که باید از یادگیری گروهی استفاده کنیم به شرح ذیل می باشد :

- مجموعه داده‌ها خیلی بزرگ یا خیلی کوچک است .
- داده‌های پیچیده (غیرخطی[٦])
- قابلیت اعتماد بالا[6][7]

---

[۳] Classification

[٤] Ensemble Learning

[۵] Ensemble

[٦]Nonlinear

٦-١- ساختار پژوهش

مطالبی که در فصول بعدی ارائه خواهد شد به شرح زیر خواهد بود:

- در فصل دوم مفاهیم پایه‌ای مانند داده‌کاوی، کلیات مربوط به الگوریتم‌های دسته بندی، مفاهیم یادگیری تجمعی، الگوریتم‌های رایج دسته‌بندی، معیارهای ارزیابی این الگوریتم‌ها و انواع روش ترکیب دسته بندها مورد بحث قرار می‌گیرد.
- در فصل سوم حاوی کارهای انجام شده و تحقیقات مرتبط با موضوع می‌باشد، همچنین فضای کلی مسأله معرفی می شود.
- در فصل چهارم مدل پیشنهادی برای دسته‌بندی بر روی مجموعه داده‌های نویسندگان اعمال و نتایج روش پیشنهادی در این زمینه مورد مقایسه و ارزیابی قرار گرفته است.
- فصل پنجم نیز حاوی خلاصه، نتیجه‌گیری و پیشنهادات می‌باشد.

فصل دوم :

ادبیات و پیشینه تحقیق

۲-۱- مقدمه

در دنیای امروزی، ما با حجم عظیمی از داده ها مواجه هستیم و تجزیه و تحلیل مجموعه داده‌های بزرگ، اساس کلیدی رقابت، زیربنای رشد بهره‌وری و نوآوری خواهد بود . افزایش حجم و جزئیات اطلاعات به‌دست‌آمده توسط شرکت‌ها، رشد چند رسانه‌ای، رسانه‌های اجتماعی و اینترنت اشیاء، رشد نمایی در داده را برای آینده قابل‌پیش‌بینی خواهد داشت . در نتیجه با توجه به این حجم از داده ها، چالش هایی برای توانایی کنترل داده ها بوجود می آید[1] . دست پیدا کردن به اطلاعات نهفته در پایگاه داده شرکت‌ها، دانشگاه‌ها، مؤسسات دولتی و سایر مراکز نیازمند مدیریتی جدید است و با به‌کارگیری سیستم‌های سنتی این امر تحقق نمی‌یابد.

در این فصل ابتدا داده کاوی و مراحل آن شرح داده می شود و سپس دسته بندی داده ها و الگوریتم های رایج آن و معیار های ارزیابی آن ها مورد بحث قرار میگیرد و در پایان توضیحاتی در مورد یادگیری تجمعی و انواع روش ترکیب دسته بندها بصورت کامل شرح داده می شود.

۲-۲- داده‌کاوی

در سال ۱۹۶۰ آماردانان اصطلاح "Data Fishing" یا "Data Dredging" به معنای "صید داده" را جهت کشف هر گونه ارتباط در حجم بسیار بزرگی از داده‌ها بدون در نظر گرفتن هیچگونه پیش فرضی بکار بردند . بعد از سی سال و با انباشته شدن داده‌ها در پایگاه داده اصطلاح داده کاوی در حدود سال ۱۹۹۰ رواج بیشتری یافت. این فرآیند یک مرحله فراتر از بازیابی ساده داده‌ها است و این اجازه را می‌دهد که دانش را در میان حجم انبوه داده‌ها کشف کرد .

داده‌کاوی، به مفهوم استخراج اطلاعات نهان یا الگوها و روابط مشخص در حجم زیادی از داده‌ها در یک یا چند بانک اطلاعاتی بزرگ گفته می‌شود. بسیاری از مردم داده کاوی را مترادف واژه‌های رایج کشف دانش از داده‌ها (KDD) می‌دانند . داده‌کاوی، پایگاه‌ها و مجموعه حجیم داده‌ها را در پی کشف و استخراج، مورد تحلیل قرار می‌دهد. در حقیقت داده کاوی یک مرحله ضروری از فرآیند بزرگ‌تر کشف دانش می‌باشد که شامل مراحل زیر می‌باشد :

۱. پاک‌سازی داده‌ها[۷] : در این مرحله ، داده‌های نویز و ناسازگار حذف می‌شوند .

۲. یکپارچگی داده[۸] : در این مرحله , چند منبع داده ترکیب می‌شوند .

۳. انتخاب داده[۹] : در این مرحله ، داده‌های مربوط به کار آنالیز از پایگاه‌داده بازیابی می‌شوند

٤. تبدیل داده ها[۱۰] : در این مرحله ، داده‌ها به اشکالی که برای بکار بردن در داده‌کاوی مناسب باشند، تبدیل یا ترکیب می شوند .

۵. داده کاوی : در این مرحله ، روش‌های هوشمند به منظور استخراج الگوهای داده‌ای بکار گرفته می‌شوند .

٦. ارزیابی الگو[۱۱] : در این مرحله الگوهای داده‌ای مورد ارزیابی قرار می‌گیرند .

۷. ارائه دانش[۱۲] : در این مرحله , دانش ارائه می‌شود [8].

داده‌کاوی از دو مرحله اصلی تشکیل شده است؛ مرحله اول پیش پردازش داده‌ها که در این مرحله ویژگی های با تأثیر بالاتر از داده‌های سطح پایین استخراج می‌شود. مرحله دوم تشخیص الگو می‌باشد که به کشف الگوی موجود در داده‌ها به کمک صفات و ویژگی های بدست آمده می‌پردازد[9].

کاوش اطلاعات، حجم عظیمی از داده‌های خام را به فرمی تغییر می‌دهد که انسان بتواند آن‌ها را به راحتی بفهمد و برای تصمیم گیری بتواند از این اطلاعات استفاده کند. در قدم اصلی داده کاوی ممکن است از چندین الگوریتم داده کاوی استفاده شود. کار اصلی الگوریتم داده کاوی با توجه به نوع مسئله‌ی کشف دانش تغییر می‌کند اما دو نوع اصلی الگوریتم‌های داده کاوی، دسته‌بندی و خوشه‌بندی است.

---

[۷] Data cleaning

[۸] Data Integration

[۹] Data selection

[۱۰] Data transformation

[۱۱] Pattern evaluation

[۱۲] Knowledge presentation

یکی از دلایلی که باعث شد داده کاوی مورد توجه قرار بگیرد، مسأله در دسترس بودن حجم وسیعی از داده‌ها و نیاز شدید به اینکه از این داده‌ها، اطلاعات و دانش استخراج شود . داده‌کاوی عبارت است از استخراج دانش از مجموعه‌ای از داده‌ها.

۳-۲- دسته‌بندی

هرگاه داده‌ها دارای خصیصه‌ای خاص باشند که مستقیماً از دیگر خصایص به وجود نیامده باشد اما بین آن مشخصه و دیگر ابعاد رابطه وابستگی وجود داشته باشد، در این صورت می‌توان با کشف مدلی بر اساس دیگر مشخصه‌ها، آن بعد مذکور (که نشان دهنده دسته خاصی از داده‌ها است) را شناسایی نمود .

این یک کار تحلیل داده و فرآیند پیدا کردن مدلی است که کلاس‌ها و مفاهیم داده‌ها را توصیف و متمایز می‌کند. دسته‌بندی مساله شناسایی مجموعه‌ای از طبقات (زیر جمعیت‌ها) و یک مشاهده جدید براساس مجموعه آموزشی داده‌ها است[10].

فرض کنید که قسمتی از نوشته های نویسندگان در پایگاه داده‌ای وجود دارد که قبلاً با استفاده از آزمایش خاص سه نویسنده مشخص شده که هرکدام از این نوشته ها به کدام نویسنده تعلق دارد، در این جا هیچ نوشته ای حق ندارد به دو نویسنده تعلق داشته باشد، به این معنی که دسته‌ها فضای مسئله را افراز می‌کند. در چنین پایگاه داده‌هایی برای هر نوشته یک رکورد خاص وجود دارد و در نهایت نام یا برچسب نویسنده ای که آن نوشته به آن تعلق دارد، قرار گرفته است . یک داده کاو تصمیم می‌گیرد سیستمی را ابداع کند که طی آن بدون آزمایش و فقط از روی برخی ویژگی های نوشته ها بتوان نام نویسنده مربوطه را تشخیص داد. این تصمیم ممکن است به هر دلیلی صورت گرفته باشد. آنچه باید انجام شود عملیات دسته بندی نامیده می‌شود.

دو مرحله مهم دسته بندی عبارتند از : اولین مرحله ساخت مدل و دومین مرحله استفاده از مدل و پیش‌بینی کلاس از طریق مدل ساخته شده است[11] . بدین منظور باید مجموعه داده‌ها را به سه دسته داده‌های آموزش[۱۳] و داده های اعتبار سنجی[۱۴] و داده‌های تست[۱۵] تقسیم کنیم . با استفاده از داده‌هایی که برچسب آموزش خورده‌اند یک دسته‌بند ایجاد می‌شود که بر اساس آن بتوان داده‌های فاقد برچسب را در دسته‌های مربوط به خودشان قرار

---

[۱۳] Train data

[۱۴] Development data

[۱۵] Test data

داد. کارایی دسته‌بند ساخته شده با داده‌های تست و داده های اعتبار سنجی(که به صورت تصادفی از میان داده‌ها انتخاب شده‌اند) مورد سنجش قرار می‌گیرد و مدل روی آن‌ها اجرا می‌شود تا دقت پیش بینی دسته‌بند بررسی گردد، چنان که مدل دارای دقت مناسبی باشد برای دسته‌بندی داده‌ها به کار می‌رود.

۲-٤- الگوریتم‌های رایج دسته‌بندی

روش‌های زیادی برای دسته‌بندی وجود دارد که از جمله مواردی که ما در این پژوهش از آنها استفاده کرده ایم می توان به موارد زیر اشاره کرد:

۱. دسته بندی کننده بردار پشتیبان خطی[۱۶]
۲. شبکه های بیزین[۱۷]
۳. رگرسیون منطقی[۱۸]
٤. درخت های اضافی[۱۹]

در هر چهار الگوریتم استفاده شده ابتدا Binary BOW[۲۰] ، سپس BOW[۲۱] و در نهایت TF-IDF[۲۲] را محاسبه نموده و با استفاده از TF-IDF صد نمونه داده ساخته ایم.

۲-٤-۱- دسته بندی کننده بردار پشتیبان خطی

به صورت کلی ماشین‌های بردار پشتیبان (SVM[۲۳]) یکی از الگوریتم های طبقه‌بندی است که فاقد پایه نظری

---

[۱۶] Linear SVC : Linear Support Vector Classifier
[۱۷] Naïve Base
[۱۸] Logistic regression
[۱۹] Extra trees
[۲۰] Binary Bag Of Word
[۲۱] Bag Of Word : کیسه کلمات
[۲۲] Term Frequency_ Inverse Document Frequency
[۲۳] Support vector machine

میباشد و نسبت به تعداد ابعاد در مجموعه داده‌ها حساس است . زمانی که مجموعه داده‌ها بصورت خطی قابل‌تفکیک باشند، ماشین‌های بردار پشتیبان صفحه ابر را پیدا می‌کنند که به بهترین شکل از هم جدا می‌شوند.

این الگوریتم یک جداکننده است که با معیار قرار دادن بردارهای پشتیبان، بهترین دسته بندی و تفکیک بین داده ها را برای ما مشخص می کنند . بردارهای پشتیبان به زبان ساده، مجموعه ای از نقاط در فضای n بعدی داده ها هستند که مرز دسته ها را مشخص می کنند و مرزبندی و دسته بندی داده ها براساس آنها انجام می شود و با جابجایی یکی از آنها، خروجی دسته بندی ممکن است تغییر کند . در فضای دو بعدی بردارهای پشتیبان، یک خط، در فضای سه بعدی یک صفحه و در فضای n بعدی یک ابر صفحه را شکل خواهند داد. در این الگوریتم فقط داده های قرار گرفته در بردارهای پشتیبان مبنای یادگیری ماشین و ساخت مدل قرار می گیرند و این الگوریتم به سایر نقاط داده حساس نیست و هدف آن هم یافتن بهترین مرز در بین داده هاست به گونه ای که بیشترین فاصله ممکن را از تمام دسته ها (بردارهای پشتیبان آنها) داشته باشد .

حال در SVC ها با کرنل پارامترِ " خطی " ، انعطاف‌پذیری بیشتری دارد و باید نسبت به تعداد زیادی از نمونه‌ها مقیاس داشته باشد .

این کلاس هم از ورودی متراکم و هم پراکنده پشتیبانی می‌کند و پشتیبانی چندکلاسه[۲۴] با توجه به یک برنامه تک مرحله‌ای انجام می‌شود .

در SVM ها، مدل چند کلاسی، با استفاده از یک طرح در مقابل یک برنامه اجرا می‌شود در حالی کهLinearSVC از یکی در مقابل بقیه استفاده می کند که در نهایت اگر ورودی پیوسته باشد یک svc می تواند با داده های متراکم هماهنگ باشد.

در حالت کلی کد استفاده شده در الگوریتم linear svc اینگونه عمل مینماید که ابتدا تابعی را تعریف نموده سپس الگوریتم را بر اساس پارمترهای خاص خود تشکیل می دهد . ما با استفاده از دستور pickle کلیه ی مدل های ساخته شده را ذخیره کرده و در مرحله بعد محاسبات از قبیل دقت(precision_recall_fscore[۲۵]) انجام شده و نتیجه و اینکه میانگیری به چه صورتی باید باشد را نمایش می دهد.

---

[۲۴] Multiclass

[۲۵] معیارهای ارزیابی امتیازدهی و دقت و صحت

۱-۱-٤-۲- مزایا:

- در فضاهایی با بعد زیاد، خوب جواب می دهد . یعنی در جاهایی که تعداد ویژگی هایمان زیاد باشد این الگوریتم بسیار مؤثر است.
- در زمانی که تعداد بعد بیشتر از تعداد نمونه ها باشد مؤثر عمل می کند .
- در بعدهای زیاد و همچنین در زمانی که تعداد داده های آموزشی مان نسبت به بعد کم باشد بسیار مؤثر است .
- توابع تصمیم گیری[۲۶] مختلفی دارد که در زیرمجموعه آموزشی[۲۷] در تابع تصمیم‌گیری استفاده می شود.
- حافظه قوی دارد.
- کرنل، تابع هایی که دارد در انتخاب تصمیم مفید است.

۲-۱-٤-۲- معایب:

- اگر تعداد feature ها و مشخصه هایمان، تعداد ابعاد xtrain و ytrain بیشتر از تعداد sample ها باشد برای جلوگیری از overfitting انتخاب نوع function و c بسیار حساس و مهم هستند که احتمال دارد به مشکل برخورد کنیم . پس زمانی که تعداد مشخصه هایمان بسیار بیشتر از تعداد نمونه هایمان باشد یا اینکه ابعادمان بیشتر از نمونه ها باشد، مفید است.

سه تا کرنل مهم دارد:

۱. کرنل های خطی : که با خط راست دیتاها را جدا میکند .
۲. کرنل های قوسی یا rbf :که با تابع های شعاعی(قوسی) دیتاها را از هم جدا می کند. یعنی یک سری موارد مانند نمودارهای قوسی باعث جداکردن دیتا می شود .
۳. کرنل های چندجمله ای : که با خطوط چندجمله ای دیتاها را از هم جدا می کند .

- ایراد دیگری که دارد این است که تخمین احتمالات برایش مهم نیست و با استفاده از five-fold-

[۲۶] Decision function

[۲۷] Train Set

cross-validation[۲۸] کار می کند و بر اساس تابع تخمین اطلاعات کار نمی کند.

۳-۱-٤-۲- پارامترهای مهم *linear SVC*:

- penalty:string,'L1' or 'L2'(default='L2')

نوعی پنالتی است که به دو نوع L1 و L2 میباشد .حالت پیش فرض آن L2 است و ازآن بهترین جواب بدست می آید . اگر L2 را جایگزین کنیم بهترین جواب را (ACCURACY[۲۹]=0.777) خواهیم داشت.

- C:float,optional(default=1.0)

فرمول محاسبه c به صورت زیر می باشد:

$$C\sum_{i=}^{n}\mathcal{L}(f(x_i),y_i)+\Omega(\omega) \qquad (۲-۱)$$

مقدار ذاتی c همان یک است . هرچقدر c بیشتر باشد، حول همسایه ها می چرخد و اگر سطحی که در می آید کمتر باشد، سطح تخت می شود(هرچه بیشتر باشد سطح خروجی پستی و بلندی بیشتری دارد) . مقدار C هر چقدر کمتر باشد میزان همواری بیشتر است . مقدار این پارامتر بسته به نوع دیتاستی است که ما دراختیار داریم.

- dual: در این الگوریتم مسئله ما مقدار dual نیست یعنی این معیار برای پارامترهای دومجهولی است اما در اینجا ما یک مجهول داریم و نیازی به تغییر این پارامتر نمی باشد.
- tol چیست؟ tol یعنی چقدر تابع optimization مان یعنی svc برای بهینه سازی است. این تابع بر اساس بهینه سازی عمل می کند . حال باید دید اگر چقدر تغییر کند ما آن را قبول می کنیم؟ یعنی از آن حد کمتر تغییر کند سیستم را break می کند . اگر عدد به اندازه tol=0.0025 کوچک باشد دقت بیشتر می شود و جواب بهتری خواهیم گرفت اما زمان محاسبات بالاتر خواهد رفت.

یکی دیگر از پارامترهایی که می تواند svc بگیرد به شکل زیر است:

- loss:string,'hinge' or 'squared_hinge'(default='squared_hinge')

که تعیین کننده تابع loss است . همانطور که در فرمول c آمده، L یا همان loss پارامتری است که نسبت f(x) به y را تعیین می کند . دیفالت تابع loss بر اساس میانگین مربعات است . اگر بر اساس میانگین مربعات یا توان ۲ مربعات نخواستیم می توانیم از hinge استفاده کنیم که تنها استاندارد را انجام بدهد.

---

[۲۸] اعتبارسنجی متقابل پنج برابر

[۲۹] صحت

- dual:bool,(default=true)

زمانی که بیش از یک پارامتر بهینه سازی داریم یعنی دو یا بیشتر پارامتر بهینه سازی داریم.

- tol:foat,optinal(default=1e-4)

که همان tolerance است که به عنوان یک مدرک stopping مورد استفاده قرار می گیرد.

- c:float,optional(default=1.0)

Float c که یک پارامتر پنالتی برای error است.

- multi_class:string ‘ovr’ or ‘crammer_singer’(default=’ovr’)

چندکلاسه برای وقتی است که کلاس دو دسته نباشد و کلاس ها از چنددسته متفاوت باشند . مثلا دسته یک، دسته دو و .... از چند کلاسه بر اساس one-vs-res (یکی و بقیه) استفاده می شود . وقتی از multi class system استفاده می کند یعنی اینگونه عمل می کند که مثلا آنهایی که پارامتر یک دارند را یک می کند و بقیه را صفر بعد می آید آنهایی که پارامتردو دارند را یک می کند و بقیه را صفر و به همین ترتیب ادامه می دهد تا آخر و بر اساس همین روش، داده ها را آموزش می دهد.

- fit_intercept:Boolean,optional(default=true)

که می توان گفت بر اساس بولین باشد یا نه که معمولاً بر اساس بولین عمل می کند.

- max_iter:int,(default=1000)

این جزو مهمترین پارامترها است . بیشترین میزان تکرار را نشان می دهد . اگر پارامتر tol با مقدار مشخص شده فوق به نتیجه نرسد (یعنی میزان تخت بیشتر از حد باشد) در اینجا max-iter می تواند آن را بشکند . هم برای اینکه میزان تخت بودن کم باشد و هم برای جوابهای بهتر استفاده می شود و زمان محاسباتی بالا است . باید هم max-iter عدد بالایی باشد و هم desineflprance عدد کمتری باشد ، زمان محاسباتی بالاتر اما جواب بهتر و درست تری به ما می دهد . پارامترهای دیگر تأثیر چندانی در جواب بهینه ندارند.

حال ما با تنظیم پارامترها به صورت (C=2.5, penalty="l2", dual=**False** ,tol=0.0025) توانستیم صد مدل با استفاده train با استفاده از tfidf بسازیم که خروجی آنها همانند جدول ذیل می باشد[13][12]:

Bow binary=f-clf0=0.7676200204
Bow=f-clf1=0.6755771195
Tfidf=fclf2=0.7946884576

f-clf	مقدار
۳	۰,۷۷۳۲۳۷۹۹۷۹

٤	٠,٦٧٨١٣٠٧٤٥٦
٥	٠,٧٦٣٥٣٤٢١٨٥
٦	٠,٧٧٤٢٥٩٤٤٨٤
٧	٠,٧٦٧٦٢٠٠٢٠٤
٨	٠,٧٦٥٠٦٦٣٩٤٢
٩	٠,٧٦٦٥٩٨٥٦٩٩
١٠	٠,٧٧٢٢١٦٥٤٧٤
١١	٠,٧٧١٧٠٥٨٢٢٢
١٢	٠,٧٧٤٧٧٠١٧٣٦
١٣	٠,٧٧١٧٠٥٨٢٢٢

٢-٤-٢- شبکه بیزین

دومین الگوریتمی که در این پژوهش از آن جهت دسته بندی استفاده شده است شبکه بیزین می باشد .

در روش‌های دسته‌بندی آماری برخلاف سایر دسته‌بندها میزان عضویت یک نمونه به هر کلاس را با یک احتمال نشان می‌دهند. روش شبکه‌های بیزین رایج‌ترین روش دسته‌بندی آماری و از روش‌های ساده و موثر محسوب می‌شود . در این روش احتمال شرطی هر صفت داده شده را توسط برچسب دسته مربوطه از داده‌های آموزشی یاد می‌گیرید . سپس عمل دسته‌بندی توسط بکار بردن قوانین بیز برای محاسبه مقدار احتمالی دسته نتیجه نمونه داده شده با دقت بالایی انجام می‌شود . در حالت معمولی این کار با تخمین احتمالاتی هر ترکیب ممکن از صفات صورت می‌گیرد ولی هنگامی که تعداد صفات خیلی زیاد باشد، این امر امکان پذیر نیست. بنابراین یک فرض مستقل قوی اتخاذ می‌شود که همه صفات با مشخص بودن مقدار صفت دسته مستقل می‌باشند. با در نظر گرفتن این فرض لازم است که فقط احتمالات حاشیه‌ای هر صفت دسته محاسبه گردد. با این حال این فرض به صورت غیرواقعی می‌باشد و شبکه‌های بیزین با مدل کردن صریح، وابستگی بین صفات آن را در نظر نمی‌گیرند[14].

مسأله یادگیری ساختار شبکه بیزین به این صورت بیان می‌شود که با داشتن یک مجموعه آموزشی $A=\{u_1,u_2,...,u_n\}$ از n نمونه u؛ یک شبکه پیدا کنیم که بهترین تطبیق را با A داشته باشد . معمول‌ترین روش

برای این مسأله معرفی یک تابع هدف است که هر شبکه با توجه به داده‌های آموزشی و جستجوی بهترین شبکه بر اساس این تابع ارزیابی شود . چالش‌های بهینه‌سازی کلیدی انتخاب تابع هدف و تعیین روال جستجو برای بهترین شبکه می‌باشد .

تئوری بیز یکی از روش های آماری برای رده بندی به شمار می آید. در این روش کلاس های مختلف، هر کدام به شکل یک فرضیه دارای احتمال در نظر گرفته می شوند. هر رکورد آموزشی جدید، احتمال درست بودن فرضیه های پیشین را افزایش و یا کاهش می دهد و در نهایت، فرضیاتی که دارای بالاترین احتمال شوند، به عنوان یک کلاس در نظر گرفته شده و برچسبی بر آن ها زده می شود. این تکنیک با ترکیب تئوری بیز و رابطه سببی بین داده ها، به دسته بندی می پردازد . روشی برای دسته‌بندی پدیده‌ها، بر پایه احتمال وقوع یا عدم وقوع یک پدیده‌است و در نظریه احتمالات با اهمیت و پرکاربرد است . اگر برای فضای نمونه‌ای مفروضی بتوانیم چنان افرازی انتخاب کنیم که با دانستن اینکه کدامیک از پیشامدهای افراز شده رخ داده‌است، بخش مهمی از عدم قطعیت تقلیل می یابد . این قضیه از آن جهت مفید است که می‌توان از طریق آن، احتمال یک پیشامد را با مشروط کردن نسبت به وقوع یا عدم وقوع یک پیشامد دیگر محاسبه کرد . در بسیاری از حالت‌ها، محاسبه احتمال یک پیشامد به صورت مستقیم کاری دشوار است . با استفاده از این قضیه و مشروط کردن پیشامد مورد نظر نسبت به پیشامد دیگر، می‌توان احتمال مورد نظر را محاسبه کرد.

این رابطه به خاطر بزرگداشت توماس بیز فیلسوف انگلیسی به نام فرمول بیز معروف است.

به طور خلاصه قضیه بیز به صورت زیر عمل می نماید:

زمانی که از قبل وقوع یک پیشامد تصادفی را بدانیم، به کمک فرمول‌های احتمال شرطی می‌توانیم مقدار احتمال برای هر پیشامد دیگر را محاسبه کنیم .

۴-۲-۴-۲- پارامترهای *NB* :

- priors:array_lik,shape(n_class)

ورودی هایی که دارد ورودی های خاصی هستند.ورودی هایی با عنوان توزیع احتمالات کلاس های قبلی است.

- var_smooting:float.optional(default=1e-9)

پارامتر هموار سازی است به نوعی پارامتر هموار سازی هر چقدر بیشتر باشد،بهتر است.طبق دیتاست جواب بهینه به ما خواهد داد.اما نسبت بیشترین variance به عنوان پارامتر پایداری در نظر می گیرد.دقت کنید که

هرچقدر پارامتر پایدارسازی بیشتر باشد سیستم smoot تر می شود (همان مقدار 1e-9)[1][15][16]

ما با استفاده از این الگوریتم صد مدل داده train با استفاده از tfidf ساخته ایم که یک نمونه از آن به شرح جدول ذیل میباشد:

Bow binary=f-clf103=0.6603677221
Bow=f-clf104=0.6613891726
Tfidf=f-clf105=0.6736465781

f-clf	مقدار
106	0.6068028600
107	0.6869254341
108	0.6675178753
109	0.6741573033
110	0.6603677221
111	0.6680286006
112	0.6527068437
113	0.6690500510
114	0.6721144024
115	0.6629213483
116	0.6690500510

۲-٤-۳- رگرسیون منطقی

یک مدل آماری رگرسیون برای متغیرهای وابسته دوسویی مانند بیماری یا سلامت، مرگ یا زندگی است . این مدل را می‌توان به عنوان مدل خطی تعمیم یافته‌ای که از تابع لوجیت به عنوان تابع پیوند استفاده می‌کند و خطایش از توزیع چندجمله‌ای پیروی می‌کند، به‌حساب‌آورد . منظور از دو سویی بودن، رخ داد یک واقعه تصادفی در دو موقعیت ممکنه است . به عنوان مثال خرید یا عدم خرید، ثبت نام یا عدم ثبت نام، ورشکسته شدن یا ورشکسته نشدن و ... متغیرهایی هستند که فقط دارای دو موقعیت هستند و مجموع احتمال هر یک آن‌ها در نهایت یک خواهد شد.

این الگوریتم برای برآورد مقادیر گسسته (مقادیر باینری مانند ۰/۱، آری/خیر، درست/نادرست) بر اساس مجموعه‌ای از متغیر (های) وابسته استفاده می‌شود . به بیان ساده‌تر این الگوریتم احتمال رخداد یک رویداد را بر حسب گنجاندن داده‌ها در یک تابع منطقیی پیش‌بینی می‌کند . از این‌رو به نام رگرسیون logit نیز شناخته می‌شود. از آنجا که این الگوریتم، احتمال را پیش‌بینی می‌کند، مقادیر خروجی آن بین ۰ تا ۱ هستند

اگر از دید ریاضیاتی به این الگوریتم نگاه کنیم خروجی به صورت یک ترکیب خطی از متغیرهای پیش‌بین مدل‌سازی شده است.

روش‌های سنتی شناسایی نویسنده نیازمند یک روش یادگیری دسته‌بندی ضعیف و پیاده‌سازی بسیار مقیاس پذیر هستند . در مقابل، می توان به دلیل ماهیت احتمالی آن به رگرسیون منطقی چند جمله‌ای رفت . از آنجا که این مدل تخمینی از احتمال تعلق ورودی به هر یک از کلاس‌های ممکن را نشان می‌دهد ، می‌توان به راحتی هزینه‌های نسبی واحدهای مختلف را هنگام تصمیم‌گیری طبقه‌بندی در نظر گرفت . اگر این هزینه‌ها تغییر کنند، طبقه‌بندی می‌تواند بدون آموزش مجدد مدل تغییر کند .

به علاوه ، دیدگاه Bayesian درباره آموزش یک مدل رگرسیون چند جمله‌ای باعث می‌شود اطلاعات آموزشی و دانش حوزه به راحتی ترکیب شوند . رگرسیون منطقی به طور گسترده در طبقه‌بندی متن استفاده شده‌است . در حالی که طرفدار ارزش‌های نزدیک است، موافق نیست که دقیقاً برابر با صفر باشد . از آنجا که مدل‌های رگرسیون چند جمله‌ای برای شناسایی مولف می‌تواند به راحتی میلیون‌ها پارامتر داشته باشد، چنین تخمین پارامتر متراکم می‌تواند منجر به طبقه‌بندی کننده‌های ناکارآمد شود . با این حال برآورده‌ای پارامترهای پراکنده را می‌توان به سادگی در چارچوب Bayesian به دست آورد [15][17].

ما با استفاده از این الگوریتم صد مدل داده train با استفاده از tfidf ساخته ایم که یک نمونه از آن به شرح جدول ذیل میباشد:

Bow binary=f-clf206= 0.7967313585
Bow=f-clf۲۰۷=0.7941777323

Tfidf=f-clf۲۰۸=0.8105209397

f-clf	مقدار
209	0.7860061287
210	0.7773237997
211	0.7808988764
212	0.7900919305
213	0.7911133810
214	0.7839417773
215	0.7819203268
216	0.7911133810
217	0.7921348314

218	0.7951991828
219	0.7870275791

۲-٤-٤- درخت های اضافی

یک طبقه‌بندی کننده درخت اضافی است . این کلاس یک estimator[۳۰] متا را پیاده می‌کند که با تعدادی از درخت‌های تصمیم‌گیری تصادفی متناسب است . که در زیر مجموعه داده‌ها چند نمونه از مجموعه داده‌ها استفاده شده‌است و از میانگین گیری برای بهبود دقت پیش‌بینی و کنترل بیش از حد استفاده می‌کند .

این الگوریتم یک طبقه‌بندی کننده درخت بسیار تصادفی است . درخت‌های اضافی از درخت‌های تصمیم‌گیری کلاسیک تفاوت دارند . هنگامی که به دنبال بهترین شکاف برای جدا کردن نمونه‌های یک نود به دو گروه بگردید ، splits تصادفی برای هر کدام از ویژگی‌ها انتخاب می‌شوند که به طور تصادفی ویژگی‌ها را انتخاب می‌کنند و بهترین شکاف بین آن‌ها انتخاب می‌شود.حداکثر ویژگی‌ها ۱ تنظیم می‌شود ، این مقدار برای ساخت یک درخت تصمیم‌گیری کاملاً تصادفی است .

هشدار : درختان اضافی تنها باید در روش‌های گروهی مورد استفاده قرار گیرند .

مقادیر پیش‌فرض برای پارامترهایی که اندازه درختان را کنترل می‌کنند (به عنوان مثال عمق ، نمونه _ نمونه‌های ، برگ و غیره) منجر به درخت‌های کاملاً بالغ و unpruned می‌شوند که می‌تواند به طور بالقوه در برخی از مجموعه‌های داده بسیار بزرگ باشد . برای کاهش مصرف حافظه، پیچیدگی و اندازه درختان باید با تنظیم مقادیر پارامتر کنترل شوند [18][19].

ما با استفاده از این الگوریتم صد مدل داده train با استفاده از tfidf ساخته ایم که به شرح جدول ذیل میباشد:

Bow binary=f-clf308= 0.65985699
Bow=f-clf309=0.6813074565
Tfidf=f-clf310=0.6884576098

f-clf	مقدار
311	0.6716033677
312	0.6894790602

[۳۰] تخمین زننده ،برآورد گر

313	0.6782431052
314	0.6710929519
315	0.6705822676
316	0.6797752808
317	0.6782431052
318	0.6889683350
319	0.6613891726
320	0.6552604698
321	0.6552604698

۲-۵- یادگیری تجمعی

یکی از اهداف اولیه در داده کاوی، پیش بینی یک مقدار نامعلوم از یک نمونه جدید براساس نمونه های مشاهده شده قبلی است . دستیابی به چنین نتیجه ای با دو گام حاصل می شود :

الف) مرحله آموزش:

ایجاد یک مدل پیش بینی از نمونه های آموزشی با استفاده از یکی از الگوریتم های یادگیری باناظر

ب) مرحله آزمون:

در گام بعدی مدل ایجاد شده با استفاده از نمونه های آزمایش، مدل مورد ارزیابی قرار میگیرد. تجربیات گوناگون از بکارگیری روشهای آموزش نشان میدهد که هیچ تک الگوریتم آموزشی مشخصی وجود ندارد که بتواند برای تمامی کاربردها بهترین و دقیقترین باشد. در واقع هر الگوریتم، مدل خاصی است که بر پایه مفروضات معینی شکل گرفته است. برخی مواقع این مفروضات برقرار بوده و گاهی نیز نقض میشوند. بنابراین، هیچ الگوریتمی به تنهایی نمیتواند درتمامی شرایط و برای همه بطور موفقی عمل نماید. برای رفع این مشکل روش آموزش دسته جمعی معرفی شده است.

انگیزه ی اصلی بر توسعه چنین روشی، کاهش نرخ خطا می باشد. فرض مبنایی این متدولوژی آن است که در حالت دسته جمعی احتمال اشتباه در تشخیص دسته یا جایگاه یک نمونه جدید خیلی کمتر از حالت پیش بینی با تنها یک مدل میباشد.

این ایده توسط هنسن بصورت نظری به ترتیب زیر اثبات شده است :

اگر N دسته بند مستقل با احتمال خطای $e<0.5$ داشته باشیم ، در این صورت میتوان نشان داد که خطای دسته جمعی E بر حسب N بصورت یکنواخت کاهش پیدا میکند. همچنین اگر از دسته بندهای وابسته استفاده کنیم، عملکرد کلی بشدت کاهش می یابد.

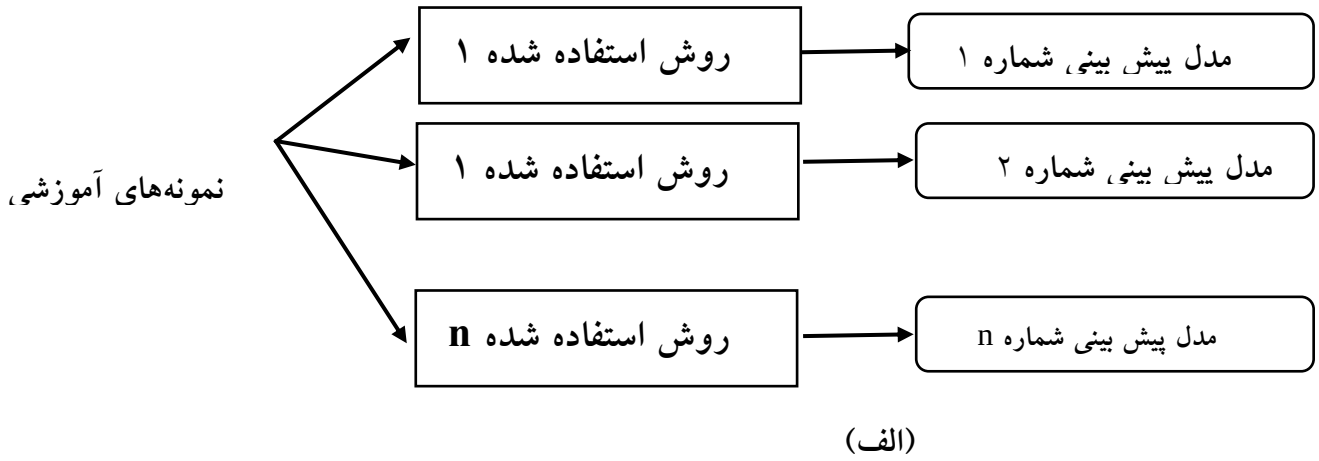
۱-۵-۲- متدولوژی آموزش دسته جمعی

متدولوژی آموزش دسته جمعی از دو مرحله متوالی تشکیل شده است :

الف) مرحله آموزش

ب) مرحله آزمایش

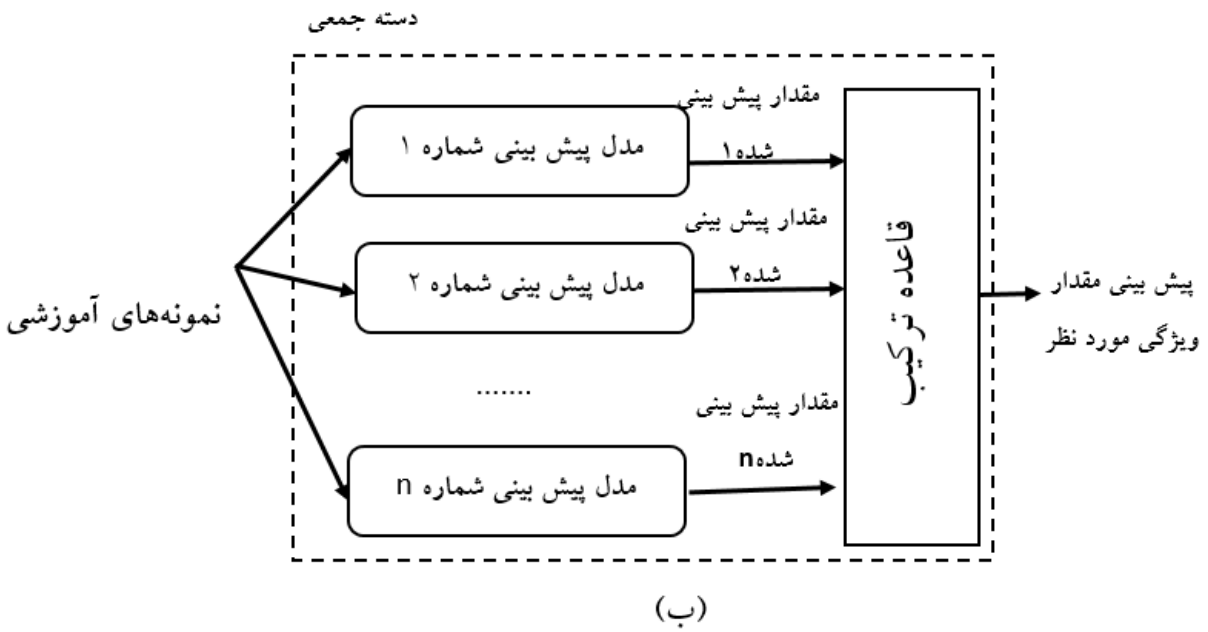
در مرحله آموزش روش دسته جمعی، چندین مدل پیش بینی با استفاده از نمونه های آموزش تولید میشوند .



(الف)

۲- ۱- مرحله آموزش روش دسته جمعی

در مرحله بعد، برای پیش بینی دسته یا جایگاه نمونه آزمایشی، روش دسته جمعی، خروجی تک تک مدل های پیش بینی را محاسبه و با یکدیگر تجمیع می کند .



۲- ۲- مرحله آزمایشی روش دسته جمعی

برای اینکه مدل دسته جمعی عملکرد بهتری نسبت به مدل تکی داشته باشد، باید هر کدام از مدل های تشکیل دهنده آن مستقل از بقیه بوده و دارای دقتی بزرگتر از ۰,۵ باشد .

شرط استقلال مدل ها مانع از همبسته شدن خطای مدل ها خواهد شد . در نهایت برای تشخیص دسته یا جایگاه نمونه آزمایشی، خروجی همه مدل ها با یکدیگر تجمیع می شوند .مدل دسته جمعی، تصمیم گیری در مورد دسته نمونه آزمایشی را با توجه به بیشترین تکرار در میان خروجی دسته بندها، انجام میدهد[23].

۲-٦ – انواع روش ترکیب دسته بندها

برای ترکیب دسته بندها روش های متفاوتی وجود دارد ،در این پژوهش روش های ترکیب زیر مورد بررسی قرار داده شد.

۱-٦-۲- ترکیب همه دسته بندها

در این نوع ترکیب برای هر نمونه با توجه به صورت مسئله تعدادی دسته بند وجود دارد که با توجه به خروجی هر دسته بند، خروجی که بیشترین تکرار را داشته باشد به عنوان خروجی نهایی در نظر گرفته می شود.

۲-٦-۲- انتخاب رو به جلو[۳۱]

انتخاب رو به جلو یک نوع رگرسیون گام به گام است که با یک مدل خالی شروع می شود و دسته بندهایی را که به صورت نزولی بر مبنای یکی از معیارهای ارزیابی مرتب شده اند را یک به یک اضافه می کند و بهترین متغیر ، با برخی از معیارهای از پیش تعیین شده تعیین می شود و به مدل اضافه می گردد. معیار استفاده شده برای تعیین اینکه کدام مدل اضافه شود ، متفاوت است. در هر مرحله رو به جلو، شما یک دسته بند را اضافه می کنید که بهترین نتیجه را برای شما به ارمغان می آورد این یکی از دو روش معمول استفاده از رگرسیون گام به گام است. تغییر بیش از حد زمانی اتفاق می افتد که ما متغیرهای بیشتری را در اختیارمان قرار دهیم تا اینکه برای مدل مناسب باشد. به طور معمول یک اطلاعات بسیار دقیق از داده های مورد استفاده در رگرسیون را نشان می دهد، اما این مدل از نقاط داده های اضافی دور خواهد ماند و برای اینترلوسیون خوب نیست.

در این روش مدل ها بر مبنای F_score به صورت نزولی مرتب شد، در ابتدا اولین مدل مورد ارزیابی قرار گرفت و سپس مدل دوم با مدل اول ترکیب شد و ترکیب این دو، مورد ارزیابی قرار گرفت اگر f-score ترکیب این دو مدل بهتر از مدل اول به تنهایی باشد که ترکیب دو مدل را ذخیره می کنیم و مدل بعد را اضافه می کنیم در غیر این صورت مدل دوم حذف می شود و سومین مدل ترکیب می شود ،به همین ترتیب تا آخرین مدل(۱۰۰ مدل) انتخاب گردید .

۳-٦-۲- حذف رو به عقب[۳۲]

در این روش ابتدا مدل ها را برمبنای F_score به صورت صعودی مرتب کردیم سپس همه ی مدل ها با هم ترکیب شد .ترکیب مدل ها ارزیابی شد و اولین مدل را حذف کردیم و نتیجه را مورد ارزیابی قرار دادیم به همین ترتیب تا آخرین مدل ادامه دادیم[24].

---

[۳۱] Forward selection

[۳۲] Backward elimination

۲-۷- معیارهای ارزیابی دسته بندها

معیارهای زیادی برای ارزیابی کارایی الگوریتم‌های دسته‌بندی ارائه می‌شود که مهم‌ترین این معیارها عبارتند از: نرخ صحت[۳۳] دسته‌بندی، سرعت یادگیری در مرحله آموزش و دسته‌بندی، سادگی و شفافیت مدل، پایداری (توانایی مدل در مواجهه با داده‌های غیر معمول یا مقادیر مفقوده)، نحوه برخورد الگوریتم با صفت‌ها با دامنه مقادیر مختلف (پیوسته گسسته و دودویی) و قابلیت تفسیر.

در این مرحله از مدل های ساخته شده در بخش قبلی روی داده های توسعه ارزیابی انجام گرفت، برای هر مدل F_score[۳٤] , precision[۳۵] و recall[۳٦] محاسبه شده و مدل ها در یک آرایه بر مبنای F_score به صورت نزولی مرتب و بهترین مدل انتخاب شد که در ذیل به روشهای ارزیابی اشاره ای خواهیم داشت.

دقت و صحت عبارت‌هایی هستند که برای بیان نتیجه اندازه‌گیری به وفور مورد استفاده قرار می‌گیرند. در گفتگوهای روزانه، بسیار پیش می‌آید که این دو عبارت را به اشتباه به جای هم به کار می‌بریم. در حالی که هریک مفهوم مشخصی دارند و باید در موقعیت مناسب خود استفاده شود.. برای شروع، بهتر است هریک از این اصطلاحات را جداگانه تعریف کنیم.

۱-۷-۲- نگاهی دقیق‌تر به مسائل دسته‌بندی

در مسائل دسته‌بندی به دنبال پیش‌بینی دسته یا گروهی هستیم که یک رکورد یا داده به آن تعلق دارد. مثلا می‌خواهیم با داشتن مشخصات مختلف یک دانشجو مانند رشته، سهمیه قبولی، وضعیت مالی خانواده، تعداد هم اتاقی‌ها ، وضعیت تحصیلی دوران دبیرستان، تعداد واحدهای گذرانده، معدل ترم‌های پیش، تعداد ترم‌های مشروطی و مانند آن تعیین کنیم که این دانشجو مشروط خواهد شد یا نه . یا به عبارتی به دسته مشروطی‌ها تعلق دارد یا دسته دانشجویان غیرمشروط.

اینکه احتمال دیابت داشتن یک بیمار با توجه به آزمایش‌های انجام شده چقدر است یعنی این بیمار جزء دسته دیابتی‌ها قرار میگیرد یا افراد غیردیابتی، باز هم به یک مدل یا الگوریتم دسته‌بندی نیاز داریم.

---

[۳۳] Accuracy

[۳٤] نرخ امتیاز

[۳۵] دقت

[۳٦] یاداوری

همانطور که از دو مثال فوق مشاهده می‌کنید، دسته‌بندی‌های دوگانه بیشترین کاربرد را در دنیای واقعی دارند و ما هم برای ساده کردن مطلب، تمرکز را بر این نوع دسته‌بندی خواهیم گذاشت اما تمامی موارد بیان شده، به راحتی قابل تعمیم به دسته‌بندی های چندگانه هم خواهند بود.

بعد از ساخت یک مدل دسته‌بندی یعنی یافتن الگوریتمی که با مشاهده یک داده جدید، دسته یا گروه (Class) آنرا مشخص کندعمل(Classification[۳۷]) ، برای سنجش میزان کارآیی و دقت مدل پیشنهادی، آنرا بر روی داده-های آموزشی یعنی داده‌هایی که از قبل دسته‌بندی آنها را می‌دانیم، اعمال می‌کنیم.

خروجی این مدل را با خروجی واقعی به صورت زیر مقایسه خواهیم کرد:

		مقادیر واقعی	
		مثبت	منفی
مقادیر پیش بینی شده	مثبت	درست مثبت TP	نادرست مثبت FP خطای نوع یک
	منفی	نادرست منفی FN خطای نوع دو	درست منفی TN

۲- ۳- ارزیابی کارایی یک مدل

به این جدول مهم که اساس تحلیل و ارزیابی کارآیی یک مدل در مباحث دسته‌بندی است، ماتریس درهم‌ریختگی یا اغتشاش گفته می‌شود . اما با توجه به نامفهوم بودن این ترجمه، از عبارت ماتریس پراکنش که پراکندگی توزیع دسته ها را از لحاظ درستی یا نادرستی نمایش می‌دهد، در ادامه این سلسله آموزش ها برای اشاره به این ماتریس استفاده خواهد شد.

تعداد سطرها و ستون‌های این ماتریس به تعداد دسته‌ها وابسته است. جدول فوق یک ماتریس دو در دو را برای یک مساله دسته بندی دوگانی مثلا دیابت داشتن یا نداشتن، مشروط شدن یا نشدن، ترک کار یا عدم ترک کار نشان می دهد. در حالت کلی برای یک مساله دسته‌بندی n حالته، یک ماتریس پراکنش n*n باید رسم شود.

در حالت کلی به دنبال این هستیم که بخشهای قرمز رنگ این ماتریس یعنی پیش‌بینی های نادرست ، به حداقل برسد.

[۳۷] دسته بندی

با فرض اینکه هدف ما پیش‌بینی دیابت یک بیمار باشد یعنی اگر پیش‌بینی مثبت باشد یعنی بیمار، مبتلا به دیابت است و اگر پیش بینی منفی باشد، یعنی بیمار به دیابت مبتلا نیست، به تحلیل سلول‌های این ماتریس میپردازیم :

- درست مثبت[۳۸] : (True Positive-TP) اگر بیمار واقعا دیابت داشته باشد و مقدار پیش‌بینی شده هم دیابت را نشان دهد.
- نادرست مثبت[۳۹] : (FP) اگر بیمار دیابت نداشته باشد اما نتیجه پیش بینی ما، نشانگر دیابت بیمار باشد.
- نادرست منفی[۴۰] : (FN) اگر بیمار دیابت داشته باشد اما پیش بینی ما، دیابت را منفی نشان دهد.
- درست منفی[۴۱] : (TN) اگر بیمار دیابت نداشته باشد و پیش بینی ما هم همین را نشان بدهد.

همانطور که مشخص است ایده آل ما این است که موارد نادرست (نادرست مثبت و نادرست منفی) صفر باشند اما در عمل این اتفاق نمی‌افتد و نیازمند مکانیزم‌ها و معیارهایی برای بررسی دقت و صحت و کارآیی مدل ایجاد شده از داده ها هستیم.

۲-۷-۲- دقت – صحت – بازخوانی

اولین معیار یا سنجه‌ای که به ذهن مان می‌رسد، معیار **دقت** یا **میزان تشخیص درست** مدل است. یعنی نسبت تشخیص های درست (TP+TN) به کل داده ها

$$\text{دقت (Accuracy)} = \frac{TP+TN}{N} = \frac{\text{تشخیص‌های درست}}{\text{کل داده‌ها}}$$

---

[۳۸] True Positive

[۳۹] False Positive

[۴۰] False Negetive

[۴۱] True Negetive

برای بسیاری از مسائل دسته‌بندی دنیای واقعی این معیار، بسیار کارآمد است چون هم داده‌های در نظر نگرفته شده را لحاظ کرده است (مخرج کسر) و هم داده‌های شناسایی شده را (صورت کسر). هدف ما هم رسیدن این عدد به مقدار یک یا همان صد در صد است. اما اگر با ادبیات یادگیری ماشین آشنا باشید این معیار چندان برایتان آشنا نخواهد بود و امروزه کمتر مورد استفاده قرار می‌گیرد.

این معیار، برای داده های نامتعادل یعنی داده هایی که تعداد برچسب های مثبت و منفی آن در دنیای واقعی از لحاظ عددی اختلاف بسیار زیادی دارند، معیار مناسبی نیست. بسیاری از مسائل دنیای واقعی هم دقیقاً جزء این گروه قرار می گیرند. اگر قرار باشد ابتلا به ایدز را از روی آزمایشات مختلف برای هزار نفر تشخیص دهید، شاید یک یا دونفر از این بین، به ایدز مبتلا باشند که اختلاف زیادی بین دسته مثبت (افراد دارای ایدز) و دسته منفی (افراد سالم) حاکم است. مسایلی مانند تشخیص هرزنامه بودن یک ایمیل، تروریست بودن یک مسافر هواپیما، شناسایی دانشجویان مشروط ، پیش‌بینی خروج یک کارمند از شرکت و مانند آن، نمونه‌های دیگری از مسایل دسته‌بندی با دسته‌های نامتعادل در دنیای واقعی هستند. این اختلاف معنی دار بین دسته‌های مختلف داده‌ها، باعث عدم کارآیی معیار دقت می‌شود.

مشکل اصلی هم نامتعادل بودن داده ها و تفاوت معنی دار تعداد نمونه های هر دسته است که باعث می‌شود یک مدل متمایل به دسته پرتعداد، دقت کلی را بالا نشان دهد. بنابراین نیاز به معیاری دقیق تر برای سنجش دقت و کارآیی الگوریتم های پیشنهادی دسته بندی هستیم.

در این گونه مسایل بهتر است بر تعداد نمونه های مثبت شناسایی شده به کل نمونه های مثبت تمرکز کنیم. یعنی ببینیم از دونفر بیمار مبتلا به ایدز، چند نفر شناسایی شده اند. معیار صحت را برای این منظور به صورت زیر تعریف می کنیم:

$$\text{بازخوانی (Recall)} = \frac{\text{تعداد های نمونه تشخیصی درست مثبت}}{\text{کل های نمونه واقعاً مثبت}} = \frac{TP}{TP+FN}$$

توضیح اینکه کل نمونه های واقعاً مثبت شامل نمونه هایی است که درست، مثبت شناسایی شده اند (TP)و نمونه هایی که مثبت بوده اند اما نادرست، منفی شناسایی شده اند.(FN)

سنجش بازخوانی برای روش پیشنهادی تشخیص ایدز، صفر است (چون هیچ نمونه مثبتی را شناسایی نکرده ایم – صورت کسر برابر صفر است) که نشانگر ضعیف بودن مدل پیشنهادی است و بنابراین آنرا می توانیم به راحتی رد کنیم.

۲-۷-۳- الگوریتم (مدل) پیشنهادی

تمام نمونه ها را مثبت اعلام کن! در این صورت تمام دو نفر بیمار ایدز را تشخیص داده ایم . یعنی بازخوانی ما برابر حداکثر ممکن یعنی ۱ شده است. توضیح اینکه تعداد داده های درست تشخیص داده شده برابر ۲ و تعداد داده های نادرست منفی اعلام شده) کسانی که ایدز دارند اما نتیجه آزمایش ایدز آنها را منفی اعلام کرده ایم – (FN، برابر صفر است و تقسیم دو بر دو هم که یک می شود. یعنی گاهی بازخوانی ما به خاطر ضعیف بودن مدل پیشنهادی، بالاست. این ضعیف بودن را با معیار دیگری باید اندازه بگیریم.

برای حل این مشکل، در کنار معیار بازخوانی معیار دیگری را به نام صحت(Precision) ، برابر تعداد نمونه های تشخیصی درست مثبت به کل نمونه های مثبت اعلام شده به صورت زیر تعریف می کنیم تا میزان مثبت های اشتباه را هم در نظر گرفته باشیم:

$$\text{صحت (Precision)} = \frac{\text{تعداد های نمونه تشخیصی درست مثبت}}{\text{تعداد کل های نمونه تشخیصی مثبت}} = \frac{TP}{TP+FP}$$

در این فرمول، وجود FP در مخرج باعث می شود که اگر تعداد تشخیص های اشتباه مان بالا باشد، صحت الگوریتم عددی نزدیک به صفر نشان دهد و بنابراین کارآیی مدل، زیر سوال برود.با این توضیحات، معیارهای بازخوانی و صحت به جای معیار اولیه دقت، کاربرد وسیع تری در دنیای امروز یادگیری ماشین پیدا کرده است.در اغلب موارد، این دو معیار با هم رشد و حرکت نمی کنند. گاهی ما صحت مدل را با الگوریتم های دقیق‌تر بالا می بریم، یعنی آنهایی را که مثبت اعلام می کنیم، اکثراً درست هستند و موارد نادرست مثبت ما بسیار کم هستند یعنی صحت الگوریتم ما بسیار بالاست اما ممکن است جنبه یا ویژگی خاصی از داده ها را در نظر نگرفته باشیم و تعداد کل نمونه های مثبت، بسیار بیشتر از نمونه های اعلام شده ما باشد یعنی بازخوانی بسیار پایینی داشته باشیم.

از طرفی ممکن است کمی الگوریتم تشخیصی خود را ساده تر بگیریم تا تعداد مثبت های تشخیصی خود را بالا ببریم، در این صورت میزان اشتباهات ما زیادتر شده، صحت الگوریتم عدد پایین تر و بازخوانی آن، عدد بالاتری را نشان می دهد.اگر بتوانیم معیاری ترکیبی از این دو معیار برای سنجش الگوریتم های دسته‌بندی به دست آوریم، تمرکز بر آن معیار به جای بررسی همزمان این دو، مناسب‌تر خواهد بود مثلا از میانگین این دو به

عنوان یک معیار جدید استفاده کنیم و سعی در بالا بردن میانگین حسابی این دو داشته باشیم. اگر بخواهیم میانگین معمولی دو معیار بازخوانی و صحت را ملاک کار درنظر بگیریم، برای حالت هایی که صحت بالا و بازخوانی پایینی داریم (و یا بالعکس)، میانگین معمولی عددی قابل قبول خواهد بود در صورتی که الگوریتم پیشنهادی نباید نمره قبولی بگیرد.

برای رفع این نقیصه و تولید یک معیار واحد که متمایل به عدد کوچکتر باشد، از میانگین هارمونیک استفاده می کنیم.این میانگین هارمونی برای دو مقدار بازخوانی و صحت را با نام F1-Score می شناسیم که طبق فرمول فوق برابر است با :

$$\text{F1-Score} = \frac{۲}{\frac{۱}{Recall}+\frac{۱}{Precision}} = ۲ \times \frac{Precision*Recall}{Precision+Recall}$$

در این فرمول همانطور که مشاهده می کنید اگر یکی از دو مقدار عددی کوچک باشد، یا حتی صفر باشد، نتیجه نهایی عددی کوچک و یا صفر خواهد بود. توضیح این امر هم ساده است چون دو معیار بازخوانی و صحت اعدادی بین صفر تا یک هستند و در صورت کسر در هم دیگر ضرب شده اند بنابراین نتیجه نهایی به سمت عدد کوچکتر، متمایل خواهد بود و اگر هر دو با هم، عددی بزرگ (نزدیک ۱) باشند، نتیجه نهایی به سمت یک حرکت خواهد کرد .

در دنیای واقعی یک حد آستانه پذیرش هم می‌توانیم برای F1-Score تعیین کنیم مثلا ۰،۹ و اعلام کنیم که مدلهایی با نمره بالاتر از این آستانه، مورد تایید نهایی قرار خواهند گرفت.

در فرمول میانگین هارمونیک ، وزنی مساوی به هر دو پارامتر داده ایم و بسته به نیاز می توانیم میانگین هارمونیک مرتبه های بالاتر یعنیF2 ، F3و غیره را هم به کار ببریم

آخرین مطلب درباره این معیارهای اصلی دسته بندی این است که این معیارها کاملاً بستگی به بستر و حوزه دسته‌بندی دارند .مثلاً در تشخیص ایدز یا تشخیص کلاه برداری در تراکنش های بانکی، ما نیاز به شناسایی تمامی موارد ایدز و کلاه‌برداری داریم یعنی نیاز داریم که بازخوانی ما بسیار بالا باشد و اگر خطایی هم تولید شد مثلاً بیماری به اشتباه ایدزی تشخیص داده شد و یا یک تراکنش سالم، متهم به کلاه برداری شد، کافی است با کمی آزمایش بیشتر، نتایج را بهبود خواهیم بخشید و موارد خطا را از لیست تشخیص داده شده‌ها حذف خواهیم کرد.اما

گاهی اوقات به دنبال صحت بیشتر هستیم مثلاً با خواندن توئیت های روزانه، قرار است آنالیز احساسات روی آنها انجام دهیم. در این حالت، صحت الگوریتم یعنی تشخیص درست و دقیق احساسات هر توئیت خوانده شده و نه همه توئیت ها (با فرض اینکه تعداد توئیت‌های بررسی شده زیاد باشد) برای ما اهمیت زیادتری از بررسی تمامی توئیت‌ها دارد. بنابراین همیشه و در همه موارد، ما از F1-Score استفاده نمی کنیم، بلکه با بررسی نیازمندیها، بهترین معیار را برای کار خود انتخاب خواهیم کرد.

در مواردی که دسته‌ها، متعادل هستند، مثلاً تعیین جنسیت ارسال کننده یک توئیت، می‌توانیم همان معیار دقت که معیار اول مورد بحث بود را هم به کار ببریم . در هرصورت باید بدانیم که دنبال چه هستیم.هدف اصلی ما در یافتن یک مدل دسته‌بندی افزایش F1-Score آن خواهد بود[21].

پس به طور خلاصه می توان اینگونه بیان کرد :

صحت(Accuracy) : به طور کلی، دقت به این معناست که مدل تا چه اندازه خروجی را درست پیش‌بینی می‌کند.

$$Accuracy = \frac{true\ positives + true\ negatives}{total\ examples}$$

با نگاه کردن به دقت ، بلافاصله می‌توان دریافت که آیا مدل درست آموزش دیده است یا خیر و کارآیی آن به طور کلی چگونه است. اما این معیار اطلاعات جزئی در مورد کارآیی مدل ارائه نمی‌دهد.

دقت (**Precision**) : وقتی که مدل نتیجه را مثبت (positive) پیش‌بینی می‌کند، این نتیجه تا چه اندازه درست است؟

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

زمانی که ارزش false positives بالا باشد، معیار صحت، معیار مناسبی خواهد بود. فرض کنید، مدلی برای تشخیص سرطان داشته باشیم و این مدل Precision پایینی داشته باشد. نتیجه این امر این است که این مدل، بیماری بسیاری از افراد را به اشتباه سرطان تشخیص می‌دهد. نتیجه این امر استرس زیاد، آزمایش‌های فراوان و هزینه‌های گزافی را برای بیمار به دنبال خواهد داشت.

یادآوری(Recall) :

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

زمانی که ارزش false negatives بالا باشد، معیارRecall ، معیار مناسبی خواهد بود. فرض کنیم مدلی برای تشخیص بیماری کشنده ابولا داشته باشیم. اگر این مدل Recall پایینی داشته باشد چه اتفاقی خواهد افتاد؟ این

مدل افراد زیادی که آلوده به این بیماری کشنده هستند را سالم در نظر می‌گیرد و این فاجعه است.

نرخ امتیاز(F1-Score) :

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

معیارF1 ، یک معیار مناسب برای ارزیابی دقت یک آزمایش است. این معیار Precision و Recall را با هم در نظر می‌گیرد. معیار F1 در بهترین حالت، یک و در بدترین حالت صفر است[22].

فصل سوم :

روش تحقیق

۳-۱- مقدمه

همان طور که بیان شد روش‌های مختلفی برای دسته‌بندی نمونه‌های ورودی ارائه شده‌اند که با توجه به معیارهای ارائه شده برای برازش کارایی این روش‌ها، هر کدام از این روش‌ها دارای مزایا و معایبی هستند .

در این فصل قصد داریم عملیات اصلی را بر روی دادگان[۴۲] انجام داده و نتایج حاصله را بررسی نماییم . روال کار بر اساس روش های منطقی انجام شده که هر کدام در نحوه انجام و انتخاب انواع روش ها و کتابخانه ها دارای جزییات منحصر به فرد می باشند . در مورد هر کدام از کتابخانه هایی که استفاده شده به طور مختصر توضیح داده خواهد شد .

هدف آن است که پس از انتخاب روش مناسب برای خواندن داده ، ابتدا در مورد دیتاست اطلاعات دقیقی بدست آورده و پس از تحلیل دقیق آن، برای ادامه کار با استفاده از روش های معمول تقسیم بندی و پاکسازی که در هر بخش بطور مجزا بررسی خواهد شد سپس دادگان آموزشی را آماده خواهد شد تا بتوان انواع مختلفی از دسته بند[۴۳]ها را بر روی آن ها پیاده سازی کرده و نتایج را بررسی کرد.

همچنین در قدم بعدی ، برخی از روش های ترکیب دسته بندهای نیز استفاده شده تا امکان بهبود نتایج توسط ترکیب دسته بندها نیز بررسی گردد.

۳-۲ روش کار

۳-۲-۱- خواندن مجموعه داده

اولین قدم برای آغاز کار هوش مصنوعی، بارگذاری داده است[25]. برای این کار کتابخانه ها و روش های متفاوتی وجود دارد . برخی از این روش ها عبارتند از :

- بارگذاری فایل های CSV با کتابخانه استاندارد پایتون
- بارگذاری فایل های CSV با کتابخانهNumpy
- بارگذاری فایل های CSV با کتابخانهPandas

[۴۲] Dataset

[۴۳] Classifier

در این پژوهش از کتابخانه pandas برای بارگذاری داده استفاده شده است . کتابخانه Pandas بر مبنای NumPy ساخته شده است و ساختارها و ابزارهای تحلیل داده با کاربری آسان را برای زبان برنامه‌نویسی پایتون ارائه می‌کند. این کتابخانه قدرتمند و انعطاف‌پذیر به طور مکرر از سوی علاقه‌مندان علم داده برای دریافت داده‌ها در ساختارهای داده‌ای که برای تحلیل‌هایشان کاملاً گویا هستند مورد استفاده قرار می‌گیرد[26].

در کدهای نوشته شده در این پژوهش برای شناخت بیشتر کتابخانه ها و مشاهده آن ها به صورت یکجا ، تمام کتابخانه های مورد استفاده در ابتدای کد های برنامه فراخوانی شده اند . اما در زمان توضیح روال برنامه، در هر زمان که از کتابخانه ای استفاده شده باشد ، دستور مورد نیاز برای فراخوانی آن نیز توضیح داده خواهد شد .

```
import re
import nltk
import pickle
import numpy as np
import pandas as pd
import seaborn as sns
from scipy import stats
from sklearn import tree
from sklearn.svm import SVC
from random import randrange
from bs4 import BeautifulSoup
from nltk import word_tokenize
import matplotlib.pyplot as plt
from operator import itemgetter
from nltk.corpus import stopwords
from sklearn.svm import LinearSVC
from nltk.stem.porter import PorterStemmer
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_fscore_support
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
NB =GaussianNB()
%matplotlib inline
```

فراخوانی کتابخانه های مورد نیاز

در تصویر زیر فراخوانی داده توسط کتابخانه pandas را مشاهده می فرمایید . برای فراخوانی کتابخانه pandas از کد import pandas as pd استفاده شده است .

```
In [3]: dataset = pd.read_csv("train.csv")
        print("Number of rows in train dataset : ",len(dataset))
        dataset.head()

        Number of rows in train dataset :  19579
```

Out[3]:

	id	text	author
0	id26305	This process, however, afforded me no means of...	EAP
1	id17569	It never once occurred to me that the fumbling...	HPL
2	id11008	In his left hand was a gold snuff box, from wh...	EAP
3	id27763	How lovely is spring As we looked from Windsor...	MWS
4	id12958	Finding nothing else, not even gold, the Super...	HPL

خواندن مجموعه داده

۳-۲-۲- تجزیه و تحلیل داده

از مهم ترین قسمت های کار با داده ، شناخت داده ورودی است . تسلط بر اینکه داده ورودی، شامل چه آیتم هایی است ، چه ویژگی[٤٤] و چه شیء[٤٥] هایی در داده وجود دارد، شامل چند شیء است و نوع ویژگی و اشیاء چیست . همچنین بررسی برچسب[٤٦] ها ، نوع آن ها و تعداد برچسب تاثیر بسیاری در نحوه کار با داده و انتخاب روش های مختلف کار با داده دارد .

داده استفاده شده در این پژوهش ، شامل ۱۹۵۷۹ شیء و سه ویژگی می باشد .

به عنوان ویژگی ،داده هایی نظیر نام نویسنده، ID و بخشی از متن کتاب قرار گرفته است . کل داده شامل ۱۹۵۷۹ جمله از کتاب های این سه نویسنده است .

```
In [4]: 1 dataset = pd.read_csv("train.csv")
        2 print("Number of rows in train dataset : ",len(dataset))
        3 dataset.head()
        4 dataset.shape

        Number of rows in train dataset :  19579

Out[4]: (19579, 3)
```

Shape داده ورودی

[٤٤] Attribute

[٤٥] Object

[٤٦] Label

از سه نویسنده که با برچسب های EAP , MWS , HPL شناسایی میشوند ، تعداد ۷۹۰۰ متن از EAP و 6044 متن از MWS و ۵۶۳۵ متن از HPL در داده قرار داده شده است . و برای هر متن یک ID خاص در نظر گرفته شده است .

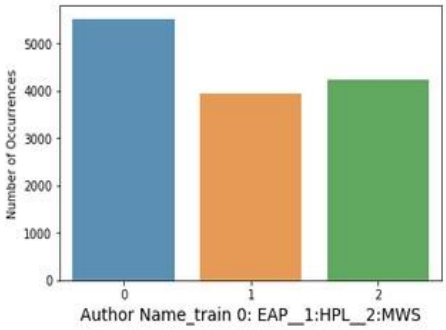
```
In [5]: 1 dataset['author'].value_counts()

Out[5]: EAP    7900
        MWS    6044
        HPL    5635
        Name: author, dtype: int64
```

تعداد متن های مربوط به نویسنده

داده از لحاظ توازن، تقریبا متوازن محسوب می شود زیرا حدود تعداد متن های در نظر گرفته شده از هر نویسنده به هم نزدیک می باشد. در شکل زیر می توان تعداد متن هایی که برای هر نویسنده در داده ورودی در نظر گرفته شده را با استفاده از کتابخانه seaborn که با کد import seaborn as sns در ایتدای برنامه فراخوانی شده است ، به صورت بصری مشاهده فرمایید .

```
In [41]: 1 cnt_srs=pd.DataFrame(ytrain)[0].value_counts()
         2 plt.figure(figsize=(6,4))
         3 sns.barplot ( cnt_srs.index , pd.DataFrame(ytrain)[0].value_counts(),alpha=0.8)
         4 plt.ylabel('Number of Occurrences', fontsize=10)
         5 plt.xlabel('Author Name_train 0: EAP__1:HPL__2:MWS', fontsize=14)
         6
         7 plt.show()
```



نمودار مربوط به تعداد متن های هر نویسنده در داده وودی

کتابخانه Seaborn یک کتابخانه بسیار مفید مصورسازی داده در پایتون محسوب می گردد. کتابخانه Seaborn روی Matplotlib ساخته می شود و قابلیت های پیشرفته بسیاری در رابطه با مصورسازی داده ارائه می کند. کتابخانه Seaborn را جهت رسم انواع چارت هایی چون نمودارهای ماتریسی، نمودارهای شبکه ای (Grid)،

نمودارهای رگرسیونی ، برای رسم نمودارهای توزیعی و غیره می توان مورد استفاده قرار داد[27].

۳-۲-۳- تقسیم بندی دیتاست

در این پژوهش تقسیم بندی دیتاست بر اساس سه دسته انجام گرفته است .

دسته اول داده آموزش۲۸ است که که ۷۰ درصد از داده اصلی را شامل میشود

دسته دوم داده آزمون۲۹ است که ۲۰ درصد از داده اصلی را شامل می شود

و دسته سوم به عنوان داده اعتبارسنجی۳۰ شناخته می شود که با نام اختصاری Dev تعریف شده و شامل ۱۰ درصد از داده اصلی می باشد .

در واقع در این روش پس از یادگیری از روی داده آموزش و انجام آزمون روی داده آزموت، ارزیابی بر روی داده اعتبارسنجی انجام میگیرد.

مهم ترین فاکتور در ساخت این سه دسته، رعایت توزیع کلاس هاست . به نحوی که وقتی از داده اصلی، ۷۰ درصد را به عنوان داده آموزشی استفاده می کنیم، توزیع کلاس ها یا برچسب ها باید به همان میزانی باشد که در داده اصلی موجود بوده است .

به عنوان مثال با فرض اینکه در داده اصلی، ۲۰ درصد از داده مربوط به کلاس A ، ۳۰ درصد مربوط به کلاس B و ۵۰ درصد مربوط به کلاس C باشد، این توزیع کلاس باید در داده های آموزش،آزمون و اعتبار سنجی نیز به همین صورت باشد.

در شکل زیر کدهای مربوط به اختصاص داده را مشاهده می فرمایید .

برای انجام عملیات بر روی داده ، با توجه به اینکه بعضی از الگوریتم های یادگیری ماشین۳۱صرفا توانایی کار با عدد را دارند، برچسب ها را برای جلوگیری از خطاهای احتمالی به عدد تبدیل کرده ایم(خطوط ۳ و ٤) .

```
In [ ]:
from sklearn import preprocessing, decomposition, model_selection, metrics, pipeline

lbl_enc = preprocessing.LabelEncoder()
y = lbl_enc.fit_transform(dataset.author.values)

xrest, xdev, yrest, ydev = train_test_split(dataset.text.values, y,
                                            stratify=y,
                                            random_state=42,
                                            test_size=0.1, shuffle=True)
Xrest=pd.DataFrame(xrest)
Yrest=pd.DataFrame(yrest)
Xrest["author"]=Yrest[0]
Xrest.columns = ['text', 'author']

y2 = lbl_enc.fit_transform(Xrest.author.values)

xtrain, xtest, ytrain, ytest = train_test_split(Xrest.text.values, y2,
                                            stratify=y2,
                                            random_state=42,
                                            test_size=0.2222, shuffle=True)

Ydev=pd.DataFrame(ydev)
z0=pickle.dump(Ydev, open('Ydev.pkl','wb'))

Ytrain=pd.DataFrame(ytrain)
z0=pickle.dump(Ytrain, open('Ytrain.pkl','wb'))

Ytest=pd.DataFrame(ytest)
zo=pickle.dump(Ytest, open('Ytest.pkl','wb'))
```

تقسیم بندی داده

در هر قسمت پس از انجام تقسیم بندی ، داده های مربوطه جهت تسهیل در فراخوانی به عنوان یک دیتافریم با استفاده از تابع pickle ذخیره شده اند .

نمایش مقادیر عددی برچسب ها بعد از تبدیل به عدد

```
In [6]: dataset['author'].value_counts(),pd.DataFrame(y)[0].value_counts()
        print(0,'\t:EAP\n',1,'\t:HPL\n',2,'\t:MWS\n')

0       :EAP
 1      :HPL
 2      :MWS
```

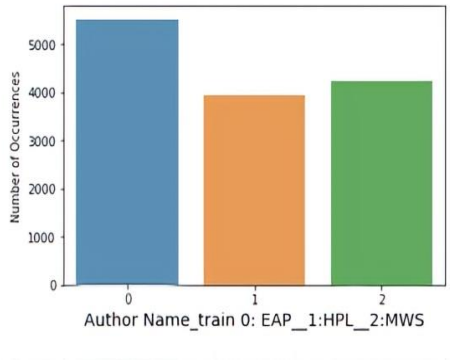
نمایش برچسب ها مقادیر عددی

همانطور که در شکل های زیر مشاهده میفرمایید ، توازن کلاس ها در تمام تقسیم بندی های داده رعایت شده است .

In [41]:

```
cnt_srs=pd.DataFrame(ytrain)[0].value_counts()
plt.figure(figsize=(6,4))
sns.barplot ( cnt_srs.index , pd.DataFrame(ytrain)[0].value_counts(),alpha=0.8)
plt.ylabel('Number of Occurrences', fontsize=10)
plt.xlabel('Author Name_train 0: EAP__1:HPL__2:MWS', fontsize=14)

plt.show()
```



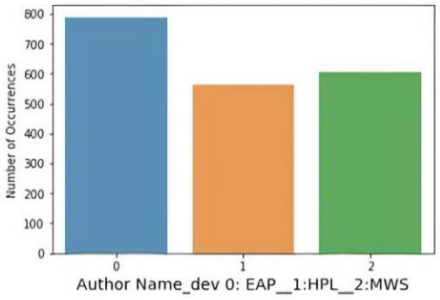
توازن کلاس ها در داده آزمایش

In [42]:

```
cnt_srs=pd.DataFrame(ydev)[0].value_counts()
plt.figure(figsize=(6,4))
sns.barplot ( cnt_srs.index , pd.DataFrame(ydev)[0].value_counts(),alpha=0.8)
plt.ylabel('Number of Occurrences', fontsize=10)
plt.xlabel('Author Name_dev 0: EAP__1:HPL__2:MWS', fontsize=14)

plt.show()

```

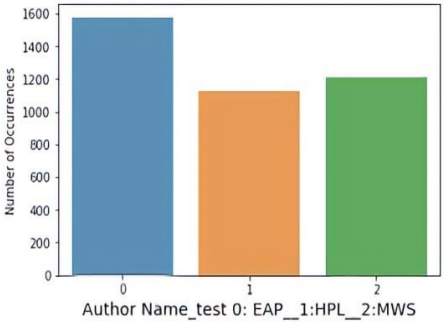


توازن کلاس ها در داده اعتبارسنجی

```
In [43]:
cnt_srs=pd.DataFrame(ytest)[0].value_counts()
plt.figure(figsize=(6,4))
sns.barplot ( cnt_srs.index , pd.DataFrame(ytest)[0].value_counts(),alpha=0.8)

plt.ylabel('Number of Occurrences', fontsize=10)
plt.xlabel('Author Name_test 0: EAP__1:HPL__2:MWS', fontsize=14)

plt.show()
```



توازن کلاس ها در داده آزمایش

٤-٢-٣- پاک‌سازی داده

پاک‌سازی داده‌ها یا تمیز کردن داده‌ها فرایند پیدا کردن، اصلاح کردن (یا حتی حذف کردن) داده‌های بی ارزش و اشتباه از دادگان یا پایگاه داده[٣٢] است. فرایند تمیز کردن داده‌ها ممکن است که از طریق ابزارهای داده‌کاوی یا پردازش دسته‌ای از طریق اسکریپت‌ها انجام شود. بعد از پاکسازی، مجموعه داده باید با سایر مجموعه داده‌های مشابه در سیستم سازگار باشد. ناسازگاری داده‌ها شناسایی و حذف (اصلاح) شده ممکن است بر اثر اشتباه انسانی هنگام ورود اطلاعات، انحراف در هنگام انتقال و ذخیره سازی اطلاعات یا به دلیل واژه نامه‌های داده مختلف باشد.

در این پژوهش برای پاکسازی داده به انجام عملیات نشانه گذاری[٣٣] ، حذف علایم ، کوچک کردن تمام حروف ، حذف کلمات کلیدی[٣٤] و یافتن ریشه لغات[٣٥] بسنده کرده ایم .

یک تابع کلی برای انجام پاکسازی در ابتدا تعریف شده است تا بعدا بتوان به سادگی با فراخوانی این تابع، تمامی عملیات مربوطه را بر روی هر سه دیتا انجام داد.

```
In [54]:
def cleaning_and_stemming(text, non_alpha=True, normalization=True, stemming=True, stopword=True):
    Beautiful_text = BeautifulSoup(text).get_text()
    words = word_tokenize(Beautiful_text)
    if non_alpha:
        words = [word for word in words if word.isalpha()]
    if normalization:
        words = [w.lower() for w in words]
    if stopword:
        stop_words = set(stopwords.words('english'))
        words = [w for w in words if w not in stop_words]
    if stemming:
        porter = PorterStemmer()
        words = [porter.stem(word) for word in words]
    return( " ".join(words ))

```

حالا این تابع را بر روی داده های آموزش ، آزمون و اعتبارسنجی پیاده سازی می کنیم .

```
In [34]:
clean_train=list(range(len(xtrain)))
clean_dev = list(range(len(xdev)))
clean_test = list(range(len(xtest)))
xTrain= pd.DataFrame(xtrain)
xDev=pd.DataFrame(xdev)
xTest=pd.DataFrame(xtest)
clean_train=xTrain[0].apply(lambda x: cleaning_and_stemming(x) ).tolist()
clean_dev=xDev[0].apply ( lambda x: cleaning_and_stemming(x) ).tolist()
clean_test=xTest[0].apply ( lambda x: cleaning_and_stemming(x) ).tolist()
```

پاکسازی داده

۵-۲-۳- تهیه ورودی برای الگوریتم های پایه با استفاده از TF-IDFو BOW(TF) و BOW(binary)

یک نمایش ساده است که در پردازش زبان‌های طبیعی و بازیابی اطلاعات ۳۶ استفاده می‌شود. همچنین به عنوان مدل فضای بردار‌یشناخته می‌شود. در این مدل، یک متن (مانند یک جمله یا سند) به صورت یک بسته چند مجموعه از کلمات آن، بی‌توجه به دستور زبان و حتی نظم کلمات نمایش داده می‌شود(McTear et al 2016, p. 167) مدل بسته کلمات برای بینایی کامپیوتر۳۷ استفاده شده‌است((Sivic, Josef (April 2009) مدل بسته کلمات معمولاً در روش‌های دسته‌بندی اسناد مورد استفاده قرار می‌گیرد که در آن وقوع هر کلمه به عنوان یک ویژگی برای آموزش طبقه‌بندی آماری استفاده می‌شود ((Harris, Zellig (1954)

برای این منظور از ماژول های TfidfVectorizer, CountVectorizer که به صورت زیر فراخوانی می شوند ، استفاده شده‌است.

```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
```

فراخوانی کتابخانه های tfidfVectorizer,CountVectorizer

برای ساخت بردار کلمات در متن از ماژول CountVectorizer پارامترهای زیر استفاده شده است:

min_df=3: در زمان ساختن دایره لغات، کلماتی که حداقل تکرار خاصی داشته باشند (در این پژوهش عدد ۳ انتخاب شده است) را حذف می‌کند .

max_features=None: مشخص کننده حداکثر تعداد کلماتی است که از ویژگی ها انتخاب می شود. می توان این محدودیت را با وارد کردن یک عدد (integer) مشخص کرد . در این پژوهش با انتخاب پارامتر None ، هیچ حداکثری برای این مورد در نظر نگرفته ایم .

strip_accents='unicode' : این پارامتر برای حذف لهجه ها و نرمال سازی کلمات به کار می رود . به سه صورت قابل تنظیم است .

حالت ASCII : در این حالت که یک روش سریع است، صرفا بر روی کاراکترهایی تمرکز می شود که به کد اسکی خاصی نگاشت می شوند.

حالت UNICODE : در این روش که نسبت به روش قبل کندتر است، عملیات حذف لهجه و نرمال سازی بر روی تمام کاراکترها انجام می شود .

حالت NONE : در این روش هیچ کاری بر روی کاراکتر ها انجام نمی شود .

analyzer='word' : با این پارامتر مشخص میکنیم که ویژگی های ما بر اساس کلمات، کاراکترها یا n-grams ساخته شده اند. با توجه به feature های دادگان و نوع عملیاتی که قصد داریم انجام دهیم، این پارامتر را word در نظر گرفته ایم .

ngram_range= (1, 3): مشخص کتنده حداقل و حداکثر تعداد کلمات چندتایی که باید استخراج شوند استفاده می شود.

```
vectorizer = CountVectorizer(min_df=3,  max_features=None, strip_accents='unicode', analyzer='word',ngram_range=(1, 3))
```

تعیین پارامترهای تابع vectorizer

پس با تابع fit_transform و تعیین کردن داده آموزش پاک‌سازی شده به عنوان پارامتر ورودی مدل بردار کلمات ساخته شده است.

```
train_BOW_bi = vectorizer.fit_transform(clean_train).toarray()
```

تابع fit_transform

در ادامه بردار داده‌های اعتبارسنجی و آزمایش با تابع fit ساخته شده است.

```
dev_BOW_bi = vectorizer.transform(clean_dev).toarray()
test_BOW_bi = vectorizer.transform(clean_test).toarray()
```

ساخت بردار دادگان اعتبارسنجی و آزمایش

در انتها با توجه به اینکه در جدول بردارکلمات، تعداد تکرار کلمات موجود می باشد و درکیسه کلمات دو دویی، معیار فقط وجود و یا عدم وجود کلمه است، با استفاده از دو حلقه for تمام مولفه‌هایی که کلمه در آن ها موجود بوده و به عبارتی مخالف صفر بوده اند، با عدد یک جایگزین شده و بدین صورت تمام داده های آموزش، اعتبارسنجی و آزمون به کیسه کلمات دودویی تبدیل شده‌است.

In [35]:
```
print ("Creating the bag of words binary...\n")
vectorizer = CountVectorizer(min_df=3,  max_features=None, strip_accents='unicode',
                             analyzer='word',ngram_range=(1, 3))

train_BOW_bi = vectorizer.fit_transform(clean_train).toarray()
dev_BOW_bi = vectorizer.transform(clean_dev).toarray()
test_BOW_bi = vectorizer.transform(clean_test).toarray()

for i in range (len(train_BOW_bi)):
    for j in range (len(train_BOW_bi[i])):
        if train_BOW_bi[i][j]!=0 :
            train_BOW_bi[i][j]=1

for i in range (len(dev_BOW_bi)):
    for j in range (len(dev_BOW_bi[i])):
        if dev_BOW_bi[i][j]!=0 :
            dev_BOW_bi[i][j]=1

for i in range (len(test_BOW_bi)):
    for j in range (len(test_BOW_bi[i])):
        if test_BOW_bi[i][j]!=0 :
            test_BOW_bi[i][j]=1
```

ساخت Binary BOW

ساخت کیسه کلمات با فرکانس کلمات

In [ ]:
```
print ("Creating the bag of words...\n")
train_BOW = vectorizer.fit_transform(clean_train).toarray()
dev_BOW = vectorizer.transform(clean_dev).toarray()
test_BOW = vectorizer.transform(clean_test).toarray()
```

ساخت BOW

ساخت فراوانی کلمه کلیدی(tf-idff)

```
In [36]:
print ("Creating the Tfidf ...\n")
vectorizer = TfidfVectorizer(min_df=3,  max_features=None,
            strip_accents='unicode', analyzer='word',
            ngram_range=(1, 3), use_idf=1,smooth_idf=1,sublinear_tf=1)


train_TFIDF = vectorizer.fit_transform(clean_train).toarray()
dev_TFIDF = vectorizer.transform(clean_dev).toarray()
test_TFIDF = vectorizer.transform(clean_test).toarray()
```

ساخت TFIDF BOW

ابتدا تمام برچسب های دادگان آموزش ، اعتبار سنجی و آزمون را در در لیست های جدیدی که تعریف می کنیم قرار داده ایم

```
In [37]:
Ytrain=pd.DataFrame(ytrain)
Ydev=pd.DataFrame(ydev)
Ytest=pd.DataFrame(ytest)
```

فراخوانی برچسب های دادگان

با توجه به اینکه تعداد مدل هایی که قصد ساخت آن ها را داریم ، بالاست و در حال حاضر بصورت دقیق قابل پیش بینی نیست، لیست هایی به طول ۱۰۰۰ برای مدل ها ، نتایج و مقادیر پیش بینی شده در نظر گرفته شده است . در آینده با توجه به تعداد مدل هایی که ساخته شده اند، میتوان برای ذخیره سازی تعداد مورد نظر را انتخاب کرد.

```
In [38]:
allModels_clf=list (range (1000)) # [(model,fscore,predict),...,(model,fscore,predict)]
Fscoer_clf=list (range (1000))
Pred_clf=list (range (1000))
```

تعریف لیست های مورد نیاز

٦-٢-٣- ساخت ۱۰۰ نمونه داده آموزش با رویکرد جایگشتی

برای آموزش هوش مصنوعی ، بهتر است تعداد داده های بیشتری به عنوان ورودی به دسته بند های متفاوت داده شود .

با توجه به اینکه داده های دیتاست دارای محدودیت هستند، می توان با استفاده از همان داده ها، تعداد بیشتری داده آموزش ساخت . روش کار به اینصورت است که به با استفاده از تابع rand در هر بار برداشت، یکی از داده

های دیتاست برداشته می شود و در جدول جدیدی قرار میگیرد . از تابع rand به این دلیل استفاده شده که داده های دیتاست اصلی به ترتیب برداشته نشده و در نتیجه ترتیب داده ها یکسان نباشند . این کار باعث می شود دسته بند ، اگر چه داده های تکراری را آموزش میبیند اما با تغییر جایگاه داده ها بتواند بصورت بهتر عمل یادگیری را انجام دهد .

توجه داشته باشید که این روش بصورت جایگشتی بوده و با توجه به اینکه داده ها بصورت رندوم انتخاب می شوند ممکن است داده تکراری داشته باشیم . این روش را sampling with replacement میگویند .

همانطور که در شکل زیر ملاحظه میفرمایید، ابتدا باید داده های آموزش، آزمون و اعتبارسنجی که پاکسازی شده بصورت دادگان جدید ذخیره شده اند را فراخوانی کرده و در لیست های جداگانه قرار دهیم . سپس با تعریف دو تابع، عملیات ساخت ۱۰۰ عدد داده آموزش را انجام می دهیم .

برای انجام این کار دو تابع Subsampleو cerat_multi_dataset به صورت زیر تبدیل شده است.

در تابع subsample یک لیست به اندازه طول اشیاء دادگان ساخته شده و با روش جایگشتی، هر یک از اشیاء به صورت رندوم انتخاب شده است تا بتوانیم از روی داده آموزش، یک نمونه جدید بسازیم.

در تابع cerat_multi_dataset عملیات فوق به تعداد ۱۰۰ بار انجام شده تا در نتیجه از روی اشیاء دادگان اصلی ۱۰۰ داده آموزش ساخته باشیم .

حال این داده ها آماده هستند تا در دسته بندهای مختلف به عنوان داده ورودی استفاده شوند .

In [40]:
```
train_TFIDF_D=pd.DataFrame(train_TFIDF)
train_TFIDF_D['lable']=ytrain
train_TFIDF_D=train_TFIDF_D.as_matrix()

dev_TFIDF_D=pd.DataFrame(dev_TFIDF)
dev_TFIDF_D['lable']=ydev
dev_TFIDF_D=dev_TFIDF_D.as_matrix()

def subsample(dataset):
    sample = list()
    for i in range (dataset.shape[0]):
        index = randrange(dataset.shape[0])
        sample.append(dataset[index])
    return sample

def cerat_muliti_dataset(dataset):
    COM_LIST=list(range(100))
    for i in range (len(COM_LIST)):
        COM_LIST[i]=subsample(dataset)
    return COM_LIST
COM_LIST=cerat_muliti_dataset(train_TFIDF_D)
```

ساخت ۱۰۰ داده train و Dev

۷-۲-۳- انتخاب الگوریتم های پایه و ساخت ۱۰۰ دسته بند با استفاده از هر الگوریتم

الگوریتم، روشی که برای جستجوی الگو در داده‌ها مورد استفاده قرار می‌گیرد را تعیین می‌کند و در واقع مانند یک روال ریاضی برای حل یک مساله خاص است. الگوریتم‌های گوناگونی برای تحلیل داده[۴۷] موجود هستند و لذا انتخاب الگوریتم داده‌کاوی[۴۸] مناسب یک مساله، برای پژوهشگران و تحلیلگران کاری دشوار است. برخی از سازمان‌ها به دلیل دشواری انتخاب الگوریتم داده کاوی مناسب، به طور مکرر از برخی الگوریتم‌های داده‌کاوی استفاده می‌کنند. هنگامی‌که یک الگوریتم نامناسب پیاده‌سازی می‌شود، دانش کشف شده اغلب برای سازمان کاربرد است زیرا از اطلاعات صحیحی استخراج نکرده و این امر می‌تواند منجر به تصمیم‌گیری‌های غلط در کسب‌وکار شود.

هیچ راهنمای مشخصی برای آنکه پژوهشگران یا تحلیلگران چگونه الگوریتم انتخاب کنند وجود ندارد . انتخاب یک الگوریتم مشخص امری بسیار پیچیده است، لذا در این پژوهش برای ارتقای نتایج داده‌کاوی از چندین الگوریتم استفاده شده و پردازش‌ها را با الگوریتم‌های مختلف تکرار و دست آخر از از تکنیک رای گیری اکثریت استفاده می‌شود.

گاه نیاز به استفاده از چندین الگوریتم برای حل یک مساله واحد جهت حل فازهای مختلف مساله است. در مجموع می‌توان گفت هدف مساله، ساختار مجموعه داده موجود، نتایج مورد انتظار در خروجی، شناخت داده‌کاو از یک الگوریتم و مولفه‌های پیکربندی پایگاه داده در انتخاب الگوریتم مناسب داده‌کاوی تاثیرگذار هستند [1].

الگوریتم‌های پایه مورد استفاده در این مقاله شامل SVC خطی، بیز ساده، رگرسیون منطقی و درختان اضافی می باشد. همه این الگوریتم‌ها با ۳ رویکرد BOW_Binary، BOW_TF و TF-IDF مدل‌سازی شده‌اند که به شرح و تفصیل هر یک می پردازیم.

کیسه کلمات فرکانس : کیسه کلمات (BOW) یک مدل در پردازش زبان های طبیعی است که با هدف دسته بندی مستندات و متون استفاده می‌شود. ایده اصلی آن، به این صورت هست که به هر کدام از کلمات یک عدد Unique نسبت می‌دهیم و Feature بدست آمده بر اساس فرکانس تکرار هر کدام از کلمات به دست خواهد

آمد. به طور متعارف، مجموعه‌ای از کلمات با استفاده از فرکانس کلمه یا اهمیت به عنوان ویژگی اغلب برای طبقه‌بندی یا بازیابی سند استفاده می‌شود.یکی از مشکلات این روش، انفجار ابعاد است که با افزایش کلمات، بعد هم افزایش می یابد[28].

کیسه کلمات دودویی : تنها بودن و یا نبودن یک کلمه در آن داده یا همان رکورد را بررسی می کنه که آن را با صفر و یک نشون میدهد.

فراوانی کلمه کلیدی[۴۹] :

جدول تکرار کلمات (TF) : روش اصلی یافتن صفحات مرتبط با یک جستجو، روش سنجش تعداد تکرار (TF) یک کلمه است. هر چه که یک کلمه در یک صفحه بیشتر تکرار شده باشد، آن صفحه ارتباط بیشتری با آن کلمه دارد بنابراین اگر کاربر کلمه ای را جستجو کرد، صفحاتی را نمایش خواهیم داد که آن کلمه در آنها بیشتر تکرار شده باشد.

جدول معکوس تکرار در صفحات (IDF) : به ازای هر لغت، باید فرمولی را استفاده کنیم که هر چه تکرار یک لغت در یک کتاب کمتر باشد، به آن امتیاز بیشتری بدهد (رابطه معکوس) مثلاً می توانیم از فرمول N/DF استفاده کنیم که N تعداد کل کتابها و DF تعداد کتابهای حاوی آن لغت است . با این فرمول هر چه یک لغت کمتر تکرار شده باشد، عدد بزرگتری تولید می شود.

جدول TF-IDF : با داشتن جدول TF و IDF ، می توانیم مرتبط بودن یک لغت با یک صفحه را با ضرب این دو در هم نمایش دهیم :

میزان ارتباط لغت با یک صفحه = TF * IDF

یعنی هر چه یک لغت در یک متن بیشتر به کار رفته باشد و در سایر متن¬ها خیلی کم به کار رفته باشد، امتیاز آن صفحه برای آن لغت بیشتر می شود که منطقی هم به نظر می رسد.

به هر کلمه در متن یک وزن اختصاص دهیم. با این کار، می‌توانیم اهمیت یک کلمه را در فرآیند مهندسی ویژگی بهتر شناسایی کرده و در نهایت ویژگی‌های بهتری را برای تزریق به الگوریتم‌های بعدی مانند طبقه‌بندی یا خوشه‌بندی داشته باشیم. مقدار TF-IDF مخفف دو کلمه است: TF به معنیِ Term Frequency یعنی تعداد تکرار یک کلمه در یک متن و عبارت IDF به معنی Inverse Document Frequency که می‌توان آن را به

[۴۹] TF-IDF

برعکس تعداد تکرار در متون ترجمه کرد.وزن دهی TF-IDF طبق فرمول (۱) محاسبه می شود.

$$f(w) = TF(w).IDF(w) = TF(w).\log N/(n(w)+1) \quad (۱$$

۳-۲-۷-۱- الگوریتم *Linear SVC* :

این الگوریتم مشابه SVC هست بدین صورت که پارامتر Kernel آن برابر با Linear می‌باشد و برای داده‌های در مقیاس بزرگ مناسب است . در ابتدا باید ماژول و کتابخانه این الگوریتم فراخوانی شود.

```
10 from sklearn.svm import SVC
```

فراخوانی کتابخانه Linear SVC

جهت ساخت مدل، پارامترهای الگوریتم LinearSVC ، به صورت زیر مقداردهی شده است:

C=2.5 : مقدار ذاتی c همان یک است.هرچقدر c بیشتر باشد،حول همسایه ها می چرخد و اگر سطحی که در می آید کمتر باشد،سطح تخت می شود.مقدار C هر چقدر کمتر باشد میزان همواری بیشتر است مقدار این پارامتر بسته به نوع دیتاستی است که ما دراختیار داریم انتخاب شده چون بهترین نتیجه را می دهد

Penalty="l2" : نوعی پنالتی است که به دو نوع L1 و L2 است .پیش فرض آن L2 است و ازآن بهترین جواب بدست می آید

dual=False : در این الگوریتم مسئله ما مقدار dual نیست یعنی این معیار برای پارامترهای دومجهولی است اما در اینجا ما یک مجهول داریم و نیازی به تغییر این پارامتر نمی باشد.

tol=0.0025 : این تابع بر اساس بهینه سازی عمل می کند.یعنی از آن حد کمتر تغییر کند سیستم را break می کند.هرچقدر عدد کوچک باشد دقت بیشتر می شود و جواب بهتری خواهیم گرفت اما زمان محاسبات بالاتر خواهد رفت. بهینه ترین مقدار ۰,۰۰۲۵ در نظر گرفته شده است.

```
2 clf0=LinearSVC(C=2.5, penalty="l2", dual=False ,tol=0.0025).fit(train_TFIDF,Ytrain)
```

ساخت مدل با Linear SVC

برای ذخیره مدل،پیش بینی‌ها و ارزیابی آن از ماژول Pickle استفاده شده است.Pickle یک ماژول از پیش ساخته شده در پایتون است که یک ذخیره سازی قابل حمل از داده های ساخت یافته را فراهم می کند. از تابع ()dumpدر ماژول pickle برای ذخیره کردن و از تابع ()load برای فراخوانی مجدد جریان داده استفاده شده است.

برای استفاده از این ماژول ابتدا باید آن را فراخوانی کنیم .

```
3 import pickle
```

فراخوانی کتابخانه pickle

تابع دامپ دو پارامتر ورودی دارد، در ابتدا باید داده‌ای که می خواهیم ذخیره کنیم را به تابع می دهیم در این جا با clf0 که همان مدل ساخته شده می‌باشد، مقداردهی شده و سپس نام فایل مورد نظر و اینکه می خواهیم برای چه کاری این فایل را باز کنیم ، ’wb‘ مشخص می‌کند که فایل جهت نوشتن باز شده است، مدل تولیده شده با نام bay_model_0.pkl ذخیره شده است.

```
3 z0=pickle.dump(clf0, open('bay_model_0.pkl','wb'))
```

و با استفاده از تابع load() امکان استفاده و فراخوانی مدل ذخیره شده فراهم می شود. ’rb‘ مشخص می کند که فایل bay_model_0.pkl جهت خواندن باز شده است.

```
4 z0=pickle.load(open('bay_model_0.pkl','rb'))
```

با استفاده از مدل ذخیره شده بر روی داده های اعتبار سنجی پیش‌بینی و ارزیابی را انجام شده و به ترتیب در لیست‌های Pred_clf[] و Fscore_clf[] قرار داده شده است. لیست allModels_clf[] نیز محتوی مدل و ارزیابی آن می باشد.معیار ارزیابی مورد استفاده F-Score می‌باشد.

```
5 p_clf0=z0.predict(dev_TFIDF)
6 F_clf0 = precision_recall_fscore_support(Ydev, p_clf0, average='micro')[2]
```

به صورت نمونه در این قسمت کد مربوط به tf-idf ذکر شده است.

In [82]:
```
#tfidf
# clf0=LinearSVC(C=2.5, penalty="l2", dual=False ,tol=0.0025).fit(train_TFIDF,Ytrain) ##only for frist run
# z0=pickle.dump(clf0, open('bay_model_2.pkl','wb'))  ##only for frist run
z0=pickle.load(open('bay_model_2.pkl','rb'))
p_clf0=np.array(z0.predict(dev_TFIDF).tolist(),dtype=float)
F_clf0 = precision_recall_fscore_support(Ydev, p_clf0, average='micro')[2]
Pred_clf[2]=p_clf0
Fscoer_clf[2]=F_clf0
print('F_clf2=',F_clf0)
```

```
F_clf2= 0.7946884576098059
```

TF-IDF

بعد از ساخت مدل‌ها با داده‌های آموزش کیسه کلمات دودویی، فرکانس کلمات و فراوانی کلمه کلیدی، طبق بررسی‌های انجام شده مدلی که با استفاده از فراوانی کلمه کلیدی ساخته شده بود بهترین F-Score را داشت. سپس در ادامه ارزیابی مدل بر روی داده‌های آزمایش نیز انجام شد.

In [83]:
```
1 p_clf1=np.array(z0.predict(test_TFIDF).tolist(),dtype=float)
2 F_clf1 = precision_recall_fscore_support(Ytest, p_clf1, average='micro')[2]
3 Pred_clf_test[2]=p_clf1
4 Fscoer_clf_test[2]=F_clf1
5 print('F_clf2=',F_clf1)
```

```
F_clf2= 0.8003064351378958
```

با توجه به بهینه بودن این رویکرد، ۱۰۰ مدل Linear SVC با استفاده از آن ساخته شد.

ساخت این مدل‌ها با استفاده از ۱۰۰ نمونه داده آموزشی که در مراحل قبل ساخته شده و در لیست COM_LIST قرار گرفته، انجام شده است، بدین صورت که ابتدا ویژگی‌ها و برچسب این داده آموزشی با دستورات زیر جدا و به ترتیب در Xcom و Ycom قرار گرفته که به عنوان ورودی به الگوریتم LinearSVC جهت ساخت ۱۰۰ داده شده است. سپس مدل های ساخته شده در فایل های جداگانه ذخیره شده است. در ضمن پیش بینی و ارزیابی مدل ها به ترتیب در لیست‌های Pred_clf[] و Fscore_clf[] قرار داده شده است.تمام این ارزیابی‌ها بر روی داده اعتبارسنجی صورت گرفته است.

In [84]:
```
#100 train with tfidf

for i in range (3,103):
##only for frist run
#     j=i-3
#     ff=np.array(COM_LIST[j],dtype=float)
#     fff=pd.DataFrame(ff)
#     num=fff.shape[1]-1
#     xcom=fff.iloc[:,0:num].as_matrix()
#     Ycom=pd.DataFrame(fff.iloc[:,-1].as_matrix())
#     clf0=LinearSVC(C=2.5, penalty="l2", dual=False ,tol=0.0025).fit(xcom,Ycom)
#     z0=pickle.dump(clf0, open('bay_model_%d.pkl'%i,'wb'))
    z0=pickle.load(open('bay_model_%d.pkl'%i,'rb'))
    p_clf0=z0.predict(dev_TFIDF)
    F_clf0 = precision_recall_fscore_support(Ydev, p_clf0, average='micro')[2]
    Pred_clf[i]=p_clf0
    Fscoer_clf[i]=F_clf0
#     print('F_clf%d='%i,F_clf0)


```

ساخت ۱۰۰ مدل LinearSvc

3-2-7-2- الگوریتم *Naïve bayes*

تکنیک نایوبیز مبتنی بر قضیه بیزین[۵۰] است و به خصوص زمانی که ابعاد ورودی بالا باشد ، مناسب است . برخلاف سادگی ، بیز می‌تواند از بسیاری روش‌های طبقه‌بندی پیچیده تر بهتر عمل کند [29].

برای استفاده از این الگوریتم، ماژول آن بدین صورت فراخوانی شده است.

```
16 from sklearn.naive_bayes import GaussianNB
```

فراخوانی کتابخانه Naïve bayes

[۵۰] Bayesian

جهت ساخت مدل، از تابع fit() برای الگوریتم نایوبیز استفاده شده‌است .NB ذاتا پارامتر زیادی جهت تغییر ندارد و به همین دلیل پرانتز خالی می باشد.

```
clf0=GaussianNB().fit(train_TFIDF,Ytrain)
```

In [90]:
```
#tfidf
# clf0=GaussianNB().fit(train_TFIDF,Ytrain) ##only for frist run
# z0=pickle.dump(clf0, open('bay_model_105.pkl','wb')) ##only for frist run
z0=pickle.load(open('bay_model_105.pkl','rb'))
p_clf0=np.array(z0.predict(dev_TFIDF).tolist(),dtype=float)
F_clf0 = precision_recall_fscore_support(Ydev, p_clf0, average='micro')[2]
Pred_clf[105]=p_clf0
Fscoer_clf[105]=F_clf0
print('F_clf105=',F_clf0)
```

```
F_clf105= 0.6736465781409602
```

ساخت مدل با Naïve bayes

سپس ۱۰۰ مدل با استفاده از ۱۰۰ نمونه داده آموزشی ساخته و بر روی داده های اعتبارسنجی ارزیابی صورت گرفته است. بعد از آن مدل‌ها ذخیره گردیدند .در ضمن پیش بینی مدل در لیست Pred_clf ،ارزیابی در لیست Fscore_clf قرار گرفته شده است. سپس ارزیابی بر روی داده‌های آزمایش صورت گرفت.

In [91]:
```
p_clf1=np.array(z0.predict(test_TFIDF).tolist(),dtype=float)
F_clf1 = precision_recall_fscore_support(Ytest, p_clf1, average='micro')[2]
Pred_clf_test[105]=p_clf1
Fscoer_clf_test[105]=F_clf1
print('F_clf105=',F_clf1)
```

```
F_clf105= 0.6557711950970377
```

In [92]:
```
for i in range (106,206):
##only for frist run
#     j=i-106
#     ff=np.array(COM_LIST[j],dtype=float)
#     fff=pd.DataFrame(ff)
#     num=fff.shape[1]-1
#     xcom=fff.iloc[:,0:num].as_matrix()
#     Ycom=pd.DataFrame(fff.iloc[:,-1].as_matrix())
#     clf0=GaussianNB().fit(xcom,Ycom)
#     z0=pickle.dump(clf0, open('bay_model_%d.pkl'%i,'wb'))
    z0=pickle.load(open('bay_model_%d.pkl'%i,'rb'))
    p_clf0=z0.predict(dev_TFIDF)
    F_clf0 = precision_recall_fscore_support(Ydev, p_clf0, average='micro')[2]
    Pred_clf[i]=p_clf0
    Fscoer_clf[i]=F_clf0
#     print('F_clf%d='%i,F_clf0)

```

ساخت ۱۰۰ مدل با Naïve bayes

۳-۲-۷-۳- الگوریتم ***logistic regression***

رگرسیون منطقی یک الگوریتم یادگیری ماشین آماری است که داده‌ها را با در نظر گرفتن متغیرهای خروجی نهایی طبقه‌بندی می‌کند و سعی می‌کند یک خط لگاریتمی برای تمایز بین آن‌ها ایجاد کند [30].

جهت ساخت مدل، پارامتر الگوریتم LogisticRegression ، به صورت زیر مقداردهی شده است:

C=8 : پارامتر پایدارسازی معکوس - یک متغیر کنترلی که اصلاح قدرت را از طریق ارتباط معکوس با تنظیم‌کننده لامبدا حفظ می‌کند .

با توجه به این Scikit آن را اینگونه توصیف می‌کند :

$C = 1/\lambda$: رابطه، این است که کاهش C - تنظیم کننده لامبدا را تقویت خواهد کرد[31].

بعد از ساخت مدل، با ماژول pickle ذخیره شده است .پیش بینی و ارزیابی بر روی داده اعتبارسنجی انجام شده و هر کدام به طور جداگانه در لیست نگه‌داری می‌شود.

In [99]:
```
#tfidf
# clf0=LogisticRegression(C=8).fit(train_TFIDF,Ytrain)  ##only for frist run
# z0=pickle.dump(clf0, open('bay_model_208.pkl','wb'))  ##only for frist run
z0=pickle.load(open('bay_model_208.pkl','rb'))
p_clf0=np.array(z0.predict(dev_TFIDF).tolist(),dtype=float)
F_clf0 = precision_recall_fscore_support(Ydev, p_clf0, average='micro')[2]
Pred_clf[208]=p_clf0
Fscoer_clf[208]=F_clf0
print('F_clf208=',F_clf0)
```

```
F_clf208= 0.810520939734423
```

ساخت مدل با logistic regression و ارزیابی بر رروی dev

سپس در ادامه ارزیابی بر روی داده‌های آزمایش انجام شده است.

```
In [100]:
p_clf1=np.array(z0.predict(test_TFIDF).tolist(),dtype=float)
F_clf1 = precision_recall_fscore_support(Ytest, p_clf1, average='micro')[2]
Pred_clf_test[208]=p_clf1
Fscoer_clf_test[208]=F_clf1
print('F_clf208=',F_clf1)
```

```
F_clf208= 0.8158835546475995
```

۱۰۰ نمونه مدل با استفاده از الگوریتم پایه لوجستیک رگرسیون ساخته شده است.ارزیابی و پیش بینی همه ۱۰۰ مدل بر روی داده اعتبارسنجی انجام شده‌است و به طور جداگانه در لیست‌ها نگه‌داری شده‌اند.

```
In [101]:
for i in range (209,308):
##only for frist run
#     j=i-209
#     ff=np.array(COM_LIST[j],dtype=float)
#     fff=pd.DataFrame(ff)
#     num=fff.shape[1]-1
#     xcom=fff.iloc[:,0:num].as_matrix()
#     Ycom=pd.DataFrame(fff.iloc[:,-1].as_matrix())
#     clf0=LogisticRegression(C=8).fit(xcom,Ycom)
#     z0=pickle.dump(clf0, open('bay_model_%d.pkl'%i,'wb'))
    z0=pickle.load(open('bay_model_%d.pkl'%i,'rb'))
    p_clf0=z0.predict(dev_TFIDF)
    F_clf0 = precision_recall_fscore_support(Ydev, p_clf0, average='micro')[2]
    Pred_clf[i]=p_clf0
    Fscoer_clf[i]=F_clf0
#     print('F_clf%d='%i,F_clf0)

```

ساخت ۱۰۰ مدل با logistic regression

۴-۷-۲-۳- الگوریتم *ExtraTrees*:

درختان اضافی اصلاح دیگری از کیسه هستند که در آن درخت‌های تصادفی از نمونه‌های مجموعه داده آموزشی ساخته می‌شوند .شما می‌توانید یک مدل درختی اضافی برای طبقه‌بندی با استفاده از کلاس ExtraTreesClassifier بسازید .

جهت استفاده از این الگوریتم، به صورت زیر فراخوانی آن انجام گرفته است.

```
from sklearn.ensemble import ExtraTreesClassifier
```

فراخوانی الگوریتم Extra trees

برای این الگوریتم، پارامتری تغییر داده نشده و از پیش فرض‌های خود الگوریتم استفاده و مدل ساخته شده است. ارزیابی مدل بر روی داده اعتبارسنجی انجام شده‌است.

In [107]:
```
#tfidf
# clf0=ExtraTreesClassifier().fit(train_TFIDF,Ytrain)  ##only for frist run
# z0=pickle.dump(clf0, open('bay_model_310.pkl','wb'))  ##only for frist run
z0=pickle.load(open('bay_model_310.pkl','rb'))

p_clf0=np.array(z0.predict(dev_TFIDF).tolist(),dtype=float)
F_clf0 = precision_recall_fscore_support(Ydev, p_clf0, average='micro')[2]
Pred_clf[310]=p_clf0
Fscoer_clf[310]=F_clf0
print('F_clf310=',F_clf0)
```

```
F_clf310= 0.7022471910112360
```

ارزیابی مدل بر روی داده های Dev

در ادامه ارزیابی بر روی داده آزمایش نیز انجام شده است.

In [108]:
```
p_clf1=np.array(z0.predict(test_TFIDF).tolist(),dtype=float)
F_clf1 = precision_recall_fscore_support(Ytest, p_clf1, average='micro')[2]
Pred_clf_test[310]=p_clf1
Fscoer_clf_test[310]=F_clf1
print('F_clf310=',F_clf1)
```

```
F_clf310= 0.6999489274770173
```

سپس ۱۰۰ نمونه مدل از این الگوریتم ساخته و ذخیره شده‌است.

In [109]:
```
for i in range (311,411):
##only for frist run
#     j=i-311
#     ff=np.array(COM_LIST[j],dtype=float)
#     fff=pd.DataFrame(ff)
#     num=fff.shape[1]-1
#     xcom=fff.iloc[:,0:num].as_matrix()
#     Ycom=pd.DataFrame(fff.iloc[:,-1].as_matrix())
#     clf0=ExtraTreesClassifier().fit(xcom,Ycom)
#     z0=pickle.dump(clf0, open('bay_model_%d.pkl'%i,'wb'))
    z0=pickle.load(open('bay_model_%d.pkl'%i,'rb'))
    p_clf0=z0.predict(dev_TFIDF)
    F_clf0 = precision_recall_fscore_support(Ydev, p_clf0, average='micro')[2]
    Pred_clf[i]=p_clf0
    Fscoer_clf[i]=F_clf0
#     print('F_clf%d='%i,F_clf0)

```

ساخت ۱۰۰ مدل با Extra trees

در انتها تمام لیست‌های Pred_clf، Fscore و allModels_clf با استفاده از ماژول pickle در فایل‌های جداگانه‌ای ذخیره شده‌است.

In [113]:
```
#result dev
#z0_Pred_clf=pickle.dump(Pred_clf[:411], open('Pred_clf0.pkl','wb')) ##only for frist run
z0_Pred_clf=pickle.load(open('Pred_clf0.pkl','rb'))

# z0_Fscoer_clf=pickle.dump(Fscoer_clf[:411], open('Fscoer_clf0.pkl','wb'))##only for frist run
z0_Fscoer_clf=pickle.load(open('Fscoer_clf0.pkl','rb'))

z0_allModels_clf=[]
for i in range (len(z0_Fscoer_clf)):
    z0=pickle.load(open('bay_model_%d.pkl'%i,'rb'))
    temp=[]
    temp.append(z0)
    temp.append(z0_Fscoer_clf[i])
    z0_allModels_clf.append(temp)
```

در ادامه ارزیابی هایی که بر روی داده‌های آزمایش نیز انجام شده بود نیز به طور جداگانه ذخیره شد.

۳-۲-۸- گرفتن میانگین F-score برای هر مدل و اعمال فیلترینگ

برای اینکه مدل دسته جمعی عملکرد بهتری نسبت به مدل تکی داشته باشد، باید هرکدام از مدلهای تشکیل

دهنده آن مستقل از بقیه بوده و دارای دقتی بیشتر از میانگین امتیاز مدل‌ها داشته باشد.

بدین سه لیست خالی به نام‌های allModels_clf، Fscore_clf و Pred_clf به ترتیب برای مدل‌ها، امتیازها و پیش بینی‌ها تعریف شده است.طبق خط شماره ۵ میانگین امتیاز همه مدل‌ها به‌دست می‌آید.سپس با استفاده از یک حلقه for و پیمایش امتیاز تمام مدل ها،امتیازی که از میانگین امتیازها بیشتر باشد به لیست Fscore_clf اضافه می‌شود. مدل و پیش بینی هم به لیست‌های allModels و Pred_clf اضافه می‌گردد. همان طور که مشاهده می‌کنید از بین ٤١١ مدل ساخته شده فقط ۲۰۵ مدل امتیازی بالاتر از میانگین امتیازها داشتند.

In [196]:
```
allModels_clf=[]
Fscoer_clf=[]
Pred_clf=[]

mean_fscore = sum (z0_Fscoer_clf) / len(z0_allModels_clf)
for i in range (len(z0_allModels_clf)):
    if z0_Fscoer_clf[i] > mean_fscore :
        Fscoer_clf.append(z0_Fscoer_clf[i])
        allModels_clf.append(z0_allModels_clf[i])
        Pred_clf.append(z0_Pred_clf[i])

len(allModels_clf),len(z0_allModels_clf)
```

Out[196]: (205, 411)

سپس رویکردها بر روی داده‌های اعتبار سنجی، اجرا و ارزیابی شده‌است.

[209]:
```
best_clasifire_test_filter=best_clasifire(allModels_clf_testf,Fscoer_clf_testf,Pred_clf_testf,'test')
print ('best_clasifire_test_filter=',best_clasifire_test_filter[0])

voting_test_filter=modle_voting(Ytest,Pred_clf_testf,'test')
print ('voting_test_filter=',voting_test_filter[0])

forward_test_filter=forward (allModels_clf_testf,Pred_clf_testf,Ytest,'test')
print ('fscore_Forward_test_filter=',forward_test_filter[0])

backward_test_filter= backward(allModels_clf_testf,Pred_clf_testf,Ytest,'test')
print ('fscore_backward_test_filter=',backward_test_filter[0])

best_predict_final_test_filter = backward_test_filter[1]+forward_test_filter[1]+[np.array(voting_test_filter[1],dtype=float)]
all_comb_test_filter=all_comb (best_predict_final_test_filter,Ytest,"test")
print('best_score_final_test_filter=',all_comb_test_filter)
```

```
best_clasifire_test_filter= 0.8158835546475995
voting_test_filter= 0.8128192032686414
fscore_Forward_test_filter= 0.813585291113381
fscore_backward_test_filter= 0.813585291113381
best_score_final_test_filter= 0.8133299284984679
```

۳-۲-۹- انتخاب بهترین دسته بند

بدین منظور تابعی برای انتخاب بهترین دسته بند با نام best_clasifire نوشته شد. ورودی این تابع لیست تمام مدل‌ها،ارزیابی‌ها و پیش‌بینی‌ها می باشد.با تبدیل لیست مدل‌ها با استفاده از تابع DataFrame کتابخانه پاندا به دیتافریم تبدیل و نام‌های Modle و Fscore برای ستون های آن درنظرگرفته شده است.با استفاده از تابع max بیشترین Fscore از بین تمام Fscoreها انتخاب شده است. سپس طبق خط شماره ٤ تکه کد زیر بیشترین امتیاز بدست آمده با امتیازهای دیتافریم all_M_D_frame مقایسه شده و مدل متناسب با آن بدست آمده است.که در best_Modle به صورت یک تاپلی از مدل و امتیاز قرار گرفته است .با استفاده از مشخصه index اندیس صفر آن که همان نام مدل است به‌دست آمده است. در انتها نام و امتیاز و پیش بینی بهترین مدل برگردانده می‌شود.

In [116]:
```
def best_clasifire(list_allModle,list_Fscore,list_pred,name):
    all_M_D_frame =pd.DataFrame(list_allModle,columns=['Modle','Fscore'])
    best_Fscore=(max(list_Fscore))
    best_Modle=all_M_D_frame[all_M_D_frame['Fscore']==best_Fscore]
    best_pre=list_pred[best_Modle.index[0]]
    return best_Fscore,best_pre,best_Modle
```

با استفاده از تابع تعریف شده بهترین مدل با امتیاز ۰,۸۱۰ انتخاب شد.

```
best_clasifire_dev=best_clasifire(z0_allModels_clf,z0_Fscoer_clf,z0_Pred_clf,'dev')
print ('best_clasifire_dev=',best_clasifire_dev[0])
```

۳-۲-۱۰- رای گیری اکثریت

هر مدل یک پیش‌بینی (رای‌گیری) برای هر نمونه آزمایش و پیش‌بینی خروجی نهایی همان چیزی است که بیش از نیمی از آرا را دریافت می‌کند . اگر هیچ یک از پیش‌بینی‌ها بیش از نیمی از آرا را به دست نیاورند، می‌توانیم بگوییم که روش گروهی نمی‌تواند یک پیش‌بینی پایدار برای این نمونه ایجاد کند . اگر چه این یک تکنیک بسیار مورد استفاده است، شما ممکن است بیش‌ترین پیش‌بینی را امتحان کنید (حتی اگر آن کم‌تر از نیمی از آرا) به عنوان پیش‌بینی نهایی باشد [32]. که در این پژوهش رویکرد بیش ترین پیش بینی انتخاب شده است.

در ابتدا ماژول رای‌گیری اکثریت به برنامه اضافه شده‌است.

```
from sklearn.ensemble import VotingClassifier
```

فراخوانی ماژول رای گیری اکثریت

فایل پیش بینی‌های که در فایل ذخیره شده بود با ماژول pickle جهت خواندن فراخوانی شده‌است. جهت

انجام رای‌گیری اکثریت یک تابع نوشته شده است،که لیست پیش بینی‌ها و برچسب‌های داده اعتبارسنجی را دریافت کرده ، سپس یک لیست خالی به نام Colmns ایجاد شده با استفاده از حلقه for ای که به اندازه طول یکی از لیست پیش بینی‌ها می باشد با تابع append لیستی از تمام شماره‌های پیش بینی‌ها به آن اضافه می‌شود . این لیست به اندازه طول تمام مدل‌هایی است که ساخته شده‌است.

```
    Columns=[]
    for i in range(len(predict_list[0])) :
        Columns.append('pred%d'%i)
```

Columns= [pred1, pred2, pred3 … pred411]

سپس لیست پیش‌هایی با استفاده‌از تابع DataFrame و با در نظر گرفتن لیست Columns به عنوان سرستون‌ها به یک دیتافریم تبدیل شده است.

```
    df=pd.DataFrame(predict_list,columns=Columns)
```

یک لیست خالی به نام pre_final برای پیش‌بینی‌های نهایی ایجاده شده‌است.

حلقه for دومی به تعداد پیش بینی های انجام شده توسط مدل‌ها که در دیتافریم موجود می‌باشد، بررسی می کند که برای نمونه تمام مدل ها چه برچسبی را پیش‌بینی کرده‌اند و طبق آن‌ها با استفاده از max_freq رای‌گیری اکثریت انجام داده است.و در انتها امتیاز Fscore و پیش بینی نهایی را برمی‌گرداند. امتیاز Fscore با این رویکرد ۰,۸۲۰ به دست آمده است.

In [117]:
```
def modle_voting (lable,predict_list,name):
    Columns=[]
    for i in range(len(predict_list[0])) :
        Columns.append('pred%d'%i)
    df=pd.DataFrame(predict_list,columns=Columns)
    pre_final=[]
    for i in range (df.shape[1]):
        a=df['pred%d'%i].tolist()
        max_freq=np.bincount(a).tolist()
        pre_final.append(max_freq.index(max(max_freq)))
    F_eclf1=precision_recall_fscore_support(lable,pre_final, average='micro')[2]
    return F_eclf1,pre_final


```

پیش بینی نهایی

۳-۲-۱۱- انتخاب رو به جلو

انتخاب رو به جلو یک نوع رگرسیون گام به گام است که با یک مدل خالی شروع می شود و طبقه بندهایی را

که به صورت نزولی بر مبنای یکی از معیارهای ارزیابی مرتب شده اند را یک به یک اضافه می کند و بهترین متغیر ، با برخی از معیارهای از پیش تعیین شده تعیین می شود و به مدل اضافه می گردد. معیار استفاده شده برای تعیین اینکه کدام مدل اضافه شود ، متفاوت است. در هر مرحله رو به جلو، شما یک طبقه بند را اضافه می کنید.در اینجا برای انجام مقایسات،Fscore ها مدنظر قرار داده شده اند[33] .

بدین منظور تابعی با نام forward نوشته شده است.این تابع لیست مدل‌ها، پیش‌بینی‌ها،برچسب و نام را به‌عنوان ورودی دریافت می‌کند.

```
def forward (list_modles,list_pred,lable,name):
```

در این تابع ابتدا با استفاده از یک حلقه for تمام مدل‌ها، پیش‌بینی‌ها و امتیازهایشان در لیست clf_forward به صورت واحد قرارگرفته‌اند . این عملیات با استفاده از تابع append لیست انجام گرفته‌است.

```
    clf_forward=[]
    for i in range (len(list_modles)):
        temp=[]
        temp.append(list_modles[i][1])
        temp.append(list_modles[i][0])
        temp.append(list_pred[i])
        clf_forward.append(temp)
```

قرار دادن تمام مدل ها و پیش بینی ها و امتیازهایشان در یک

سپس لیست به صورت نزولی براساس Fscore مرتب شده است. این مرتب‌سازی با استفاده از تابع Sorted انجام گرفته است . در این تابع reverse=False لیست داده شده clf_forward را به صورت نزولی مرتب می کند. اولین مدل و Fscore لیست را به عنوان بهترین مدل و امتیاز درنظرگرفته شده است.

```
    sorted(clf_forward, key = itemgetter(0),reverse=False)
    best_score_forward=clf_forward[0][0]# افسکور
    # print(best_score_forward)
    best_pre_forward=[list_pred[0]]
```

مرتب کردن لیست به صورت نزولی

همان‌طور که مشاهده می‌کنید با استفاده از یک حلقه مدل‌ها یکی یکی به لیست templist اضافه شده‌اند و رای‌گیری اکثریت انجام شده، امتیاز و پیش‌بینی ها در دو لیست score_forward و pre_forward قرار گرفته است.

```
    for i in range(len(list_pred)):
        templist=best_pre_forward+[clf_forward[i][2]]
        p_voting= modle_voting (lable,templist,name)
        score_forward = p_voting[0]
        pre_forward =p_voting[1]
```

ذخیره کردن امتیاز و پیش بینی ها در دو لیست

در انتها با یک حلقه for مقایسه بین امتیازها صورت گرفته و تابع forward بهترین امتیاز و پیش‌بینی را برمی‌گرداند. بهترین امتیاز این رویکرد ۰,۸۲۲ بدست آمد.

```
        if score_forward > best_score_forward:
            best_score_forward = score_forward
            best_pre_forward=templist
        #     print(best_score_forward)
#     print('fscore_Forward%s='%name,best_score_forward)
    return  best_score_forward,best_pre_forward
```

مقایسه بین امتیاز ها

۳-۲-۱۲- حذف رو به عقب

انتخاب ویژگی به عقب مرتبط است ، و ممکن است با انتخاب کل مجموعه مدل‌ها شروع می‌شود و از آنجا به عقب کار می‌کند ، ویژگی‌هایی برای پیدا کردن زیر مجموعه بهینه از یک اندازه از پیش تعریف‌شده را حذف می‌کند [33].

برای این رویکرد هم یک تابع با نام backward تعریف شده‌است.

```
def backward (list_modles,list_pred,lable,name):
```

همانند رویکرد قبلی تمام مدل‌ها به همراه پیش‌بینی و امتیازشان به طور مجتمع در یک لیست به اسم clf_backward قرارگفته است.

```
clf_backward=[]
for i in range (len(list_modles)):
    temp=[]
    temp.append(list_modles[i][1])
    temp.append(list_modles[i][0])
    temp.append(list_pred[i])
    clf_backward.append(temp)
```

قرار دادن تمام مدل ها و پیش بینی ها و امتیازهایشان در یک لیست

در این رویکرد لیست ایجاد شده،به صورت صعودی مرتب شده‌است.

```
sorted(clf_backward, key = itemgetter(0),reverse=True)
```

مرتب کردن لیست به صورت صعودی

یک لیست خالی به نام predictbl ایجاد شده‌است که با استفاده از یک حلقه for تمام پیش‌بینی ها در آن قرارداده شده‌است.

```
predictbl=[]
for i in range(len(list_pred)):
    predictbl.append(clf_backward[i][2])
```

ذخیره کردن پیش بینی ها در لیست

رای‌گیری اکثریت روی پیش‌بینی تمام مدل‌ها و برچسب داده اعتبار سنجی انجام گرفته‌است. در ابتدا امتیاز و پیش‌بینی حاصل به عنوان بهترین در نظرگرفته شده‌است.

```
backward= modle_voting(lable,list_pred,name)
best_score_back=backward[0]
best_pre_back = [backward[1]]
```

```
templist=list (range (len(predictbl)))
for i in range (1,len(predictbl)):
    templist[i]=predictbl[:-i]
```

یک حلقه for به اندازه طول تمام پیش بینی ها ایجاد شده است .بر روی templist رای گیری اکثریت انجام شده است در صورتی که این امتیاز از بهترین امتیاز قبلی بیشتر است با آن جایگزین می شود .بهترین امتیاز و

پیش‌بینی برگردانده می‌شود. امتیاز نهایی بدست آمده با این رویکرد ۰,۸۲۳ می باشد.

```
    for i in range(1,len(predictbl)):#
        p_voting= modle_voting (lable,templist[i],name)
        if best_score_back < p_voting[0]:
            best_score_back = p_voting[0]
#             print(best_score_back)
            best_pre_back=templist[i]
    return best_score_back,best_pre_back
```

۳-۲-۱۳- ترکیب همه رویکردها:

در این رویکرد پیش بینی تمام رویکردها با هم جمع و در یک لیست به نام best_score_final قرار داده شده و بر روی این پیش‌بینی‌ها و برچسب داده اعتبارسنجی رای‌گیری اکثریت انجام گرفته است. Fscore این رویکرد۰,۸۲۳ بدست آمده است.

In [186]:
```
def all_comb (list_allbestpred,lable,name):
    best_score_final = modle_voting(lable,list_allbestpred,name)
    return best_score_final[0]
```

امتیاز نهایی در روش ترکیب همه رویکردها

فصل چهارم :

محاسبات و یافته های تحقیق

٤-١- داده های مورد استفاده

برای تشخیص نویسنده از مجموعه داده ی Spooky_Author_Identification، از مخزن مسابقه سایت Kaggle استفاده شده است. این مجموعه داده شامل ۲ ویژگی به نامهای id (یک شناسه منحصر به فرد برای هر جمله است) و text (متون نوشته شده توسط نویسندگان) و یک مقدار خروجی به نام Author(نویسندگان جملات) است. متون استفاده شده از آثار نویسندگان رمانهای ترسناک Spooky ، می باشد. این مجموعه داده شامل سه کلاس (EAP[۵۱]، HPL[۵۲] ، MWS[۵۳]) می باشد.

	id	text	author
0	id26305	This process, however, afforded me no means of...	EAP
1	id17569	It never once occurred to me that the fumbling...	HPL
2	id11008	In his left hand was a gold snuff box, from wh...	EAP
3	id27763	How lovely is spring As we looked from Windsor...	MWS
4	id12958	Finding nothing else, not even gold, the Super...	HPL

کلاس EAP تعداد ۷۹۰۰ متن و کلاس HPL ٦۰٤٤ متن و MWS شامل ٥٦۳٥ متن می باشد. که برای هر متن یک شناسه در نظر گرفته شده است.

٤-۲- تنظیم پارامترها

٤-۲-۱- پارامترهای الگوریتمsvc:

۱. C:float,optional(default=1.0)

فرمول محاسبه c به صورت زیر می باشد:

---

[۵۱] EAP : Edgar Allan Poe

[۵۲] HPL : HP Lovecraft

[۵۳] MWS : Mary Shelley

$$C\sum_{i=}^{n}\mathcal{L}(f(x_i),y_i)+\Omega(\omega)$$

مقدار ذاتی c همان یک است.هرچقدر c بیشتر باشد،حول همسایه ها می چرخد و اگر سطحی که در می آید کمتر باشد،سطح تخت می شود (هرچه بیشتر باشد سطح خروجی پستی و بلندی بیشتری دارد) مقدار C هر چقدر کمتر باشد میزان هواری بیشتر است مقدار این پارامتر بسته به نوع دیتاستی است که ما دراختیار داریم.

LinearSVC(C=2.5, penalty="l2", dual=False ,tol=0.0025)

۲. penalty:string,'L1' or 'L2'(default='L2')

نوعی پنالتی است که به دو نوع L1 و L2 است .پیش فرض آن L2 است و ازآن بهترین جواب بدست می آید.اگر L2 را جایگزین کنیم همانظور که در تصویر زیر می بینیم بهترین جواب را (Fscore=0.794) خواهیم داشت.

۳. dual: در این الگوریتم مسئله ما مقدار dual نیست یعنی این معیار برای پارامترهای دومجهولی است اما در اینجا ما یک مجهول داریم و نیازی به تغییر این پارامتر نمی باشد.

٤. tol : یعنی چقدر تابع svc ، برای بهینه سازی است. این تابع بر اساس بهینه سازی عمل می کند . مقدار حد آستانه برای این پارامتر tol=0.0025 است، یعنی از آن حد کمتر تغییر کند سیستم را break می کند. هرچقدر عدد tol=0.0025 کوچک باشد دقت بیشتر می شود و جواب بهتری خواهیم گرفت اما زمان محاسبات بالاتر خواهد رفت.

لیست پارامترهای تنظیم شده توسط کاربر، برای الگوریتم LinearSVC را در جدول (۱-٤) مشاهده می کنید.

جدول ٤-۱: پارامترهای تنظیم شده توسط کاربر

الگوریتم LinearSVC	
نام پارامتر	**مقدار**
C	2.5
Penalty	L2
Dual	False

Tol	0.0025

٤-٢-٢- پارامترهای الگوریتم Logistic Regression :

این مدل حتی بااینکه در نام خود " regression " دارد ، logistic regression یک روش دسته بندی است. در این جا ما هیچ hyperparameters را تنظیم نمی کنیم . ما فقط باید مدل را بسازیم و آموزش دهیم .

LogisticRegression(C=8)

پارامتر های تنظیم شده برای الگوریتم Logistic Regression را در جدول ذیل مشاهده می کنید.

جدول ٤-٢: پارامترهای تنظیم شده توسط کاربر

الگوریتم Logistic Regression	
نام پارامتر	**مقدار**
c	8
Class Weight	None

٤-٢-٣- Extra trees :

برای این الگوریتم، پارامتری تغییر داده نشده و از پیش فرض‌های خود الگوریتم استفاده و مدل ساخته شده است. ارزیابی مدل بر روی داده اعتبارسنجی انجام شده‌است.

ExtraTreesClassifier()

٤-٢-٤- Naïve Bayes :

ذاتا پارامتر زیادی جهت تغییر ندارد و به همین دلیل پرانتز خالی می باشد.

GaussianNB()

مقدار این پارامترها ی ذکر شده به وسیله آزمون و خطا بر رروی داده های اعتبارسنجی به دست آمده است.

یعنی با آزمایش مقادیر مختلف، مقداری را برای پارامتر ورودی انتخاب می‌کنیم که بهترین کارایی را داشته است.

٤-٣- روش‌های استفاده شده به منظور مقایسه

روش‌هایی که برای مقایسه با روش پیشنهادی در نظر گرفته شده‌اند، عبارتند از:

- SVM:

الگوریتم SVM یکی از الگوریتم‌های معروف در زمینه یادگیری با نظارت است که برای دسته‌بندی و رگرسیون استفاده می‌شود. این الگوریتم به طور هم‌زمان حاشیه‌های هندسی را بیشینه کرده و خطای تجربی دسته‌بندی را کمینه می‌کند لذا به عنوان دسته‌بندی حداکثر حاشیه[٥٤] نیز نامیده می‌شود [23].

- NB :

یک الگوریتم یادگیری ماشین برای حل مشکلات طبقه بندی است. برای ساخت مدل و پیش بینی سریع از قضیه احتمال استفاده می کند. اساس قضیه Bayes ، بر پایه استقلال ویژگی ها است. یک روش مبنا برای طبقه بندی متن و حل مشکل قضاوت اسناد درباره تعلق یک سند به یک نویسنده است که از روش تعیین فرکانس کلمه به عنوان ویژگی استفاده می کند.

- ET :

مجموعه ای از درخت تصمیم گیری یا رگرسیون غیرمعمول را ایجاد می کند با توجه به روش کلاسیک بالا به پایین .دو تفاوت عمده نسبت به سایر روش های گروهی، بر پایه دسته بندی دارد . اولا گره ها را با انتخاب نقطه های برش به صورت تصادفی جدا می کند و دوما از کل نمونه یادگیری (بوت استرپ)برای رشد درخت استفاده می کند.

Logistic Regression :

[٥٤] maximum margin

به بیان ساده‌تر این الگوریتم احتمال رخداد یک رویداد را بر حسب گنجاندن داده‌ها در یک تابع منطقیی پیش‌بینی می‌کند . از این‌رو به نام رگرسیون logit نیز شناخته می‌شود. از آنجا که این الگوریتم، احتمال را پیش‌بینی می‌کند، مقادیر خروجی آن بین ۰ تا ۱ هستند.

- رای گیری اکثریت:

رای گیری اکثریت ما بین پیش‌بینی برای هر نمونه توسط مدل های ساخته شده ، بدین صورت که هر پیش بینی که بیشترین تکرار را داشته باشه به عنوان پیش بینی اصلی انتخاب میشود.

- Forward :

انتخاب رو به جلو با یک مدل خالی شروع می شود و طبقه بندهایی را که به صورت نزولی بر مبنای یکی از معیار ارزیابی F1 مرتب شده اند را یک به یک اضافه می کند و در هر بار رای گیری اکثریت صورت می پذیرد و بهترین متغیر ، با برخی از معیارهای از پیش تعیین شده، تعیین می شود و به مدل اضافه می گردد.

- Back Ward :

انتخاب ویژگی به عقب مرتبط است ، و با انتخاب کل مجموعه مدل‌ها شروع می‌شود و از آنجا به عقب کار می‌کند ، ویژگی‌هایی برای پیدا کردن زیر مجموعه بهینه از یک اندازه از پیش تعریف‌شده را حذف می‌کند .

همانند رویکرد قبلی تمام مدل‌ها در هر بار رای گیری اکثریت صورت می پذیرد و بهترین متغیر ، با برخی از معیارهای از پیش تعیین شده، تعیین می شود و به مدل اضافه می گردد.

٤-٤- نتایج

روش پیشنهادی در پایتون[٥٥] ۳,۷ با استفاده از Jupyter پیاده‌سازی و در یک کامپیوتر شخصی با پردازشگر پنج هسته‌ای اینتل[٥٦] با سرعت ۵,۲۰۰ گیگا هرتز و حافظه با دست‌یابی ۸ گیگا بایت به اجرا در آمده است. همچنین این روش بر روی مجموعه داده ، برای تشخیص نویسنده از مجموعه داده ی Spooky_Author_Identification، از مخزن مسابقه سایت Kaggle محک زده شده است.

مهم‌ترین معیارهایی که برای برازش کارایی الگوریتم پیشنهادی در نظر گرفته شده است، عبارتند از: نرخ افسکور F1. همان‌طور که در فصل دوم بیان شد؛(میانگین هارمونی برای دو مقدار بازخوانی و صحت)

جدول ۳-۴- بهترین نتیجه بدست آمده از الگوریتم های ترکیب بر روی مجموعه داده تست

نام مجموعه داده	نرخ صحت
Test	۰,۸۲۳۵

جدول ۴- ۴:مقایسه نتایج بدست آمده برای مجموعه داده Spooky_Author_Identification با سایر روش‌ها

نام روش	نرخ سنجش-F BOW_bin	نرخ سنجش-F BOW	نرخ سنجش-F TFIDF	نرخ سنجش-F TFIDFبا اعمال فیلتر
LinearSVC	۰,۷٦٥٥	0.7630	0.8003	---------
NB	۰,٦٥١٥	0.6521	0.6557	---------
LR	۰,۸۰٦٤	0.8082	۰,۸۱٥۸	---------

[٥٥] PYTHON 3.7

[٥٦] Intel Core i3

ER	0.6695	0.6640	0.6999	--------
رای گیری اکثریت	--------	--------	0.8222	0.8128
Forward	--------	--------	0.8225	0.8135
Backward	--------	--------	0.8235	0.8135
رای گیری کل	--------	--------	0.8220	0.8133

ارزش استفاده از الگوریتم با توجه به نتایج به وضوح قابل رویت می باشد.

٤-٥- نتیجه گیری

نتایج بدست آمده نشان می‌دهد دقت روش پیشنهادی ترکیب مدل ها در دسته‌بندی مجموعه داده‌های تشخیص نویسنده بهتر یا حداقل قابل رقابت با الگوریتم‌های مطرحی مانند NB و LR وET وSVM می‌باشد. البته الگوریتم ET نیز در اصل استفاده از گروهی از الگوریتم های درخت تصمیم می باشد.

در ارزیابی الگوریتم های ترکیبی نیز دو رویکرد با اعمال فیلتر بر اساس معیار F1 مورد سنجش قرار گرفت .بدین صورت که مدل هایی که دارای معیارهای F1 بالاتر از حد میانگین F1 ها مربوط به مدل ها انتخاب، و در روش های ترکیب مورد استفاده قرار گرفت .

تعداد مدل های انتخاب شده از مجوع ٤١١ مدل ساخته شده برابر ٢٠٥ مدل می باشد.

نتایج بدست آمده از این رویکرد بهبودی نسبت به حالت بدون این رویکرد نداشته است و شاید بتوان این نتیجه را استنباط کرد که در روش های ترکیب هرچه تعداد مدل های رای گیری بیشتر باشد نتیجه بهتری می توان گرفت که البته این موضوع را نباید نادیده گرفت که در این پژوهش الگوریتم های استفاده شده اکثرا دارای نتایج بالای ٠,٦٥ می باشند و شاید اگر الگوریتم های دیگری مانند درخت تصمیم که نتایج آن ها حداکثر ٠,٥٥بوده (بر اساس آزمون های انجام شده) در این پژوهش دخیل بودند شاهد چنین نتایجی نبودیم .

فلذا این فرضیه را باتوجه به ویژگی های مجموعه داده انتخابی و نوع الگوریتم های مورد استفاده باید در نظر گرفت .

یعنی زمانی که الگوریتم ها همگی دارای نتایج نسبتا بالایی می باشند استفاده از رویکرد فیلتر کردن بر اساس حد آستانه میانگین معیار ارزیابی F1 مدل ها ، بهبودی در نتایج ندارد.

لیکن در استفاده از روش های ترکیب ،می توان نتیجه گرفت کاربرد آن ها ، نتایج را بهبود داده و در درنیای واقعی کمتر دچار خطا می شود ویا به عباراتی دارای پایداری نسبی در نتایج می باشند .

فصل پنجم :

نتیجه گیری و مشاهدات

۵-۱- خلاصه و نتیجه گیری

در این پژوهش روشی را برای واکشی دانش مطرح کردیم. دانش مورد نظر توسط فرایند داده‌کاوی و به کمک روش دسته‌بندی به دست آمده است. دسته‌بند طراحی شده توسط معیارهای **F1** ارزیابی شدند.

که با استفاده از روش های معمول مجموعه داده آموزشی به قسمت های آموزش و اعتبارسنجی و آزمایش تقسیم و پاکسازی و نرمال سازی بر روی آن ها صورت پذیرفت. سپس با استفاده از سه رویکرد کیسه کلمات دودویی و کیسه کلمات با فرکانس کلمات و فراوانی کلمه کلیدی ویژگی های هر متن استخراج و به بردار تبدیل گردید .که با استفاده از الگوریتم های داده کاوی و استفاده از روش های یادگیری تجمعی بر روی مجموعه داده های اعتبارسنجی سنجیده و سپس برروی داده آزمایش مورد آزمایش قرار گرفت.

نتایج بدست آمده نشان دادند که استفاده از روش قراردادن میانگین **F1** مدل های ساخته شده به عنوان حد آستانه برای انتخاب مدل ها بهبودی در نتایج نسبت به حالت بدون استفاده از این تکنیک نداشته و شاید بتوان این نتیجه را استنباط کرد که در روش های ترکیب هرچقدر تعداد مدل های رای گیری بیشتر باشد نتیجه بهتری می توان گرفت ، که البته این موضوع را نباید نادیده گرفت که در این پژوهش الگوریتم های استفاده شده اکثرا دارای نتایج بالای بودند و شاید اگر الگوریتم های دیگری که دارای نتایج پایین تری بودند در این پژوهش مورد استفاده قرار می گرفت از مقایسه نتایج می توانست نتیجه ای عکس گرفت.

فلذا این فرضیه را می بایست باتوجه به ویژگی های مجموعه داده انتخابی و نوع الگوریتم های مورد استفاده باید در نظر گرفت .یعنی زمانی که الگوریتم ها همگی دارای نتایج نسبتا بالایی می باشند استفاده از رویکرد فیلتر کردن بر اساس حد آستانه میانگین معیار ارزیابی **F1** مدل ها ، بهبودی در نتایج ندارد.

لیکن در مقایسه نتایج روش های ترکیب با سایر روش ها در این پژوهش، می توان نتیجه گرفت کاربرد آن ها ، در بهبود نتایج موثر هستند و در درنیای واقعی به دلیل به کارگیری رای گیری اکثریت مابین مدل های ساخته شده کمتر دچار خطا می شوند ویا به عباراتی دارای پایداری نسبی در نتایج می باشند .

۵-۲- پیشنهادات

برای کارهای آتی در این زمینه می‌توان پیشنهادات زیر را در نظر گرفت:

- با توجه به اینکه در این پژوهش مدل ها اکثرا با ستفاده از فراوانی کلمه کلیدی ساخته شده است ،اضافه نمودن مدل های بیشتر با استفاده از کیسه کلمات دودویی و کیسه کلمات با فرکانس می تواند نتایج متفاوت تری را نشان بدهد.
- یادگیری روش‌های ترکیبی با در نظر گرفتن اینکه در رای گیری اکثریت هریک از پیش بینی‌ها می بایست بیش از نیمی از آرا را به دست بیاورند.
- استفاده از ویژگی های ظاهری مانند تعداد کل کلمات ، تعداد کلمات اضافه ، میانگین طول کلمات ، تعداد کلمات استپ ورد ، تعداد کلمات شاخص و .... در هرمتن جهت ساخت مدل مربوطه و ترکیب یا روش های یادگیری تجمعی که در این پژوهش آورده شده.
- به کارگیری قواعد انجمنی و کشف روابط با استفاده از ویژگی ها با سطح تکرار مشخص و فروانی بالا

مراجع

[1] O. Maimon and L. Rokach, “Data Mining and Knowledge Discovery Handbook,” pp. 1–15, 2005.

[2] H. Jang, S. Kim, and T. Lam, “Kaggle Competitions : Author Identification & Statoil / C-CORE Iceberg Classifier Challenge,” pp. 1–21, 2017.

[3] M. Kestemont *et al.*, “Overview of the author identification task at PAN-2018: Cross-domain authorship attribution and style change detection,” *CEUR Workshop Proc.*, vol. 2125, 2018.

[4] https://searchsqlserver.techtarget.com/definition/data-mining

[5] https://www.geeksforgeeks.org/basic-concept-classification-data-mining/

[6] https://hub.packtpub.com/what-is-ensemble-learning/

[7] https://simplicable.com/new/ensemble-learning

[8] https://www.tutorialspoint.com/data_mining/dm_knowledge_discovery.htm

[9] https://www.techopedia.com/definition/1181/data-mining

[10] https://www.geeksforgeeks.org/basic-concept-classification-data-mining/

[11] https://www.tutorialride.com/data-mining/classification-in-data-mining.htm

[12] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[13] https://scikit-learn.org/stable/

[14]D. Heckerman and J. S. Breese, “Causal independence for probability assessment and inference using Bayesian networks,” *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans.*, vol. 26, no. 6, pp. 826–831, 1996.

[15] A. Genkin, D.Lewis, “Author Identification on the Large Scale”, 2005

[16] https://www.edureka.co/blog/naive-bayes-tutorial/

[17] https://www.geeksforgeeks.org/understanding-logistic-regression/

[18] scikit-learn.org/stable/.../sklearn.ensemble.ExtraTreesClassifier.html

[19] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

[20] https://blog.faradars.org/

[21] http://www.bigdata.ir/1397/03/

[22] https://howsam.org/

[23] http://research-moghimi.ir/1395/05/15/ensemble-learning/

[24] M.Brusco, D.Steinley, J.Cradit,” An exact algorithm for hierarchically well-formulated subsets in second-order polynomial regression”, 2009

[25] https://dataio.ir/how-to-import-dataset-with-pandas-r1gsggvg9xx2

[26] https://blog.faradars.org/python-excel-tutorial/

[27] https://www.datopia.ir/

[28] K. Matsumoto, “Classification of Emoji Categories from Tweet Based on Deep Neural Networks,” pp. 17–25, 2018

[29] http://www.statsoft.com/textbook/naive-bayes-classifier
[30] https://towardsdatascience.com/logistic-regression-b0af09cdb8ad
[31] https://www.quora.com/What-is-the-C-parameter-in-logistic-regression
[32] https://www.toptal.com/machine-learning/ensemble-methods-machine-learning
[33] https://xavierbourretsicotte.github.io/subset_selection.html
[34] M.Rastegar, S.Hosseinzadeh, E.Bakhshi, "Application of Logistic Regression with Missclassified Variables in Diabetes Data", 2018