دانشکده علوم و فنون نوین

تمرین اول درس شبکه های عصبی مصنوعی

دانشکده علوم و فنون نوین

محمد هادی رهجو
rahjooh@gmail.com
09126780521

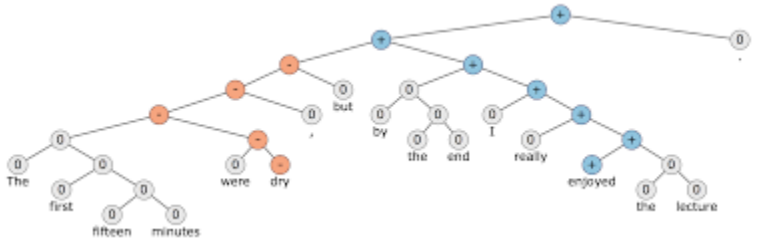
اسفند 97

۱. (٪۱۰) [**مطالعه و موضوع پروژه**] شبکه‌های عصبی مصنوعی در کاربردهای مختلفی و به شیوه‌های متنوعی به کار گرفته شده‌اند. هدف این تمرین، مروری بر کاربردها و نحوه به‌کارگیری آنها در زمینه مربوط به رشته شماست. برای این کار، هر دانشجو، حداقل دو کاربرد مختلف را که در مقالات علمی آمده‌اند، مورد مطالعه قرار داده و گزارش آن را به صورت مکتوب (الکترونیکی) ارسال کند. پاسخ خود را به صورت یک گزارش علمی تهیه کرده و در آن، منابع علمی مورد استفاده (با تاکید بر ژورنال‌ها و پایان‌نامه‌ها) را به صورت استاندارد بیان کنید. منابع مورد استفاده خود را نیز به همراه گزارش تحویل دهید. از میان موضوع‌های مطالعه شده، کدامیک را به عنوان موضوع پروژه برمی‌گزینید؟

همانطور که می دانیم اولین عصب مصنوعی در سال 1943 توسط "وارن مک کلوچ" فیزیولوژیست عصبی و "والتر پیتز" منطق شناس معرفی گردید اما فناوری موجود اجازه کار بیشتر به آنها نداد. تا اینکه شبکه‌های عصبی مصنوعی در دهه 50 به معنی ایجاد یک شبکه برای محاسبه‌ی وظایف منطقی و برای شناخت الگوها به کار گرفته شد . در دهه 60 پیشرفت شبکه‌های عصبی به علت محدودیت‌های مدل‌های پرسپترون تک‌لایه کند شد . در سال 69 " میسنکی و پپرت" به محدودیت‌های تعمیم یافته پرسپترون تک لایه به نظام‌های چندلایه اشاره کردند .اواخر دهه 70 و دهه 80 همراه بود با شکوفایی مجدد و پدیدار شدن شبکه‌های عصبی کارا ، تا اینکه امروزه شبکه عصبی به یکی از پیشرفته ترین و کارامدترین شاخه های هوش مصنوعی وارد دانشگاه‌ها و صنعت گردیده است .

مدیریت مشتری Customer experience

یادگیری ماشین از قبل توسط بسیاری از شرکت‌ها برای افزایش مشتری‌مداری استفاده می‌شود.به عنوان مثال شامل راه‌حل‌های خدمات خودکار آنلاین و ایجاد جریان‌های کاری قابل . امروزه مدل‌های یادگیری عمیقی وجود دارند که برای این کار مورد استفاده قرار می‌گیرند، و همانطور که یادگیری عمیق به بلوغ ادامه می‌دهد، ما انتظار داریم که این یک یادگیری عمیق باشد که برای بسیاری از شرکت‌ها مورد استفاده قرار خواهد گرفت.(Marr, 2018)



پردازش زبان طبیعیNLP

پردازش زبان طبیعی یکی از مهم‌ترین تکنولوژی‌های عصر اطلاعات است. درک گفتار پیچیده زبان نیز بخش مهمی از هوش مصنوعی است. کاربردهای of در همه جا هستند زیرا مردم اغلب همه چیز را به زبان ارسال می‌کنند: جستجوی وب، تبلیغ، ایمیل، خدمات مشتری، ترجمه زبان، گزارش رادیولوژی، و غیره. اخیرا رویکردهای یادگیری عمیق عملکرد بسیار بالایی را در بسیاری از وظایف NLP مختلف به دست آورده‌اند. این مدل‌ها غالبا می‌توانند با یک مدل تک انتها به انتها آموزش داده شوند و نیازی به مهندسی ویژگی سنتی و مخصوص به کار نداشته باشند..(stanford.edu, 2017)

در ادامه به چند نمونه از پروژه‌های پرطرفدار این حوزه که با یادگیری عمیق به دستاوردهای خوبی رسیده اند می پردازیم.

ماشین‌های چت خودکار Chat Bots

زمانی که روی لینک پشتیبانی وب سایت bank's یا وب سایت خرید مورد علاقه خود کلیک کنید، یک ربات چت می‌تواند فعال باشد. "چطور می‌توانم به شما کمک کنم؟" پاسخ می‌تواند یک برنامه کاملا خودکار باشد که متن شما را می‌خواند و پاسخ‌های

مربوط به آن را بررسی می‌کند، یا به ساده‌ترین شکل می‌تواند شما را به یک نماینده زنده مناسب هدایت کند. از آنجا که روبات‌های پیچیده تری با استفاده از DNN نوشته می‌شوند، توانایی آن‌ها برای درک اظهارات شما، و مهم‌تر از آن، ربات قادر خواهد بود مکالمات معنی‌دار طولانی‌تر را بدون اینکه شما درک کنید که با یک شخص واقعی صحبت نمی‌کنید، ادامه دهند.[Joel Marcey , Yang qing , jae youn kim, 2017)]

تشخیص زبان Language recognition

ماشین‌های یادگیری عمیق شروع به متمایز کردن گویش‌های زبان می‌کنند.به طور مثال یک ماشین به که یک نفر انگلیسی صحبت می‌کند گوش می‌دهد و سپس یک هوش مصنوعی را درگیر می‌کند که موجب تشخیص تفاوت بین زبان و گویش‌ها را بیان کند. هنگامی که گویش تعیین می‌شود، دیگر می‌توان از بسیاری ابزارها برای اهداف مورد انتظار بهره گرفت. همه اینها بدون دست داشتن از یک انسان رخ می‌دهند. [(Marr, 2018)]

فیلتر کردن اخبار بر پایه احساسات News aggregator based on sentiment

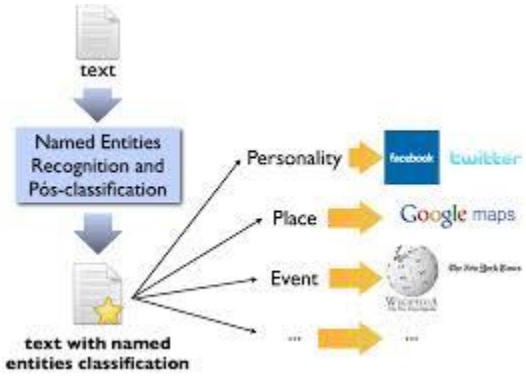
زمانی که می‌خواهید منفی را به دنیای خود فیلتر کنید، پردازش زبان طبیعی پیشرفته و یادگیری عمیق می‌تواند کمک کند. مدل‌های مربوط به استفاده از این فن‌آوری جدید می‌تواند اخبار را براساس آنالیز احساسی فیلتر کند، بنابراین شما می‌توانید جریان‌های خبری ایجاد کنید که فقط اخبار خوب را پوشش می‌دهند. [(Marr, 2018)]

تشخیص گفتار Speech Recognition

بسیاری از خوانندگان ممکن است در معرض Siri اپل قرار داشته باشند. این تعامل اصلی دستیار دیجیتال با کاربران از طریق شناسایی صدا است. برای قرار دادن قرارها در تقویم خود، از Siri تقاضا کنید، و اطلاعات را به دست آورید. توانایی آن برای درک انواع صداها در زبان انگلیسی، چه رسد به تنظیمات و قابلیت‌های چندزبانه آن، براساس بسیاری از پیشرفت‌ها در Siri از سال ۲۰۱۴ است. این پیشرفت از طریق استفاده از شبکه‌های عصبی عمیق ، شبکه‌های عصبی مصنوعی، و دیگر پیشرفت‌ها در یادگیری ماشین انجام شد. (Joel Marcey , Yang qing , jae youn kim, 2017)

ایجاد متن Text Generation

امروزه با ساخت ، توسعه و آموزش مدل‌های یادگیری عمیق می‌توان با آموزش دستورزبان، اصول نگارش، سبک یک قطعه از متن و موضوع ماشین را قادر به ایجاد متن و محتوایی با دستورزبان و سبک متن و املای مناسب ساخت . هر چیزی از نمایشنامه های شکسپیر تا مدخل‌های ویکی پدیا را می‌توان با این مدل‌های تولید نمود. (Marr, 2018)



شناخت نوع موجودیت در متون Named-entity recognition

یکی از شاخه های پردازش زبان طبیعی محسوب می شود که عبارت است از استخراج اطلاعات است که به دنبال تعیین مکان و طبقه‌بندی نام نهاد به عنوان نام فردی، سازمان‌ها، مکان‌ها،نمادهای پزشکی، عبارات زمان، مقادیر پولی، درصد و غیره است.

(DANIEL W. OTTER,JULIAN R. MEDINA,JUGAL K. KALITA, July 2018)

بینایی ماشین Computer vision

یادگیری عمیق دقت انسان را برای طبقه‌بندی تصویر، تشخیص اشیا، بازیابی تصویر و تقسیم‌بندی تصویر ارائه کرده‌است ، حتی ارقام دست‌نوشته شده می‌تواند به رسمیت شناخته شود. یادگیری عمیق با استفاده از شبکه‌های عصبی بزرگ به ماشین‌ها جهت خودکار کردن وظایف انجام‌شده توسط سیستم‌های بصری انسان می‌پردازد. (Marr, 2018)

شبکه‌های عصبی کاربردهای بینایی رایانه‌ای را به طور قابل‌توجهی بهبود بخشیده‌اند. پردازش عکس برای تشخیص شی مورد استفاده قرار می‌گیرد، به سوالاتی مانند "آیا یک گربه یا یک سگ است؟" . پردازش ویدئو برای خودکار کردن طبقه‌بندی صحنه یا تشخیص افراد، پاسخ دادن به پرسش‌هایی مانند " آیا این یک هلیکوپتر است؟ آیا کسی در بالگرد هست؟ این شخص کیست؟ " .

(Joel Marcey , Yang qing , jae youn kim, 2017)

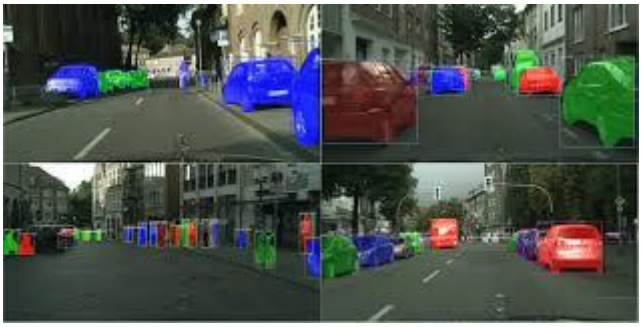
در ادامه به چند نمونه از پروژه‌های پرطرفدار این حوزه که با یادگیری عمیق به دستاوردهای خوبی رسیده اند می پردازیم.

ترجمه Translations

اگرچه ترجمه ماشینی خودکار ، صنعت جدیدی نیست، ولیکن یادگیری عمیق کمک می‌کند که ترجمه خودکار متن با استفاده از شبکه‌های روی هم انباشته (RNN) ترجمه متون در تصاویر را نیز ممکن سازد. (Marr, 2018)

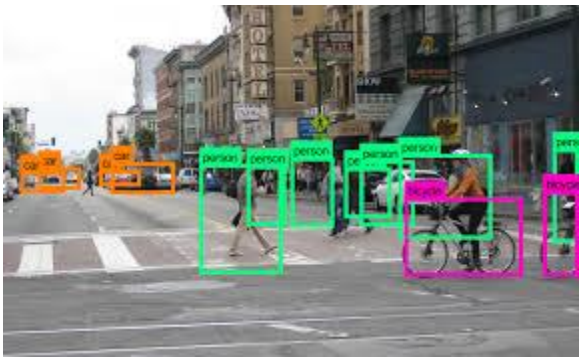
رنگی کردن تصاویر یا فیلم‌های سیاه و سفید

چیزی که عادت داشتیم یک فرآیند بسیار وقت گیر باشد که در آن انسان‌ها باید رنگ به تصاویر و فیلم‌های سیاه و سفید اضافه کنند ،می‌تواند به طور خودکار با مدل‌های یادگیری عمیق انجام شود. (Marr, 2018)



خودروهای خودران Autonomous vehicles

در زمینه رانندگی بدون نیاز به انسان به عنوان راننده شاخه‌های مختلفی وجود دارد که یادگیری عمیق در آن ها در حال بهره برداری قرار گرفته است . برخی از مدل‌هایی که در این زمینه پیاده سازی شده اند به شناسایی علائم راهنمایی و رانندگی پرداخته اند در حالی که برخی دیگر به تشخیص عابر پیاده تمرکز دارند. میلیون‌ها مدل هوش مصنوعی می تواند به کارگرفته شود در حالیکه یک خودرو در خیابان حرکت می کند. (Marr, 2018)



برچسب زدن تصاویر Image caption generation

یک قابلیت قابل‌توجه دیگر یادگیری عمیق، شناسایی یک تصویر و ایجاد یک نوشته منسجم با ساختار زمان مناسب برای آن تصویر به خوبی عملکرد یک انسان است. (Marr, 2018)

پزشکی

ادارات گمرک از پردازش تصویر حرارتی برای شناسایی افرادی استفاده کرده‌اند که ممکن است از تب به منظور اعمال فشار روانی و محدود کردن شیوع بیماری عفونت رنج ببرند. تقسیم‌بندی تصویر یک کار مشترک برای تصویربرداری پزشکی برای کمک به شناسایی انواع مختلف بافت، اسکن برای بی قاعدگی‌ها و ارائه کمک به پزشکان در تجزیه و تحلیل تصاویر در انواع رشته‌ها مانند رادیولوژی و تومورشناسی است. سوابق پزشکی را می توان با ML و DNN پردازش کرد تا بینش و همبستگی در این مجموعه داده‌های حجیم پیدا کند. (Joel Marcey , Yang qing , jae youn kim, 2017)

ربات ها Deep-learning robots

کاربردهای عمیق یادگیری برای ربات‌ها بسیار زیاد و قدرتمند از یک سیستم یادگیری عمیق است که می‌تواند یک ربات را تنها با مشاهده اقدامات یک انسان در یک کار به یک ربات خانگی که با ورودی چندین هوش مصنوعی دیگر به منظور انجام عمل تامین می‌شود، آموزش دهد. درست مانند نحوه ورودی فرایندهای مغز انسان از تجربیات گذشته، ورودی فعلی از حواس و هر اطلاعات اضافی که ارایه می‌شود، مدل‌های یادگیری عمیق به روبات‌ها کمک خواهند کرد تا کارها را براساس ورودی‌های بسیاری از عقاید اطلاعاتی مختلف انجام دهند. (Marr, 2018)

اینترنت اشیاء IoT

زمانی که ما تاثیر کامل و قابلیت‌های اینترنت اشیا را بررسی می‌کنیم، که در آن تکنولوژی مشترک با شما ارتباط دارد - از یخچال شما، به سیستم امنیتی شما، به طور خودکار قادر به تشخیص خودکار بین صاحب‌خانه، میهمان و سارق، و تنظیم روشنایی، موسیقی و صداهای زنگ خطر است. نحوه تمایز این سیستم بین گروه‌ها را می توان با آموزش یک سیستم DNN و سپس انواع مختلفی از سیستم‌ها از قبیل AWS's IoT می‌تواند این ردیاب مرکزی را برای ارایه پاسخ‌ها و اعمال پوشش دهد. (Joel Marcey , Yang qing , jae youn kim, 2017) .

References

DANIEL W. OTTER,JULIAN R. MEDINA,JUGAL K. KALITA. (July 2018). A Survey of the Usages of Deep Learning in Natural. *Deep Learning in Natural Language Processing*, 35.

Joel Marcey , Yang qing , jae youn kim. (2017). *caffe2.ai*. Retrieved from https://caffe2.ai/docs/applications-of-deep-learning.html

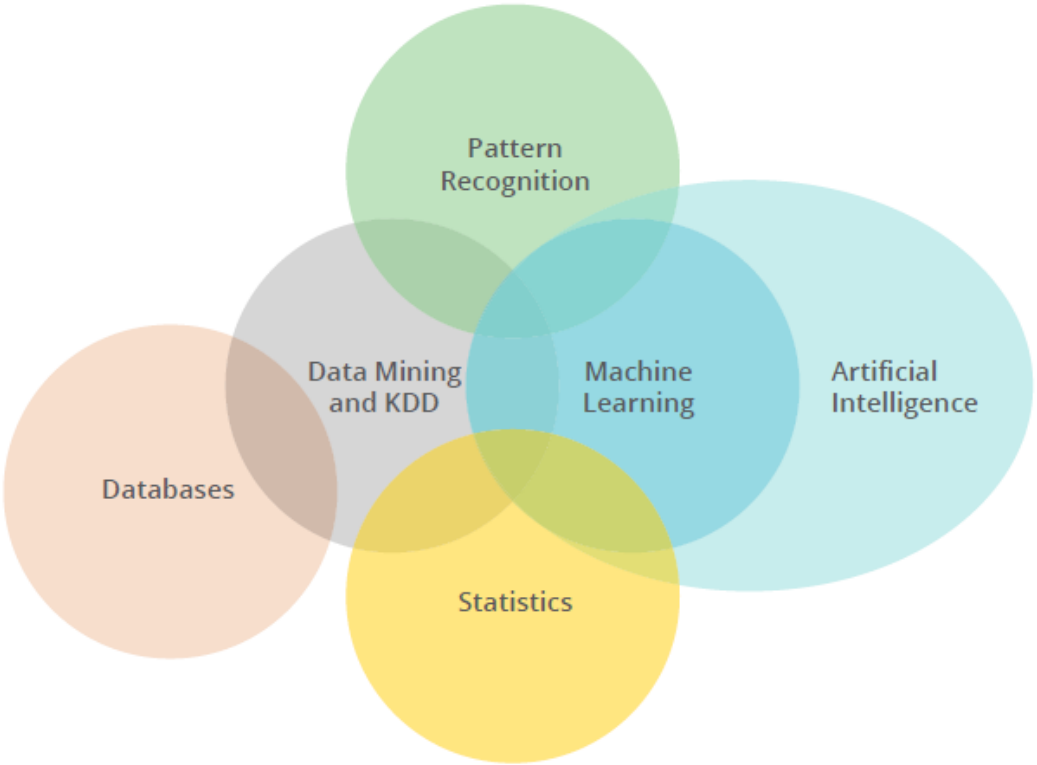
Marr, B. (2018, Aug 20). 10 Amazing Examples Of How Deep Learning AI Is Used In Practice. *Forbes*, p. 10. Retrieved from forbes.com: https://www.forbes.com/sites/bernardmarr/2018/08/20/10-amazing-examples-of-how-deep-learning-ai-is-used-in-practice/#3709bfcbf98a

stanford.edu. (2017, Winter 01). CS224d: Deep Learning for Natural Language Processing. (S. B. James Hong, Interviewer) Retrieved from http://cs224d.stanford.edu/

Wikipedia. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Named-entity_recognition

فکر میکنم یک پروژه NLP با مشاورت با دکتر ویسی میتونه بسیار خوب باشه برای پروژه.

۲. (۱۰٪) [**مطالعه و تحلیل**] تفاوت‌های میان یادگیری ماشین، داده‌کاوی و بازشناسی الگو را ذکر کنید.



هوش مصنوعی

هوش مصنوعی (AI) امکان یادگیری از تجربیات، تطبیق یافتن با ورودیهای جدید و انجام وظایف مانند انسان را برای ماشین‌ها فراهم می آورد. بیشتر مثال‌های AI که امروز درباره آنها می‌شنوید (از کامپیوترهایی که شطرنج بازی می کنند تا خودروهای

خودراننده) اتکای زیادی بر یادگیری عمیق و پردازش زبان طبیعی دارند. با استفاده از این تکنولوژیها، کامپیوترها را می توان برای اجرای ماموریتهای ویژه از طریق پردازش حجم بالایی از داده ها و تشخیص الگوها در داده ها آموزش داد.

داده کاوی

داده کاوی (Data Mining) علم و فنی است که در سالهای اخیر و با گسترش استفاده از فناوری اطلاعات و سیستم های اطلاعاتی مورد توجه سازمان ها قرار گرفت. داده کاوی کاربردهای مختلفی برای سازمان ها دارد و می تواند برای شرکت ها در زمینه های مهمی مثل بازاریابی و فروش بسیار راهگشا و مفید باشد.

یادگیری ماشین

یادگیری ماشین، که از اُلگوشناسی و نظریه یادگیری محاسباتی الهام گرفته شده است، مطالعه و ساخت الگوریتم هایی را که می توانند بر اساس داده ها یادگیری و پیش بینی انجام دهند بررسی می کند – چنین الگوریتم هایی از دستورات برنامه پیروی صرف نمی کنند و از طریق مدلسازی از داده های ورودی نمونه، پیش بینی یا تصمیم گیری می کنند. یادگیری ماشین در کارهای محاسباتی که طراحی و برنامه نویسی الگوریتم های صریح با عملکرد مناسب در آن ها سخت یا نشدنی است، استفاده می شود؛ برخی کاربردها عبارت اند از فیلترینگ ایمیل، شناسایی مزاحم های اینترنتی یا بدافزارهای داخلی که قصد ایجاد رخنه اطلاعاتی دارند، نویسه خوان نوری(OCR) ، یادگیری رتبه بندی، و بینایی ماشین.

تشخیص الگو

بخشی از هوشمند سازی سیستم ها در ایجاد توانایی در آن ها برای تفکیک و دسته بندی الگوهاست. شناسایی الگو یکی از مهم ترین مباحث در زمینه هوش مصنوعی است و طیف وسیعی از مسائل و کاربردها را در بر می گیرد. از این رو مبحث شناسایی الگو در کنار درس یادگیری ماشین از جمله مهم ترین دروس مقطع کارشناسی ارشد است که دیدگاه خوبی در زمینه استفاده از روش های آماری و ریاضی در مدل سازی الگوها و روابط بین آنها با هدف دسته بندی آنها فراهم می نماید. این مبحث دارای کاربردهای فراوانی در حوزه های مختلف از جمله کاربردهای حفاظتی و امنیتی، ایجاد واسط کاربری برای سیستم های کامپیوتری جهت سهولت استفاده از این سیستم ها، ایجاد واسط برای افراد معلول و ناتوان، استفاده در کاربردهای تشخیص پزشکی، ارائه برنامه های کاربردی پردازش تصویر و گفتار و ... می باشد

تفاوت سه موضوع

سردرگمی قابل‌توجهی در زمینه داده کاوی، یادگیری ماشین، و تشخیص الگو بین پژوهشگران و متخصصان ، به دلیل همپوشانی قابل‌توجه در زمینه اهداف و روش‌های این علوم وجود دارد. همیشه یک چالش برای توضیح تفاوت بین این سه حوزه وجود دارد.

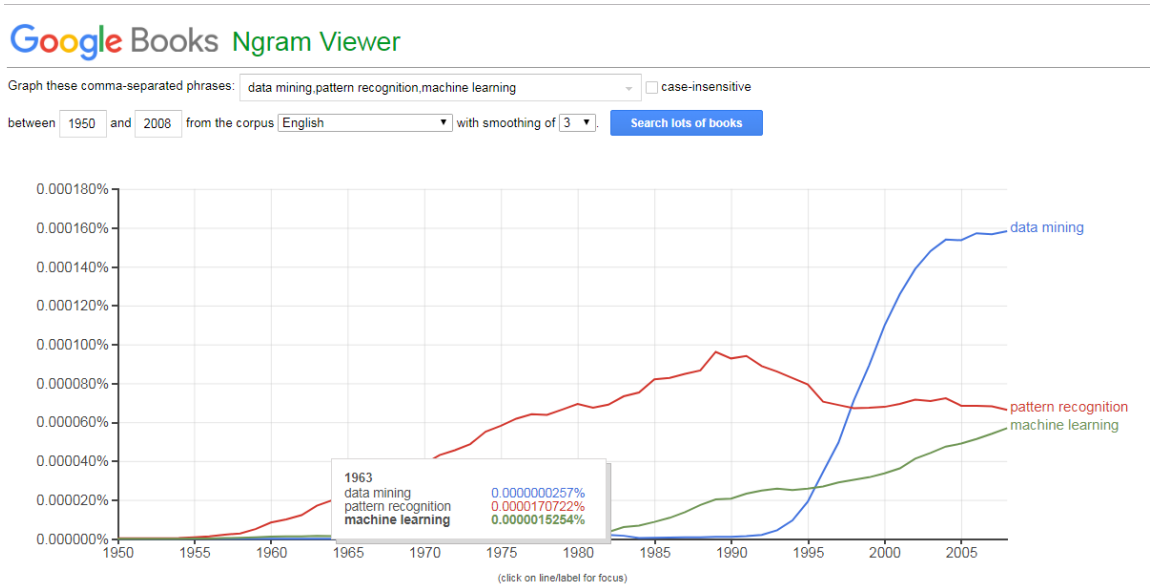
بازشناسی الگو قدیمی‌ترین این سه زمینه است که به اوایل دهه ۱۹۵۰ بر می‌گردد زمانی که محققان در تلاش برای توسعه ماشین‌آلات برای OCR و بازشناسی گفتار بودند. توجه من به شناسایی الگو این است که این حوزه‌ای است که مربوط به طراحی و توسعه سیستم‌ها برای تشخیص و یا الگوهای گروهی - اشیا، سیگنال‌ها، و فرایندهایی است که از طریق برخی از مکانیزم‌های حسی به خود جذب می‌شوند؛ این یک خاصیت مهندسی دارد.

اصطلاح یادگیری ماشین از جامعه هوش مصنوعی ناشی می‌شود و تمرکز آن بر یادگیری روابط حاضر در داده‌ها جهت ساخت مدل‌های طبقه‌بندی است. تاکید بر یادگیری ماشین بر روی مدل‌های الگوریتمی برای یادگیری و خواص آن‌ها است.

اصطلاح استخراج داده در این بازی دیر ظاهر شد و برای مشخص کردن فعالیت‌هایی که دارای تمرکز کاربردی قوی بودند و هدف از آن استخراج الگوهای مفید از داده‌ها، بیشتر در داده‌های تجاری بود، به کار گرفته شد. هر دو روش تشخیص الگو و روش‌های یادگیری ماشین جز مهمی از هر اقدام استخراج داده را تشکیل می‌دهند.

هر سه علم اگرچه اهداف و روش‌هایی نزدیک به یکدیگر دارند ولیکن خواستگاه های آنان جدا بوده است

- علم یادگیری ماشین از دل هوش مصنوعی بوجود آمده است .
- علم بازشناسی الگو از دل سیگنال و پردازش تصویر بوجود آمده است.
- علم داده کاوی از دل پایگاه داده بوجود آمده است .



می توان یک تصویر جالب از این که چگونه این سه زمینه با رفتن به نمایشگر **Ngram** گوگل ظاهر شدند، به دست آورد.

این ابزار نتیجه تلاش عظیم digitalization است که توسط گوگل انجام می‌شود. شما می‌توانید از Ngram بخواهید تا فرکانس استفاده از عبارت‌های بهره را در طول سال‌ها در مجموعه کتاب‌های دیجیتالی از گوگل ترسیم کند. نمودار زیر توسط نمایشگر Ngram برای سه عبارت "استخراج داده + استخراج داده"، "شناسایی الگوی + شناسایی الگو"، و "یادگیری ماشین + یادگیری ماشین" تولید شد. از آنجا که بیننده Ngram حساس است، من هم موارد پایین و هم بالایی از سه عبارت مورد توجه را در نظر گرفتم. همانطور که نمودار نشان می‌دهد، تشخیص الگو در اوایل دهه پنجاه شروع به ظهور کرد. هر دو دوره یادگیری ماشین و هم شرایط داده کاوی بسیار بعدتر مشخص شدند. با آغاز دهه نود، یادگیری ماشین و داده کاوی در محبوبیت افزایش‌یافته است، در حالی که بازشناسی الگو کم‌تر مد روز شده‌است.

۳. (٪۷۰) **[پیاده‌سازی: یک دسته‌بند ساده و معیارهای ارزیابی]** در این مسئله شما یک دسته‌بند ساده را به منظور تشخیص سه زبان فارسی، عربی و کُردی در یک متن طراحی می‌کنید. داده‌های مربوط به این تمرین در فایل ANN-HW1-Data.xlsx قرار گرفته است که حاوی ۵۰ جمله برای هر زبان است. از این داده، برای هر زبان، ٪۸۰ جملات اول (۴۰ جمله اول) را برای آموزش و مابقی ٪۲۰ (۱۰ جمله آخر) را برای آزمون جدا کنید.

الف) برای تشخیص این سه زبان، برای هر جمله تعداد چهار نویسه (کاراکتر) «ژ، پ، ل، ع» را به تعداد کل نویسه‌های آن جمله تقسیم کنید و بر اساس مقدار آن در مورد نوع زبان تصمیم بگیرید. برای این منظور، این بردار چهار بعدی را برای همه جملات آموزشی در هر زبان حساب کنید و میانگین این بردارهای آموزشی هر زبان به عنوان نماینده آن زبان استفاده کنید. بدین صورت که برای هر داده آزمون، هر زبانی که میانگین بردارهای نویسه آن بیشترین شباهت را با بردار جمله آزمون بر اساس کسینوسی به صورت زیر است که در آن d1 وd2 دو بردار مورد مقایسه هستند. صورت کسر، ضرب داخلی دو بردار و مقادیر مخرج اندازه‌های دو بردار است.

$$\cos(\theta) = \frac{\mathbf{d}_1 . \mathbf{d}_2}{|\mathbf{d}_1| \, |\mathbf{d}_2|}$$

داده‌های مجموعه آزمون را برای ارزیابی روش خود به سیستم ارائه دهید و نتایج حاصل را با معیارهای صحت (Accuracy)، دقت (Precision)، یادآوری (Recall) و F-Measure گزارش کنید.

برای این قسمت قطعه کد 1.py به ضمیمه پیوست می گردد که شامل پیاده سازی این کل قسمت الف سوال سوم است .

این کد با پایتون 3.6 پیاده سازی شده و روی تمامی نسخه ها به درستی کار می کند. تنها ملاحظه موجود نصب بودن دو کتابخانه Numpy و Pandas و Matplotlib می باشد .

این قطعه کد شامل قسمت های مختلفی است که به ذیل توضیح داده می‌شود

```
def CosinosSimilarity(Vec1,Vec2) :
    if len(Vec1)==len(Vec2) :
        d1d2 = sum(i[0] * i[1] for i in zip(Vec1, Vec2))
        ld1l = sum(i**2 for i in Vec1)**(.5)
        ld2l = sum(i**2 for i in Vec2)**(.5)
        if ld1l*ld2l == 0 : return 99999
        return d1d2/ld1l*ld2l
```

یک متد که دو وکتور دریافت و شباهت کسینوسی آنان را در صورت هم سایز بودن بر می گرداند.

```
ds = pd.read_excel("ST-HW1-Data.xlsx",sheet_name="Sheet1")

ds.columns = ['sentence','lang']
ds['zhe'] = ds['sentence'].str.count('ۀ')/ds['sentence'].str.len()
ds['pe'] = ds['sentence'].str.count('ٱ')/ds['sentence'].str.len()
ds['ein'] = ds['sentence'].str.count('ۼ')/ds['sentence'].str.len()
ds['lam'] = ds['sentence'].str.count('ـ')/ds['sentence'].str.len()
ds['tool'] = ds['sentence'].str.len()
```

خواندن داده اکسل و محاسبه میزان فراوانی 4 حرف خواسته شده برای هر سطر .

```
#define Fractions
fractions = np.array([0.8, 0.2])
```

در اینجا یک آرایه با نوع Numpy Array ساخته می شود که در حقیقت نحوه تقسیم هر کلاس داخل دیتاست را به دو دسته آموزش و آزمون مشخص میکند . در اینجا 80 درصد ابتدایی (40 جمله اول هر سه زبان) به عنوان داده آموزش و 20 درصد انتهایی (10 جمله آخر هر سه زبان) به عنوان داده آزمون لحاظ گردیده است .

```
# Arabic Train and Test
df = ds[ds['lang']==arabic]
# split into 2 parts
trainArabic,  testArabic = np.array_split(
    df, (fractions[:-1].cumsum() * len(df)).astype(int))
```

ابتدا با فیلتر کردن دیتافریم pandas سطرهای زبان عربی جدا شده و سپس با توجه به Fraction داده شده داده های تست و آموزش برای زبان عربی استخراج می گردد. روند برای زبان کردی و فارسی نیز به همین شکل است

```
# merge train of all lang and test of them together
train = pd.concat([trainArabic,trainFarsi, trainKurdi], ignore_index=True)
test = pd.concat([testArabic,testFarsi, testKurdi], ignore_index=True)
```

داده های آموزش هر سه زبان را ادغام و یک دیتافریم آموزش می سازد . خط دوم همین عمل را برای دیتافریم آزمون انجام می دهد.

```
# Calculate Mean Vector for each language
FarsiMeanVector = (trainFarsi.drop('sentence', axis=1).drop('lang', axis=1).drop('tool', axis=1).apply(lambda x: x.mean()))
KurdiMeanVector = (trainKurdi.drop('sentence', axis=1).drop('lang', axis=1).drop('tool', axis=1).apply(lambda x: x.mean()))
ArabicMeanVector = (trainArabic.drop('sentence', axis=1).drop('lang', axis=1).drop('tool', axis=1).apply(lambda x: x.mean()))
```

بدون در نظر گرفتن ستون های طول ، زبان و جمله ، میانگین فراوانی هر 4 حرف خواسته شده را برای هر سه زبان از دیتاست آموزش محاسبه و در وکتور مخصوص آن زبان میریزد .

```
predict = { 'FarsiTrue' : 0 , 'FarsiFalse': 0 ,'KurdiTrue' : 0 , 'KurdiFalse': 0 ,'ArabicTrue' : 0 , 'ArabicFalse': 0 , 'Unpredicted' : 0}
pred = []
```

المان predict از نوع dictionary برای ذخیره میزان تشخیص درست و خطای هر زبان ساخته می گردد. طبیعی است که کلید unpredicted برای سطر هایی است که الگوریتم نمی تواند در مورد زبان آنان پیش بینی کند.

```
for index, row in test.drop('sentence', axis=1).drop('lang', axis=1).drop('tool', axis=1).iterrows():

    SimilarityToFarsi = CosinosSimilarity(list(row),FarsiMeanVector)
    SimilarityToKurdi = CosinosSimilarity(list(row), KurdiMeanVector)
    SimilarityToArabic = CosinosSimilarity(list(row), ArabicMeanVector)

    if SimilarityToFarsi > SimilarityToKurdi and SimilarityToFarsi > SimilarityToArabic :
        pred.append('Farsi')
        if test.iloc[index]['lang'] == farsi :
            predict['FarsiTrue'] += 1
        else:
            predict['FarsiFalse'] += 1
```

به ازای تمام سطر های در دیتاست آزمون و با استفاده از متد فوق فاصله کسینوسی چهار ستون مورد نظر را محاسبه می کند

چنانچه میزان شباهت کسینوسی سطر مورد نظر به وکتور فارسی از دو زبان دیگر بیشتر بود تشخیص الگوریتم برای جمله زبان فارسی خواهد بود پس :

- اگر واقعا زبان سطر مورد نظر فارسی باشد FarsiTrue یکی اضافه می شود.
- اگر واقعا زبان سطر مورد نظر فارسی نباشد FarsiFalse یکی اضافه می شود.

اعمال برای دو زبان دیگر نیز به همین صورت خواهد بود و اگر شباهت سطر به هیچ وکتوری نسبت به دو وکتور دیگر برتری نداشت آنگاه Unpredicted یکی اضافه خواهد شد .

```
predict['Accuracy'] =  (predict['FarsiTrue'] + predict['KurdiTrue']+predict['ArabicTrue']) /\
                       (predict['FarsiTrue'] + predict['KurdiTrue']+predict['ArabicTrue']+predict['FarsiFalse']
                            + predict['KurdiFalse']+predict['ArabicFalse']+predict['Unpredicted'])


epsilon = .0000000000000001
predict['Precision'] = ( predict['FarsiTrue'] /(predict['FarsiTrue']  +predict['FarsiFalse']  +epsilon )+
                         predict['KurdiTrue'] /(predict['KurdiTrue']  +predict['KurdiFalse']  +epsilon )+
                         predict['ArabicTrue']/(predict['ArabicTrue'] +predict['ArabicFalse'] +epsilon )  ) / 3

predict['recall'] = ( predict['FarsiTrue'] / 10 +
                         predict['KurdiTrue'] /10 +
                         predict['ArabicTrue']/ 10
                       ) / 3

predict['Fmeasure']=2*(predict['Precision']*predict['recall'] )/(predict['Precision']+predict['recall'] )
```

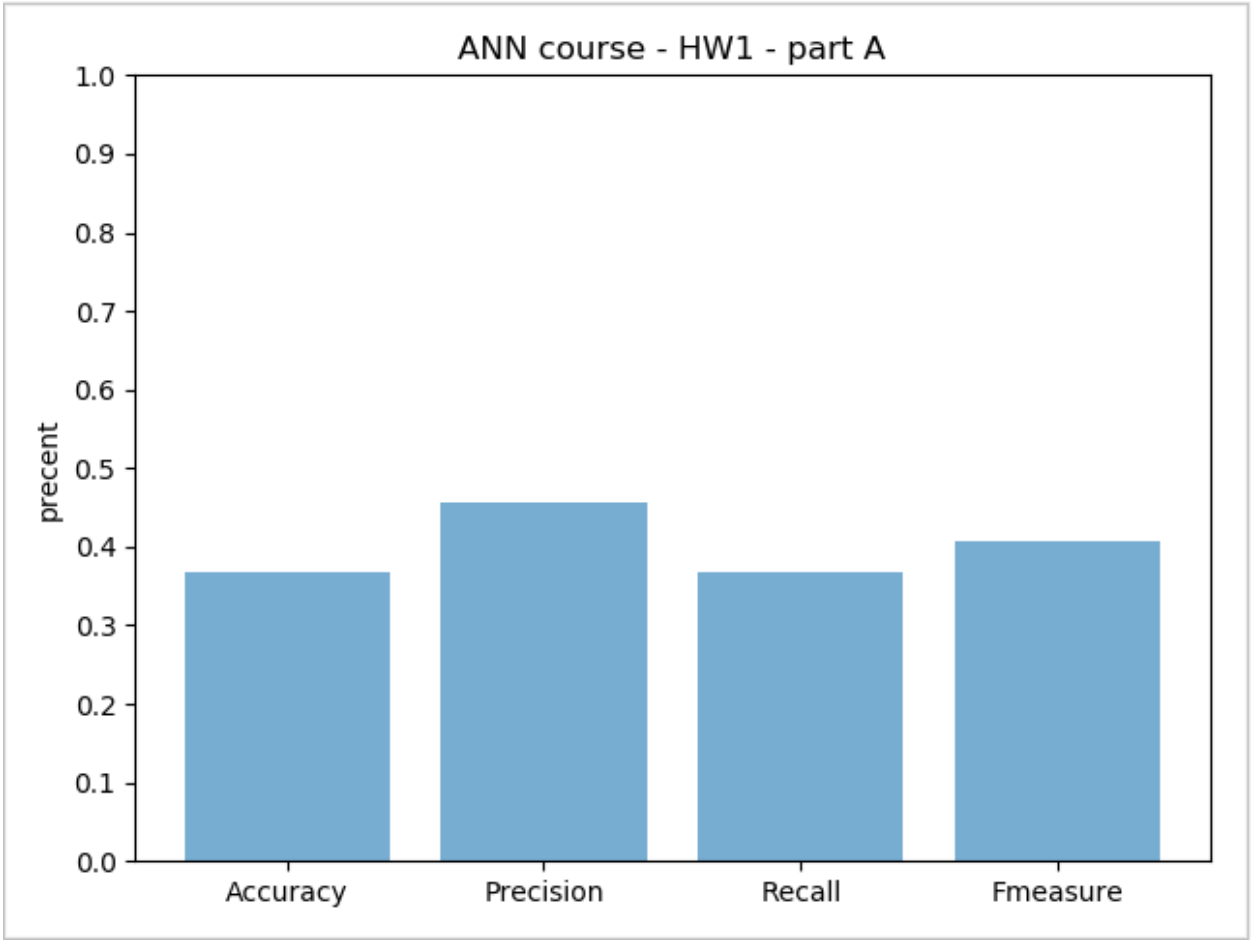
محاسبه 4 معیار خواسته شده در سوال برای نتایج پیش بینی الگوریتم . دقت شود که در محاسبه Precision مخرج کسر با یک مقدار اپسیلون جمع شده است تا چنانچه الگوریتم هیچ پیش بینی ای برای یک زبان نداشته باشد مخرج صفر نگردد و برنامه با خطا مواجه نشود

```
with pd.ExcelWriter('1.xlsx') as writer:
    ds.to_excel(writer, 'Dataset' )
    train.to_excel(writer, 'Trainset')
    test.to_excel(writer, 'Testset')
    test['Predition'] = pred
    test.to_excel(writer, 'Testresult')
    pd.DataFrame.from_dict([predict]).to_excel(writer, 'Measures')
    writer.save()
print(predict)
```

پس از محاسبه معیارهای سنجش الگوریتم خروجی تمامی مراحل به تفکیک در Sheet های مختلف فایل 1.xlsx در مسیر جاری جهت بررسی های بیشتر ذخیره می گردد.همینطور در خروجی چاپ می شود.

```
import matplotlib.pyplot as plt;
plt.rcdefaults()
import matplotlib.pyplot as plt

objects = ('Accuracy', 'Precision', 'Recall', 'Fmeasure')
print(len(objects))
y_pos = np.arange(len(objects))
performance = [predict['Accuracy'], predict['Precision'], predict['recall'], predict['Fmeasure']]

plt.bar(y_pos, performance, align='center', alpha=0.6)
plt.xticks(y_pos, objects)
plt.yticks(np.arange(0, 1.1, step=0.1))
plt.ylabel('precent')
plt.title('ANN course - HW1 - part A')

plt.show()
```

در نهایت این قطعه کد چهار معیار سنجش را به صورت نمودار میله ای برمیگرداند .



همانطور که ملاحظه می نمایید معیارهای سنجش الگوریتم حکایت از ضعیف بودن الگوریتم مذکور بوده است .

{ 'FarsiTrue': 0 , 'FarsiFalse': 0 , 'KurdiTrue': 1 , 'KurdiFalse': 0 , 'ArabicTrue': 10 , 'ArabicFalse': 17,

'Unpredicted': 2, 'Accuracy': 0.36666666666666664, 'Precision': 0.4567901234567901

, 'recall': 0.3666666666666667 , 'Fmeasure': 0.4067966016991504 }

خروجی برنامه حاکی از ان است که هیچ یک از سطور به عنوان فارسی توسط الگوریتم حدس زده نشده است و برای زبان کردی تنها یک سطر حدس زده شده که درست نیز بوده است اما برای عربی 27 سطر حدس زده شده که تنها 10 سطر آن درست می باشد . دو سطر نیز پیش بینی ای برای آن ها انجام نشده است . با مراجعه به فایل 1.xlsx و شیت TestResult در می یابم که دو سطر مذکور فاقد هر 4 حرف بوده اند .

	sentence	lang	zhe	pe	ein	lam	tool	Predition
10	تاجیکستان دارای رودهای بسیاری است	فارسی	0	0	0	0	33	Unpredicted
11	حکومت سامانی نیز از تاجیکستان برخاسته‌است	فارسی	0	0	0	0	41	Unpredicted

ب) می‌خواهیم کارایی سیستم را مقداری بهبود دهیم و یک بردار ویژگی بزرگ‌تر استفاده کنیم. به این منظور، از کل تعداد نویسه‌های متن به عنوان ویژگی استفاده می‌کنیم. به منظور طراحی این دسته‌بند ابتدا لازم است برداری با نام Character Frequency (CF) را معرفی کنیم. تعداد عناصر (مولفه‌های) این بردار برابر تعداد کل نویسه‌های موجود در تمامی سه زبان است. آنگاه برای هر جمله، عناصر این بردار برابر با فراوانی نرمال شده تعداد نویسه‌های آن متن استفاده می‌کنیم. به عنوان مثال فرض کنید نویسه‌های (A,B,C,D,E) تمامی نویسه‌های مجاز در سه زبان باشند. در این صورت، بردار CF متناظر با متن فرضی "ABBDCDEDEABB" به صورت (2, 4, 1, 3, 2) خواهد بود. حال با استفاده از بردار CF، بردار دیگری با نام Normal CF (NCF) با استفاده از فرمول $NCF(i) = CF(i)/\mathrm{Sum}(CF)$ تعریف می‌شود. بعد از نرمال کردن، بردار این جمله به صورت (0.17, 0.33, 0.08, 0.25, 0.17) خواهد بود (تقسیم مولفه‌ها بر ۱۲).

با داشتن این بردار برای هر جمله، میانگین همه بردارهای داده آموزش را برای هر زبان محاسبه کنید و از آن به عنوان معیار مقایسه داده‌های آزمون استفاده کنید. بدین صورت که برای هر داده آزمون، هر زبانی که میانگین بردارهای NCF آموزش آن، کم‌ترین فاصله کسینوسی را با NCF جمله آزمون داشته باشد، به عنوان زبان آن جمله تشخیص داده می‌شود.

معیارهای صحت (Accuracy)، دقت (Precision)، یادآوری (Recall) و F-Measure را در این حالت نیز محاسبه کنید و با نتایج بخش الف مقایسه کنید و مشاهده و تحلیل خود را گزارش کنید.

برای این قسمت قطعه کد 2.py به ضمیمه پیوست می گردد که شامل پیاده سازی این کل قسمت ب سوال سوم است .

این کد با پایتون 3.6 پیاده سازی شده و روی تمامی نسخه ها به درستی کار می کند. تنها ملاحظه موجود نصب بودن دو کتابخانه Numpy و Pandas و Matplotlib می باشد .

این قطعه کد شامل قسمت های مختلفی است که به ذیل توضیح داده می‌شود

```
def CosinosSimilarity(Vec1,Vec2) :
    if len(Vec1)==len(Vec2) :
        d1d2 = sum(i[0] * i[1] for i in zip(Vec1, Vec2))
        ld1l = sum(i**2 for i in Vec1)**(.5)
        ld2l = sum(i**2 for i in Vec2)**(.5)
        if ld1l*ld2l == 0 : return 99999
        return d1d2/ld1l*ld2l
```

یک متد که دو وکتور دریافت و شباهت کسینوسی آنان را در صورت هم سایز بودن بر می گرداند.

```
ds = pd.read_excel("ST-HW1-Data.xlsx",sheet_name="Sheet1")
ds.columns = ['sentence','lang']
ds['sentence'] = ds['sentence'].str.strip()
CF = ds['sentence'].str.cat().replace(u'\u200c','').replace(u'\xa0','').replace(u"'",'')
CF = list(set(CF))

for i,char in enumerate(CF):
    ds['char '+str(i)] =  ds['sentence'].str.count(char)/ds['sentence'].str.len()
ds['tool'] = ds['sentence'].str.len()
```

خواندن داده اکسل و حذف حروف فاصله و enter و یونیکدهای بلا معنی از تمامی سطور . پس از آن محاسبه Character Frequency به ازای تمامی سطور . دستور ds['sentence'] تنها ستون جملات را جدا کرده و داخل یه وکتور از نوع Pandas DF میریزد . متد str آن را تبدیل به رشته و متد cat تمامی رشته های را یکی می کند . دستور set رشته را تبدیل به مجموعه ای از کارکتر میکند که تمامی اعضا منحصر بفرد هستند و دستور list ان مجموعه را به لیست پایتون تبدیل میکند حال در یک iteration تمامی اعضای لیست اندیس دریافت میکنند و میزان فراوانی آن های در هرجمله در ستون Char i ثبت می گردد .

```
#define Fractions
fractions = np.array([0.8, 0.2])
```

در اینجا یک آرایه با نوع Numpy Array ساخته می شود که در حقیقت نحوه تقسیم هر کلاس داخل دیتاست را به دو دسته آموزش و آزمون مشخص میکند . در اینجا 80 درصد ابتدایی (40 جمله اول هر سه زبان) به عنوان داده آموزش و 20 درصد انتهایی (10 جمله آخر هر سه زبان) به عنوان داده آزمون لحاظ گردیده است .

```
# Arabic Train and Test
df = ds[ds['lang']==arabic]
# split into 2 parts
trainArabic,  testArabic = np.array_split(
    df, (fractions[:-1].cumsum() * len(df)).astype(int))
```

ابتدا با فیلتر کردن دیتافریم pandas سطرهای زبان عربی جدا شده و سپس با توجه به Fraction داده شده داده های تست و آموزش برای زبان عربی استخراج می گردد. روند برای زبان کردی و فارسی نیز به همین شکل است

```
# merge train of all lang and test of them together
train = pd.concat([trainArabic,trainFarsi, trainKurdi], ignore_index=True)
test = pd.concat([testArabic,testFarsi, testKurdi], ignore_index=True)
```

داده های آموزش هر سه زبان را ادغام و یک دیتافریم آموزش می سازد . خط دوم همین عمل را برای دیتافریم آزمون انجام می دهد.

```
# Calculate Mean Vector for each language
FarsiMeanVector = (trainFarsi.drop('sentence', axis=1).drop('lang', axis=1).drop('tool', axis=1).apply(lambda x: x.mean()))
KurdiMeanVector = (trainKurdi.drop('sentence', axis=1).drop('lang', axis=1).drop('tool', axis=1).apply(lambda x: x.mean()))
ArabicMeanVector = (trainArabic.drop('sentence', axis=1).drop('lang', axis=1).drop('tool', axis=1).apply(lambda x: x.mean()))
```

بدون در نظر گرفتن ستون های طول ، زبان و جمله ، میانگین فراوانی هر 4 حرف خواسته شده را برای هر سه زبان از دیتاست آموزش محاسبه و در وکتور مخصوص آن زبان میریزد .

```
predict = {'FarsiTrue' : 0 , 'FarsiFalse': 0 ,'KurdiTrue' : 0 , 'KurdiFalse': 0 ,'ArabicTrue' : 0 , 'ArabicFalse': 0 , 'Unpredicted' : 0}
pred = []
```

المان predict از نوع dictionary برای ذخیره میزان تشخیص درست و خطای هر زبان ساخته می گردد. طبیعی است که کلید unpredicted برای سطر هایی است که الگوریتم نمی تواند در مورد زبان آنان پیش بینی کند.

```
for index, row in test.drop('sentence', axis=1).drop('lang', axis=1).drop('tool', axis=1).iterrows():

    SimilarityToFarsi = CosinosSimilarity(list(row),FarsiMeanVector)
    SimilarityToKurdi = CosinosSimilarity(list(row), KurdiMeanVector)
    SimilarityToArabic = CosinosSimilarity(list(row), ArabicMeanVector)

    if SimilarityToFarsi > SimilarityToKurdi and SimilarityToFarsi > SimilarityToArabic :
        pred.append('Farsi')
        if test.iloc[index]['lang'] == farsi :
            predict['FarsiTrue'] += 1
        else:
            predict['FarsiFalse'] += 1
```

به ازای تمام سطر های در دیتاست آزمون و با استفاده از متد فوق فاصله کسینوسی چهار ستون مورد نظر را محاسبه می کند

چنانچه میزان شباهت کسینوسی سطر مورد نظر به وکتور فارسی از دو زبان دیگر بیشتر بود تشخیص الگوریتم برای جمله زبان فارسی خواهد بود پس :

- اگر واقعا زبان سطر مورد نظر فارسی باشد FarsiTrue یکی اضافه می شود.
- اگر واقعا زبان سطر مورد نظر فارسی نباشد FarsiFalse یکی اضافه می شود.

اعمال برای دو زبان دیگر نیز به همین صورت خواهد بود و اگر شباهت سطر به هیچ وکتوری نسبت به دو وکتور دیگر برتری نداشت آنگاه Unpredicted یکی اضافه خواهد شد .

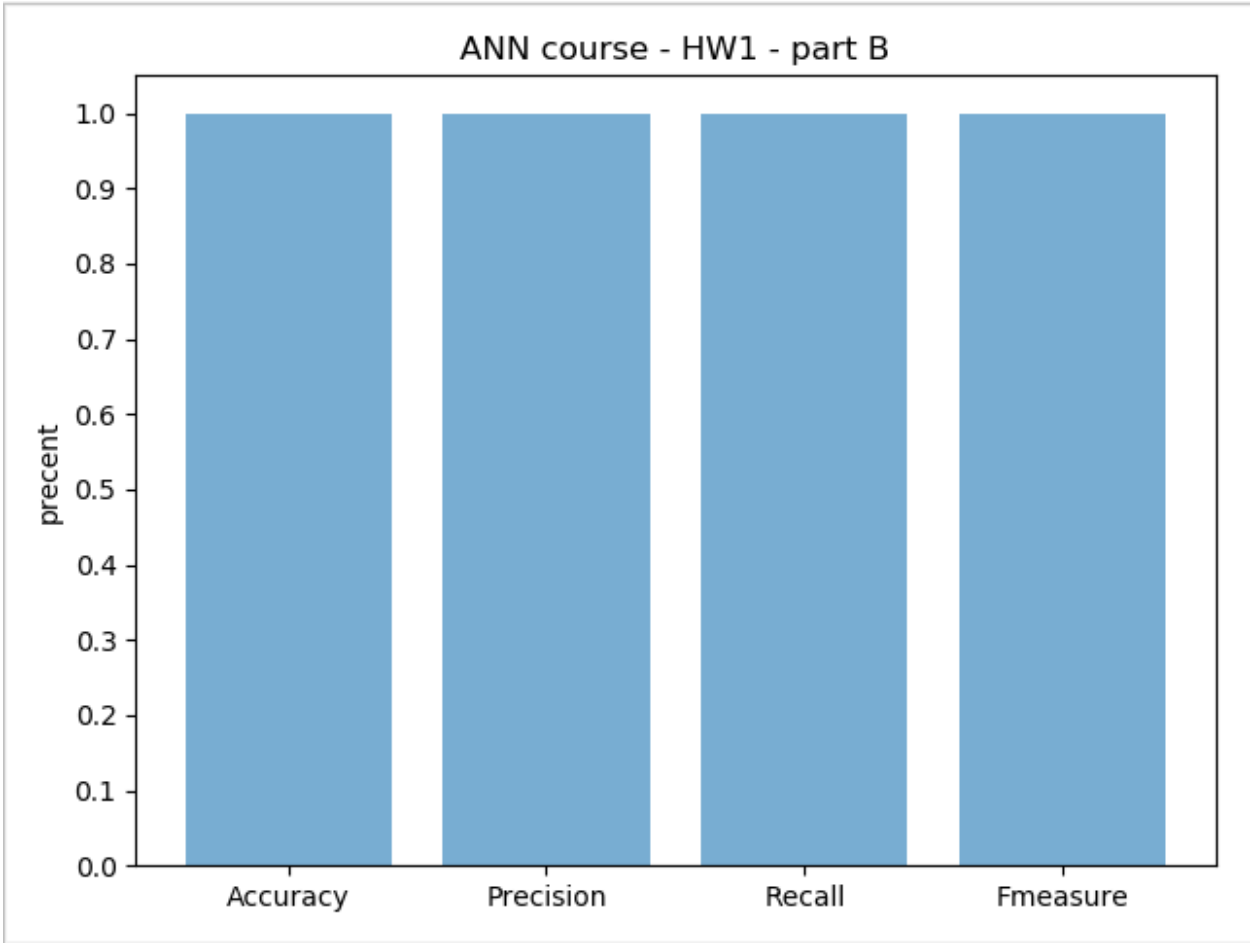
```
predict['Accuracy'] =  (predict['FarsiTrue'] + predict['KurdiTrue']+predict['ArabicTrue']) /\
                       (predict['FarsiTrue'] + predict['KurdiTrue']+predict['ArabicTrue']+predict['FarsiFalse']
                            + predict['KurdiFalse']+predict['ArabicFalse']+predict['Unpredicted'])


epsilon = .0000000000000001
predict['Precision'] = ( predict['FarsiTrue'] /(predict['FarsiTrue']  +predict['FarsiFalse']  +epsilon )+
                         predict['KurdiTrue'] /(predict['KurdiTrue']  +predict['KurdiFalse']  +epsilon )+
                         predict['ArabicTrue']/(predict['ArabicTrue'] +predict['ArabicFalse'] +epsilon )  ) / 3

predict['recall'] = ( predict['FarsiTrue'] / 10 +
                         predict['KurdiTrue'] /10 +
                         predict['ArabicTrue']/ 10
                       ) / 3

predict['Fmeasure']=2*(predict['Precision']*predict['recall'] )/(predict['Precision']+predict['recall'] )
```

محاسبه 4 معیار خواسته شده در سوال برای نتایج پیش بینی الگوریتم . دقت شود که در محاسبه Precision مخرج کسر با یک مقدار اپسیلون جمع شده است تا چنانچه الگوریتم هیچ پیش بینی ای برای یک زبان نداشته باشد مخرج صفر نگردد و برنامه با خطا مواجه نشود

```
with pd.ExcelWriter('2.xlsx') as writer:
    ds.to_excel(writer, 'Dataset' )
    train.to_excel(writer, 'Trainset')
    test.to_excel(writer, 'Testset')
    test['Predition'] = pred
    test.to_excel(writer, 'Testresult')
    pd.DataFrame.from_dict([predict]).to_excel(writer, 'Measures')
    writer.save()
```

پس از محاسبه معیارهای سنجش الگوریتم خروجی تمامی مراحل به تفکیک در Sheet های مختلف فایل 2.xlsx در مسیر جاری جهت بررسی های بیشتر ذخیره می گردد.همینطور در خروجی چاپ می شود.

```
import matplotlib.pyplot as plt;
plt.rcdefaults()
import matplotlib.pyplot as plt

objects = ('Accuracy', 'Precision', 'Recall', 'Fmeasure')
print(len(objects))
y_pos = np.arange(len(objects))
performance = [predict['Accuracy'], predict['Precision'], predict['recall'], predict['Fmeasure']]

plt.bar(y_pos, performance, align='center', alpha=0.6)
plt.xticks(y_pos, objects)
plt.yticks(np.arange(0, 1.1, step=0.1))
plt.ylabel('precent')
plt.title('ANN course - HW1 - part A')

plt.show()
```



در نهایت این قطعه کد چهار معیار سنجش را به صورت نمودار میله ای برمیگرداند .

FarsiTrue': 10, 'FarsiFalse': 0, 'KurdiTrue': 10, 'KurdiFalse': 0, 'ArabicTrue': 10, 'ArabicFalse': 0, '}

'Unpredicted': 0, 'Accuracy': 1.0, 'Precision': 1.0, 'recall': 1.0, 'Fmeasure': 1.0}

هم نمودار و هم خروجی حاکی از دقت بالای الگوریتم دارد و کارایی الگوریتم در اینجا صد در صد ارزیابی شده است .

ج) نتایج بخش ب را با ارزیابی مبتنی بر روش 5-fold Cross-Validation گزارش کنید. برای این کار کل مجموعه داده را استفاده کنید (و نه فقط مجموعه آموزش).

برای این قسمت قطعه کد 3.py به ضمیمه پیوست می گردد که شامل پیاده سازی این کل قسمت ج سوال سوم است .

این کد با پایتون 3.6 پیاده سازی شده و روی تمامی نسخه ها به درستی کار می کند. تنها ملاحظه موجود نصب بودن دو کتابخانه Numpy و Pandas و Matplotlib می باشد .

این قطعه کد شامل قسمت های مختلفی است که به ذیل توضیح داده می‌شود

```
def CosinosSimilarity(Vec1,Vec2) :
    if len(Vec1)==len(Vec2) :
        d1d2 = sum(i[0] * i[1] for i in zip(Vec1, Vec2))
        ld1l = sum(i**2 for i in Vec1)**(.5)
        ld2l = sum(i**2 for i in Vec2)**(.5)
        if ld1l*ld2l == 0 : return 99999
        return d1d2/ld1l*ld2l
```

یک متد که دو وکتور دریافت و شباهت کسینوسی آنان را در صورت هم سایز بودن بر می گرداند.

```
ds = pd.read_excel("ST-HW1-Data.xlsx",sheet_name="Sheet1")
ds.columns = ['sentence','lang']
ds['sentence'] = ds['sentence'].str.strip()
CF = ds['sentence'].str.cat().replace(u'\u200c','').replace(u'\xa0','').replace(u"'",'')
CF = list(set(CF))

for i,char in enumerate(CF):
    ds['char '+str(i)] =  ds['sentence'].str.count(char)/ds['sentence'].str.len()
ds['tool'] = ds['sentence'].str.len()
```

خواندن داده اکسل و حذف حروف فاصله و enter و یونیکدهای بلا معنی از تمامی سطور . پس از آن محاسبه Character Frequency به ازای تمامی سطور . دستور ds['sentence'] تنها ستون جملات را جدا کرده و داخل یه وکتور از نوع Pandas DF میریزد . متد str آن را تبدیل به رشته و متد cat تمامی رشته های را یکی می کند . دستور set رشته را تبدیل به مجموعه ای از کارکتر میکند که تمامی اعضا منحصر بفرد هستند و دستور list ان مجموعه را به لیست پایتون تبدیل میکند حال در یک iteration تمامی اعضای لیست اندیس دریافت میکنند و میزان فراوانی آن های در هرجمله در ستون Char i ثبت می گردد .

```
#define Fractions
fractionsList = [[0.8, 0.2 , 0],[0.6, 0.2 , 0.2],[0.4, 0.2 , 0.4],[0.2, 0.2 , 0.6],[0 , 0.2 , 0.8]]
measures = {'Accuracy' : 0 , 'Precision' : 0 , 'Recall' : 0 , 'Fmeasure' : 0}
for i in range(5):
    fractions = np.array(fractionsList[i])
```

مهم ترین تفاوت کد 2 و 3 این قسمت است که در آن لیستی از تقسیم بندی ها برای هربار اجرا روی یک تقسیم بندی در نظر گرفته می شود. در اینجا دیتاست 3 تکه می شود عدد اول نشان دهنده قسمت اول داده آموزش و عدد دوم نشان دهنده داده ازمون و عدد سوم نشان دهنده قسمت دوم داده اموزش خواهد بود یعنی :

- در دور اول [0, 0.2, 0.8] به این معناست که 80 درصد اول جملات هر زبان و 0 درصد اخر جملات هر زبان به عنوان داده آموزش و 20 درصد میانی به عنوان داده آزمون در نظر گرفته می شود
- در دور اول [0.2, 0.2, 0.6] به این معناست که 60 درصد اول جملات هر زبان و 20 درصد اخر جملات هر زبان به عنوان داده آموزش و 20 درصد میانی به عنوان داده آزمون در نظر گرفته می شود
- در دور اول [0.4, 0.2, 0.4] به این معناست که 40 درصد اول جملات هر زبان و 40 درصد اخر جملات هر زبان به عنوان داده آموزش و 20 درصد میانی به عنوان داده آزمون در نظر گرفته می شود
- در دور اول [0.6, 0.2, 0.2] به این معناست که 20 درصد اول جملات هر زبان و 60 درصد اخر جملات هر زبان به عنوان داده آموزش و 20 درصد میانی به عنوان داده آزمون در نظر گرفته می شود
- در دور اول [0.8, 0.2, 0] به این معناست که 4 درصد اول جملات هر زبان و 80 درصد اخر جملات هر زبان به عنوان داده آموزش و 20 درصد میانی به عنوان داده آزمون در نظر گرفته می شود

هر بار تمام مراحل الگوریتم بخش قبل بر روی هر Fraction اجرا می شود.

```
# Arabic Train and Test
df = ds[ds['lang']==arabic]
# split into 2 parts
trainArabic,  testArabic = np.array_split(
    df, (fractions[:-1].cumsum() * len(df)).astype(int))
```

ابتدا با فیلتر کردن دیتافریم pandas سطرهای زبان عربی جدا شده و سپس با توجه به Fraction داده شده داده های تست و آموزش برای زبان عربی استخراج می گردد. روند برای زبان کردی و فارسی نیز به همین شکل است

```
# merge train of all lang and test of them together
train = pd.concat([trainArabic,trainFarsi, trainKurdi], ignore_index=True)
test = pd.concat([testArabic,testFarsi, testKurdi], ignore_index=True)
```

داده های آموزش هر سه زبان را ادغام و یک دیتافریم آموزش می سازد . خط دوم همین عمل را برای دیتافریم آزمون انجام می دهد.

```
# Calculate Mean Vector for each language
FarsiMeanVector = (trainFarsi.drop('sentence', axis=1).drop('lang', axis=1).drop('tool', axis=1).apply(lambda x: x.mean()))
KurdiMeanVector = (trainKurdi.drop('sentence', axis=1).drop('lang', axis=1).drop('tool', axis=1).apply(lambda x: x.mean()))
ArabicMeanVector = (trainArabic.drop('sentence', axis=1).drop('lang', axis=1).drop('tool', axis=1).apply(lambda x: x.mean()))
```

بدون در نظر گرفتن ستون های طول ، زبان و جمله ، میانگین فراوانی هر 4 حرف خواسته شده را برای هر سه زبان از دیتاست آموزش محاسبه و در وکتور مخصوص آن زبان میریزد .

```
predict = { 'FarsiTrue' : 0 , 'FarsiFalse': 0 ,'KurdiTrue' : 0 , 'KurdiFalse': 0 ,'ArabicTrue' : 0
                                                , 'ArabicFalse': 0 , 'Unpredicted' : 0}
predict['State'] = str(i + 1) + ' : ' + str(fractions)
pred = []
```

المان predict از نوع dictionary برای ذخیره میزان تشخیص درست و خطای هر زبان ساخته می گردد. طبیعی است که کلید unpredicted برای سطر هایی است که الگوریتم نمی تواند در مورد زبان آنان پیش بینی کند. به ازای اینکه در چندمین دور اجرای CrossValidation هستیم و نحوه Fraction در این دور چگونه است کلید State پر می‌شود.

```
for index, row in test.drop('sentence', axis=1).drop('lang', axis=1).drop('tool', axis=1).iterrows():

    SimilarityToFarsi = CosinosSimilarity(list(row),FarsiMeanVector)
    SimilarityToKurdi = CosinosSimilarity(list(row), KurdiMeanVector)
    SimilarityToArabic = CosinosSimilarity(list(row), ArabicMeanVector)

    if SimilarityToFarsi > SimilarityToKurdi and SimilarityToFarsi > SimilarityToArabic :
        pred.append('Farsi')
        if test.iloc[index]['lang'] == farsi :
            predict['FarsiTrue'] += 1
        else:
            predict['FarsiFalse'] += 1
```

به ازای تمام سطر های در دیتاست آزمون و با استفاده از متد فوق فاصله کسینوسی چهار ستون مورد نظر را محاسبه می کند

چنانچه میزان شباهت کسینوسی سطر مورد نظر به وکتور فارسی از دو زبان دیگر بیشتر بود تشخیص الگوریتم برای جمله زبان فارسی خواهد بود پس :

- اگر واقعا زبان سطر مورد نظر فارسی باشد FarsiTrue یکی اضافه می شود.

- اگر واقعا زبان سطر مورد نظر فارسی نباشد FarsiFalse یکی اضافه می شود.

اعمال برای دو زبان دیگر نیز به همین صورت خواهد بود و اگر شباهت سطر به هیچ وکتوری نسبت به دو وکتور دیگر برتری نداشت آنگاه Unpredicted یکی اضافه خواهد شد .

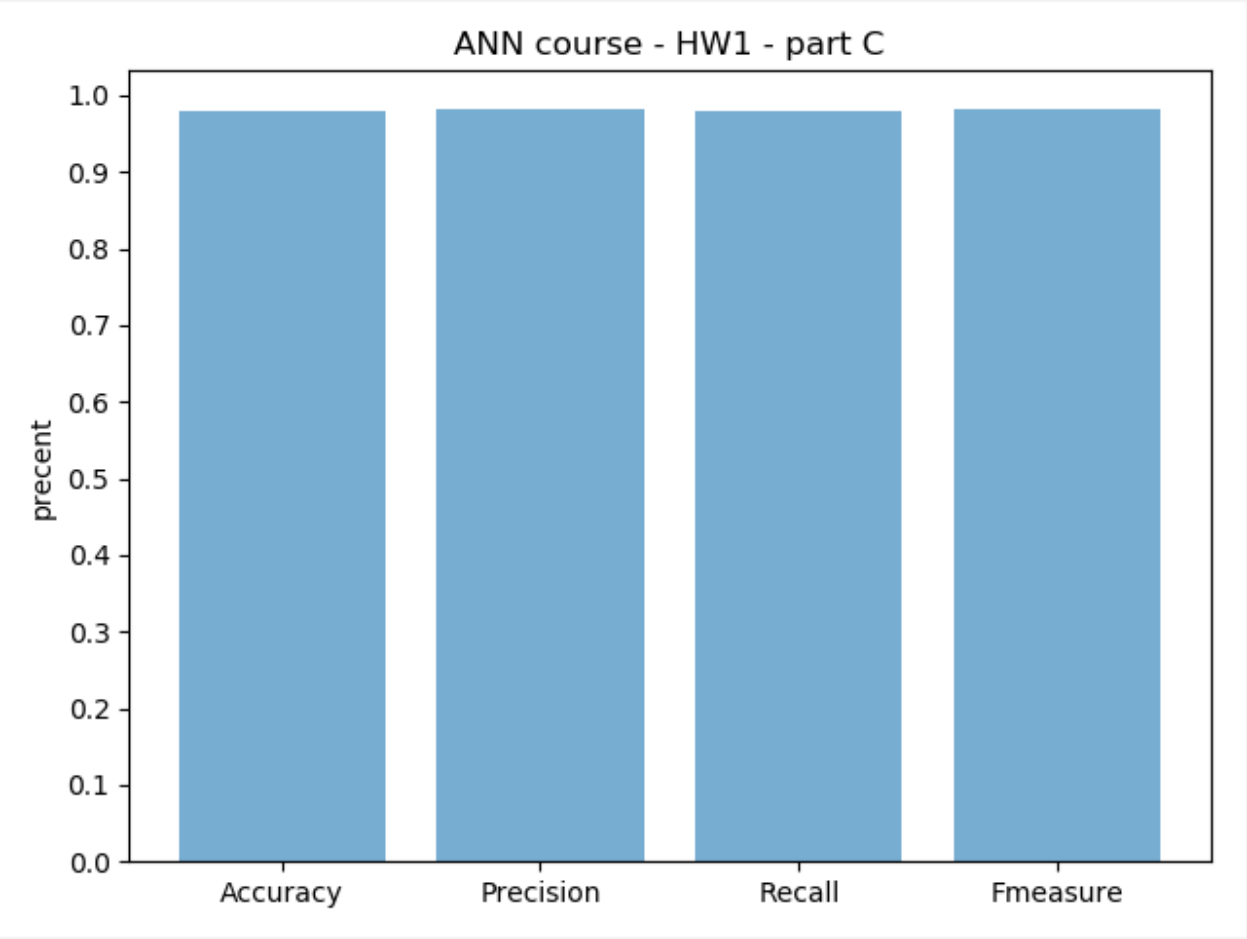
```
predict['Accuracy'] =  (predict['FarsiTrue'] + predict['KurdiTrue']+predict['ArabicTrue']) /\
                       (predict['FarsiTrue'] + predict['KurdiTrue']+predict['ArabicTrue']+predict['FarsiFalse']
                            + predict['KurdiFalse']+predict['ArabicFalse']+predict['Unpredicted'])


epsilon = .0000000000000001
predict['Precision'] = ( predict['FarsiTrue'] /(predict['FarsiTrue']  +predict['FarsiFalse']  +epsilon )+
                         predict['KurdiTrue'] /(predict['KurdiTrue']  +predict['KurdiFalse']  +epsilon )+
                         predict['ArabicTrue']/(predict['ArabicTrue'] +predict['ArabicFalse'] +epsilon )  ) / 3

predict['recall'] = ( predict['FarsiTrue'] / 10 +
                         predict['KurdiTrue'] /10 +
                         predict['ArabicTrue']/ 10
                       ) / 3

predict['Fmeasure']=2*(predict['Precision']*predict['recall'] )/(predict['Precision']+predict['recall'] )
```

محاسبه 4 معیار خواسته شده در سوال برای نتایج پیش بینی الگوریتم . دقت شود که در محاسبه Precision مخرج کسر با یک مقدار اپسیلون جمع شده است تا چنانچه الگوریتم هیچ پیش بینی ای برای یک زبان نداشته باشد مخرج صفر نگردد و برنامه با خطا مواجه نشود

```
with pd.ExcelWriter('3_'+str(i+1)+'.xlsx') as writer:
    ds.to_excel(writer, 'Dataset')
    train.to_excel(writer, 'Trainset')
    test.to_excel(writer, 'Testset')
    test['Predition'] = pred
    test.to_excel(writer, 'Testresult')
    pd.DataFrame.from_dict([predict]).to_excel(writer, 'Measures')
    writer.save()
print(fractionsList[i],predict)
measures['Accuracy'] += predict['Accuracy']
measures['Precision'] += predict['Precision']
measures['Recall'] += predict['Recall']
measures['Fmeasure'] += predict['Fmeasure']
```

پس از محاسبه معیارهای سنجش الگوریتم خروجی تمامی مراحل به تفکیک در Sheet های مختلف فایل 3_i.xlsx (بسته به دور اجرا i متفاوت است مثلا در دور اول فایل خروجی 3_1.xlsx و در دو دوم 3_2.xlsx و به همین ترتیب تا دور آخر و پنجم که فایل خروجی 3_5.xlsx میباشد) در مسیر جاری جهت بررسی های بیشتر ذخیره می گردد.همینطور معیارهای سنجش و ارزیابی در هر دور در خروجی چاپ می شود.

و در اخر معیار های سنجش جهت میانگین گیری و بدست آوردن سنجش کلی الگوریتم در حالت 5 fold Cross Validation داخل دیکشنری measures انباشت می گردد.



در نهایت این قطعه کد چهار معیار سنجش را به صورت نمودار میله ای برمیگرداند .

over all accuracy : 0.9800000000000001 '

over all Precision : 0.9828282828282828

over all Recall : 0.9800000000000001

over all Fmeasure : 0.9814070925663557

هم نمودار و هم خروجی حاکی از دقت بالای الگوریتم دارد و کارایی الگوریتم در اینجا نزدیک به صد در صد ارزیابی شده است .

با تشکر و سپاس فراوان