



درس : یادگیری ماشین

دانشجو : امیرمحمد خرازی

شماره دانشجویی : ۴۰۱۵۲۵۲۱۰۰۲

استاد درس : دکتر منصور رزقی آهق

دانشکده علوم ریاضی ، گروه علوم کامپیوتر، گرایش داده‌کاوی

تمرین سری اول

## لینک‌های مفید

منبع بیشتر از scholar google و wikipedia در یافت شده اند. علاوه بر این موارد می‌توانید اگر بنابر هر دلیلی فایل‌های ارائه شده، قابل به استفاده نبود، آن‌ها را از لینک زیر دانلود کنید. در نظر داشته باشید که در بین داده‌ها چون داده‌های pavia حجم هستند، این داده را باید دستی دانلود کنید. این موارد برای زمانی است که فایل‌های اصلی که در سایت تمرین آپلود شده‌اند، اجرا نشوند یا با مشکل دانلود شوند.

لینک : google drive

لینک گیت‌هاب GitHub

## تمرین اول

### پیش‌گفتار

برای پاسخ این تمرین، با توجه به محدودیت‌های اینترنتی و عدم دسترسی به تمامی محتوای اینترنت، اغلب جواب‌ها از سایت ویکیپدیا جمع‌آوری شده‌اند. این موضوع که سایت ویکیپدیا<sup>۱</sup> لزوماً اطلاعات کامل و درستی را در دسترس قرار نمی‌دهد را قبول دارم ولی با توجه به شرایط و بررسی اطلاعات بدست آمده، نظر بر این شد که مرجع این پاسخ‌ها را، سایت ویکیپدیا، قرار دهم.

همچنین در خصوص سوال که از دانشجو درخواست شده بود علاوه بر مفهوم، شباهت‌ها و عدم شباهت‌ها، اشتراکات و تفاوت‌ها و غیره را با مثال بررسی کند، در هر بخش اگر شباهت خاصی با بخش‌های دیگر مشاهده شود، به آن ذکر شده و اگر توانسته باشم مثالی در این خصوص پیدا کنم، آن نیز در همان بخش ذکر شده است. منظور از این بند این است که بخش جدایی برای این بررسی‌ها در نظر گرفته نشده است و در همان جایی که مفهوم این عبارات شرح داده می‌شود، این موضوعات نیز بررسی می‌گردند.

### علم داده<sup>۲</sup>

ابتدا در سال ۱۹۶۲ آقای جان توکی<sup>۳</sup> موضوع آنالیز داده را معرفی کرد که بسیار به علم داده امروزی شبیه بود. در سال ۱۹۸۵ در سمیناری که در پکن برگزار شد، فردی به نام جف وو<sup>۴</sup> برای اولین بار از واژه علم داده به عنوان جایگزینی برای آمار استفاده کرد. البته پیش از آن در سال ۱۹۷۴، آقای پیتر ناور<sup>۵</sup> این واژه را به عنوان نامی جایگزین برای علوم کامپیوتر پیشنهاد داده بود. بعداً در سال ۱۹۹۲ شرکت کنندگان در نشست آمار که در دانشگاه مونپلیه<sup>۶</sup> برگزار شد، با تلفیق یافته‌های آماری، آنالیز داده و محاسبات به ظهور یک رشته جدید که بر روی انواع داده‌های متفاوت تمرکز دارد، موافقت کردند. تا بدین جا علم داده را

C. F. Jeff Wu<sup>۴</sup>  
Peter Naur<sup>۵</sup>  
Montpellier<sup>۶</sup>

Wikipedia<sup>۱</sup>  
Data Science<sup>۲</sup>  
John Tukey<sup>۳</sup>

بیشتر به عنوان نامی جایگزین مطرح میکردند اما در سال ۱۹۹۸، هایاشی چیکو<sup>۷</sup> بر این موضوع که علم داده یک مفهوم جدید و میان رشته‌ای با سه جنبه: طراحی داده، جمع‌آوری و آنالیز است، تأکید کرد. نکته جالب در این تاریخچه این است که همزمان در دهه ۱۹۹۰، عناوین محبوبی که بر یافتن الگو در دیتاست‌ها تأکید داشتند شامل کشف دانش<sup>۸</sup> و داده‌کاوی<sup>۹</sup> بودند. اما امروزه بیشتر ما علم داده را با کمک ویلیام کلیولند<sup>۱۰</sup> می‌شناسیم. بعد از مقاله ایشان که در سال ۲۰۰۱ چاپ شد و بر گسترش آمار فراتر از تئوری به حوزه‌های فنی اشاره داشت، علم داده بیشتر و بیشتر مورد استفاده قرار گرفت.

البته با گذشت این همه مدت هنوز تعریفی از علم داده که مورد تأیید همه زمینه‌ها باشد وجود ندارد و برخی آن را یک کلید واژه<sup>۱۱</sup> در نظر می‌گیرند. کلید واژه به این معنی که در بازه زمانی خاصی محبوبیت دارند. کلان داده‌ها<sup>۱۲</sup> نیز یک اصطلاح بازاری مرتبط است. افرادی که با علم داده کار می‌کنند وظیفه دارند که این کلان داده‌ها را به اطلاعات قابل استفاده شکسته و نرم‌افزارها و الگوریتم‌هایی را طراحی کنند که به شرکت‌ها و موسسات برای انتخاب عمل بهینه کمک می‌کند.

بطور خلاصه علم داده دانشی میان رشته‌ای است که از روش‌های علمی، فرآیندها، الگوریتم‌ها و سیستم‌ها برای استخراج دانش و فهم از میان داده‌های نویزی، با ساختار یا بدون ساختار استفاده می‌کند و این دانش بدست آمده را در طیف وسیعی از کاربرد ها اعمال می‌کند. از نمونه موضوعاتی که با علم داده مرتبط هستند می‌توان به داده‌کاوی، یادگیری ماشین و کلان داده اشاره کرد.

علم داده مفهومی است که آنالیز داده، انفورماتیک، آمار و متد های آن‌ها را متحد می‌کند برای اینکه پدیده‌های واقعی داده را بفهمد و آنالیز کند و برای اینکار نیز از تکنیک‌ها و تئوری‌های متفاوتی که از حوزه‌های مختلفی چون: آمار، ریاضی، علوم کامپیوتر، علوم اطلاعات و غیره بدست آمده‌اند، استفاده می‌کند.

لذا علم داده یک دانش میان رشته‌ای است که بر روی استخراج اطلاعات از (معمولا) دیتاست‌های بزرگ و اعمال این دانش بدست آمده برای حل مسائل مختلف، تمرکز دارد. این حوزه شامل آماده کردن داده برای آنالیز، فرموله کردن مسائل علم داده، آنالیز داده، ایجاد راه حل‌های داده-محور و ارائه یافته‌ها برای تصمیم‌گیری می‌باشد. برای همین هم هست که از مهارت‌های مختلفی از علوم کامپیوتر، آمار، ریاضی، نمایش داده و غیره استفاده می‌کند.

بسیاری از آماری‌ها از جمله نیت سیلور<sup>۱</sup> بر این موضوع که علم داده یک حوزه جدید نیست، بلکه اسمی دیگر برای آمار است، تأکید داشتند. اما به هر حال این موضوع مهم است که علم داده ارتباط زیادی بر آمار دارد.

## داده‌کاوی<sup>۲</sup>

در دهه ۱۹۶۰، آماریان و اقتصادی‌ها از اصطلاحاتی چون ماهیگیری داده<sup>۳</sup> یا لایروبی داده<sup>۴</sup> برای اشاره به چیزی که آن‌ها آن را عملکرد بد تجزیه داده بدون فرض پیشین<sup>۵</sup> می‌دانستند، استفاده می‌کردند. اصطلاح داده‌کاوی در مقاله‌ای که در سال ۱۹۸۳ از اقتصاددان مایکل لاول<sup>۶</sup> چاپ شد، استفاده شده. اصطلاح داده‌کاوی در دوره ۱۹۹۰ در جامعه پایگاه داده شروع به ظهور کرد. البته برای یک دوران کوتاهی در دهه ۱۹۸۰، عنوان پایگاه داده کاوی<sup>۷</sup> استفاده می‌شد. اسامی دیگری که استفاده می‌شدند مانند باستان‌شناسی داده<sup>۸</sup>، برداشت اطلاعات<sup>۹</sup>، کشف اطلاعات<sup>۱۰</sup>، استخراج دانش<sup>۱۱</sup> و غیره بودند. نکته مهم که در پایگاه داده‌ها وجود دارد و به داده‌کاوی نیز بسیار مربوط است، کشف دانش در پایگاه داده<sup>۱۲</sup> یا کی دی دی (KDD) است که نخست توسط گریگوری پیاتتسکی-شاپیرو<sup>۱۳</sup> استفاده شد و بعدا در جوامع دیگری نیز چون هوش مصنوعی و یادگیری ماشین محبوبیت زیادی پیدا کرد. امروزه اصطلاحات داده‌کاوی و KDD به جای هم استفاده می‌شوند.

a-priori<sup>۵</sup>  
Michael Lovell<sup>۶</sup>  
Database Mining<sup>۷</sup>  
data archaeology<sup>۸</sup>  
information harvesting<sup>۹</sup>  
information discovery<sup>۱۰</sup>  
knowledge extraction<sup>۱۱</sup>  
Knowledge Discovery in Databases<sup>۱۲</sup>  
Gregory Piatetsky-Shapiro<sup>۱۳</sup>

Hayashi Chikio<sup>۷</sup>  
Knowledge Discovery<sup>۸</sup>  
Data Mining<sup>۹</sup>  
William S. Cleveland<sup>۱۰</sup>  
Buzzword<sup>۱۱</sup>  
Big Data<sup>۱۲</sup>  
Nate Silver<sup>۱</sup>  
Data Mining<sup>۲</sup>  
Data fishing<sup>۳</sup>  
Data dredging<sup>۴</sup>

بطور خلاصه، داده‌کاوی فرآیند استخراج و کشف الگوها در دیتاست‌های بزرگ با استفاده از متدهایی که در اشتراک یادگیری ماشین، آمار و سیستم‌های پایگاه داده هستند، است. داده‌کاوی یک زیر رشته میان رشته‌ای از علوم کامپیوتر و آمار است که هدف نهایی آن استخراج اطلاعات با استفاده از روش‌های هوشمند از دیتاست‌ها و تبدیل این اطلاعات به ساختاری قابل درک که در آینده قابل استفاده باشد، است. لذا در همینجا تفاوت آن با علم داده قابل مشاهده است. داده‌کاوی در حقیقت یکی از گام‌های تحلیل در KDD می‌باشد. جدای گام آنالیز، شامل جنبه‌های پایگاه داده و مدیریت داده، پیش پردازش داده، مدل، ماتریس، پیچیدگی، پس پردازش ساختارهای کشف شده، دیداری سازی و آپدیت کردن برخط می‌باشد.

واژه داده‌کاوی یک واژه غلط<sup>۱۴</sup> است چراکه هدف آن استخراج الگو و دانش از داده‌های زیاد است و استخراج از خود داده منظورمان نیست. همچنین این واژه نیز یک کلید واژه است مانند علم داده و اغلب به هر شکلی از داده‌های بزرگ مقیاس یا فرآیندهای اطلاعات (جمع‌آوری، استخراج، انبارداده، آنالیز، آمار) و همچنین کاربردهایی از سیستم‌های کامپیوتری کمک به تصمیم<sup>۱</sup> شامل هوش مصنوعی (مثل یادگیری ماشین) و هوش اقتصادی و غیره، اعمال می‌شود.

کار واقعی داده‌کاوی آنالیز اتوماتیک یا نیمه اتوماتیک یکسری داده‌های زیاد برای کشف الگوهای جالب مثل آنالیز خوشه، یافتن ناهنجاری‌ها، قوانین انجمنی و غیره است. این الگوها بعداً مورد استفاده قرار می‌گیرند اما مسئله مهم اینجاست که جمع‌آوری داده، آماده سازی داده، تفسیر نتیجه و گزارش بخشی از گام داده‌کاوی نیستند ولی متعلق به KDD می‌باشند.

تفاوت میان آنالیز داده و داده‌کاوی در اینجاست که آنالیز داده برای آزمون مدل و فرض روی دیتاست استفاده می‌شود. در مقابل داده‌کاوی از یادگیری ماشین و مدل‌های آماری برای یافتن الگوهای پنهان در داده‌های زیاد استفاده می‌کند.

استخراج الگوها از داده‌ها بصورت دستی از صدهای گذشته انجام می‌شده اما امروزه از آنجایی که دیتاست‌ها بزرگتر شده‌اند به جای آنالیز دستی، اینکار بصورت اتوماتیک با کمک کشفیاتی که در علوم کامپیوتر صورت گرفته است به ویژه یادگیری ماشین (شبکه‌های عصبی، آنالیز خوشه، الگوریتم‌های ژنتیک، درخت تصمیم، قوانین تصمیم، SVM و غیره) انجام می‌شود. لذا داده‌کاوی فرآیند استفاده از این متدها روی داده با هدف پیدا کردن الگوهای مخفی می‌باشد. در داده‌های بزرگ، داده‌کاوی، پلی میان آمار کاربردی و هوش مصنوعی (که معمولاً پایه ریاضی را تشکیل می‌دهند) و مدیریت پایگاه داده ایجاد می‌کند.

اگرچه در سوال از ما در خصوص KDD، مطلبی نخواستیم، برای روشن‌تر شدن مفهوم داده‌کاوی لازم است مختصری از KDD مطرح شود. بطور خلاصه KDD را با ۵ گام می‌شناسیم:

۱. انتخاب

۲. پیش پردازش

۳. تبدیل

۴. داده‌کاوی

۵. تفسیر و ارزیابی

اگرچه لزوماً KDD را با این ۵ گام نمی‌شناسند، ساده‌ترین شکلی که KDD را تفسیر می‌کنند شامل ۳ مرحله: پیش‌پردازش، داده‌کاوی و ارزیابی نتیجه است.

پیش پردازش یعنی اینکه داده قبل از اینکه قابل استفاده در الگوریتم‌های داده‌کاوی باشد، باید مرتب و تمیز شود. داده‌کاوی نیز معمولاً یکی از این ۶ کار را انجام می‌دهد:

۱. یافتن ناهنجاری‌ها

۲. قوانین انجمنی

۳. خوشه بندی

۴. طبقه بندی

<sup>۱</sup> Computer decision support systems

<sup>۱۴</sup> misnomer

۵. رگرسیون

۶. خلاصه سازی

از آنجایی که داده‌کاوی می‌تواند سهوا اشتباه استفاده شود و نتیجه ای را ایجاد کند که خیلی خوب است به نظر برسد ولی قابل استفاده روی داده‌های جدید نباشد (مثل بیش برآزش<sup>۲</sup> در یادگیری ماشین)، نیاز به ارزیابی نتیجه داریم. برای اینکار نیز از یک مجموعه آزمون استفاده می‌کنیم. مثلا داده‌کاوی سعی دارد تفاوت میان ایمیل های اسپم و واقعی را تشخیص دهد. ابتدا روی یک مجموعه داده آموزش می‌بیند. پس از اینکه آموزش دید و الگوی آموزش دیده بدست آمد، آن را روی مجموعه آزمون که قبلا آن‌ها را ندیده است، اعمال می‌کنند و سپس دقت آن بررسی می‌شود.

## هوش مصنوعی<sup>۱</sup>

موجودات مصنوعی با هوش در داستان‌های زیادی استفاده شده‌اند. مطالعه استدلال با فیلسوفان و ریاضیدانان شروع شد. مطالعه منطق ریاضی منجر به پیدایش تئوری محاسبات توسط آلن تورینگ<sup>۲</sup> شد. در دهه های ۱۹۵۰ دو دید از نحوه پیدایش هوش ماشین<sup>۳</sup> ظاهر شد. دید اول، هوش مصنوعی نمادین<sup>۴</sup>، این بود که از کامپیوترها برای ساخت ظاهری نمادین از دنیا و سیستم‌ها که بتواند در خصوص دنیا استدلال کند، استفاده شود. موضوعی که بسیار به این دید نزدیک بود، جستجو ابتکاری<sup>۵</sup> بود. دید دوم، رویکرد پیوندی<sup>۶</sup>، تلاش بر ایجاد هوش از طریق یادگیری داشت. مثالی از این دید شامل پرسپترون‌ها است. رشته هوش مصنوعی در سال ۱۹۵۶ متولد شد و رشد خوبی داشت تا سال ۱۹۷۴ که با ظهور مشکلات جدید، پژوهش‌ها در این حوزه کم رنگ شد. این دوره به زمستان هوش مصنوعی معروف بود چرا که فاند های هوش مصنوعی بسیار دشوار پیدا می‌شد. در اوایل دهه ۱۹۸۰، هوش مصنوعی با موفقیت تجاری سیستم‌های خبره<sup>۷</sup> زندگی دوباره گرفت. اما دوباره در سال ۱۹۸۷ افت کرد و زمستانی دوباره و طولانی‌تر برای هوش مصنوعی شروع شد. در اواخر دهه ۱۹۹۰ و اوایل صده ۲۱ام، هوش مصنوعی کم کم شهرت از دست رفته خود را از طریق یافتن راه حل های خاص برای مسائل خاص، بدست آورد. در دهه‌های اول قرن ۲۱ام یادگیری ماشین این رشته را تقریبا پر کرد و به حل بسیاری از مسائل کمک کرد. هوش مصنوعی یا AI هوشی است که توسط ماشین‌ها نمایش داده می‌شود و در مقابل هوش طبیعی که توسط حیوانات و انسان نمایش داده می‌شود، قرار دارد. پژوهش هوش مصنوعی به مطالعه عوامل هوشمند<sup>۸</sup> تعریف می‌شود و به هر سیستمی که پیرامون خود را درک می‌کند و عملی را انجام می‌دهد که به هدفش نزدیک شود، اشاره می‌کند. نکته جالب اینجاست که قبلا هوش مصنوعی را به عنوان ماشین‌هایی که رفتار شناختی و ذهن انسان و را شبیه‌سازی میکنند، مانند یادگیری و حل مسائل، می‌شناختند اما امروزه این تعریف توسط اغلب پژوهشگران هوش مصنوعی رد شده است و امروزه آن را به اصطلاحاتی چون عقلانیت<sup>۹</sup> و رفتار عاقلانه تعریف میکنند که نحوه بیان هوش را محدود نمی‌کند. هر چه ماشین‌ها توانا تر می‌شوند، کارهایی که نیاز به هوش دارند معمولا از هوش مصنوعی حذف می‌شوند؛ به این پدیده اثر هوش مصنوعی می‌گویند. برای مثال نویسه‌خوان نوری<sup>۱۰</sup> یا OCR مکررا از چیزهایی که آن‌ها را هوش مصنوعی می‌دانیم مستثنی می‌شوند زیرا که یک تکنولوژی عادی شده‌اند. هدف سنتی هوش مصنوعی شامل استدلال، دانش، نمایش، برنامه‌ریزی، یادگیری، پردازش زبان طبیعی، درک و امکان حرکت و تغییر اشیاء بود. توانایی حل یک مشکل دلخواه از هدف‌های بلند مدت این رشته است. لذا مسئله کلی شبیه‌سازی یا ساخت هوش را به زیر مسائل ذکر شده در بالا شکسته‌اند و در ادامه مختصری از هر کدام آورده خواهد شد:

۱. استدلال - حل مسئله: در ابتدا پژوهشگران الگوریتم‌هایی را پیشنهاد دادند که استدلال انسان را بصورت گام به گام تقلید می‌کرد. در اواخر دهه‌های ۱۹۸۰ و ۱۹۹۰، روش‌هایی برای حل مسائلی که در آن‌ها عدم قطعیت وجود داشت و یا اطلاعات کامل وجود نداشت ارائه شد. با گذشت زمان و بزرگتر شدن مسائل، این روش کارایی خود را از دست داد زیرا انسان نیز به ندرت از استدلال های گام به گام استفاده می‌کند.

Connectionist approach<sup>۶</sup>  
Expert Systems<sup>۷</sup>  
Intelligent Agents<sup>۸</sup>  
rationality<sup>۹</sup>  
Optical character recognition<sup>۱۰</sup>

overfitting<sup>۲</sup>  
Artificial Intelligence<sup>۱</sup>  
Alan Turing<sup>۳</sup>  
Machine Intelligence<sup>۴</sup>  
Symbolic AI<sup>۵</sup>  
heuristic search<sup>۵</sup>

۲. نمایش دانش : نمایش دانش و مهندسی دانش به هوش مصنوعی این امکان را می‌دهد تا به سوالات هوشمندانه پاسخ دهد. نمایشی از آنچه که وجود دارد ، مثلا روابط، مجموعه اشیاء، مفاهیم، و غیره که عامل (هوش مصنوعی) آن‌ها را تفسیر می‌کند.

۳. برنامه ریزی: یک عامل هوش مصنوعی که بتواند برنامه ریزی کند، شرایط دنیا را نمایش می‌دهد، پیشبینی می‌کند اگر عملی انجام شود چه تغییری صورت می‌گیرد و انتخاب می‌کند که چه عملی را انجام دهد که ارزش آن را بیشینه کند. در حالت کلاسیک، عامل فرض دارد خودش در دنیا تنهاست اما اگر این فرض نباشد و عوامل دیگری روی دنیا تاثیر داشته باشند باید این عدم قطعیت‌ها را نیز دخالت دهد.

۴. یادگیری : یکی از اساسی ترین مفهومات هوش مصنوعی از شروع این رشته، یادگیری ماشین است. مطالعه الگوریتم‌های کامپیوتری که با گذشت زمان و با کسب تجربه، بطور خودکار بهبود می‌یابند. انواع یادگیری‌ها مانند یادگیری با ناظر (رگرسیون ، طبقه‌بندی) و بدون ناظر (خوشه‌بندی)، یادگیری تقویتی و غیره .

۵. پردازش زبان طبیعی : این شاخه به ماشین‌ها این امکان را می‌دهد تا زبان انسان را بفهمد. یک سیستم قوی در این حوزه می‌تواند یک رابط کاربری باشد که دانش را مستقیم از نوشته‌های انسان بدست می‌آورد.

۶. درک : درک ماشین، یعنی توانایی استفاده از ورودی‌های سنسورها (مثل دوربین، میکروفون و غیره) برای استدلال جنبه‌های دنیا؛ مثلا تشخیص گفتار، تشخیص چهره و غیره .

۷. حرکت و دستکاری : در یک حرف خلاصه می‌شود . روباتیک .

۸. هوش اجتماعی : سیستم‌هایی که احساسات انسان را تشخیص، تفسیر ، پردازش و شبیه‌سازی می‌کنند. حوزه مرتبط با این موضوع ارتباط انسان-کامپیوتر<sup>۱</sup> می‌باشد.

۹. هوش کلی : یک ماشین با هوش جامع می‌توان مسائل بیشتری را حل کند و به هوش انسان نزدیک تر است.

به طور کلی هوش مصنوعی مانند علم داده است یعنی یک رشته جدا است. البته هوش مصنوعی و علم داده اشتراکاتی دارند و شاید این اشتراکات در آمار و یادگیری ماشین بیشتر دیده شود . همین‌طور داده‌کاوی نیز با هوش مصنوعی و علم داده اشتراکاتی دارد ولی هر دو این‌ها مفاهیم جامع‌تر می‌باشند و محتوای زیادی را در بر می‌گیرند. همان‌طور که در بخش قبل ذکر شد، یادگیری ماشین و پردازش زبان طبیعی بخشی از هوش مصنوعی هستند و هوش مصنوعی جامع‌تر از این‌هاست.

## یادگیری ماشین<sup>۲</sup>

اصطلاح یادگیری ماشین در سال ۱۹۵۹ توسط آرتور ساموئل<sup>۳</sup> ساخته شد ، البته در همین دوران نیز هم معنی کامپیوترهای خودآموز استفاده می‌شد. در اوایل دهه ۱۹۶۰ ماشین یادگیری بصورت آزمایشی و با استفاده از حافظه نواری پانچی<sup>۴</sup> ، سایبرترون<sup>۵</sup> ، توسط شرکتی برای آنالیز سیگنال‌های خورشیدی، الکتروگرام و الگوهای گفتار با استفاده از یادگیری تقویتی ابتدایی ساخته شد.

امروزه یادگیری ماشین دو هدف در نظر دارد. یکی اینکه داده را بر طبق مدل ساخته شده، طبقه بندی کند و دیگری برای اتفاقات آینده بر اساس مدل ساخته شده پیش‌بینی ارائه دهد.

با تلاش‌های علمی، یادگیری ماشین از تلاش‌های دسترسی به هدف هوش مصنوعی، رشد کرد. اگرچه بعدا با تاکید بیشتر بر روی رویکرد منطقی و دانش-محور باعث ایجاد بریدگی میان هوش مصنوعی و یادگیری ماشین شد. در دهه ۱۹۹۰ یادگیری ماشین شروع به شناخته شدن به عنوان یک رشته جدید کرد و هدف خود را از رسیدن به هوش مصنوعی به رسیدگی به مسائل

<sup>۴</sup>punched tape memory  
<sup>۵</sup>Cybertron

<sup>۱</sup>Human-Computer Interaction  
<sup>۲</sup>Machine Learning  
<sup>۳</sup>Arthur Samuel

قابل حل با ماهیت عملی تغییر داد و تمرکز خودش را از رویکردهای نمادین که از هوش مصنوعی به ارث برده بود به متدها و مدل‌هایی که از آمار، منطق فازی و نظریه احتمال قرض گرفته بود، تغییر داد.

یادگیری ماشین یک رشته تحقیقاتی در خصوص فهمیدن و ساختن روش‌هایی است که یاد بگیرند. یعنی روش‌هایی که به داده نفوذ کرده تا کارایی یکسری فعالیت‌ها را بهبود دهند. همان طور که ذکر شد، یادگیری ماشین را بخشی از هوش مصنوعی در نظر می‌گیرند. الگوریتم‌های یادگیری ماشین مدل را بر اساس داده‌های آموزش ساخته تا بعداً پیش‌بینی انجام دهند. از الگوریتم‌های یادگیری ماشین در زمینه‌های زیادی چون دارو، فیلتر ایمیل (اسپم و واقعی)، تشخیص گفتار و بینایی کامپیوتر<sup>۶</sup>، جاهایی که الگوریتم‌های مرسوم امکان انجام آن کار خاص را ندارند یا بسیار سخت‌اند، استفاده می‌شود.

تفاوت میان هوش مصنوعی و یادگیری ماشین معمولاً اشتباه فهمیده می‌شود. یادگیری ماشین یاد می‌گیرد و پیش‌بینی می‌کند براساس مشاهدات منفعل<sup>۱</sup>؛ این درحالیست که هوش مصنوعی دلالت بر عاملی دارد که با محیط برای یادگیری و انجام عمل مناسب که شانس بدست آوردن هدفش را بیشینه می‌کند، در حال تعامل است.

امروزه بسیاری همچنان یادگیری ماشین را زیررشته‌ای از هوش مصنوعی در نظر می‌گیرند درحالیکه دیگران می‌گویند همه یادگیری ماشین در هوش مصنوعی نیست بلکه فقط آن بخش هوش آن زیر مجموعه‌ای از هوش مصنوعی حساب می‌شود. یادگیری ماشین و داده‌کاوی نیز معمولاً متدهای یکسانی را استفاده می‌کنند و به طور قابل توجهی همپوشانی دارند. اما از آنجایی که یادگیری ماشین بر پیش‌بینی بر اساس ویژگی‌های شناخته شده که از داده‌های آموزش یادگرفته است، تمرکز دارد؛ داده‌کاوی بر کشف ویژگی‌هایی که قبلاً شناخته نشده‌اند تمرکز دارد. عملاً داده‌کاوی از روش‌های یادگیری ماشین با هدف دیگری استفاده می‌کند. از طرفی یادگیری ماشین هم از روش‌های داده‌کاوی به عنوان یادگیری بدون ناظر یا در بخش پیش پردازش داده برای افزایش دقت یادگیری استفاده می‌کند. در یادگیری ماشین، عملکرد معمولاً با توجه به توانایی تولید دوباره دانش شناخته شده ارزیابی می‌شود ولی در داده‌کاوی و KDD هدف اصلی یافتن دانشی است که قبلاً ناشناخته است.

یادگیری ماشین همچنین با بهینه‌سازی ارتباط اساسی دارد. بسیاری از مسائل یادگیری ماشین به صورت کمینه کردن تابع زیان<sup>۲</sup> روی داده‌های آموزش عمل می‌کنند. تفاوت میان یادگیری ماشین و بهینه‌سازی در هدف کلی سازی ظاهر می‌شود. یعنی بهینه‌سازی تابع زیان را روی داده‌های آموزش کمینه می‌کند ولی یادگیری ماشین هدفش کمینه کردن این تابع روی داده‌هایی است که تا کنون آن‌ها را ندیده است (داده‌های آزمون).

یادگیری ماشین و آمار نیز از نظر روش‌ها بسیار به هم نزدیک هستند اما در هدف اصلیشان تفاوت دارند. آمار از نمونه یک استنباط آماری جامعه را بدست می‌آورد درحالیکه یادگیری ماشین الگوهای پیش‌بینی کلی را پیدا می‌کند. در هر صورت برخی از آماردانان روش‌هایی را از یادگیری ماشین بدست آورده‌اند که به رشته‌ای به نام یادگیری آماری<sup>۳</sup> منجر شد. هدف اصلی یک یادگیرنده این است که بتواند از تجربه خود تعمیم دهد. یعنی در یادگیری ماشین، این توانایی را داشته باشد که روی نمونه‌ها یا کارهایی که تاکنون آن‌ها را ندیده است با توجه به تجربه بدست آورده از داده‌های آموزش، با دقت خوب عمل کند.

برای بهترین عملکرد در خصوص تعمیم این یادگیرنده‌ها، پیچیدگی فرض با پیچیدگی تابع داده‌ها باید همخوانی داشته باشد. بطور واضح تر باید یک مصالحه وجود داشته باشد. و در غیر اینصورت مسائلی چون کم برازش<sup>۴</sup> یا بیش برازش رخ می‌دهد. از طرفی نیز باید در زمان چندجمله‌ای قابل حل باشد.

معمولاً یادگیری ماشین را به دسته‌های متفاوتی تقسیم می‌کنند از جمله یادگیری با ناظر که در آن معلم داریم. یعنی هدف پیدا کردن مدلی است که ورودی را به خروجی وصل می‌کند. در یادگیری بدون ناظر برچسب نداریم. عملاً خود یادگیری بدون ناظر هدف ماست (یعنی پیدا کردن الگو پنهان) مثل خوشه‌بندی. یادگیری های نیمه ناظر نیز هستند. یادگیری ها تقویتی دسته دیگری از یادگیری ماشین است که در یک محیط پویا صورت می‌گیرد و فیدبک‌هایی را دریافت می‌کند و سعی دارد جایزه خود را بیشینه کند.

Statistical Learning<sup>۳</sup>  
Underfitting<sup>۴</sup>

Computer Vision<sup>۶</sup>  
passive observations<sup>۱</sup>  
Loss Function<sup>۲</sup>

## کلان داده ۵

کلان داده به دیتاست‌هایی اشاره دارد که بسیار بزرگ هستند یا برای ابزارهای سنتی پردازش داده بسیار پیچیده هستند. داده با ردیف‌های بیشتر یعنی قدرت آماری بیشتر درحالی‌که داده با پیچیدگی بیشتر (یعنی ستون‌های بیشتر) ممکن است به میزان کشف غلط<sup>۶</sup> بیشتر منجر شود. آنالیز کلان داده‌ها شامل چالش‌هایی چون گرفتن داده، ذخیره سازی داده، آنالیز داده، جستجو، اشتراک، انتقال، دیداری‌سازی، درخواست، بروزرسانی، امنیت اطلاعات و منبع داده می‌شود.

کلان داده‌ها ابتدا با سه مفهوم اصلی ساخته شدند: مقدار، گوناگونی و سرعت. بعداً این مفاهیم گسترش یافتند. امروزه از کلان داده برای اشاره به تحلیل‌های پیشگویانه و تحلیل‌های رفتار کاربر یا سایر روش‌های پیشرفته تحلیل داده که مقادیر را از کلان داده‌ها استخراج می‌کنند، استفاده می‌شود.

عنوان کلان داده از دهه ۱۹۹۰ به بعد، با کمک جان مشی<sup>۱</sup>، استفاده می‌شد. کلان داده‌ها را معمولاً شامل آن دسته از دیتاست‌ها هستند که اندازه آن‌ها فراتر از توانایی ابزارهای شناخته شده برای گرفتن، مدیریت، پردازش و غیره است. فلسفه کلان داده شامل داده‌های بدون ساختار، با ساختار و نیمه ساختار می‌باشد ولی تمرکز اصلی آن روی داده‌های بدون ساختار است. کلان داده را می‌توان با مشخصات زیر توصیف کرد (V ها):

۱. Volume: مقدار داده تولید یا ذخیره شده.

۲. Variety: گونه و طبیعت داده. طبیعت داده از با ساختار به نیمه ساختار و به بدون ساختار تفاوت دارد.

۳. Velocity: سرعتی که داده تولید و پردازش می‌شود تا چالش خاص برطرف شود. کلان داده‌ها معمولاً بطور پیوسته تولید می‌شوند.

۴. Veracity: صحت و اطمینان داده که به کیفیت و ارزش داده اشاره دارد. یعنی عملاً کلان داده‌ها نباید فقط زیاد باشند.

۵. Value: ارزش بدست آمده از پردازش و آنالیز این دیتاست‌های بزرگ.

۶. Variability: شاخصه تغییر فرمت، ساختار و منبع کلان داده. یعنی عملاً کلان داده می‌تواند شامل داده‌های با ساختار و بدون ساختار باشد و از منابع گوناگون بیاید.

۷. Exhaustive: آیا کل سیستم گرفته شده است یا خیر. کلان داده‌ها ممکن است همه داده‌ها را از منابعش در اختیار قرار ندهد.

۸. Relational: اگر داده جمع‌آوری شده شامل فیلدهای رایج که قابل به هم پیوستن و غیره هستند، است.

۹. Extensional: اگر فیلد جدیدی در هر بخش از داده جمع‌آوری شده به راحتی قابل اضافه شدن یا تغییر است.

۱۰. Scalability: اگر اندازه سیستم انبار کلان داده بطور سریع قابل گسترش و افزایش است.

کلان داده‌ها در شکل‌های مختلف وجود داشته‌اند و کاربردهای مختلفی دارند. از جمله در دولت، سود، سلامت، آموزش، رسانه، بیمه، اینترنت اشیا، فناوری اطلاعات و غیره کاربرد دارد.

## پردازش زبان طبیعی<sup>۲</sup>

پردازش زبان طبیعی یا NLP یک زیر رشته‌ای از علوم کامپیوتر، زبان شناسی<sup>۳</sup> و هوش مصنوعی است که بیشتر به تعاملات میان زبان انسان و کامپیوتر مربوط می‌شود. یعنی چگونه کامپیوترها را برنامه‌ریزی کنیم تا داده‌های زبان طبیعی را آنالیز و پردازش

<sup>۲</sup>Natural Language Processing  
<sup>۳</sup>Linguistics

<sup>۵</sup>Big Data  
<sup>۶</sup>false discovery rate  
<sup>۱</sup>John Mashey

کند. هدف یک کامپیوتری است که بتواند محتویات یک متن را، به همراه تفاوت‌های ظریف زبانی آن‌ها، بفهمد. این تکنولوژی می‌تواند اطلاعات و مفاهیم را که درون متون وجود دارند را استخراج کند و همچنین آن‌ها را نیز مرتب و گروه‌بندی کند. از نمونه چالش‌های پردازش زبان طبیعی می‌توان به تشخیص گفتار، فهمیدن زبان طبیعی و تولید زبان طبیعی اشاره کرد. ریشه پردازش زبان طبیعی به دهه ۱۹۵۰ برمی‌گردد. در سال ۱۹۵۰، تورینگ در مقاله‌ای آزمون تورینگ<sup>۴</sup> را به عنوان معیار هوش ارائه داد. البته در آن زمان این موضوع به عنوان مسئله‌ای جدا از هوش مصنوعی در نظر گرفته نمی‌شد. از تاریخچه پردازش زبان طبیعی می‌توان به پردازش زبان طبیعی نمادین در دهه‌های ۱۹۵۰ تا اوایل دهه ۱۹۹۰ اشاره کرد بدین صورت که با داشتن مجموعه‌ای از قوانین، کامپیوتر فهم از زبان طبیعی را با اعمال این قوانین روی داده‌هایی که مشاهده می‌کند، تقلید می‌کند. از دهه ۱۹۹۰ به بعد، تا دهه ۲۰۱۰، پردازش زبان طبیعی آماری رشد یافت. تا ۱۹۸۰ بیشتر پردازش‌های زبان طبیعی براساس مجموعه‌های پیچیده قوانینی بود که بصورت دستی نوشته می‌شدند. در اواخر دهه ۱۹۸۰ انقلابی در این حوزه رخ داد و این انقلاب با معرفی یادگیری ماشین برای پردازش زبان همراه بود. امروزه آنچه در پردازش زبان طبیعی معروف است، پردازش زبان طبیعی عصبی<sup>۱</sup> می‌باشد. در دهه ۲۰۱۰، شبکه‌های عصبی عمیق به پردازش زبان طبیعی کمک کردند. از نمونه فعالیت‌های پردازش زبان طبیعی می‌توان به تشخیص گفتار، تبدیل متن به گفتار و غیره اشاره کرد. همانطور که ویکیدیا می‌گوید:

”چالش اصلی و عمده در این زمینه درک زبان طبیعی و ماشینی کردن فرایند درک و برداشت مفاهیم بیان‌شده با یک زبان طبیعی انسانی است. به تعریف دقیق‌تر، پردازش زبان‌های طبیعی عبارت است از استفاده از رایانه برای پردازش زبان گفتاری و زبان نوشتاری. بدین معنی که رایانه‌ها را قادر سازیم که گفتار یا نوشتار تولید شده در قالب و ساختار یک زبان طبیعی را تحلیل و درک نموده یا آن را تولید نمایند. در این صورت، با استفاده از آن می‌توان به ترجمه زبان‌ها پرداخت، از صفحات وب و بانک‌های اطلاعاتی نوشتاری جهت پاسخ دادن به پرسش‌ها استفاده کرد، یا با دستگاه‌ها، مثلاً برای مشورت گرفتن به گفت‌وگو پرداخت. هدف اصلی در پردازش زبان طبیعی، ایجاد تئوری‌هایی محاسباتی از زبان، با استفاده از الگوریتم‌ها و ساختارهای داده‌ای موجود در علوم رایانه است. بدیهی است که در راستای تحقق این هدف، نیاز به دانشی وسیع از زبان است و علاوه بر محققان علوم رایانه، نیاز به دانش زبان‌شناسان نیز در این حوزه می‌باشد. با پردازش اطلاعات زبانی می‌توان آمار مورد نیاز برای کار با زبان طبیعی را استخراج کرد. کاربردهای پردازش زبان طبیعی به دو دسته کلی قابل تقسیم است: کاربردهای نوشتاری و کاربردهای گفتاری. از کاربردهای نوشتاری آن می‌توان به استخراج اطلاعاتی خاص از یک متن، ترجمه یک متن به زبانی دیگر یا یافتن مستندات خاص در یک پایگاه داده نوشتاری (مثلاً یافتن کتاب‌های مرتبط به هم در یک کتابخانه) اشاره کرد. نمونه‌هایی از کاربردهای گفتاری پردازش زبان عبارتند از: سیستم‌های پرسش و پاسخ انسان با رایانه، سرویس‌های اتوماتیک ارتباط با مشتری از طریق تلفن، سیستم‌های آموزش به فراگیران یا سیستم‌های کنترلی توسط صدا. در سالهای اخیر این حوزه تحقیقاتی توجه دانشمندان را به خود جلب کرده‌است و تحقیقات قابل ملاحظه‌ای در این زمینه صورت گرفته‌است. از مهمترین موانع اساسی این حوزه می‌توان به: (۱) نیاز به درک معانی: رایانه برای آن که بتواند برداشت درستی از جمله‌ای داشته باشد و اطلاعات نهفته در آن جمله را درک کند، گاهی لازم است که برداشتی از معنای کلمات موجود در جمله داشته باشد و تنها آشنایی با دستور زبان کافی نباشد. (۲) دقیق نبودن دستور زبان‌ها: دستور هیچ زبانی آن‌قدر دقیق نیست که با استفاده از قواعد دستوری همیشه بتوان به نقش هریک از اجزای جمله‌های آن زبان پی برد.

پردازش زبان‌های طبیعی رهیافت بسیار جذابی برای ارتباط بین انسان و ماشین محسوب می‌شود و در صورت عملی شدنش به‌طور کامل می‌تواند تحولات شگفت‌انگیزی را در پی داشته باشد. سیستم‌های قدیمی محدودی مانند SHRDLU که با واژه‌های محدود و مشخصی سر و کار داشتند، بسیار عالی عمل می‌کردند، به‌طوری‌که پژوهشگران را به شدت نسبت به این حوزه امیدوار کرده‌بودند. اما در تقابل با چالش‌های جدی‌تر زبانی و پیچیدگی‌ها و ابهام‌های زبان‌ها، این امیدها کم‌رنگ شدند. مسئله پردازش زبان‌های طبیعی معمولاً یک مسئله AI-Complete محسوب می‌شود، چرا که محقق شدن آن به‌طور کامل مستلزم سطح بالایی از درک جهان خارج و حالات انسان برای ماشین است.“



## حفاری متن<sup>۲</sup>

حفاری متن یا داده‌کاوی متن یا تحلیل متن فرآیندی است که در آن اطلاعات با کیفیت از متن استخراج می‌شود و شامل کشف اطلاعات جدید، قبلاً ناشناخته می‌باشد. اطلاعات با کیفیت معمولاً با تشکیل الگوها و ترندهایی<sup>۳</sup> از طریق یادگیری الگو آماری بدست می‌آید. بر اساس هوتو و همکاران<sup>۴</sup> در سال ۲۰۰۵، می‌توانیم میان سه دیدگاه از حفاری متن، تمایز قائل شویم. این سه دیدگاه شامل استخراج اطلاعات، داده‌کاوی و فرآیند KDD می‌باشد. حفاری متن معمولاً شامل ساختار سازی ورودی‌های متنی (معمولاً تجزیه کردن)، استنتاج الگوها در داده‌های ساختاری و در نهایت ارزیابی و تفسیر خروجی می‌باشد. کیفیت بالا در حفاری متن به معمولاً تلفیقی از علاقه، ارتباط<sup>۵</sup> و تازگی<sup>۱</sup> اشاره دارد. از نمونه کارهای حفاری متن می‌توان به گروه‌بندی متون، خوشه‌بندی متن، استخراج مفهوم، آنالیز و تحلیل عقیده، خلاصه‌سازی متون و مدل‌سازی روابط موجودیت‌ها اشاره کرد. آنالیز متن شامل بازیابی اطلاعات<sup>۲</sup> و آنالیز واژگان<sup>۳</sup> می‌باشد تا توزیع فراوانی کلمات، الگوها، اطلاعات و غیره را مطالعه کند. تکنیک‌های داده‌کاوی شامل آنالیز روابط و انجمن‌ها، دیداری‌سازی و تحلیل پیش‌بینی می‌باشد. هدف نهایی، تبدیل متن به داده‌ای برای آنالیز می‌باشد که توسط مواردی چون NLP و یا الگوریتم‌های دیگر انجام می‌گردند. حفاری متن امروزه در موارد زیادی توسط دولت، پژوهشگران و غیره استفاده می‌شود. بنابر ویکیپدیا :

”یک کاربرد معمول، جهت اسکن مجموعه‌ای از اسناد نوشته شده در یک زبان طبیعی و مدل کردن مجموعه اسناد برای اهداف کلاس‌بندی پیشگویانه یا پرکردن یک پایگاه داده یا ایندکس جستجو با اطلاعات استخراج شده است.

اصطلاح آنالیز متن یک مجموعه از تکنیک‌های زبانشناسی، آمار و یادگیری ماشینی را توضیح می‌دهد که محتوای اطلاعات منابع متنی را برای هوشمند سازی کسب و کار، آنالیز اکتشافی داده، تحقیقها یا سرمایه‌گذاری ساختار داده و مدل می‌کند. این اصطلاح تقریباً مترادف متن کاوی است. اصطلاح آنالیز متن بیشتر در کسب‌وکار مورد استفاده قرار می‌گیرد در حالی که متن کاوی حوزه کاربرهای قدیمتر به ویژه تحقیقها علوم وابسته به زندگی و هوشمند سازی دولت‌ها استفاده می‌شود.

اصطلاح آنالیز متن همچنان شرح می‌دهد که کاربرد آنالیز متن برای پاسخ به مشکل‌های کسب و کار، چه وابسته یا مستقل از پرس و جو و آنالیزهای میدانی و داده‌های عددی باشد. واضح است که ۸۰ درصد از اطلاعات وابسته به کسب و کار در شکلی بدون ساختار و متنی است. این تکنیک‌ها و فرایندها دانشی، حقایق، قواعد کسب و کار و ارتباطات را کشف و ارائه می‌نمایند که در غیر این صورت در ساختاری متنی، غیرقابل نفوذ برای فرایندهای خودکار باقی مانده بودند.“

## تمرین دوم

### پیش‌گفتار

فایل‌های پایتون هر بخش در گوگل کولب نوشته شده و لینک اشتراک آن در ادامه هر بخش قرار گرفته است، علاوه بر آن فایل اصلی آن‌ها نیز ارسال شده‌اند ولی پیشنهاد می‌کنم ابتدا کولب آن‌ها را مشاهده کنید.

### بیماران پارکینسون

اغلب اطلاعاتی که در سوال از ما خواسته شده است، به طور کامل در فایل parkinsons.names آورده شده‌اند و در اینجا بطور خلاصه شرح داده می‌شوند.

novelty<sup>۱</sup>  
Information Retrieval<sup>۲</sup>  
Lexical Analysis<sup>۳</sup>

Text Mining<sup>۲</sup>  
Trends<sup>۳</sup>  
Hotho et al<sup>۴</sup>  
relevance<sup>۵</sup>

• مشخصات دیتاست : چند متغیره<sup>۴</sup>

• تعداد نمونه‌ها : ۱۹۷ (۱۹۵)

• حوزه : سلامت

• نوع ویژگی‌های : اعداد حقیقی

• تعداد ویژگی‌ها : ۲۳

• زمان اهدا : 2008-06-26

• وظیفه : طبقه‌بندی

• مقدار گمشده : ندارد

این دیتاست توسط مکث لیتل<sup>۵</sup> از دانشگاه آکسفورد<sup>۶</sup> با همکاری مرکز ملی صدا و گفتار (در دنور<sup>۱</sup>) که ضبط سیگنال‌های گفتار را برعهده داشت، ساخته شده است. مطالعه اصلی روی روش‌های استخراج ویژگی<sup>۲</sup> برای اختلالات کلی صدا بود. اطلاعات دیتاست : این دیتاست شامل محدوده صداهای زیستی اندازه‌گیری شده از ۳۱ انسان، ۲۳ نفر با بیماری پارکینسون<sup>۳</sup> (PD). هر ستون از این جدول شامل یک اندازه‌گیری خاص صدا از این افراد می‌باشد. هدف تبعیض قائل شدن میان افراد سالم و ناسالم است.

فرمت داده از نوع ASCII CSV است. هر یک از ۱۹۵ ردیف CSV به یکی از صداهای ضبط شده باز می‌گردد. برای هر مریض ۶ صدا ضبط شده داریم. اسم افراد نیز در ستون اول به همراه شماره صدا ضبط شده آورده شده است. البته به نظرم در اینجا اشتباه کرده است و اینگونه نیست. اطلاعات ویژگی‌ها :

• name : اسم و شماره ضبط صدا

• MDVP:F0(Hz) : میانگین فرکانس صوتی

• MDVP:Fhi(Hz) : بیشینه فرکانس صوتی

• MDVP:Flo(Hz) : کمینه فرکانس صوتی

• MDVP:Jitter(%) تا Jitter:DDP : چندین اندازه از فرکانس‌های اساسی گوناگون

• MDVP:Shimmer تا Shimmer:DDA : چندین اندازه از دامنه‌های گوناگون

• NHR,HNR : دو اندازه از نرخ نویز به اجزای تونال<sup>۴</sup> در صدا

• status : وضعیت سلامتی بیمار، آنچه که در طبقه‌بندی استفاده می‌شود. ۱ اگر بیمار و ۰ اگر سالم باشد.

• RPDE,D2 : دو اندازه پویا غیرخطی پیچیدگی

• DFA : توان مقیاس فراکتال<sup>۵</sup> سیگنال

• spread1,spread2,PPE : سه اندازه غیر خطی از فرکانس‌های اساسی

Feature Extraction<sup>۲</sup>  
Parkinson's Disease<sup>۳</sup>  
tonal<sup>۴</sup>  
fractal<sup>۵</sup>

Multivariate<sup>۴</sup>  
Max Little<sup>۵</sup>  
University of Oxford<sup>۶</sup>  
Denver<sup>۱</sup>

برای استفاده از این دیتاست ابتدا باید کتابخانه‌های لازم را اضافه کنیم برای اینکار کد زیر را اجرا می‌کنیم :

```
import pandas as pd
```

سپس می‌توانیم داده‌های خود را از سایت مورد نظر (یا دانلود کرده یا مستقیماً) وارد کد خود کنیم و از آنجایی که در بخش قبل دانستیم جنس آن از نوع CSV است با کد `read_csv` که در `pandas` موجود است، آن را به یک `DataFrame` تبدیل می‌کنیم.

```
path = '../Data/Parkinson/parkinsons.data'
data = pd.read_csv(path)
```

بعد از آن می‌توانیم به راحتی با دو کد زیر، داده‌های نمونه و هدف را از هم جدا کنیم. برای اینکار از `drop` که از ویژگی‌های `DataFrame` است، استفاده می‌کنیم. با اینکار می‌توانیم ستون `status` را از `DataFrame` خود حذف کرده و آنچه باقی می‌ماند (از جنس `DataFrame`) همان `x` های ماست. از طرفی می‌توانیم با `data['status']` ستون `status` را از دیتافریم اصلی دیافت کنیم، منتها جنس خروجی این کد از نوع سری می‌باشد و برای راحتی آن را به طریق زیر به یک دیتافریم تبدیل می‌کنیم.

```
x = data.drop(['status'],axis=1)
y = pd.DataFrame({'status': data['status']})
```

سپس می‌توانیم با دستوراتی چون

```
data.head()
```

```
print("Type : " , type(data) , "\n\nShape" , data.shape )
```

```
Type : <class 'pandas.core.frame.DataFrame'>
```

```
Shape (195, 24)
```

مشخصات دیتافریم اولیه و کلی را نمایش دهیم : همانطور که مشاهده می‌شود ۱۹۵ ردیف شامل ۲۴ ستون است که ستون `status` آن متغیر وابسته و بقیه متغیرهای مستقل هستند. یعنی `status` به مقادیر `x` وابسته است. البته دستوراتی چون

```
data.info()
data.describe().T
```

نیز اطاعات مفیدی از نوع داده‌ها و تعداد، بیشینه، کمینه، میانگین، انحراف از معیار و غیره هر ستون به ما می‌دهند. همچنین می‌توان بطور مشابه :

```
x.head()
```

```
print("Type : " , type(x) , "\n\nShape" , x.shape)
# x = x.values # to change to numpy array
```

```
Type : <class 'pandas.core.frame.DataFrame'>
```

```
Shape (195, 23)
```

```
y.head()
```

```
print("Type : " , type(y) , "\n\nShape" , y.shape)
# y = y.values # to change to numpy array
```

Type : <class 'pandas.core.frame.DataFrame'>

Shape (195, 1)

را برای متغیرهای x و y نمایش داد. می‌توان آن‌ها را نیز به آرایه تبدیل کرد که در اینجا اینکار انجام نشده است. می‌بینیم که y شامل یک ستون status و x شامل سایر ستون‌ها (۲۳) می‌باشد. پیشنهاد می‌شود کد مربوط به این قسمت را مشاهده و اجرا نمایید. در خصوص بخش دیگر این سوال که از دانشجو خواسته شده است سه کلاس‌بندی در سه مقاله (بعد از ۲۰۱۸) را با هم فقط بررسی کند، جدول زیر آورده خواهد شد.

طبقه‌بندی بیماری پارکینسون		
DOI	روش	بهترین دقت
10.1109/ACCESS.2019.2936564	یادگیری عمیق و شبکه‌های عصبی پیچشی	82.50
10.1109/EBBT.2019.8741725	ماشین‌های یادگیری هیبریدی، تلفیق جنگل تصادفی و SMOTE	94.89
10.32604/iasc.2022.022037	۶ روش از جمله جنگل تصادفی و KNN و درخت تصمیم	97.00

البته لازم است این موضوع را ذکر کنم که در سه مقاله بالا روش‌ها تعمیم یافته شده‌اند و همچنین دقت بدست آمده‌ای که در جدول بالا ذکر شده است، بالاترین دقتی است که بدست آمده و دقت میانگین این طبقه‌بندی‌ها نمی‌باشند.

## NetworkX

از این کتابخانه برای رسم گراف استفاده می‌کنی. ابتدا کتابخانه‌های لازم را اضافه می‌کنیم.

```
import numpy as np
import ast
import pandas as pd
import matplotlib.pyplot as plt
import networkx as nx
```

سپس لازم از محتویات فایل‌های txt را فراخوانی می‌کنیم. ابتدا آن‌ها را با حالت readonly باز می‌کنیم. سپس خط اول که تنها خط آن است را می‌خوانیم (read()). و در نهایت با کمک کتابخانه ast آن‌را به لیست تبدیل می‌کنیم. این کار را برای هر دو فایل انجام می‌دهیم.

```
path_edge_idx = '../Data/NetworkX/edge_idx.txt'
path_node_lbls = '../Data/NetworkX/node_lbls.txt'

edge_idx = ast.literal_eval(open(path_edge_idx, 'r').read())
node_lbls = ast.literal_eval(open(path_node_lbls, 'r').read())
```

سپس می‌توانیم گراف را بسازیم. برای اینکار بعد از ساخت اولیه محیط گراف، یال‌های آن‌را که در فایل edge\_idx موجود است، به آن اضافه می‌کنیم. با اینکار نودهای مربوطه نیز اضافه می‌شوند. ولی اگر نودی هیچ یالی نداشته باشد، با این روش اضافه نمی‌شود. از آنجایی که ۳۵ نود (از ۰ تا ۳۴) داریم، این تعداد را نیز به صورت یک لیست به نودهای گراف اضافه می‌کنیم. البته در اینجا این حرکت لزوم نداشت.

```
G = nx.Graph()
G.add_edges_from(edge_idx)
G.add_nodes_from(range(0, 35)) # not necessary
```

سپس می‌توانیم اطلاعات گراف را همانطور که سوال خواسته برای خود نمایش دهیم. این گراف شامل ۳۵ نود و ۷۴ یال می‌باشد.

```
nx.info(G)
```

می‌توانیم این اطلاعات را نیز از نحوه‌های دیگری بدست آوریم:

```
print("Graph Edges : ", G.edges)
print("Number of Edges : ", len(G.edges))
print("Graph Nodes : ", G.nodes)
print("Number of Nodes : ", len(G.nodes))
print("Each node's Degree : ", G.degree)
```

با اجرای این کد در محیط پایتون، ابتدا لیستی از تاپل‌های یال‌ها و سپس تعداد یال‌ها نوشته می‌شود. لیستی از نودها و تعداد نودها در ادامه آورده می‌شوند. درجه هر نود هم بصورت یک تاپل که اولین مقدار آن عنوان نود و مقدار دوم آن درجه نود است، آورده می‌شود.

می‌توان برای هر نود یا گره همسایه تعریف کرد بدین معنی که نود مورد نظر با آن نود ارتباط دارد (یال میان آن‌ها برقرار است). بدین صورت می‌توانیم همسایگان هر نود را نمایش دهیم و بررسی کنیم که آیا نودی بدون همسایه است یا خیر. این موضوع با روش‌های دیگری چون `G.adj` نیز قابل انجام است.

```
for node in G.nodes:
    print("Node ", node ,
          "Neighbors are (have a connection with) : ",
          end = " ")
    for neighbor in G.neighbors(node):
        print(neighbor, end = " ")
    print("")
```

اما برای اینکه به هدف اصلی سوال برسیم باید بخش‌های زیر را پیاده‌سازی کنیم. اگر بخواهیم برچسب هر نود را با استفاده از فایل `node_labels.txt` بدست بیاوریم، نیاز به یک دیکشنری داریم. لذا برای هر نود در گراف، مقدار برچسب‌اش را در دیکشنری آن ذخیره می‌کنیم. از طرفی نوشتن اعداد زیاد روی تصاویر جنبه خوش‌آیندی ندارد و بهتر است از رنگ‌های متفاوت برای برچسب‌های متفاوت استفاده شود. برای اینکار مجموعه رنگی تعریف می‌شود که برای هر نود در گراف اگر برچسب آن ۱ بود سبز رنگ و اگر ۰ بود قرمز می‌شود.

در سوال از ما خواسته شده است که اندازه هر نود بر اساس درجه آن تغییر کند. برای اینکار لیستی از اندازه‌ها به عنوان `_node_size_1` می‌سازیم که در آن هر نود به اندازه‌اش وصل می‌شود. یعنی شماره نودها از ۰ تا ۳۴ است لذا مقدار عنصر ۰ام این لیست برابر با درجه راس ۰ است. از آنجایی که این مقادیر کم هستند، یک عدد دلخواه در همه آن‌ها ضرب می‌شود تا جنبه ظاهری بهتری پیدا کند. این مقدار دلخواه را با `weight` نمایش می‌دهیم. از طرفی در آخرین درخواست سوال از ما خواسته شده است تا در محاسبه اندازه نودها، آن همسایگانی را در نظر بگیریم که از یک برچسب یکسان هستند یعنی اگر نود اصلی ما برچسب ۱ دارد، فقط آن تعداد همسایگانی مد نظر است که برچسب آن‌ها ۱ است. برای اینکار برای هر نود اندازه ۰ در نظر گرفته می‌شود، سپس با محاسبه هر همسایه و مقدار برچسب آن، اندازه آن افزایش می‌یابد. در نهایت مانند روش قبل، در یک عدد دلخواه ضرب می‌شود. این روش با عنوان `_node_size_2` در کد آورده شده است.

```
labels_dict = {}
for node in G.nodes:
    labels_dict[node] = node_labels[node]

_node_color = []
```

```

for node in G.nodes:
    if node_lbls[node]== 1:
        _node_color.append('springgreen')
    else:
        _node_color.append('lightcoral')

_node_size_1 = []
weight = 15
for node in G.degree:
    _node_size_1.append(node[1]*weight)

_node_size_2 = []
weight = 15
for node in G.nodes:
    size = 0
    for neighbor in G.neighbors(node):
        if node_lbls[neighbor] == node_lbls[node]:
            size+=1
    _node_size_2.append(size*weight)

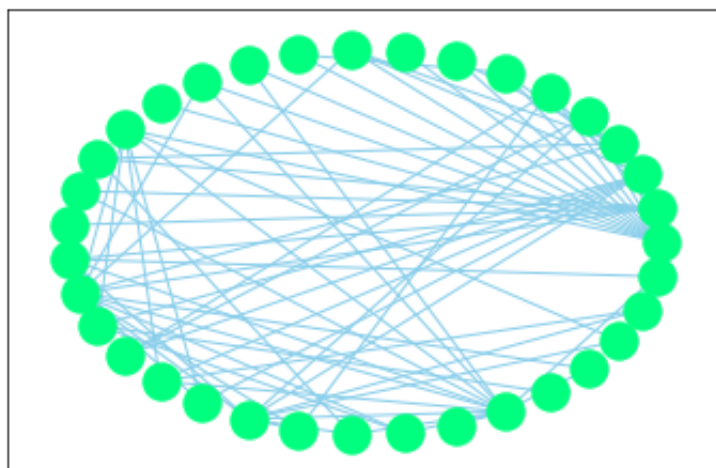
```

در ادامه لازم است این گراف ها را دیداری سازی کنیم. کد اصلی برای تولید یک گراف ساده بدون هیچگونه برجسب یا تغییر رنگ و اندازه. تنها با کمک یال ها.

```

plt.figure(1)
nx.draw_networkx(
    G,
    node_color='springgreen',
    edge_color='skyblue',
    pos = nx.circular_layout(G),
    with_labels = False
)
plt.show()

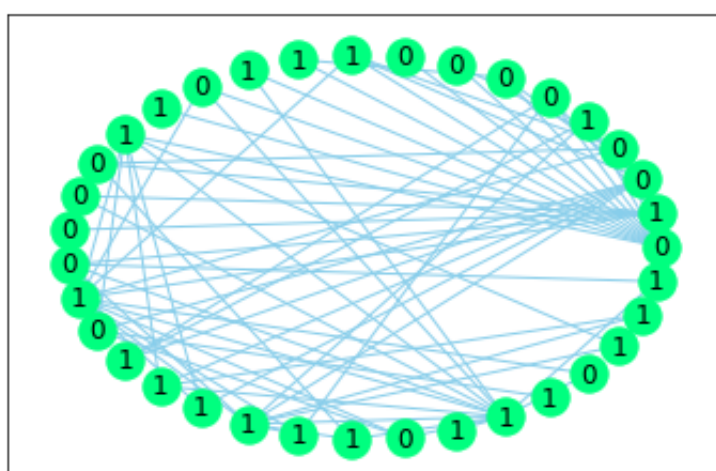
```



گراف ابتدائی

می‌توانیم برچسب هر نود را به آن اضافه کنیم و نتیجه آن گراف زیر می‌باشد:

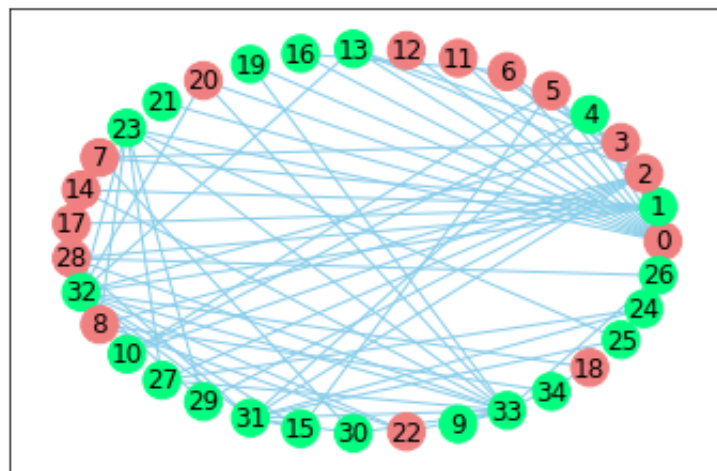
```
plt.figure(2)
nx.draw_networkx(
    G,
    node_color='springgreen',
    edge_color='skyblue',
    pos = nx.circular_layout(G),
    labels = labels_dict,
    with_labels = True
)
plt.show()
```



گراف ابتدائی برچسب‌دار

می‌توان بجای عدد از رنگ برای تمایز ۰ و ۱ های برچسب‌ها استفاده کرد:

```
plt.figure(3)
    nx.draw_networkx(
        G,
        node_color= _node_color,
        edge_color='skyblue',
        pos = nx.circular_layout(G),
        with_labels = True
    )
plt.show()
```



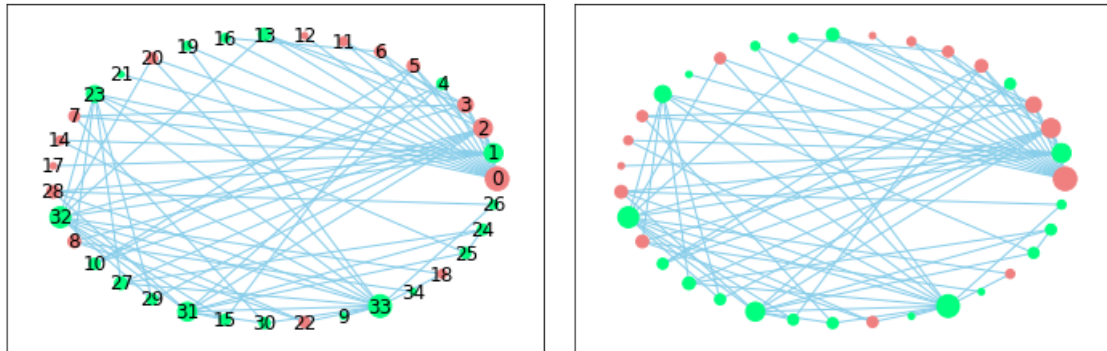
گراف رنگارنگ برجسب‌دار

می‌توان اندازه هر نود را با روش اولی، یعنی بر اساس درجه آن، تغییر داد:

```
plt.figure(4)
nx.draw_networkx(
    G,
    node_size = _node_size_1,
    node_color= _node_color,
    edge_color='skyblue',
    pos = nx.circular_layout(G),
    with_labels = False
)
plt.figure(5)
nx.draw_networkx(
    G,
    node_size = _node_size_1,
    node_color= _node_color,
    edge_color='skyblue',
    pos = nx.circular_layout(G),
    with_labels = True
```



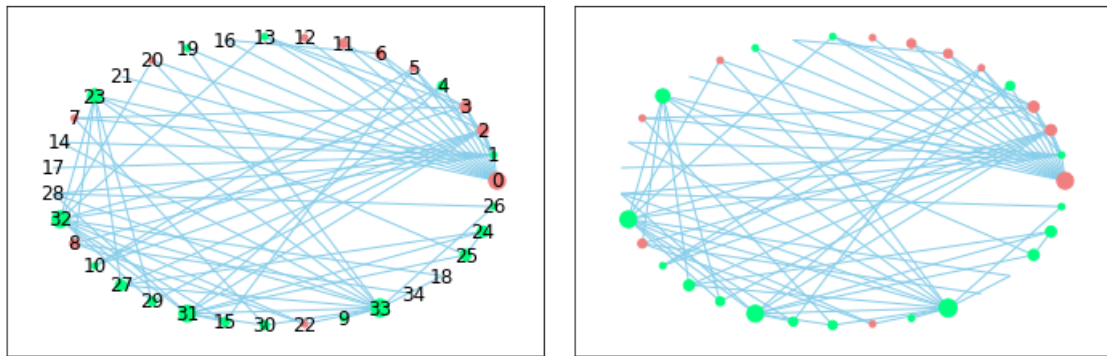
```
)
plt.show()
```



گراف رنگارنگ با اندازه متفاوت بر اساس درجه

می‌توان اندازه را با روش دوم تغییر داد، یعنی بر اساس تعداد همسایگانی که برچسب مشابه دارند.

```
plt.figure(6)
nx.draw_networkx(
    G,
    node_size = _node_size_2,
    node_color= _node_color,
    edge_color='skyblue',
    pos = nx.circular_layout(G),
    with_labels = False
)
plt.figure(7)
nx.draw_networkx(
    G,
    node_size = _node_size_2,
    node_color= _node_color,
    edge_color='skyblue',
    pos = nx.circular_layout(G),
    with_labels = True
)
plt.show()
```



گراف رنگارنگ با اندازه متفاوت بر اساس برجسب همسایه

برای راحتی پیشنهاد می‌کنم کدهای این بخش را که در پوشه کد هستند، مشاهده و بررسی کنید.

## Pavia

داده‌هایی که در این بخش استفاده می‌شود مستقیم از خود سایت جمع‌آوری شده‌اند. یعنی از همان لینک Pavia University Scene در بخش ۳-۲. و لذا فرمت فایل‌ها .mat می‌باشد. فایل اول شامل PaviaU.mat می‌باشد. تصاویر فراطیفی<sup>۱</sup> دانشگاه پاویا توسط سنسورهای ROSIS گرفته شده‌اند. این تصاویر شامل ۱۰۳ باند طیفی، ۶۱۰ در ۳۴۰ پیکسل می‌باشد؛ اما نمونه‌های طیف تصاویر هیچ اطلاعاتی ندارد و لذا ۰ تعریف شده‌اند و قبل از آنالیز دور ریخته می‌شوند. رزولوشن هندسی آن ۳.۱ متر است و حقیقت اصلی این تصاویر فراطیفی به ۹ کلاس خلاصه می‌شود.

این کلاس‌ها شامل آسفالت، درخت، سایه و غیره می‌باشد. فایل دیگر PaviaU\_gt.mat می‌باشد. دیداری سازی حقیق اصلی<sup>۲</sup> این داده‌های فراطیفی است. بعداً در عکس مشخص می‌شود که هرچه رنگ آن روشن تر شود، به یکی از آن ۹ مورد نزدیک تر می‌شود و نقاط کاملاً تیره، هیچ اطلاعاتی موجود نیست. تصویربرداری فراطیفی نوعی از تصویربرداری طیفی است که مانند سایر انواع تصویربرداری طیفی داده‌ها را از بخشی یا تمام طیف الکترومغناطیسی گردآوری و پردازش می‌کند. در حقیقت یک شیء در طول موج‌ها ی مربوط به رنج فراطیفی که بالاتر از طول موج‌های طیف مرئی هستند مورد تصویر برداری قرار می‌گیرد. هدف اصلی در تصویر برداری فرا طیفی، بدست آوردن محتوای طیفی یا به اصطلاح اثر طیفی برای هر پیکسل از تصویر است.

برخلاف چشم انسان که تنها می‌تواند نور قابل دیدار در سه باند (قرمز، سبز و آبی) را ببیند، تصویربرداری فراطیفی دامنه دید را به باندهای بسیار بیشتری گسترش می‌دهد. این نکته حائز اهمیت است که رنگ‌هایی که چشم انسان می‌بیند در واقع بازتاب طول موج‌های متفاوت در رنج طیف مرئی انسان است و وقتی که در مورد گسترش قدرت دید فرای این طیف مرئی صحبت می‌کنیم این تفاوت و برتری خود را به شکل اعداد به ما نشان می‌دهند و نه رنگ‌ها! چون بالاخره چشم انسان رنج مرئی را می‌بیند. تفاوت ظاهری در این است که تصاویر فراطیفی گرفته شده با به اصطلاح رنگ ساختگی آشکارسازی می‌شوند.

ground truth<sup>۲</sup>

HCI<sup>۱</sup>

label	class	Samples
1	Water	824
2	Trees	820
3	Asphalt	816
4	Self-Blocking Bricks	808
5	Bitumen	808
6	Tiles	1260
7	Shadows	476
8	Meadows	824
9	Bare Soil	820

تصاویر فراطیفی کاربردهای زیادی دارند اگرچه ابتدا برای زمین‌شناسی و داده‌کاوی ساخته شده بودند. از نمونه کاربردهای آن می‌توان به کشاورزی، تجزیه و مرتب‌سازی زباله، مراقبت از چشم، پردازش غذا، کانی‌شناسی<sup>۱</sup>، نظارت، فضانوردی، تصویربرداری شیمیایی، محیط زیست و مهندسی عمران اشاره کرد. لذا کاربردهای زیادی در رشته‌های مختلفی دارند. اما برای خواندن این فایل‌ها در پایتون ابتدا کتابخانه‌های لازم را اضافه می‌کنیم:

```
from scipy.io import loadmat
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt
```

سپس مسیرهای این داده‌ها را مشخص و آن‌ها را می‌خوانیم:

```
path_pavia = '../Data/Pavia/PaviaU.mat'
path_paviagt = '../Data/Pavia/PaviaU_gt.mat'
path_npy = '../Data/PVI/pavia.npy'

pavia = loadmat(path_pavia)
paviagt = loadmat(path_paviagt)
pavia_npy = np.load(path_npy)
```

می‌توانیم بعداً با کدهایی چون `pavia` و `paviagt` یا `pavia_npy` فایل‌های خوانده شده را بررسی کنیم. اما در این فایل‌ها (دو فایل اول) تنها به بخش خاصی از آن نیاز داریم. لذا این بخش‌ها را از آن جدا می‌کنیم. سپس می‌توانیم ابعاد داده بدست آمده را چاپ کنیم و آن‌را بررسی کنیم.

```
pavia = pavia['paviaU']
pavia.shape
```

```
(610, 340, 103)
```

```
paviagt = paviagt['paviaU_gt']
paviagt.shape
```

```
(610, 340)
```

```
pavia_npy.shape
```

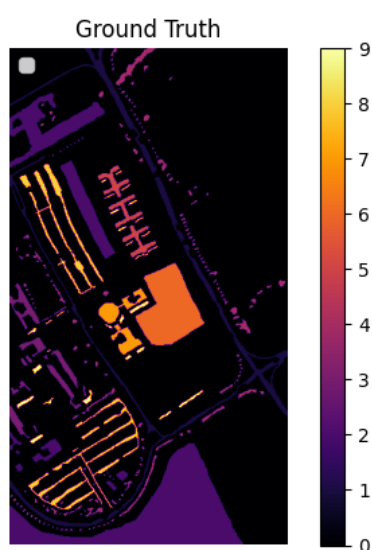
<sup>۱</sup>Mineralogy

(1096, 715, 102)

فایل gt شامل باندهای یکتایی است که این تقاط دارند. با کمک آن می‌توانیم تصویری مشترکی از باندها در هر نقطه بصورت

زیر رسم کنیم:

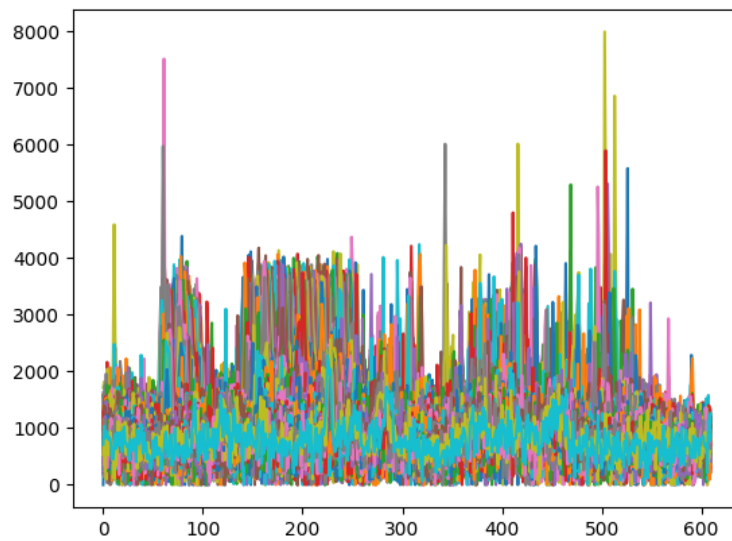
```
plt.figure(1)
plt.imshow(paviagt, cmap='inferno')
plt.colorbar()
plt.axis('off')
plt.title('Ground Truth')
plt.gca().legend(np.unique(paviagt),
                 loc='upper left')
plt.show()
```



می‌توانیم در اینجا آن‌ها را به مقیاس مورد نظر ببریم که من اینکار را نکردم. می‌توانیم تصویر هر بخش را نیز بصورت زیر بدست آوریم.

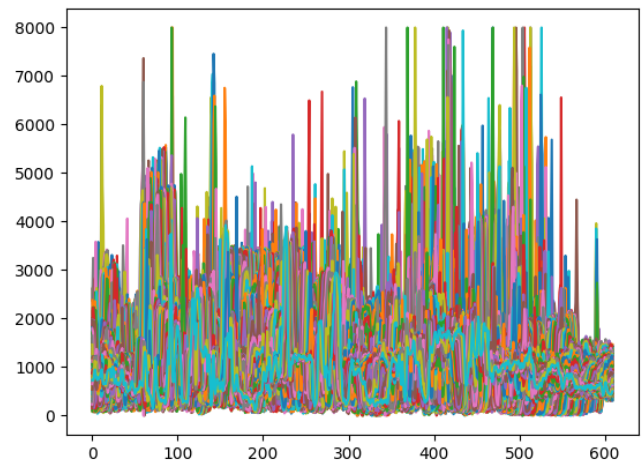
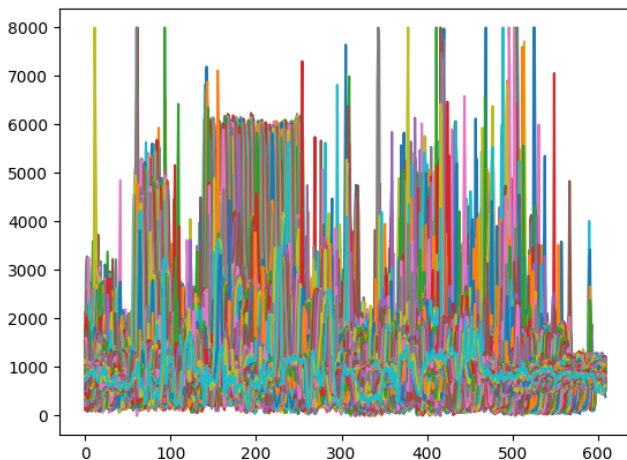
```
pavia[:, :, 0]
```

سپس می‌توانیم این تصویر بدست آمده را چاپ کنیم.



می‌توانیم بصورت رندم نیز برای دیگر بخش‌ها این تصاویر را ایجاد کنیم:

```
plt.plot(pavia[:, :, random.randrange(0, 103)])
plt.show
```



در نهایت می‌توانیم این داده‌ها را بصورت یک دیتافریم درآوریم. خوبی اینکار این است که می‌توانیم کارهای طبقه‌بندی و خوشه‌بندی روی این دیتافریم‌ها انجام دهیم. برای اینکار لازم است که ابتدا آن را به ۲ بعدی تبدیل کنیم.

```
pavia_rshp = pavia.reshape(207400, 103)
pavia_df = pd.DataFrame(pavia_rshp)
pavia_df
```

```
paviagt_df = pd.DataFrame(paviagt)
paviagt_df
```

خروجی این کد ها را می‌توانید در فایل‌های کد ارائه شده در این گزارش مشاهده کنید و خودتان اجرا بگیرید.

## تمرین سوم

### پیش‌گفتار

اطلاعات جامعی از لینک اصلی این چالش یعنی NIPS 2003 قابل دسترسی می‌باشد. در اینجا خلاصه‌ای از آن مطالب آورده خواهد شد. البته لازم به ذکر است اغلب مواردی را که سوال خواسته است در همین سایت کامل آورده شده است و در اینجا من فقط آن‌ها را ترجمه کرده‌ام.

### پس‌زمینه

اخیرا (که مال سال ۲۰۰۳ است) تلاش‌های زیادی روی استخراج ویژگی<sup>۱</sup> صورت گرفته است. در سال‌های گذشته، مقالات متعددی در این حوزه، شامل ساخت ویژگی‌ها<sup>۲</sup>، کاهش ابعاد (فضایی)<sup>۳</sup>، نمایش تنکی<sup>۴</sup> و انتخاب ویژگی<sup>۵</sup>، بیش از ۱۰ درصد ارسالی‌های NIPS را در بر می‌گرفتند و کاربرهای زیادی در موضوعاتی چون بیوانفورماتیک<sup>۶</sup>، شیمی، پردازش متن، تشخیص الگو، تشخیص گفتار و دیدار داشتند.

هدف از این کارگاه اشتراک تکنیک‌ها و متدها است. بخشی از این کارگاه به نمایش و بحث در خصوص نتیجه چالش‌های انتخاب ویژگی می‌پردازد. نتایج بدست آمده در انتخاب ویژگی مربوط به دوران گذشته، روی دیتاست‌های مختلف و شکاف‌های مختلف، می‌باشند لذا مقایسه آن‌ها را سخت می‌کند. برای اینکار یکسری دیتاست (در اینجا ۵ تا) بصورت آزمون محک برای انتخاب ویژگی استفاده می‌شوند و لذا امکان کنترل آن فراهم می‌گردد. این دیتاست‌ها هم به اندازه کافی بزرگ هستند تا نتایج بدست آمده امکان تایید از سمت امار را داشته باشد. ورودی‌ها متغیرهایی پیوسته یا دودویی، بصورت تنک یا چگال، هستند همه مسائل بصورت مسئله طبقه‌بندی دوکلاسه است.

### چالش اصلی

چالش اصلی در پیدا کردن الگوریتمی برای انتخاب ویژگی است که از متدهایی که از همه ویژگی‌ها استفاده می‌کنند، بهتر عمل کند. برای اینکار از ۵ دیتاست استفاده می‌شود و برای سادگی بیشتر نیز مسئله مورد بررسی یک طبقه‌بندی دوکلاسه است.

### نمره‌دهی

ارسالی‌هایی برای نمره‌دهی در نظر گرفته می‌شدند که پیش‌بینی کلاس (برچسب) برای مجموعه‌های آموزش<sup>۷</sup> و ارزیابی<sup>۸</sup> یا Val و آزمون<sup>۹</sup>، برای هر ۵ دیتاست، و لیستی از ویژگی‌های استفاده شده ارائه داده باشند. ارائه مقدار اطمینان طبقه‌بندی دلخواهی بوده است. عملکرد آن‌ها توسط این معیارها سنجیده می‌شود:

- BER: نرخ بالانس خطا<sup>۱۰</sup>. که در حقیقت میانگین نرخ خطای کلاس‌های مثبت و نرخ خطای کلاس‌های منفی است. چون برخی دیتاست‌ها از جمله Dorothea ناهمگن هستند، از این معیار استفاده شده است.
- AUC: مساحت زیر منحنی ROC. ROC منحنی با تغییر یک آستانه در مقادیر متمایز (خروجی‌های) طبقه‌بند بدست می‌آیند. این منحنی نمایانگر کسر مثبت درست به کسر منفی غلط است. در طبقه‌بند های دودویی،  $BER = 1 - AUC$  می‌باشد.

- Ffeat: کسر ویژگی‌های انتخاب شده

bioinformatics<sup>۶</sup>  
training<sup>۷</sup>  
validation<sup>۸</sup>  
test<sup>۹</sup>  
balance error rate<sup>۱۰</sup>

feature extraction<sup>۱</sup>  
feature construction<sup>۲</sup>  
space dimensionality reduction<sup>۳</sup>  
sparse representation<sup>۴</sup>  
feature selection<sup>۵</sup>

- Fprob : کسر prob های پیدا شده در مجموعه ویژگی‌های انتخاب شده.
- از این معیارها برای نمره‌دهی استفاده می‌شود. برای رتبه‌بندی شرکت‌کنندگان، از نتایج مجموعه آزمون، به وسیله نمره‌ای که از تلفیق BER، Ffeat و Fprob بدست می‌آید، استفاده می‌کنند.
- دو به دو طبقه‌بند ها را برای هر دیتاست مقایسه می‌کنند.
- از آزمون McNemar با نرخ ریسک ۵ درصد، برای مقایسه بهتر بودن یک طبقه‌بند نسبت به طبقه‌بند دیگر، استفاده می‌کنند. نمره بدست آمده بصورت ۱ (اگر بهتر بود)، ۰ (اگر نمی‌دانیم) و ۱- (اگر بدتر بود) است.
- اگر نمره ۰ بود (در بالا)، یعنی آزمون قبل آن‌ها را یکی تشخیص داد، از Ffeat در صورتی که اختلاف آن‌ها از ۵ درصد بیشتر باشد، استفاده می‌کنیم.
- اگر نمره ۰ بود، از Fprob استفاده می‌کنیم.
- نمره نهایی شامل جمع نمره‌های مقایسه‌های دوبه‌دو است که توسط بالاترین نمره بدست آمده نرمال‌سازی می‌شود. (برای هر دیتاست)
- نمره نهایی جامع، میانگین نمره‌های نهایی دیتاست‌هاست.
- با این روش نمره‌های مثبت و منفی و حتی ۰ بدست می‌آوریم. یکی از مشکلات این شکل نمره‌دهی این است که اگر یک طبقه‌بند جدید وارد چالش شود، همه نمره‌ها تغییر می‌کند.

## نتایج

در این بخش رتبه‌های برتر در تاریخ ۱ دسامبر، نمایش داده می‌شود. همچنین نمره نهایی آن‌ها و مقدار هر معیار برای آن‌ها، نام فردی که روی آن کار کرده است و روش استفاده آن آورده خواهد شد. نمره‌ها در ۱۰۰ ضرب شده‌اند. BER شامل نرخ خطای همگن (به درصد) می‌باشد. AUC مساحت زیر منحنی ROC می‌باشد که در ۱۰۰ ضرب شده است. Fe درصد ویژگی‌های استفاده شده است. Pr درصد prob های در ویژگی‌های انتخاب شده است.

method	people	score	BER	AUC	Fe	Pr
BayesNN-DFT	Neal/Zhang	88.0	6.84	97.22	80.3	47.8
BayesNN-DFT	Neal/Zhang	86.2	6.87	97.21	80.3	47.8
BayesNN-small	Neal	68.7	8.20	96.12	4.7	2.9
BayesNN-large	Neal	59.6	8.21	96.36	60.3	28.5
RF+RLSC	Torkkola/Tuv	59.3	9.07	90.93	22.5	17.5
final2	Chen	52.0	9.31	90.69	24.9	12.0
SVMBased3	Zhili/Li	41.8	9.21	93.60	29.5	21.7
SVMBased4	Zhili/Li	41.1	9.40	93.41	29.5	21.7
final1	Chen	40.4	10.38	89.62	6.2	6.1
transSVM2	Zhili	36.0	9.60	93.21	29.5	21.7
BayesNN-E	Neal	29.5	8.43	96.30	96.8	56.7
Collection2	Saffari	28.0	10.03	89.97	7.7	10.6
Collection1	Saffari	20.7	10.06	89.94	32.3	25.5

بطور مشابه هم برای تاریخ ۸ دسامبر این رتبه‌بندی انجام شده است و در مقاله اصلی قابل مشاهده است. همچنین می‌توان تیم‌های شرکت کننده را بطور کلی (روش‌های استفاده شده توسط آن‌ها) به جدول زیر تقسیم کرد. لازم به ذکر است که حروفی که در جدول زیر آورده می‌شود، ترجمه شوند.

• classifier : نوع طبقه‌بند (۴ گروه)

- N : شبکه‌های عصبی
- K : مثل SVM یا سایر روش‌های کرنل
- T : درخت
- O : دیگر

• Fengine : موتور انتخاب ویژگی‌ها (۳ گروه)

- C : معیار تک متغیری شامل ضرایب همبستگی
- T : درخت ، مثل RF به عنوان فیلتر
- E : روش‌های تعبیه شده<sup>۱</sup>

• Fsearch : روش‌های جستجو (۴ گروه)

- E : تعبیه شده
- R : رتبه ویژگی
- B : حذف عقب‌رو<sup>۲</sup>
- S : جستجو مفصل

Team	Classifier	Fengine	Fsearch	Ensemble	Transduction
Neal/Zhang	N/O	C/E	E	Yes	Yes
Torkkola/Tuv	K	T	R	Yes	No
Chen/Lin	K	C/T/E	R/E	No	No
Zhili/Li	K	C/E	E	No	Yes
Saffari	N	C	R	Yes	No
Ghostminer	K	C/T	B	Yes	No
Lal et al	K	C	R	No	No
CBAGroup	K	C	R	No	No
Bachrach/Navot	K/O	E	S	No	No

برنده

در هر دو تاریخ، نفرات اول رادولف نیل<sup>۳</sup> و جیانگو ژنگ<sup>۴</sup> بودند. روش آن‌ها شامل تلفیق شبکه‌های عصبی بیزی<sup>۵</sup> و درخت پخش دریخت<sup>۶</sup> بودند. نه تنها روش آن‌ها نمره خوبی به نسبت سایر شرکت کنندگان داشت بلکه مجموعه ویژگی‌هایی که استفاده کرده بودن نیز کوچکترین بود. توضیحات بیشتر در مورد روش آن‌ها در مقاله این چالش آورده شده است.

دیتاست

مشخصات دیتاست شامل نام ، دامنه استفاده، نوع داده (تنگ و چگال)، تعداد ویژگی‌ها، درصد prob ها ، تعداد داده‌های آموزش ، داده‌های ارزیابی و داده‌های تست آورده شده است.

<sup>۴</sup> Jianguo Zhang  
<sup>۵</sup> Bayesian neural networks  
<sup>۶</sup> Dirichlet diffusion trees

<sup>۱</sup> embedded  
<sup>۲</sup> backward elimination  
<sup>۳</sup> Radford Neal



Dataset	Domain	Type	Fe	Pro	Tr	Val	Te
Arcene	Mass Spectrometry	Dense	10000	30	100	100	700
Dexter	Text classification	Sparse	20000	50	300	300	2000
Dorothea	Drug discovery	Sparse binary	100000	50	800	350	800
Gisette	Digit recognition	Dense	5000	30	6000	1000	6500
Madelon	Artificial	Dense	500	96	2000	600	1800

همه دیتاست‌ها در ابتدا به سه زیر مجموعه آموزش، ارزیابی و آزمون شکسته می‌شوند. از مجموعه تست برای نمره‌دهی نهایی استفاده می‌شود و مجموعه ارزیابی برای، ارزیابی عملکرد استفاده می‌شود. همانطور که گفتیم تا اواخر دهه ۱۹۹۰، مقاله‌هایی که در مورد انتخاب ویژگی بودند، دیتاست‌هایی با کمتر از ۴۰ ویژگی را مورد بررسی قرار می‌دادند اما سال‌های بعد این شرایط تغییر کرد. این ۵ دیتاست نیز برای آزمون الگوریتم‌های انتخاب، مرتب شده‌اند.

دیتاست‌ها از دامنه‌ها و سختی‌های متفاوتی می‌آیند. متغیرهای ورودی آن‌ها در برخی پیوسته یا گسسته، تنگ یا چگال است؛ یکی از دیتاست‌ها ناهمگن است. یکی دیگر، MADELON، بصورت مصنوعی برای نشان دادن یک مشکل خاص، طراحی شده است (انتخاب مجموعه‌ای از ویژگی‌ها بطوری که هیچ ویژگی به تنهایی اطلاعاتی ندارد). همانطور که قبلاً نیز اشاره شد، دیتاست‌ها مثال‌های زیادی دارند تا آزمون آن از نظر آماری به مشکل نخورد. برای اینکه افرادی که قبلاً این دیتاست‌ها را دیده‌اند، برتری بر دیگران نداشته باشند، هویت این دیتاست‌ها در این آزمون محک، پنهان شده است.

چندین پیش پردازش و فرمت کردن داده روی داده‌ها برای متمایز شدن مبدا آن‌ها انجام داده شده است. بطور خاص، ما تعدادی از ویژگی‌ها را به عنوان probe معرفی می‌کنیم. در ابتدا آن‌ها بصورت تصادفی انتخاب شده‌اند (بصورت تصادفی از توزیعی که به توزیع ویژگی‌ها شبیه باشد)، اما هیچ اطلاعاتی در مورد برجسب‌ها (کلاسها) با خود همراه ندارند. این prob‌ها در ارزیابی عملکرد استفاده می‌شوند.

## پیش پردازش

مرکزی کردن و مقیاس کردن ویژگی‌ها از جمله پیش پردازش‌هایی بود که بیشترین استفاده را داشت. برخی روش‌ها نیاز به گسسته‌سازی ویژگی‌ها داشتند. یک گروه از الگوها را نرمال کرد. آنالیز مولفه‌های اصلی<sup>۱</sup> یا PCA توسط چندین گروه انجام شد (همچنین برنده هم از این روش برای ساخت ویژگی‌ها استفاده کرده بود). سه روش نرمال سازی شامل:

- $normalize_1$ : هر ویژگی جداگانه استاندارد شده (منهای میانگین تقسیم بر std به علاوه عامل fudge). زمانی که داده تنگ یا sparse است، دیگر میانگین کسر نمی‌شود.

- $normalize_2$ : هر مثال به نرم ۲ آن تقسیم می‌شود

- $normalize_3$ : هر مثال به میان نرم ۲های همه مثال‌ها تقسیم می‌شود.

بررسی کردن که  $normalize_2$  به نسبت بقیه بهتر عمل می‌کند.

## نتیجه‌گیری

این چالش نشان داد که انتخاب ویژگی می‌تواند به خوبی (بطور موثر) انجام شود و حذف ویژگی‌های بی معنی برای بدست آوردن عملکرد خوب در طبقه‌بندی واجب نیست. در طراحی دیتاست‌ها تعداد زیادی ویژگی‌های پرت، که آن‌ها را prob نامیدیم، آورده شده‌اند. در مقابل با ویژگی‌های اضافی که ممکن هم هست دقت را بهبود ندهند اما اطلاعات حمل می‌کنند، آن‌ها نیز

<sup>۱</sup>Principal Component Analysis

سفید<sup>۱</sup> هستند. این چالش توانست قدرت روش‌های فیلترینگ را نمایش دهد. از طرفی می‌گویند روش‌های تعبیه شده بسیار توانا هستند ولی از نظر محاسباتی پر هزینه هستند. نتیجه مهم دیگری که بدست آمد این بود که طبقه‌بند های غیر خطی لزوماً بیش‌برازی ندارند.

---

<sup>۱</sup> pure noise