
بسم الله الرحمن الرحیم

عنوان پروژه: استفاده از رویکردهای ترکیبی یادگیری ماشین برای تشخیص اختلال اوتیسم با استفاده از داده های FMRI

اساتید محترم راهنما: آقای دکتر حکیم داوودی ، خانم دکتر مالکی

دانشجو: سارا سلطانی گردفرامرزی

تاریخ ارائه پروژه: تیر ماه ۱۴۰۳

قدردانی و سپاس

ستایش مخصوص خدایی است که سلطان ملک وجود است. اوست که بی‌عوض می‌بخشد و چیزی نیست که در جهان پدید آید و از نظر عنایتش پنهان گردد. بینای امور عالم است و به لحظه لحظه اشیا و موجودات بصیر و آگاه است. اول و آخر و هرچه هست از اوست. پس چگونه می‌توان شکر نعمات بی‌شمارش را به جای آورد؟

از اساتید شایسته، جناب آقای دکتر حکیم داوودی و سرکار خانم دکتر مالکی که در کمال سعه صدر، با حسن خلق و فروتنی، از هیچ کمکی در این عرصه بر من دریغ ننمودند و زحمت راهنمایی این پروژه را به عهده گرفتند؛ به علاوه اینکه به من در اعطای این پروژه اعتماد کردند، تشکر می‌نمایم. باشد که این خردترین، بخشی از زحمات آنان را سپاس گوید.

فهرست

مقدمه

تشخیص اختلالات روانی ناهمگون مانند اختلال طیف اوتیسم (ASD) به خصوص در کودکان بسیار دشوار است. فرآیند تشخیص روانپزشکی کنونی صرفاً مبتنی بر مشاهده علائم رفتاری است و ممکن است مستعد تشخیص اشتباه باشد. برای اینکه زمینه را به سمت تشخیص کمی تر حرکت دهیم، به استفاده از داده های تصویر برداری مغز در کنار زیرساخت های یادگیری ماشینی پیشرفته و مقیاس پذیر نیاز داریم که به ما امکان می دهد نشانگرهای زیستی قابل اعتماد اختلالات سلامت روان را شناسایی کنیم.

در این گزارش یک مسئله طبقه بندی افراد مبتلا به اوتیسم از افراد سالم با استفاده از داده های rs_fmri از دیتاست ABIDE(Autism Brain Imaging Data Exchange) [۱]تعریف شده و رویکردهای یادگیری ماشین متفاوتی برای بهبود تشخیص این بیماری در افراد پیاده سازی و بررسی شده است. این رویکرد ها به تفصیل بررسی شده و سپس عملکرد آن روی این دیتاست بررسی می شود.

ادبیات موضوع

ابتدا به تعریف برخی از مفاهیم مهم می پردازیم.

داده های RS_FMRI

FMRI حالت استراحت یک روش تصویر برداری رزونانس مغناطیسی عملکردی FMRI است که در نقشه برداری مغز برای ارزیابی فعل و انفعالات ناحیه ای که در حالت استراحت یا حالتی که کار تعیین شده ای انجام نمی شود استفاده می شود.

وکسل

در داده‌های FMRI ، حجم مغز توسط گروهی از عناصر مکعبی کوچک به نام وکسل نشان داده می‌شود.

اتصال عملکردی

اتصال عملکردی در داده های fMRI به ارتباط آماری یا همبستگی زمانی بین رویدادهای عصبی فیزیولوژیکی از راه دور اشاره دارد، مانند سری زمانی سیگنال‌های BOLD (وابسته به سطح اکسیژن خون) که برای پی بردن به نحوه تعامل مناطق مختلف مغز با یکدیگر استفاده می‌شود.

اطلس

اطلس‌های موجود برای داده‌های rs_fMRI مجموعه‌ای از نقشه‌ها یا مدل‌های مختلف هستند که نواحی مختلف مغز را به دسته‌هایی تقسیم می‌کنند. این دسته‌بندی‌ها بر اساس ویژگی‌های مختلفی مانند وظایف عصبی، ساختار آناتومیکی یا الگوهای فعالیت مغزی انجام می‌شود. به هر کدام از این نواحی اصطلاحا ROI یا Region Of Interest گفته می شود.

اطلس AAL

این اطلس هر نمونه یا هر حجم مغز را به ۱۱۶ ROI تقسیم می کند.

اطلس TT

این اطلس هر نمونه یا هر حجم مغز را به ۱۶۱ ROI تقسیم می کند.

اطلس CC_۲۰۰

این اطلس هر نمونه یا هر حجم مغز را به ۲۰۰ ROI تقسیم می کند.[۲]

اطلس POWER_۲۶۴

این اطلس هر نمونه یا هر حجم مغز را به ۲۶۴ ROI تقسیم می کند.[٤] [۳]

اطلس CC_۴۰۰

این اطلس هر نمونه یا هر حجم مغز را به ۳۹۲ ناحیه مجزا تقسیم می کند.

استراتژی Filt_Global

این استراتژی برای پیش پردازش داده های FMRI استفاده شده و شامل دو پارامتر band_pass_filtering و global_signal_regression می باشد. در این استراتژی این دو پارامتر هر دو اعمال می شوند. پارامتر ابتدایی فرکانس های بی فایده را حذف کرده و عبور نمی دهد و پارامتر دومی سیگنال هایی با منشا غیر عصبی را حذف می کند.

استراتژی Filt_NoGlobal

این استراتژی نیز همانند استراتژی Filt_Global است با این تفاوت که صرفا band_pass_filtering اعمال می شود و global_signal_regression اعمال نمی شود.

خط لوله C_PAC

این بسته نیز برای پیش پردازش داده های FMRI استفاده می شود که مخفف Pipeline for Configurable the Analysis of Connectomes می باشد. این خط لوله طیفی از مراحل پیش پردازش را برای تمیز کردن و آماده سازی داده های fMRI برای تجزیه و تحلیل فراهم می کند .این شامل مراحلی مانند تصحیح زمان بندی برش، تصحیح حرکت، نرمال سازی فضایی و هموارسازی می شود[۵] .

کتابخانه nilearn

این کتابخانه یک ماژول پایتون برای تجزیه و تحلیل سریع و آسان آماری داده های تصویربرداری عصبی می باشد. این کتابخانه امکان بصری سازی داده های سه بعدی و ۴ بعدی ، تجزیه و تحلیل اتصال عملکردی میان مناطق مغز و استخراج سیگنال های عصبی از قسمت های مختلف و همچنین ارائه امکاناتی مانند پیش پردازش داده ها و استاندارد سازی سیگنال ها و اعمال فیلتر های مختلف را فراهم می کند.[٦]

بررسی دقیق تر دیتاست ABIDE

این دیتاست شامل داده های fmri از ۱۰۳۵ نفر یا سمپل از ۱۷ مرکز مختلف تصویربرداری مغز می باشد. لازم به ذکر است که هر مرکز از پارامترها و پروتکل‌های مختلفی برای اسکن داده‌ها استفاده کرده است. پارامترهایی مانند زمان تکرار(TR) ، زمان اکو(TE) ، تعداد وکسل‌ها، تعداد حجم‌ها، و باز یا بسته بودن چشم‌ها هنگام اسکن، بین مراکز مختلف تفاوت دارند. بنابراین لازم است پیش پردازش های مورد نیاز روی این دیتاست انجام شود تا آماده اعمال مدل ها روی آن باشد.

پیش پردازش روی دیتاست

مراحل پیش‌پردازش شامل اصلاح زمان برش، اصلاح حرکت، حذف سیگنال‌های مزاحم، تصحیح انحرافات فرکانس پایین، و نرمال‌سازی شدت وکسل‌ها (واحدهای حجمی کوچک در تصویر) است. برای انجام تمامی این مراحل باید

از خط لوله C_PAC و همچنین استراتژی Filt_Global هنگام دریافت این دیتاست از طریق تابع nilearn.datasets.fetch_abide_pcp از کتابخانه nilearn استفاده شود.

استراتژی دیگری مانند Filt_Noglobal نیز اعمال و نتایج حاصل از آن بررسی و با استراتژی Filt_Global مقایسه شده است. اما در نهایت استراتژی Filt_Global به دلیل نتایج بهتر انتخاب می شود. بعد از اعمال پیش پردازش های بیان شده در نهایت ۸۷۱ سمپل خواهیم داشت که ۴۰۳ نمونه مبتلا به اوتیسم و ۴۶۸ نمونه سالم هستند.

فصل اول: معرفی مدل اول[۷]

اعمال اطلس ها روی نمونه ها

مرحله بعدی استفاده از اطلس ها برای تقسیم بندی هر سمپل(حجم مغز) به یک تعداد ناحیه مشخصی می باشد. در این پیاده سازی ما از اطلس های CC_۲۰۰ و AAL و TT برای بررسی نتایج استفاده کرده ایم. این اطلس ها به ترتیب هر سمپل یا حجم مغز را به ۲۰۰ و ۱۱۶ و ۱۶۱ ناحیه مجزا تقسیم بندی و مپ می کند. برای اعمال اطلس های TT , AAL , CC_۲۰۰ روی دیتاست از تابع nilearn.datasets.fetch_abide_pcp استفاده می شود که یکی از آرگومان های ورودی آن derivative است. این آرگومان نوع اطلس اعمالی روی دیتاست را مشخص می کند. برای اعمال این سه اطلس مقادیر rois_cc۲۰۰ و rois_dosenbach۱۶۰ و rois_aal به این آرگومان ورودی داده می شود. [۸]

انتخاب ویژگی ها

گام اول انتخاب ویژگی

در ابتدا نیاز است ویژگی های مفید بعد از اعمال هر اطلس استخراج شود. از آنجایی که اتصال عملکردی (ارتباط بین نواحی مختلف مغز) حاوی الگوهای تبعیض آمیز برای طبقه بندی بهتر داده ها است می توان از همبستگی پیرسون برای تقریب این اتصال عملکردی استفاده کرد. این ماتریس همبستگی ارتباط خطی بین هر زوج ناحیه را نشان می دهد. اما با توجه به متقارن بودن این ماتریس و تکراری بودن نیمی از اطلاعات آن فقط قسمت مثلث بالایی آن جدا و استفاده می شود. سپس داده های روی قطر نیز حذف شده و مابقی داده ها فشرده شده و تبدیل به آرایه تک بعدی می شود. در این حالت با استفاده از اطلس های CC_۲۰۰ و AAL و TT و power۲۶۴ به ترتیب = ۲/ ۱۹۹ * ۲۰۰ ۱۹۹۰۰ و ۶۶۷۰ = ۲ / ۱۱۵ * ۱۱۶ و ۱۲۷۲۰ = ۲ / ۱۵۹ * ۱۶۰ و ۳۴۷۱۶ = ۲ / ۲۶۳ * ۲۶۴ویژگی خواهیم داشت.

با توجه به دو سری زمانی، u و v، هر کدام از طول T، همبستگی پیرسون را می توان با استفاده از معادله زیر محاسبه کرد:

$$\rho_{uv} = \frac{\sum_{t=1}^{T}(u_t - \bar{u})(v_t - \bar{v})}{\sqrt{\sum_{t=1}^{T}(u_t - \bar{u})^2}\sqrt{\sum_{t=1}^{T}(v_t - \bar{v})^2}}$$

$\bar{v}$ و $\bar{u}$ به ترتیب میانگین سری های زمانی u و v هستند.

گام دوم انتخاب ویژگی

حال بعد از اعمال این مرحله همچنان تعداد ویژگی ها زیاد بوده (به عنوان مثال ۱۹۹۰۰ ویژگی در یکی از اطلس ها) در نتیجه احتمال بیش برازش وجود دارد. پس لازم است تعدادی دیگر از ویژگی ها را کاهش داد. در این روش در جهت اینکه همبستگی های اصلی و مهم و متمایزکننده از دست نرود برای هر ویژگی میانگین همبستگی ها را برای همه نمونه های متناظر (۸۷۱ نمونه) محاسبه کرده. پس برای هر ۱۹۹۰۰ ویژگی یک مقدار میانگینی داریم که محاسبه شده است. حال این ۱۹۹۰۰ مقدار میانگین را مرتب کرده و اندیس های ¼ بزرگترین مقادیر و ¼ کوچکترین مقادیر را مشخص و انتخاب کرده. در واقع اندیس های بزرگترین مقادیر مثبت و منفی به دست می آید. حال از هر وکتور ویژگی برای هر سمپل صرفا این اندیس های به دست آمده انتخاب شده و به عنوان وکتور نهایی ویژگی هر سمپل در نظر گرفته می شود. پس با این اوصاف تعداد ویژگی ها نیز مجدد نصف شده و به ۹۹۵۰ می رسد و با توجه به اینکه بزرگترین مقادیر مثبت و منفی انتخاب شده است در واقع مهم ترین اتصالات عملکردی حفظ شده و مابقی

که اهمیت کمتری داشتند دور ریخته شده است. در اطلس های AAL نیز در نهایت منجر به حفظ ۳۳۳۵ و در اطلس TT منجر به حفظ ۶۳۶۰ ویژگی و در اطلس power۲٦٤ منجر به حفظ ۱۷۳۵۸ ویژگی می شود.

گام سوم انتخاب ویژگی

اما همچنان ویژگی ها زیاد است. به منظور کاهش بیشتر اندازه ویژگی‌ها، از یک انکودر خودکار برای استخراج ویژگی ها با ابعاد پایین‌تر استفاده می کنیم .انکودر خودکار نوعی مدل شبکه عصبی با ساختار پیش رو (feed-forward) است که ورودی x را به ابعاد پایین‌تر انکود می‌کند.

$$h_{enc} = \phi_{enc}(x) = \tau(W_{enc}\, x + b_{enc})$$

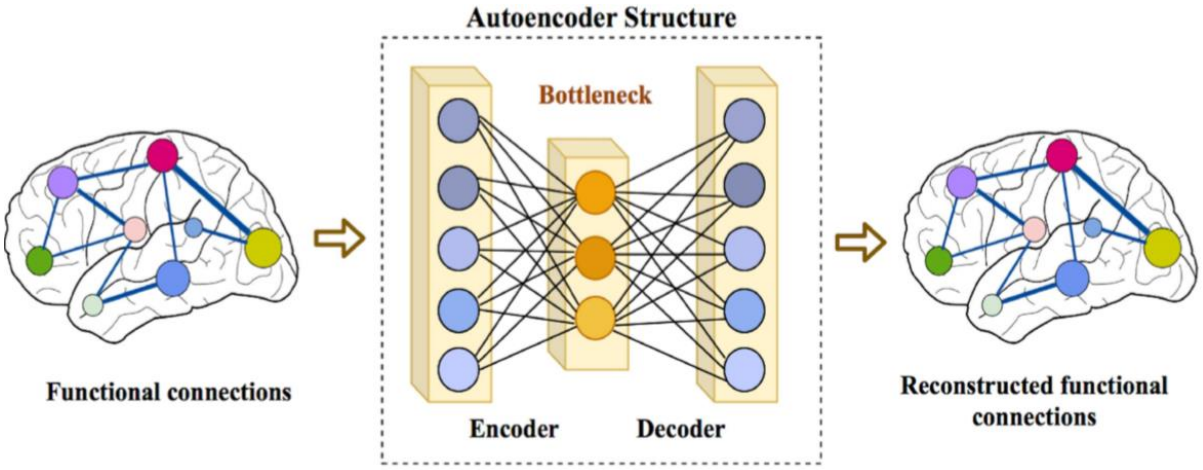
W_{enc} ماتریس وزن ها و b_{enc} بایاس انکودر است. تاو نیز همان تابع فعالسازی مماس هایپربولیک انکودر میباشد که خروجی آن در محدوده -۱ تا ۱ است. مجددا یک دیکودر لازم است که سیگنال اصلی x را بسازد. به این شکل:

$$\acute{x} = \phi_{dec}(h_{enc}) = W_{dec}\, h_{enc} + b_{dec}$$

که مجددا وزن ها و بایاس دیکودر را داریم. در این پیاده سازی وزن های انکودر خودکار گره خورده می باشد به این معنی که این رابطه برقرار است:

$$W_{dec} = W_{enc}^{T}$$

انکودر خودکار را می توان آموزش داد تا خطای بازسازی خود را به حداقل برساند، که به عنوان (MSE) بین x و بازسازی شده آن، x′ محاسبه می شود. دلیل انتخاب استفاده از انکودر خودکار به جای سایر تکنیک‌های استخراج ویژگی مانند PCA، توانایی آن در کاهش ابعاد ویژگی‌ها به روش غیرخطی است. داده‌های ابعاد پایین‌تر تولید شده در طول فرآیند انکودر حاوی الگوهای مفیدی از داده‌های ورودی اصلی با اندازه کوچک‌تر هستند و می‌توانند به عنوان ویژگی‌های جدید برای طبقه‌بندی استفاده شوند.



شکل ۱ـشمایی کلی از کاهش بعد در مرحله سوم

آموزش مدل

حال برای بخش طبقه بندی از یک پرسپترون تک لایه استفاده میشود که از لایه گلوگاه انکودر خودکار یعنی خروجی آن (h_{enc}) به عنوان ورودی استفاده میکند و احتمال تعلق یک داده به کلاس بیماران مبتلا به اوتیسم را با استفاده از یک تابع فعالسازی (sigmoid) بدست می آورد. W_{slp} و b_{slp} به ترتیب وزن ها و بایاس آن هستند. پس $f(x)$ احتمال تعلق هر نمونه به کلاس مبتلایان به اوتیسم است. رابطه این پرستپترون به شرح زیر است:

$$f(x) = \sigma\,(W_{slp}\, h_{enc} + b_{slp})$$

h_{enc} را می توان با رابطه شماره ۱ به دست آمده جایگزین کرده:

$$f(x) = \sigma\,(W_{slp}\, \tau(W_{enc}\, x + b_{enc}) + b_{slp})$$

مدل پرسپترون به گونه ای آموزش داده می شود که خطای کراس آنتروپی را مینیمم کند. در محاسبه این خطا برچسب اصلی داده ها (y) و احتمال تعلق هر نمونه به کلاس مبتلایان به اوتیسم ($f(x)$) می باشد. محاسبه خطای کراس آنتروپی به شرح زیر است:

$$H(y, f(x)) = -(y * f(x) + (1 - y) * (1 - f(x)))$$

در نهایت نیز کلاس متناظر با هر نمونه با توجه به سطح آستانه ۰.۵ برای احتمال تعلق هر نمونه به کلاس بیماران مبتلا به اوتیسم مشخص میشود. به این شکل که اگر احتمال حاصل از پرسپترون بزرگتر یا مساوی ۰.۵ باشد نمونه ، مبتلا به اوتیسم و اگر کوچکتر از ۰.۵ باشد نمونه ، سالم برچسب گذاری می شود. این رابطه به شرح زیر است:

$$\hat{y} = \begin{cases} 1, & \text{if } f(x) \geq 0.5 \\ 0, & \text{otherwise} \end{cases}$$

به طور معمول، انکودر خودکار به طور کامل آموزش داده می شود که خطای بازسازی آن به حداقل برسد. سپس، ویژگی های لایه گلوگاه یا خروجی آن به عنوان ورودی برای آموزش لایه پرستپترون ، به طور جداگانه استفاده می شود. اما در این روش پیاه سازی شده ، انکودر خودکار و پرسپترون تک لایه را به طور همزمان آموزش می‌دهیم. این به طور بالقوه می تواند منجر به به دست آوردن ویژگی هایی با ابعاد کم شود که هم مفید برای بازسازی داده های اصلی است و هم حاوی اطلاعات متمایز کننده برای کار طبقه بندی می باشد. پس در هر مرحله آموزش نتایج حاصل از دو تابع خطای متفاوت (میانگین مربعات خطا برای خطای بازسازی در جهت آموزش انکودر و خطای متقابل آنتروپی برای خطای حاصل از طبقه بندی هر نمونه در جهت آموزش پرسپترون تک لایه) محاسبه و با یکدیگر جمع می گردند تا هر دو مدل همزمان با یکدیگر آموزش داده شوند. بعد از آموزش مشترک هر دو مدل پرسپترون را طی ایپاک های دیگری آموزش می دهیم تا به درستی تنظیم شود.

افزایش تعداد داده با استفاده از درونیابی خطی

هنگام کار کردن با مدل های یادگیری ماشین یا یادگیری عمیق باید تعداد مناسبی داده داشته باشیم تا دچار بیش برازش نشویم. از طرف دیگر مجموعه های آموزشی بزرگ همیشه در دسترس نیستند و جمع آوری داده های جدید ممکن است مانند حوزه تصویربرداری پزشکی پرهزینه باشد. پس میتوان از تکنیک های افزایش داده با هوش مصنوعی استفاده کرد.

روشی که در این پیاده سازی برای افزایش داده بکار رفته است SMOTE است.[۹] این روش داده های مصنوعی را در فضای ویژگی با استفاده از نزدیکترین همسایگان نمونه تولید می کند. این روش برای هر نمونه تمامی نمونه هایی که از کلاس مشترک آن هستند را بررسی کرده و ۵ تا از نزدیکترین نمونه های به آن نمونه را به عنوان همسایگان آن نمونه در نظر میگیرد. پس از یافتن ۵ تا از نزدیکترین همسایه‌های نمونه مانند p ({q۱,q۲,...,q۵})، یک همسایه تصادفی انتخاب می‌شود (q_r) و بردار ویژگی سمپل مصنوعی با استفاده از معادله زیر به صورت خطی محاسبه می‌شود: (در این معادله، α یک عدد تصادفی است که به طور یکنواخت در محدوده [۰،۱] انتخاب شده است.)

$$p' = \alpha \times p + (۱ - \alpha) \times q_r$$

در نتیجه برای هر سمپل یک سمپل مصنوعی با همان برچسب نمونه منتخب تولید می شود. پس تعداد داده ها در نهایت دو برابر می شود. نتایج حاصل از تعداد همسایه های دیگری به غیر از ۵ تا همسایه نیز در نظر گرفته شد اما تعداد ۵ تا کارآمد تر بود هم از نظر راندمان محاسبات و هم از نظر معیار های ارزیابی مدل.

اما دیتاست ما دارای بردارهای ویژگی با اندازه ۹۹۵۰ می باشد. محاسبه فاصله های اقلیدسی با ۹۹۵۰ ویژگی کارآمد نیست. پس برای محاسبه فاصله ها از معیاری به نام هنجار فروبنیوس توسعه یافته (EROS) استفاده کردیم. این اندازه گیری شباهت بین دو سری زمانی چند متغیره (MTS) را محاسبه می کند. از طرفی داده های fMRI از چندین ناحیه تشکیل شده است که هر کدام دارای یک سری زمانی هستند، بنابراین می توانیم آن را به عنوان یک سری زمانی چند متغیره در نظر بگیریم. بنابراین این روش برای داده های ما مفید است.

EROS شباهت بین دو سری زمانی چند متغیره (A , B) را بر اساس مقادیر ویژه و بردارهای ویژه ماتریس های کوواریانس تمامی سمپل ها با استفاده از معادله زیر محاسبه می کند:

$$\begin{aligned} \text{EROS}\ (A, B, w) &= \sum_{i=1}^{n} w_i |\langle a_i, b_i \rangle| \\ &= \sum_{i=1}^{n} w_i |\cos\ \theta_i| \end{aligned}$$

θ_i زاویه بین بردارهای ویژه مربوط به ماتریس های کوواریانس سری های زمانی چند متغیره A و B است.

ازدیدگاه عملی *EROS* با این روند پیاده سازی شده است:

```
۱: for i = ۱ to N do
    ۲: si ← si / sum_j sij
۳: end for
٤: for i = ۱ to n do
    ٥: wi ← f(s∗i)
٦: end for
۷: for i = ۱ to n do
    ۸: wi ← wi / sum_j wj
۹: end for
```

در این سودوکد ابتدا بردار وزن های *W* محاسبه می شود به این شکل:

فرض شده که ماتریس *S* با ابعاد *N* ∗ *n* نشان دهنده مقادیر ویژه تمامی آیتم های چند متغیره است. در واقع هر ستون (*si*) نشان دهنده مقادیر ویژه آیتم *i* ام است. پس sij نشان دهنده مقدار این ماتریس در i امین ستون و j امین سطر است. s*i اما i امین سطر از این ماتریس و si* از طرف دیگر i امین ستون در این ماتریس است.

ابتدا لازم است هر ستون از این ماتریس نرمالایز شود به شکلی که هر مقداری به مجموع تمامی این مقادیر موجود در ستون تقسیم شود. در نهایت مجموع مقادیر نرمالایز شده در هر ستون معادل یک می شود. پس به این شکل مقادیر ویژه نرمالایز شده برای هر آیتم محاسبه و نگهداری می شود.

در مرحله بعدی لازم است روی مقادیر هر سطر که متناظر با هر ویژگی است تابع تجمیع شده ای (که در این پیاده سازی میانگین در نظر گرفته شده است) اعمال شود. پس با میانگین گیری روی تمامی مقادیر هر سطر i*s المان i ام از بردار وزن ها به دست می آید. پس در واقع المان i ام این بردار وزن ها معادل میانگین مقادیر ویژه نرمالایز شده روی تمامی آیتم ها متناظر با متغیر i ام است. بعد از اتمام حلقه فور دومی بردار وزن ها به دست آمده. حال مجدد باید این بردار را نیز نرمالایز کرده و مقدار هر المان را تقسیم بر مجموع تمامی المان ها کنیم.

حال که بردار وزن ها به دست آمد می توان برای محاسبه شباهت میان دو سمپل از آن استفاده کرد. در اینجا باید از بردار های ویژه سمپل ها استفاده کرد. در این پیاده سازی ما برای کاهش زمان اجرا و بازدهی بیشتر ، صرفا از دو بردار ویژگی از هر سمپل استفاده کردیم. پس ابتدا متناسب با مقادیر ویژه بردار های ویژگی هر نمونه را مرتب کردیم. حال برای هر نمونه دو تا اولین بردار ویژگی را انتخاب کرده و با یکدیگر ضرب داخلی میکنیم. سپس حاصل به دست آمده را در بردار وزن های نرمالایز شده که براساس مقادیر ویژه به دست آمده بود ضرب می کنیم. پس به این شکل ضرب داخلی وزن دار دو بردار ویژگی هر دو نمونه به عنوان شباهت این دو نمونه به یکدیگر در نظر گرفته می شود. براساس این شباهت ۵ تا از نزدیکترین همسایگان به هر نمونه مشخص می شود که بعدا می توان برای ساختن نمونه جدید یکی را به صورت رندوم انتخاب کرد.

جداسازی داده های آموزشی و آزمایشی

برای اینکه می خواهیم آموزش روی کل دیتاست انجام گیرد و در واقع تمامی اطلاعات ارزشمند داخل دیتاست برای آموزش مدل استفاده شود از ۱۰ fold cross validation استفاده می شود تا هر بار ۷۸۳ داده برای آموزش و ۸۸ نمونه برای ارزیابی مدل و به عنوان مجموعه تست استفاده شود. در نهایت نیز بعد از ۱۰ بار اجرای کامل نتایج به دست آمده میانگین گیری می شود و به عنوان نتایج نهایی اعلام شده است. اگر از این روش استفاده نشود ممکن است تمامی اطلاعات ارزشمند و نمونه های مفید داخل مجموعه تست قرار گیرد و برای آموزش مدل استفاده نشود. در نتیجه منجر به زیر برازش شود.

لازم به ذکر است که برای بررسی واقعی تر تاثیر افزایش داده روی مدل صرفا لازم است افزایش داده روی داده های آموزشی صورت گیرد. پس با پیاده سازی تکنیک افزایش داده در هر مرحله تعداد داده های آموزشی دو برابر می شود. با توجه به اینکه هر بار ۱۰٪ داده ها مربوط به مجموعه آزمایش است پس ۷۸۴ نمونه برای مجموعه آموزشی خواهیم داشت. از طرف دیگر به ازای هر نمونه یک نمونه مصنوعی تولید می شود. پس در هر مرحله ۱۵۶۸ داده برای فرآیند آموزش بعد از افزایش داده خواهیم داشت.

بیس لاین ها

برای اینکه مدل پیاده سازی شده خود را مورد ارزیابی قرار دهیم بیس لاین هایی مانند ماشین بردار پشتیبان و جنگل تصادفی (پیاده سازی شده با استفاده از کتابخانه scikit-learn) و همچنین نتایج حاصل از یک مقاله دیگر روی همین دیتاست و با هدف مشترک را در نظر گرفته ایم[۱۰] . مدل های ماشین بردار پشتیبان و جنگل تصادفی روی دیتاست پیش پردازش شده (پیش پردازش برای این دو بیس لاین و مدل پیشنهادی ما دقیقا یکسان و به شکل شرح داده شده است) اجرا شده اند اما با توجه به اینکه ابعاد ویژگی کاهش داده نشده است این دو بیس لاین روی دیتاست با تعداد ویژگی ۱۹۹۰۰ اجرا می شوند. بهترین مقدار برای پارامترهای این دو مدل(مقادیر C و gamma و نوع کرنل برای ماشین بردار پشتیبان و تعداد درخت در جنگل های تصادفی) نیز با استفاده از روش Grid Search به دست آمده است.

معیار های ارزیابی مدل

معیارهایی که توسط آن عملکرد مدل خود را می سنجیم *accuracy* و *sensitivity* و *specificity* می باشد.

Accuracy یا دقت نشان دهنده تعداد سمپل هایی است که در کل به درستی طبقه بندی شده اند. یعنی مبتلایان به اوتیسم که به درستی بیمار تشخیص داده شده اند و افراد سالمی که به درستی سالم تشخیص داده شده اند.

Sensitivity یا حساسیت نشان دهنده تعداد نمونه هایی که واقعا مبتلا به اوتیسم بوده و مدل نیز آن ها را بیمار تشخیص داده است. *specificity* یا شفافیت نیز نشان دهنده نمونه هایی است که در واقع سالم بوه اند و مدل نیز به درستی سالم تشخیص داده است.

sensitivity = TP / (TP + FN)

$$specificity = TN / (TN + FP)$$

$$accuracy = (TP + TN) / (TP + TN + FN + FP)$$

از طرف دیگر برای بررسی کارایی مدل به محاسبه و بررسی مقدار *AUC* مدل که نشان دهنده سطح زیر منحنی *ROC* مدل است پرداختیم. برای رسم منحنی *ROC* به محاسبه دو مولفه *True Positive Rate* و *False Positive Rate* به ترتیب برای محور عمودی و افقی نیازمندیم. این دو مقدار از طریق روابط زیر محاسبه می شوند:

TPR = TP / TP + FN = SENSITIVITY

FPR = FP / FP + TN = ۱ - SPECIFICITY

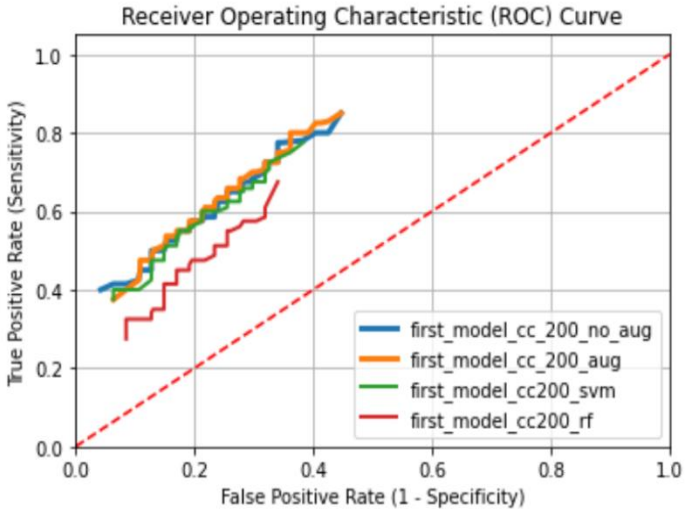
منحنی ROC هر چقدر از خط وسط دورتر باشد و به عبارتی سطح زیر منحنی بیشتر باشد نشان دهنده کارایی بالاتر و نزدیک به ۱ آن است.

نتایج نهایی

در این قسمت نتایج نهایی حاصل از اجرای مدل پیاده سازی شده و بیس لاین ها روی سه اطلس به تفصیل توضیح داده شده است.

نتایج نهایی روی اطلس CC_۲۰۰

نتایج بصری و عددی حاصل از اجرای مدل های روی CC_۲۰۰ به شرح زیر می باشد. همانطور که از شکل نیز مشخص است منحنی نارنجی که مربوط به اجرای مدل با افزایش داده است بالاتر از مابقی بوده. هر چند همپوشانی زیادی با مدل آبی رنگ (که حاصل از اجرای مدل بدون افزایش داده است) دارد. مدل ماشین بردار پشتیبان معادل با منحنی سبز رنگ کمی پایین تر از دو مدل دیگر است. اما مدل جنگل های تصادفی کارایی خیلی پایینی روی این مدل دارد.



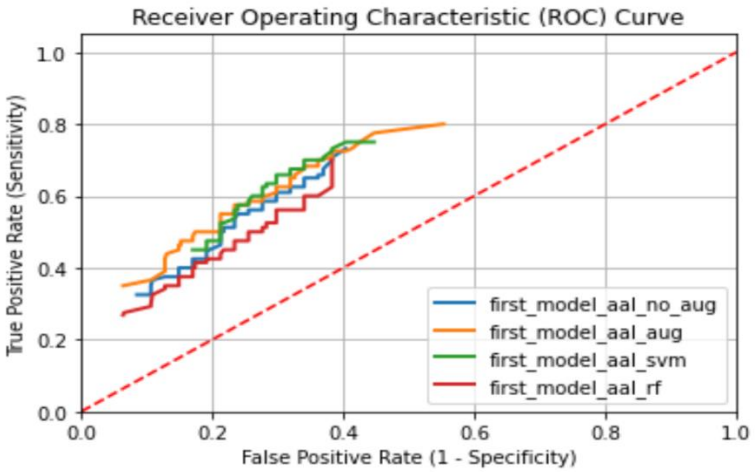
شکل ۲_منحنی ROC مدل به همراه بیس لاین ها روی CC_200

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
مدل پیاده سازی شده با افزایش داده	۷۰.۳	۶۸	**۷۳**
مدل پیاده سازی شده بدون افزایش داده	۶۹.۲	۶۹	**۷۰**
ماشین بردار پشتیبان	۶۸.۲	۶۳	**۷۲.۵**
جنگل های تصادفی	۶۲	۶۰	**۷۱.۵**

جدول ۱_نتایج نهایی مدل اول روی CC_200

نتایج نهایی روی اطلس AAL

نتایج بصری و عددی حاصل از اجرای مدل ها روی اطلس aal نیز به شرح زیر می باشد. همانطور که از نتایج نیز مشخص است در اطلس aal مدل های ماشین بردار پشتیبان و مدل پیاده سازی شده با افزایش داده عملکرد بسیار مشابهی دارند و منحنی هایشان نیز با یکدیگر هم پوشانی زیادی دارد و البته هر دو منحنی بالاتر از مابقی منحنی ها قرار گرفته اند. اما مدل بدون افزایش داده کمی پایین تر از این دو قرار گرفته و از طرف دیگر مدل جنگل های تصادفی نیز مانند اطلس CC_۲۰۰ بدترین عملکرد را داشته است.



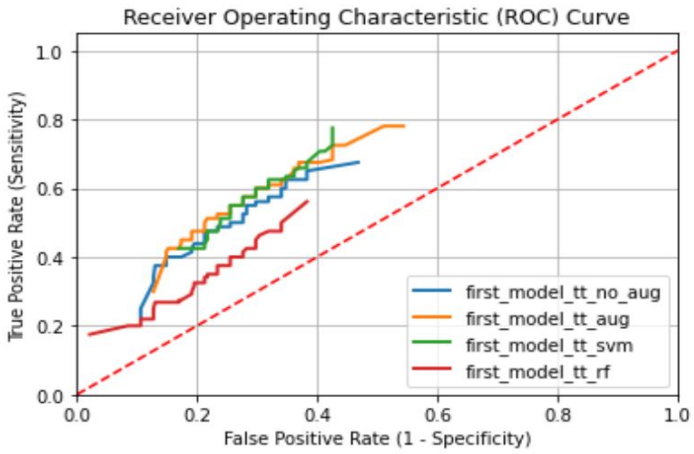
شکل ۳_منحنی ROC مدل به همراه بیس لاین ها روی اطلس AAL

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
مدل پیاده سازی شده با افزایش داده	۶۷	۶۳.۴	**۷۳**
مدل پیاده سازی شده بدون افزایش داده	۶۴.۵	۶۰	**۶۹.۵**
ماشین بردار پشتیبان	۶۷	۶۳.۹	**۷۲**
جنگل های تصادفی	۶۳	۵۳	**۷۰.۵**

جدول ۲_نتایج نهایی مدل اول روی اطلس AAL

نتایج نهایی روی اطلس TT

نتایج بصری و عددی حاصل از اجرای مدل ها روی اطلس TT نیز به شرح زیر می باشد. همانطور که از نتایج نیز مشخص است در این اطلس هر چقدر به سمت نقطه (۱ و ۱) پیش می رویم منحنی سبز رنگ کمی بالاتر از نارنجی و نارنجی کمی بالاتر از آبی است. پس می توان نتیجه گرفت مدل ماشین بردار پشتیبان عملکرد بهتری نسبت به مدل پیاده سازی شده با افزایش داده دارد و بعد آز این دو مدل بدون افزایش داده عملکرد بهتری داشته و جنگل تصادفی نیز مانند مابقی اطلس ها بدترین عملکرد را داشته است.



شکل ۴_منحنی ROC مدل اول به همراه بیس لاین ها روی اطلس TT

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
مدل پیاده سازی شده با افزایش داده	۶۵	۶۳.۵	**۶۸**
مدل پیاده سازی شده بدون افزایش داده	۶۴.۵	۶۱	**۶۹**
ماشین بردار پشتیبان	۶۶	۶۲	**۷۲**
جنگل های تصادفی	۶۰.۲	۵۲	**۷۱**

جدول ۳_نتایج نهایی مدل اول روی اطلس TT

با توجه به نتایج فوق و مقایسه آن ها با یکدیگر اطلس CC_۲۰۰ بهترین نتیجه را روی مدل پیاده سازی شده ما با افزیش داده با بالاترین مقادیر برای سه معیار مورد نظر داشته است. با توجه به اینکه این اطلس مغز را به تعداد نواحی بیشتری تقسیم می کند پس اطلاعات متمایز کننده و مهم تری را در اختیار ما قرار می دهد و در واقع مغز را ریز تر و جزئی تر کرده و الگوهای مهم را آشکار می کند در نتیجه نتایج بهتری در پی خواهد داشت.

نتایج نهایی با به کار گیری استراتژی Filt_NoGlobal

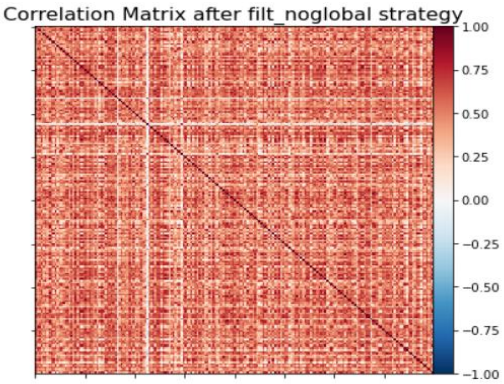
در این مرحله لازم است بهترین مدل به دست آمده تا اینجای کار روی داده های اولیه که با استراتژی Filt_Noglobal پیش پردازش شده بودند اعمال شود و نتایج نهایی حاصل از آن بررسی و ارزیابی شوند. با توجه به اینکه استراتژی Filt_Global سیگنال های اضافی را حذف می کند ممکن است برخی سیگنال های مفید را به اشتباه حذف کرده باشد. در نتیجه لازم است تا این استراتژی بدون حذف سیگنال ها نیز بررسی شود. نتایج حاصل از این استراتژی در بهترین حالت یعنی اعمال مدل پیاده سازی با افزایش داده به صورت زیر است:

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
مدل پیاده سازی شده با افزایش داده	۶۶.۷	۶۱.۹	**۷۰.۹**
مدل پیاده سازی شده بدون افزایش داده	۶۵	۶۰	**۷۱.۶**

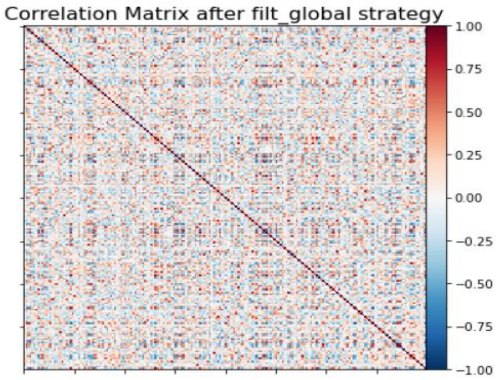
جدول ۴_نتایج نهایی مدل اول روی اطلس CC_200 با استراتژی FILT_NOGLOBAL

همانطور که از نتایج نیز مشخص است این استراتژی اصلا نتایج خوبی نداشته و در ادامه کار نیز از استراتژی Filt_Global باید استفاده شود. در زیر به شرح دلیل ناکارا بودن این استراتژی می پردازیم.

در شکل زیر نتایج این دو استراتژی را روی یکی از سمپل ها مشاهده می کنید:



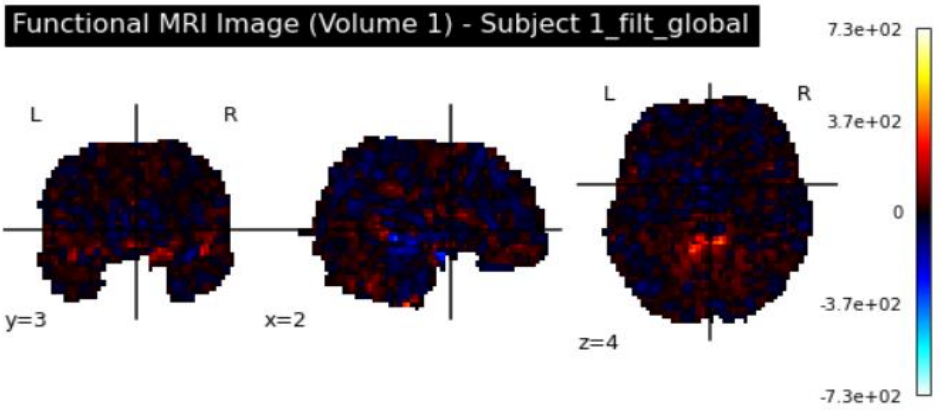
شکل ۵_ماتریس همبستگی نمونه با اعمال استراتژی FILT_NOGLOBAL



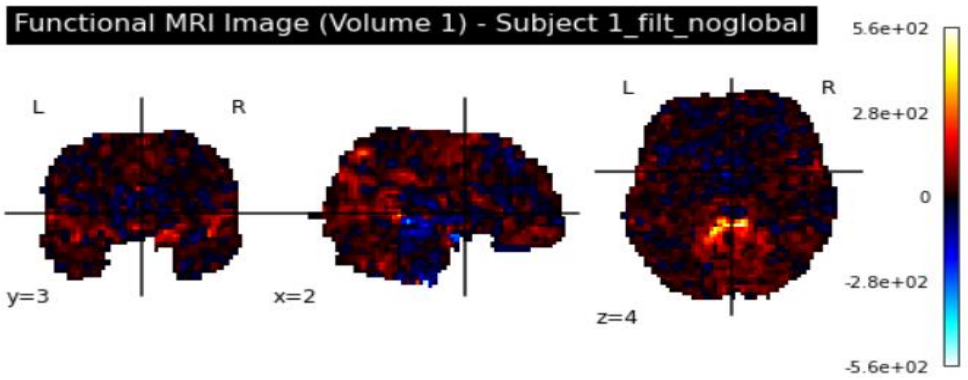
شکل ۶_ماتریس همبستگی نمونه با اعمال استراتژی FILT_GLOBAL

تفاوت این دو استراتژی در مقدار global_signal_regression است. هدف از این متغیر کاهش تأثیر نوسانات کلی در داده ها است که منشأ عصبی ندارند، مانند رانش اسکنر، نویز فیزیولوژیکی (ضربان قلب، تنفس، و مصنوعات حرکتی و همچنین اعوجاج میدان مغناطیسی) .پس نتیجه حذف نویز و نوسانات، افزایش حساسیت تشخیص فعالیت عصبی خاص منطقه و بهبود تفسیرپذیری داده‌های fMRI است.

همانطور که در اشکال بالا نیز مشخص است این نوسانات در سمپل ها زیاد بوده که با حذف آن ها می توان به نتیجه بهتری رسید. در ماتریس همبستگی سمت راست نویز باعث به وجود آمدن همبستگی های غیر واقعی شده .



شکل ۷_شمایی از حجم مغز نمونه با اعمال استراتژی FILT_GLOBAL



شکل ۸_شمایی از حجم مغز نمونه با اعمال استراتژی FILT_NOGLOBAL

برای مقایسه بهتر نیز نتیجه حاصل از دو استراتژی روی یکی از سمپل ها بصری سازی شده است. رنگ ها در این تصاویر نشان دهنده شدت مقادیر در داده های هر سمپل است. رنگ های گرم نشان دهنده شدت بالاتر و رنگ های سرد نشان دهنده شدت کمتر مقادیر هستند. این مقادیر بازتاب کننده سطح اکسیژن خون ، جریان خون مغزی ، حجم خون مغزی ، تغییرات متابولیک و سطوح انتقال دهنده عصبی و همچنین نویز می باشد. با توجه به اینکه در تصویر پایینی که مرتبط با استراتژی filt_noglobal است رنگ های گرم مانند زرد و قرمز بیشتر به چشم می آید گواهی بر وجود نویز و نوسانات بیشتر در این استراتژی است.

در نظر گرفتن اطلاعات جانبی

در کنار داده های مغزی که از افراد جمع آوری شده است این دیتاست دارای ۲۸ ویژگی جانبی افراد مانند سن افراد ، بیماری پیش زمینه ای ، داروهای مصرفی ، جنسیت و همچنین چپ دست بودن یا راست دست بودن و برخی ویژگی های اسکنر و غیره می باشد. در این بخش سعی کردیم با اضافه کردن اطلاعات جانبی به داده های مختص مغز متریک های ارزیابی مدل را بالا ببریم. در اینجا نیز بهترین مدل به دست آمده از گام های قبلی به همراه اطلاعات جانبی در نظر گرفته شده است.

پیش پردازش اطلاعات جانبی

ابتدا لازم بود پیش پردازش روی این داده ها انجام شود. در این گام ویژگی هایی با تعداد بیش از حدودا ۳۰۰ مقدار Null و همچنین دارای مقادیر یکسان برای تمامی نمونه ها و همچنین ویژگی هایی با تعداد ۸۷۱ مقدار یکتا (دارنده یک مقدار متمایز به ازای هر نمونه) حذف شدند و در نهایت ۱۵ ویژگی مفید باقی ماندند. این ویژگی ها به همراه تعداد مقادیر Null هر کدام در لیست زیر مشخص شده اند:

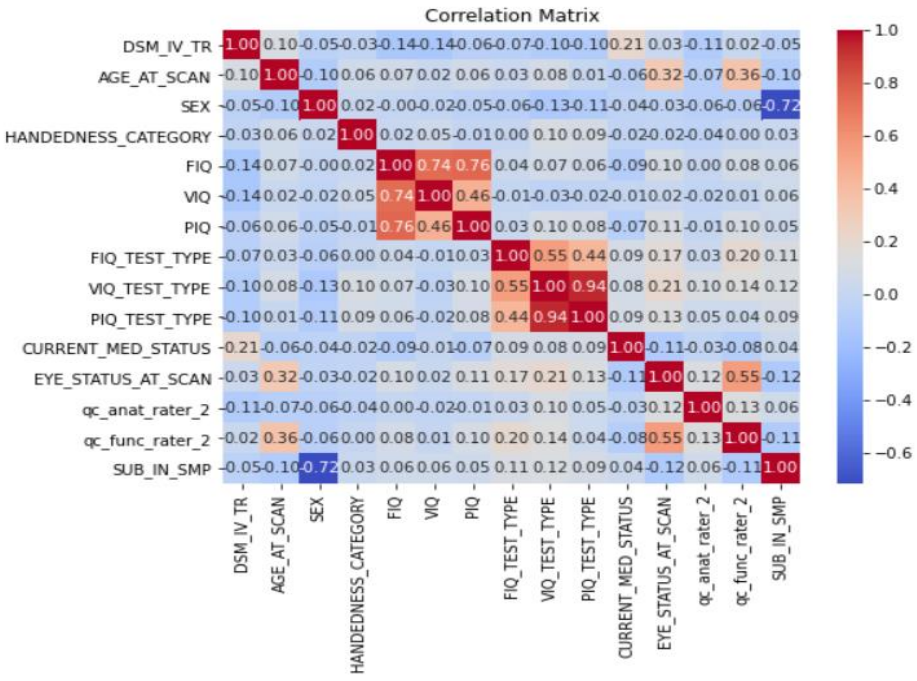
DSM_IV_TR	0
AGE_AT_SCAN	0
SEX	0
HANDEDNESS_CATEGORY	265
FIQ	28
VIQ	133
PIQ	115
FIQ_TEST_TYPE	137
VIQ_TEST_TYPE	221
PIQ_TEST_TYPE	202
CURRENT_MED_STATUS	204
EYE_STATUS_AT_SCAN	0
qc_anat_rater_2	0
qc_func_rater_2	0
SUB_IN_SMP	0

شکل ۹_۱۵ ویژگی مفید جانبی

مقادیر Null در ویژگی های , FIQ_TEST_TYPE , VIQ_TEST_TYPE ,
HANDEDNESS_CATEGORY PIQ_TEST_TYPE , CURRENT_MED_STATUS
با بیشترین مقادیر این ویژگی ها جاگذاری شده. سپس مقادیر Null در ویژگی های PIQ , VIQ , FIQ نیز با میانگین تمامی مقادیر عددی این ویژگی ها جاگذاری شدند.

در گام بعدی با استفاده از Label Encoder ویژگی های طبقه بندی شده به ویژگی های عددی تبدیل شدند.

در این گام لازم است ویژگی هایی که بنظر مفیدتر هستند از ویژگی های کم اهمیت جدا شده و نتایج روی هر دو دسته به صورت جداگانه به دست آید. برای این جداسازی همبستگی میان ویژگی ها ملاک قرار داده شد. در شکل زیر ماتریس همبستگی این ۱۵ ویژگی مشخص شده است:



شکل ۱۰_ماتریس همبستگی ۱۵ ویژگی باقی مانده

ویژگی هایی که همبستگی کمی با یکدیگر دارند شامل , DSM_IV_TR , AGE_AT_SCAN
HANDEDNESS_CATEGORY ,CURRENT_MED_STATUS ,qc_anat_rater_۲ می باشد که در دسته اول قرار گرفته.

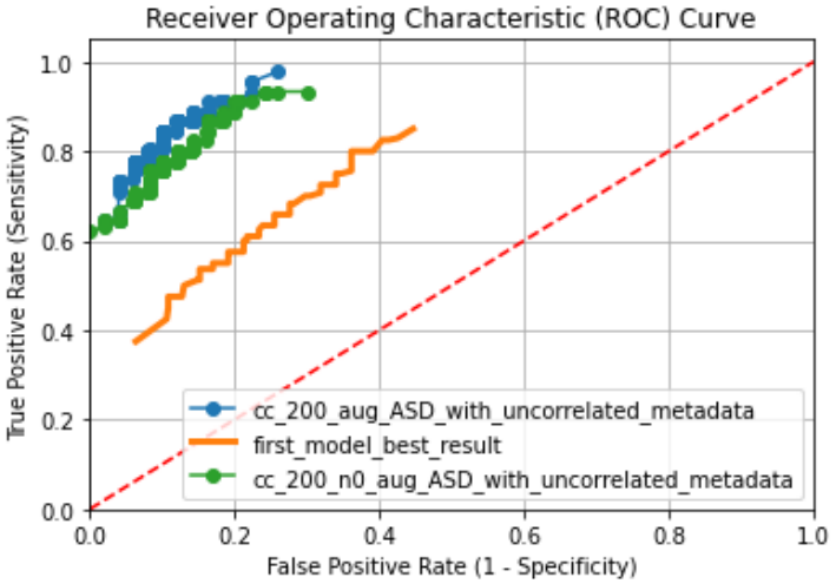
ویژگی هایی که همبستگی بیشتری با یکدیگر دارند شامل SEX ,FIQ , VIQ , PIQ , FIQ_TEST_TYPE ,VIQ_TEST_TYPE , PIQ_TEST_TYPE , SUB_IN_SMP , qc_func_rater_۲ ,EYE_STATUS_AT_SCAN هستند که در دسته دوم قرار می گیرند.

نتایج نهایی با در نظر گرفتن اطلاعات جانبی

نتایج حاصل از اعمال مدل بهترین مدل تا اینجای کار که روی اطلس CC_۲۰۰ است به صورت زیر می باشد:

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
مدل پیاده سازی شده با افزایش داده با اطلاعات جانبی دسته اول	۸۵.۹	۸۳	**۸۸.۵**
مدل پیاده سازی شده بدون افزایش داده با اطلاعات جانبی دسته اول	۸۳.۲	۷۸.۵	**۸۷.۶**
مدل پیاده سازی شده بدون افزایش داده با اطلاعات جانبی دسته دوم	۶۳.۱	۵۹	**۶۷**

جدول ۵_نتایج نهایی حاصل از اعمال بهترین مدل در کنار اطلاعات جانبی



شکل ۱۱_مقایسه مدل با اطلاعات جانبی و بدون آن

همانطور که از نتایج به دست آمده نیز مشخص است وقتی از اطلاعات جانبی ای که همبستگی زیادی با یکدیگر دارند استفاده می کنیم کارایی و همچنین دقت مدل پایین تر از حتی حالت بدون در نظر گرفتن اطلاعات جانبی می شود و تاثیر عکس دارد. چون می دانیم که اگر ویژگی ها بسیار به یکدیگر وابسته باشند چون تکراری هستند مدل را گیج و سردرگم می کند و نمی تواند به خوبی روی داده ها عمل کند. اما از طرف دیگر وقتی اطلاعات جانبی ای که همبستگی کمی با یکدیگر داشتند استفاده شدند دقت و همچنین کارایی مدل بسیار بهبود یافت و با افزایش داده نیز این کارایی و دقت در حد چندین درصد افزایش یافتند.

اما می دانیم که همواره اطلاعات جانبی افراد در دسترس ما نیست و لازم است مدل ها به گونه ای طراحی شوند که صرفا روی داده های مختص مغز عملکرد خوبی داشته باشند و بتوانند بیماری را تشخیص دهند.

فصل دوم: پیاده سازی مدل دوم

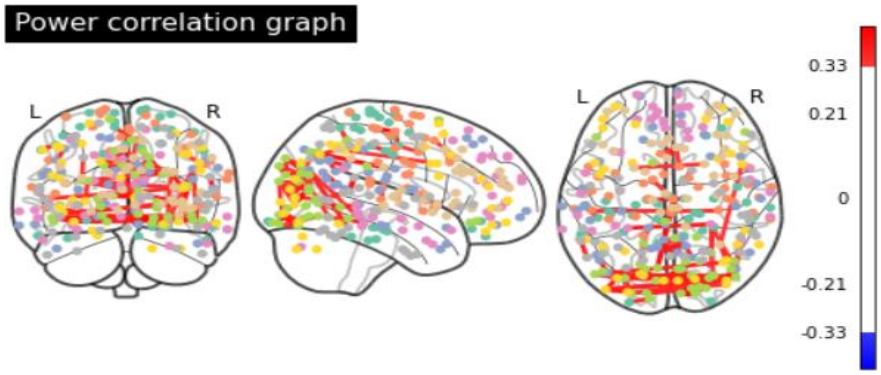
برای اینکه معیارها را بهبود داده پیاده سازی دیگری داشته و روش های دیگری برای تقسیم بندی مغز به مناطق مختلف ، انتخاب ویژگی های مفید و کاهش ابعاد و در مرحله بعدی آموزش مدل و در نهایت افزایش داده ها به کار گرفته شد که به نتایجی بسیار بهتر از پیاده سازی قبلی انجامید. در این پیاده سازی نیز نتایج مجدد روی هر سه اطلس CC_۲۰۰ و power۲٦٤ و CC_٤۰۰ به دست آمده و مقایسه شدند. با توجه به اینکه در مدل پیاده سازی شده اول با افزایش تعداد نواحی تقسیم شده مغز به نتایج بهتری رسیدیم در این مدل نیز اطلس های CC_۲۰۰ و اطلس های با تعداد نواحی بیشتر از ۲۰۰ برای ارزیابی مدل انتخاب شده است.

بیس لاین مدل

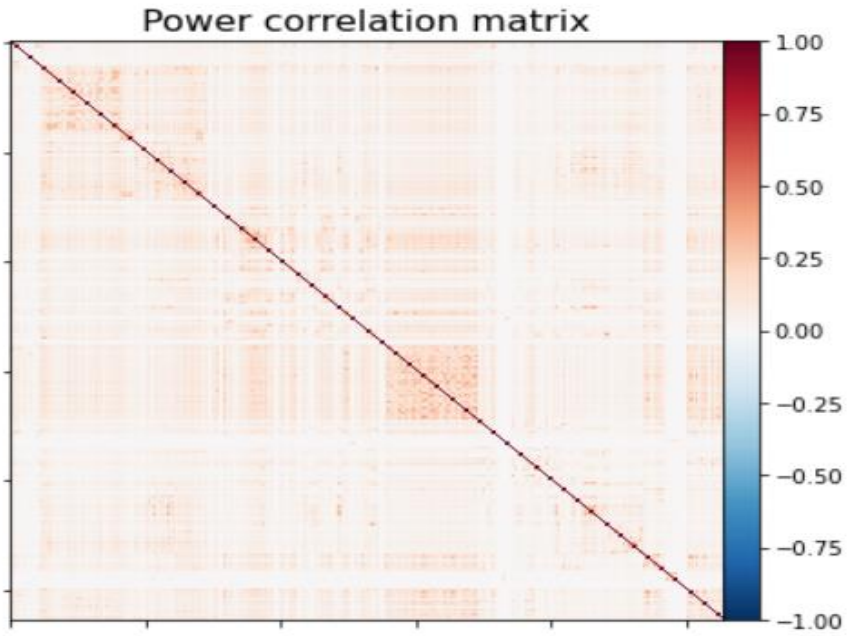
هدف از پیاده سازی مدل دوم بهبود معیار های به دست آمده از بهترین نتیجه حاصل از پیاده سازی مدل اول است. پس بیس لاین را مدل اول در نظر گرفتیم و قصد داریم در این پیاده سازی همان معیار های ارزیابی قبلی را بهبود دهیم.

انتخاب اطلس های مناسب

ابتدا لازم به ذکر است که اطلس CC_٤۰۰ نیز از تابع nilearn.datasets.fetch_abide_pcp و با مقدار rois_cc٤۰۰ برای آرگومان ورودی derivative آن استفاده می شود. اما اطلس power۲٦٤ را نمی توان با استفاده از این کتابخانه اعمال کرد و باید به صورت مستقیم روی تک تک سمپل ها در حالت اولیه که به صورت داده ۴ بعدی هستند اعمال شود. ابتدا با استفاده از تابع datasets.fetch_coords_power_۲۰۱۱ از کتابخانه nilearn مختصات ۲۶۴ ناحیه این اطلس که شامل مولفه های سه بعد x , y , z می باشد استخراج شده تا روی هر داده اعمال شود و به ۲۶۴ ناحیه مپ شود. سپس از تابع NiftiSpheresMasker از این کتابخانه استفاده شده تا سیگنال های سری زمانی از مناطق کروی هر ناحیه مغز استخراج شود.



شکل ۱۲_شمایی از مهم ترین اتصالات عملکردی در حجم مغز نمونه با اعمال اطلس POWER_264



شکل ۱۳_ماتریس همبستگی یک نمونه بعد از اعمال اطلس POWER_264

در تصویر بالا شمایی کلی از اعمال اطلس power۲۶۴ روی یکی از نمونه ها مشاهده می شود. تصویر بالایی اتصال عملکردی و مقدار این اتصال را میان نواحی مختلف مغز نشان می دهد. برای اینکه صرفا اتصالات مهم نمایش داده شود و شناسایی شوند اتصالات با مقادیر بیشتر از ۰.۳۳ فیلتر شده اند.

با توجه به اینکه در هر سه اطلس تعداد نواحی مغز زیاد است ماتریس همبستگی نیز بزرگ خواهد بود. در اطلس های CC_۲۰۰ و power۲٦٤ و CC_٤۰۰ بعد از محاسبه ماتریس همبستگی و حذف نیمه پایین مثلثی آن به دلیل متقارن بودن این ماتریس به ترتیب ۱۹۹۰۰ و ۳۴۷۱۶ و ۷۶۶۳۶ ویژگی باقی خواهد ماند. این تعداد ابعاد بسیار زیاد است و باید همچنان کاهش یابد تا از بیش برازش جلوگیری شود.

انتخاب ویژگی

روش Extra_Tree

برای کاهش بعد در این پیاده سازی از روش Extra Tree یا Extremly Randomized Tree (درختان فوق العاده تصادفی) استفاده شد. این الگوریتم به نام های جنگل های بسیار تصادفی یا جنگل های تصادفی با آستانه های تصادفی نیز شناخته می شود. این یک روش یادگیری گروهی مبتنی بر درخت تصمیم است. [۱۱]

این الگوریتم مانند جنگل های تصادفی در طول آموزش چندین درخت تصمیم می سازد. اما با در نظر گرفتن آستانه های تصادفی برای هر ویژگی، به جای جستجو برای بهترین آستانه های ممکن، سطح بیشتری از تصادفی بودن را وارد کار خود می کند. این تصادفی بودن اجازه می دهد تا Extra-Trees حساسیت کمتری نسبت به نویز در داده های آموزشی داشته باشد و بیش برازش را کاهش می دهد. این روش همچنین از نظر محاسباتی کارآمد تر است زیرا فرآیند یافتن بهترین تقسیم برای هر گره در مقایسه با درخت‌های تصمیم سنتی یا جنگل‌های تصادفی بسیار سریع‌تر است.

این روش می تواند بهترین و مفید ترین ویژگی ها را نیز تشخیص و میزان اهمیت هر کدام را محاسبه کند. اهمیت ویژگی بر اساس میانگین کاهش ناخالصی محاسبه شده از تمام درختان ساخته شده محاسبه می شود. برای هر درخت، کاهش ناخالصی (اهمیت جینی) برای هر ویژگی محاسبه می شود .کاهش ناخالصی سهم هر ویژگی را در همگنی گره های درخت اندازه گیری می کند. سپس اهمیت جینی برای هر ویژگی بر روی تمام درختان میانگین گرفته می شود تا امتیاز اهمیت ویژگی نهایی به دست آید. هر چه امتیاز بالاتر باشد، ویژگی برای طبقه بندی داده ها موثرتر است.

در این پیاده سازی ما نیز از این خاصیت استفاده کرده و ۱۹۳۵ ویژگی ابتدایی که این الگوریتم مفید و با امتیاز بالا تشخیص داده را به عنوان ویژگی های نهایی دیتاستمان برگزیدیم.[۱۲]

در اینجا رابطه محاسبه اهمیت ویژگی یا feature importance ویژگی i ام مشاهده می شود:

$$FI(i) = \frac{\sum_{K=1}^{N} FI_k(i)}{N}$$

$$i = 1, \ldots, m$$

در این رابطه N تعداد درخت های تصادفی و $FI_k(i)$ اهمیت ویژگی i ام را در k امین درخت تصادفی مشخص می کند. مقدار FI نیز توسط تابع مورد نظر در کتابخانه sklearn محاسبه می شود.

اجرای مدل

مدلیی که استفاده می کنیم ماشین بردار پشتیبان است. حال این که کاهش بعد انجام شد بهترین مقدار برای پارامترهای مدل ماشین بردار پشتیبان با استفاده از روش Grid Search محاسبه شده و دیتاست کاهش بعد داده شده به این مدل داده شد. معیارهای ارزیابی در حد سه الی ۴ درصد نسبت به بهترین نتیجه پیاده سازی قبلی بهبود یافت. روی هر سه اطلس CC_۴۰۰, power۲۶۴, CC_۲۰۰ این مدل اجرا شد اما نتایج روی CC_۴۰۰ بهتر از power۲۶۴ و power۲۶۴ بهتر از CC_۲۰۰ بود. مانند پیاده سازی قبلی نیز fold cross validation ۱۰ داشتیم و در نهایت نتایج روی ۱۰ بار اجرا میانگین گیری شد و به عنوان نتیجه نهایی اعلام شده است. نتایج حاصل به صورت زیر است:

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
SVM (test result)	۷۳	۶۶	**۷۹**
SVM (train result)	۹۱	۸۷.۸	**۹۳.۷**

جدول ۶_نتایج حاصل از اعمال مدل دوم روی اطلس CC_200

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
SVM (test result)	۷۴.۳	۷۰	**۷۸.۴**
SVM (train result)	۹۱	۸۷.۱	**۹۴**

جدول ۷_نتایج حاصل از اعمال مدل دوم روی اطلس power264

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
SVM (test result)	۷۴.۱	۶۹	**۷۸.۹**
SVM (train result)	۹۲.۳	۸۹.۴	**۹۴.۷**

جدول ۸_نتایج حاصل از اعمال مدل دوم روی اطلس CC_400

همانطور که از نتایج مشخص است نتایج روی هر سه اطلس نسبت به بهترین نتیجه به دست آمده از پیاده سازی قبلی به میزان قابل توجهی افزایش یافته است. روی معیار دقت حداکثر تا حدودا ۴ درصد و روی معیار حساسیت حداکثر تا ۶ درصد و روی معیار شفافیت حداکثر تا ۴ درصد افزایش داشته است. البته این نکته حائز اهمیت است که نتایج بدون افزایش داده الان با نتایج حاصل از افزایش داده پیاده سازی قبلی مقایسه شده اند و تا این اندازه تفاوت وجود دارد. پس بنظر می رسد بهتر است افزایش داده نیز روی این پیاده سازی هم داشته باشیم تا نتایج این دو پیاده سازی دقیق تر با یکدیگر مقایسه شوند.

پیاده سازی روش افزایش داده SMOTE

در این قسمت از استراتژی SMOTE در جهت افزایش داده همراه با تغییرات اندک استفاده شده است.

پیاده سازی نسخه اولیه SMOTE

ابتدا تکنیک افزایش داده استفاده شده در فصل اول مورد بررسی قرار می گیرد. مانند پیاده سازی قبلی نیز برای افزایش داده از روش SMOTE با معیار EROS استفاده کرده تا تاثیر آن روی معیارهای ارزیابی بررسی شود. با توجه به اینکه مجددا مانند پیاده سازی قبلی ۱۰ fold cross validation استفاده شده است در هر مرحله ۱۰٪ داده ها یعنی ۸۷ نمونه به عنوان مجموعه تست در نظر گرفته می شود و ۷۸۴ نمونه آموزشی داریم که به ازای هر نمونه یک نمونه دیگر تولید می شود پس در کل ۱۵۶۸ نمونه برای آموزش خواهیم داشت. در این پیاده سازی افزایش داده

روی اطلس های CC_۲۰۰ , power۲٦٤ موثر نبوده اما روی CC_٤۰۰ منجر به نتایج بهتری می شود. نتایج حاصل به صورت زیر است:

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
SVM (test result)	۷۲.۹	۶۷.۴	**۷۷.۸**
SVM (train result)	۹۴	۹۳	**۹۵**

جدول ۹_نتایج حاصل از اعمال مدل دوم با افزایش داده SMOTE روی اطلس CC_200

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
SVM (test result)	۷۴.۱	۷۱.۱	**۷۶.۶**
SVM (train result)	۹۲	۹۱.۱	**۹۴**

جدول ۱۰_نتایج حاصل از اعمال مدل دوم با افزایش داده SMOTE روی اطلس power264

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
SVM (test result)	۷۴.۹	۷۱.۹	**۷۷.۴**
SVM (train result)	۹۲.۵	۹۱.۵	**۹۳.۵**

جدول ۱۱_نتایج حاصل از اعمال مدل دوم با افزایش داده SMOTE روی اطلس CC_400

همانطور که از نتایج نیز مشخص است با افزایش داده و استفاده از اطلس CC_٤۰۰ به بهترین نتایج تا اینجای کار رسیدیم. پس بنظر می رسد با افزایش داده با بکار گیری این اطلس می توان مدل های بهتری را آموزش داد. پس لازم است روش های دیگری برای افزایش داده نیز در نظر گرفته شود. با توجه به اینکه روی اطلس های CC_۲۰۰ و power۲٦٤ نتایج خوبی با افزایش داده به دست نیاوردیم در آزمایشات بعدی تکنیک های افزایش داده صرفا روی اطلس CC_٤۰۰ اعمال شد.

پیاده سازی تغییر اول روی روش

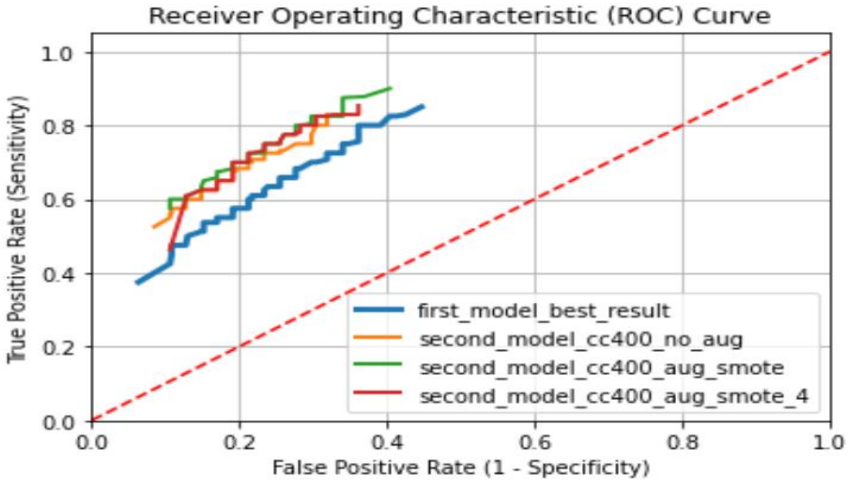
با توجه به اینکه ممکن است با افزایش داده بیشتر مدل بهتری آموزش دیده شود که به خوبی الگوهای موجود در داده را یاد گرفته باشد در این روش به ازای هر نمونه به جای اینکه فقط یک نمونه مصنوعی تولید کنیم ، سه نمونه جدید تولید کردیم. در واقع برای هر نمونه موجود در مجوعه آموزشی از بین ۵ بهترین همسایه منتخب ، ۳ همسایه به صورت تصادفی انتخاب شده و سپس با روش SMOTE سه نمونه جدید تولید شد. بنابراین با توجه به اینکه ۷۸۴ نمونه آموزشی داشتیم بعد از تولید داده ها ۳۱۳۶ = ۴ * ۷۸۴ داده خواهیم داشت. نتایج به دست آمده به صورت زیر است:

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
SVM (test result)	۷۴.۹	۷۲.۸	**۷۶.۶**
SVM (train result)	۹۳	۹۳.۲	**۹۳.۵**

جدول ۱۲_نتایج حاصل از اعمال مدل با افزایش داده SMOTE با تعداد داده ۴ برابر روی اطلس CC_400

همانطور که از نتایج به دست آمده نیز مشخص است در مقایسه با زمانی که ۱۵۶۸ داده در کل داشتیم دقت هیچ تغییری نکرده است. شفافیت در حد چند دهم کاهش یافته اما حساسیت ۱ درصد افزایش یافته است. می توان از این مدل برای زمان هایی که می خواهیم فردی که اوتیسم دارد حتما مبتلا به اوتیسم تشخیص داده شود تا هر چه سریعتر درمان شود و خطرات ناشی از آن را کم کند استفاده شود. چون نسبت به مدل های ارائه شده تا کنون این مدل حساسیت بهتری نسبت به مابقی داشته است.

اما اگر برایمان مهم است که افراد سالم حتما سالم تشخیص داده شوند و به اشتباه مبتلا به اوتیسم تشخیص داده نشوند میتوان از مدل قبلی با توجه به میزان بالاتر معیار شفافیت آن استفاده کرد.



شکل ۱۴_منحنی ROC مدل دوم با افزایش داده با نسخه های متفاوت SMOTE

همانطور که از ROC های این سه روش به همراه بیس لاین مدل نیز مشخص است ، اولین مدل پیاده سازی شده در این گزارش (مدل آبی رنگ) نسبت به مدل دوم کارایی بسیار پایین تری دارد و به خط قرمز رنگ وسط که سطح آستانه است نزدیک تر از مابقی است. مدل نارنجی که نشان دهنده نتیجه مدل دوم روی اطلس CC_٤٠٠ بدون افزایش داده است بعد از بیس لاین قرار گرفته است اما با فاصله زیاد از آن که نشان دهنده کارایی بسیار بالاتر آن نسبت به مدل اول است. اما مدل های سبز و قرمز با یکدیگر همپوشانی زیادی دارند. هر دو مدل مربوط به افزایش داده با روش SMOTE هستند با این تفاوت که مدل سبز با ۱۵۶۸ داد و مدل قرمز با ۳۱۳۶ داده است. مدل سبز کمی از نظر عملکرد بهتر از مدل قرمز می باشد با توجه به اینکه کمی بالاتر از آن قرار گرفته و خط قرمز رنگ وسط دورتر است.

پیاده سازی تغییر دوم روی روش

در روش قبلی برای افزایش داده ، بعد از مشخص شدن ۵ تا از بهترین همسایگان هر نمونه آموزشی با استفاده از معیار EROS یکی از همسایگان به صورت تصادفی انتخاب شده تا یک نمونه جدید با استفاده از نمونه اصلی و این همسایه تولید شود. اما در این روش بر اساس میزان شباهت هر همسایه به نمونه یک وزنی متناسب با آن به هر همسایه تخصیص داده می شود که انتخاب همسایه بر اساس این وزن صورت می گیرد و هر چقدر همسایه وزن بیشتری داشته

باشد احتمال انتخاب آن برای تولید داده مصنوعی بیشتر است. در این حالت به میزان کمی معیار های ارزیابی بالاتر می رود و بنظر روش مناسبی بوده است:

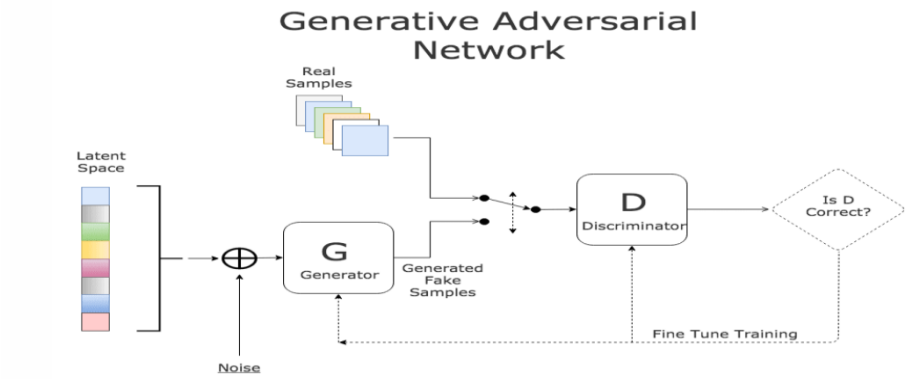
مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
SVM (test result)	۷۵.۱	۷۲.۲	**۷۷.۸**
SVM (train result)	۹۲.۷	۹۱.۶	**۹۳.۵**

جدول ۱۳_نتایج حاصل از اعمال مدل دوم با افزایش داده وزن دار SMOTE روی اطلس CC_400

همانطور که مشخص است با این روش معیارها در حد چند دهم افزایش یافتند اما تغییر فاحشی نیست. مجددا سعی می شود تکنیک های دیگری بررسی شود.

پیاده سازی مدل GAN ساده در جهت افزایش داده

شبکه‌های متخاصم مولد (GAN) یا Generative Adversarial Network دسته‌ای از فریم ورک های یادگیری ماشین هستند که از دو شبکه عصبی به نام‌های Generator (G) یا مولد و Discriminator (D) یا متمایز کننده تشکیل شده‌اند که در یک بازی حاصل جمع صفر (zero_sum) با یکدیگر رقابت می‌کنند. [۱۳] به این شکل که مولد سعی می‌کند داده‌های جعلی بهتر و بهتری ایجاد کند تا متمایزکننده را فریب دهد، در حالی که متمایزکننده تلاش می‌کند تا در تشخیص داده‌های جعلی بهتر شود. در واقع ، هدف مولد این است که احتمال تشخیص درست خروجی‌های جعلی توسط متمایزکننده را به حداقل برساند. برعکس، هدف متمایزکننده به حداکثر رساندن دقت خود در تشخیص داده های واقعی از جعلی است.[۱٤]



شکل ۱۵_ساختار کلی مدل های GAN

در این بخش از پیاده سازی نسخه های مختلف این نوع مولد ها بررسی شده و در نهایت بهترین نتیجه از مدل مولد ساده با دو لایه پنهان به دست آمد.[۱۵] این پیاده سازی از یک انکودر ، دیکودر ، متمایز کننده و متمایزکننده کد استفاده شده است. ابتدا در انکودر داده های ورودی که در واقع داده های آموزشی هستند به فضای ویژگی کوچکتری مپ می شوند که به این فضا کد پنهان گفته می شود. را بطه بین فضای ورودی و خروجی به صورت زیر است:

$$z = E(x)$$

x داده های ورودی یا همان مجموعه آموزشی و z فضای مپ شده نهایی است.

سپس وظیفه دیکودر این است که فضای اولیه ورودی را از فضای نهایی مپ شده بازسازی کند. لازم به ذکر است که در این پیاده سازی نیز تبدیل ها خطی می باشند. رابطه بین فضای ورودی و خروجی به صورت زیر است:

$$\hat{x} = D(z)$$

متمایزکننده نیز یک مدل شبکه عصبی است که میان داده های جعلی و واقعی تمایز قائل می شود. این مدل یک خروجی برمی گرداند که احتمال تعلق به داده های واقعی و جعلی است. هر چقدر این احتمال به مقدار یک نزدیکتر

باشد به معنای واقعی تر بودن داده ها و هرچقدر به صفر نزدیکتر باشد به معنای جعلی بودن آن است. رابطه میان ورودی و خروجی این مدل به صورت زیر است:

$$Y = D(x)$$

اما درنهایت متمایز کننده کد مشابه متمایزکننده عمل میکند با این تفاوت که باید احتمال واقعی یا جعلی بود داده ها در فضای ویژگی مپ شده نهایی با ۱۲۸ ویژگی را تشخیص دهد. رابطه بین ورودی و خروجی به صورت زیر است:

$$Y = C(z)$$

در تمامی این چهار مدل لایه ها به صورت تبدیل خطی با وزن و بایاس مشخص بوده و با رابطه زیر می باشد:

$$Linear(x)=Wx+b$$

حال به مرحله آموزش مدل می رسیم. در این مرحله لازم است ابتدا متمایز کننده و متمایز کننده کد با توابع لاس کراس آنتروپی باینری آموزش داده شوند. رابطه این تابع لاس متمایز کننده به صورت زیر است:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim p_{\text{data}}}[\log\ D(x)] - \mathbb{E}_{\hat{x}\sim p_{\text{fake}}}[\log\ (1 - D(\hat{x}))]$$

منطق این رابطه به این صورت است که با توجه به اینکه هدف متمایز کننده این است که تواناییش در تشخیص داده های واقعی از جعلی را افزایش دهد در نتیجه هر چقدر $D(x)$ که احتمال واقعی بودن داده ها است به یک نزدیکتر باشد بدین معناست که تشخیص بهتری داشته است و متمایز کننده به خوبی عمل کرده است چون داده های واقعی x به احتمال بالایی واقعی تشخیص داده شده اند. در این صورت مقدار لگاریتم این احتمال نیز افزایش یافته و با توجه به اینکه در محاسبه خطا این مقدار باید کمینه شود منفی می شود. اما از طرف دیگر هر چقدر متمایز کننده داده های جعلی را با احتمال نزدیک به صفر و کمتر تشخیص دهد به معنای عملکرد بهتر آن در تشخیص داده های جعلی بوده است. در نتیجه هر چقدر $D(\hat{x})$ کوچکتر باشد مقدار $1 - D(\hat{x})$ نیز افزایش یافته و در ادامه مقدار لگاریتم آن نیز

بیشتر می شود و در تابع خطا باید این مقدار کمینه شود پس این مقدار منفی می شود. در نهایت این مقادیر با یکدیگر جمع شده تا عملکرد متمایز کننده هم روی داده های جعلی و هم روی داده های واقعی بهبود داشته باشد.

در مرحله بعدی لازم است تابع خطای متمایز کننده کد در جهت آموزش این مدل به دست آید. رابطه تابع لاس متمایز کننده کد به شرح زیر است:

$$\mathcal{L}_C = -\mathbb{E}_{z\sim p_{\text{latent}}}[\log\ C(z)] - \mathbb{E}_{z'\sim p_{\text{noise}}}\left[\log\left(1 - C(z')\right)\right]$$

منطق این رابطه نیز کاملا مشابه رابطه تابع لاس متمایز کننده می باشد با این تفاوت که متمایز کننده روی داده های واقعی فضای اولیه عمل می کند اما متمایز کننده کد روی داده های مپ شده با فضای ویژگی نهایی و کاهش بعد داده شده عمل می کند تا عملکرد متمایز کننده کد را در تشخیص داده های واقعی از این فضای مپ شده و داده های جعلی یا نویز در این فضای مپ شده را بسنجد. در این رابطه z داده های واقعی فضای مپ شده و z' مقادیر نویز تصادفی در این فضای ویژگی می باشد.

در مرحله آخر آموزش لازم است انکودر و دیکودر همراه با یکدیگر آموزش داده شوند. تابع خطای مولد یا همان دیکودرکراس آنتروپی باینری و به صورت زیر است:

$$\mathcal{L}_G = -\mathbb{E}_{\hat{x}\sim p_{\text{fake}}}[\log\ D(\hat{x})]$$

می دانیم که هدف مولد این است که داده ها را به گونه ای تولید کند که متمایز کننده نتواند آن را از داده های واقعی تشخیص دهد. بدین ترتیب می خواهیم متمایزکننده در مواجهه با داده های جعلی یا بازسازی شده که واقعی نیستند احتمال نزدیک به یک را داشته باشد و در واقع آن ها را داده های واقعی تشخیص دهد. پس هر چقدر احتمال بیشتر باشد مقدار لگاریتم نیز بیشتر می شود و برای اینکه کمینه شود خطا این مقدار باید منفی شود.

از طرف دیگر خطای بازسازی دیکودر که تعیین کننده تفاوت بین داده واقعی و داده بازسازی شده است با استفاده از خطای میانگین مربعات یا *MSE* محاسبه شده و به صورت زیر می باشد:

$$\mathcal{L}_{rec} = \mathbb{E}_{x\sim p_{\text{data}}}\left[\| x - \hat{x} \|^{2}\right]$$

در نهایت این دو مقدار با یکدیگر جمع شده تا از مقدار لاس تجمیع شده برای آپدیت کردن پارامترهای انکودر و دیکودر و آموزش این دو مدل استفاده شود.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{G} + \mathcal{L}_{\text{rec}}$$

در نهایت بعد از اینکه این ۴ مدل طی ۱۰۰ ایپاک آموزش داده شدند و مقدار لاس مینیمم شد از طریق دیکودر در فضای ویژگی های مپ شده با ۱۲۸ ویژگی به تعداد داده های آموزشی ، سمپل جنریت شده و سپس بازسازی شده و به فضای ویژگی اولیه با ۱۹۳۵ ویژگی برگردانده می شود که به عنوان داده های مصنوعی برای آموزش بهتر مدل استفاده می شود.

در مرحله نهایی ، مدل ماشین بردار پشتیبان در کنار افزایش داده پیاده شده و متریک ها ارزیابی شدند. برای آموزش مدل ۱۰ fold cross validation روی دیتاست اعمال شده. در هر مرحله ۸۷ تعداد داده برای آزمایش و ۷۸۴ تعداد داده برای آموزش داشتیم که با توجه به اینکه به ازای هر سمپل آموزشی یک سمپل مصنوعی جنریت می شد در نهایت به تعداد ۱۵۶۸ = ۷۸۴ * ۲ سمپل برای آموزش داشتیم.

ابتدا یک مدل *GAN* با یک لایه پنهان پیاده شد و نتایج به صورت زیر به دست آمدند:

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
SVM (test result)	۷۴.۸	۷۰.۵	**۷۸.۴**
SVM (train result)	۹۰.۷	۸۷.۴	**۹۳.۶**

جدول ۱۴_نتایج حاصل از اعمال مدل دوم با افزایش داده با GAN یک لایه روی اطلس CC_400

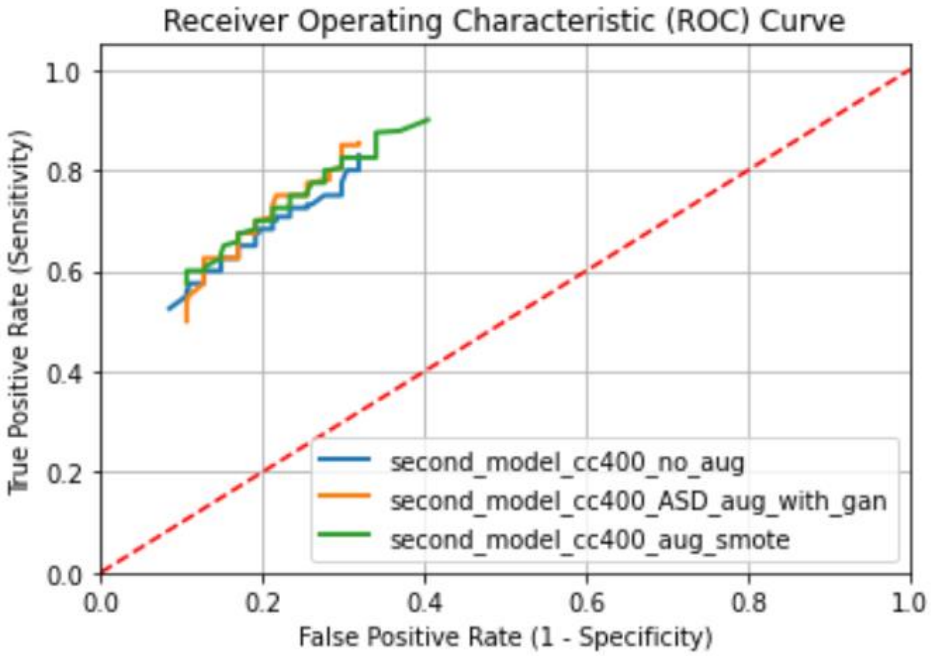
متریک های ارزیابی مدل نسبت به حالت بدون افزایش داده در حد چند دهم افزایش یافتند اما نتوانستند بهتر از روش افزایش داده SMOTE عمل کنند. بنظر می رسد بهتر است کمی مدل GAN را پیچیده تر کرد تا عملکرد بهتری داشته باشد. در این راستا یک لایه دیگر به این مدل اضافه می کنیم و مجددا آزمایش را روی مدل GAN با دو لایه پنهان انجام می دهیم.

در این مدل دو لایه کاهش بعد به این شکل عمل می کند که داده ها از ۱۹۳۵ ویژگی به فضایی با ۵۱۲ ویژگی در لایه ابتدایی مپ شده. سپس در لایه دوم به ۲۵۶ ویژگی مپ می شود و در نهایت در لایه انتهایی به ۱۲۸ ویژگی مپ می شوند. سپس در دیکودر ابتدا ۱۲۸ ویژگی را به فضایی با ۲۵۶ ویژگی و بعد ۵۱۲ ویژگی و در نهایت به همان ۱۹۳۵ ویژگی مپ می کند.

نتایج حاصل از اعمال مدل روی این GAN پیاده شده نشان می دهد که این ساختار عملکرد خوبی روی مدل داشته و تا اینجای کار بهترین نتیجه را داشته است. نتایج حاصل از این استراتژی افزایش داده به صورت زیر می باشد:

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
SVM (test result)	۷۵.۳	۷۰.۸	**۷۹.۲**
SVM (train result)	۹۲	۸۸	**۹۵.۴**

جدول ۱۵ _نتایج حاصل از اعمال مدل دوم با افزایش داده با GAN دو لایه روی اطلس CC_400



شکل ۱۶_منحنی ROC مقایسه میان افزایش داده با SMOTE و مدل GAN دولایه

همانطور که از نتایج بالا نیز مشخص است افزایش داده عملکرد مدل را بهبود داده است. افزایش داده با GAN پیاده سازی شده و همچنین SMOTE با یکدیگر از نظر عملکردی هم پوشانی و اشتراک زیادی دارند اما پیاده سازی شده با GAN در نهایت و انتهای کار در سطح بالاتری نسبت به روش SMOTE قرار گرفته است. پس تا به اینجای کار GAN دولایه بهترین عملکرد و همچنین دقت را در مقایسه با مابقی مدل ها داشته است.

مجدد لایه ها را افزایش دادیم تا با پیچیده تر کردن معماری آن نتایج بهتر شود اما از دولایه به بعد نتایج افت کرده و موثر واقع نشدند.

پیاده سازی مدل CW_GAN در جهت افزایش داده

نسخه بهبود یافته ای از GAN ساده در این بخش در جهت افزایش داده به کار گرفته و پیاده سازی شده است. این معماری ترکیبی از WGAN و Conditional GAN می باشد.[۱۸] [۱۷] [۱٦] تابع Gradient Penalty وظیفه محاسبه فاصله Wasserstein که نمایانگر فاصله میان دو توزیع متفاوت است در ساختار WGAN را دارا

است. Gradient Penalty در ساختار WGAN یک تکنیک منظم سازی است که برای اعمال محدودیت Lipschitz بر روی مدل متمایز کننده یا Discriminator استفاده می شود. این محدودیت به بهبود پایداری آموزش کمک می کند و باعث همگرایی بهتر مدل های Generator , Discriminator می شود. محدودیت Lipschitz روی مدل متمایزکننده تضمین می کند که مدل خیلی سریع روی ورودی های مختلف تغییر نکند و شرایط پایدار تری داشته باشد. این محدودیت به این شکل تعریف می شود:

تابعی مانند f به صورت $f: R^n \to R$ پیوسته Lipschitz با مقدار ثابت Lipschitz مانند L است اگر یک مقدار ثابت $L \geq 0$ وجود داشته باشد و برای هر $X, Y \in R^n$ رابطه زیر برقرار باشد:

$$|f(x) - f(y)| \leq L \parallel x - y \parallel$$

که در این رابطه $\parallel x - y \parallel$ نشان دهنده فاصله بین x , y در فضای ورودی است.

نحوه کار gradient_penalty به این صورت است که داده های واقعی و جعلی را دریافت کرده و با استفاده از درون یابی خطی داده های جدید با استفاده از داده های واقعی و جعلی تولید می کند. رابطه تولید این داده های جدید به صورت زیر است:

$$\hat{\mathbf{x}} = \alpha \mathbf{x}_{\text{real}} + (1 - \alpha)\mathbf{x}_{\text{fake}}$$

آلفا به صورت رندوم از یک توزیع یکنواخت انتخاب می شود.

سپس خروجی متمایزکننده روی این داده های جدید درون یابی شده به دست آمده:

$$D(\hat{\mathbf{x}})$$

در این مرحله گرادیان های خروجی متمایزکننده روی داده های جدید درون یابی شده محاسبه می شود:

$$\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})$$

سپس نرم L_2 یا نرم اقلیدوسی این گرادیان ها باید محاسبه گردد:

$$\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2$$

تابع gradient_penalty محدودیت Lipschitz را با جریمه کردن انحرافات نرم گرادیان از یک اعمال می کند. میانگین مربعات این جریمه به صورت زیر محاسبه می شود:

$$\text{gradient penalty} = \mathbb{E}_{\hat{\mathbf{x}} \sim p_{\hat{\mathbf{x}}}} \left[\left(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1 \right)^2 \right]$$

از طرف دیگر تولید داده در یک Conditional GAN مشروط به اطلاعات ورودی خاص است که می تواند برچسب ها، اطلاعات کلاس یا هر ویژگی مرتبط دیگری باشد. این شرطی سازی امکان تولید داده های دقیق تر و هدفمندتر را فراهم می کند. در این پیاده سازی ما کلاس های داده ها را به عنوان اطلاعات اضافی در مولد و همچنین متمایز کننده به کار گرفتیم. پس نیاز بود تا در مولد و متمایزکننده لایه های تعبیه شده برای کلاس ها در نظر گرفته شود. این لایه تعبیه شده مانند یک جدول جستجو عمل می کند که هر کلاس را (صفر یا یک) در این مسئله به بردارهایی با اندازه مشخص مپ می کند. وزن های متناظر این لایه در حین فرآیند آموزش به دست آمده و به روزرسانی می شوند. پس این لایه تعبیه شده لازم است تا مقادیر گسسته متناظر با هر کلاس را به بردارهایی با مقادیر پیوسته تبدیل کند. در مولد هر کلاس به برداری با اندازه فضای ویژگی مپ شده نهایی تبدیل می شود و از طرف دیگر در متمایزکننده هر کلاس به برداری با اندازه فضای ویژگی اولیه داده های اصلی تبدیل می شود.[۲۰] [۱۹]

ساختار مولد

ورودی مولد یک بردار نویز تصادفی با اندازه ۱۲۸ و همچنین کلاس ها است. در مولد ابتدا لایه تعبیه شده برای کلاس های دریافت شده ایجاد شده که در واقع بردارهایی پیوسته به اندازه ۱۲۸ برای هر کلاس است. این دو بردار با یکدیگر ادغام شده و به برداری با اندازه ۲۵۶ تبدیل می شود. سپس در طی لایه های متوالی این ۲۵۶ ویژگی به ۵۱۲ و در لایه بعدی به ۱۰۲۴ و در لایه بعد به ۲۰۴۸ و در آخر از ۲۰۴۸ به فضایی با ۱۹۳۵ ویژگی مپ می شود. این لایه ها از تبدیل خطی استفاده می کنند به این صورت:

$$h = Wx + b$$

که W و b به ترتیب بردار وزن ها و بایاس هر لایه و x ورودی هر لایه می باشد. سپس بعد از تبدیل خطی در هر لایه تابع فعالسازی RELU روی خروجی هر لایه اعمال می شود تا مقداری غیر خطی بودن به لایه ها اضافه شود و پیچیده تر کردن مدل امکان پذیر شود:

$$a = RELU(h)$$

در نهایت نیز در آخرین لایه نیاز به اعمال تابع فعالسازی Tanh است تا تضمین کند که مقادیر خروجی در یک محدوده مشخص قرار گرفته اند.

ساختار متمایزکننده

ورودی متمایزکننده داده هایی با فضای ویژگی اصلی یعنی ۱۹۳۵ و همچنین کلاس ها است. در متمایزکننده ابتدا لایه تعبیه شده برای کلاس های دریافت شده ایجاد شده که در واقع بردارهایی پیوسته به اندازه ۱۹۳۵ برای هر کلاس است. این دو بردار با یکدیگر ادغام شده و به برداری با اندازه ۳۸۷۰ تبدیل می شود. در نهایت طی لایه های متوالی این ابعاد به فضایی با ۱۰۲۴ ویژگی و سپس در لایه بعدی ۵۱۲ و بعد ۲۵۶ و بعد از آن به فضایی با ۱۲۸ ویژگی و در نهایت به یک مقدار مپ می شود که احتمال واقعی بودن یا جعلی بودن داده ی ورودی را مشخص می کند. این لایه ها نیز مانند لایه های مولد از تبدیل خطی به همراه تابع فعالسازی در جهت اضافه کردن مقداری غیر خطی بودن به این لایه ها استفاده می کنند.

نحوه آموزش متمایزکننده

برای آموزش متمایزکننده به این شکل عمل می کنیم که در هر ایپاک ابتدا یک بردار نویز به صورت تصادفی با اندازه فضای ویژگی مپ شده نهایی یعنی ۱۲۸ تولید می شود و به همراه کلاس های واقعی داده ها به مولد داده می شود. بعد از دریافت خروجی از مولد ، خروجی به دست آمده که در واقع یک داده جعلی است به متمایزکننده داده می شود و نتایج حاصل از آن ذخیره می شود. از طرف دیگر داده های واقعی که در واقع همان داده های آموزشی

هستند به همراه کلاس های داده ها نیز به متمایزکننده داده می شوند و خروجی حاصل از آن ذخیره می شود. حال باید داده های واقعی و جعلی به همراه کلاس ها و همچنین متمایزکننده به تابع gradient_penalty داده شود و خروجی آن ذخیره شده تا بعدا مورد استفاده قرار گیرد. در نهایت مجموع خطای کلی متمایزکننده در هر ایپاک از رابطه زیر محاسبه می شود:

$$\mathcal{L}_D = -\mathbb{E}[D(x)] + \mathbb{E}\big[D\big(G(z)\big)\big] + \lambda.\,\text{Gradient Penalty}$$

از آنجایی که می خواهیم متمایزکننده احتمال بالاتری در مواجهه با داده های ورودی واقعی به دست آورد پس هر چقدر بالاتر باشد باید خطای کمتری ایجاد کند در نتیجه مینیمم می شود. اما از طرف دیگر متمایزکننده باید مقدار خروجی با احتمال کمتری را برای داده های جعلی داشته باشد در نتیجه باید این مقدار تاثیر مثبتی روی خطای کلی داشته باشد. هر چقدر کمتر باشد خطا کتر شود و هرچقدر بیشتر باشد خطا نیز بیشتر باشد. از طرف دیگر اگر هنجار گرادیان از مقدار یک منحرف شود، نتیجه حاصل از gradient_penalty متمایز کننده را جریمه می کند. این اطمینان را ایجاد می کند که گرادیان متمایزکننده دارای هنجاری نزدیک به یک باشد، که برای ارضای محدودیت Lipschitz که مورد نیاز مجاسبه فاصله Wasserstein می باشد بسیار مهم است. λ نیز وزن این مقدار جریمه است که برای اینکه مناسب داده های ما باشد مقدار بزرگ ۲۰ برای آن انتخاب شده است.

نحوه آموزش مولد

برای آموزش مولد به این شکل عمل می کنیم که در هر ایپاک مجددا بردار نویز تصادفی با اندازه ۱۲۸ به همراه کلاس های داده ها به مولد داده می شود. سپس خروجی مولد به همراه کلاس ها به متمایزکننده داده می شود و از این خروجی به دست آمده به این شکل برای محاسبه خطای مولد استفاده می شود:

$$\mathcal{L}_G = -\mathbb{E}[D(G(z))]$$

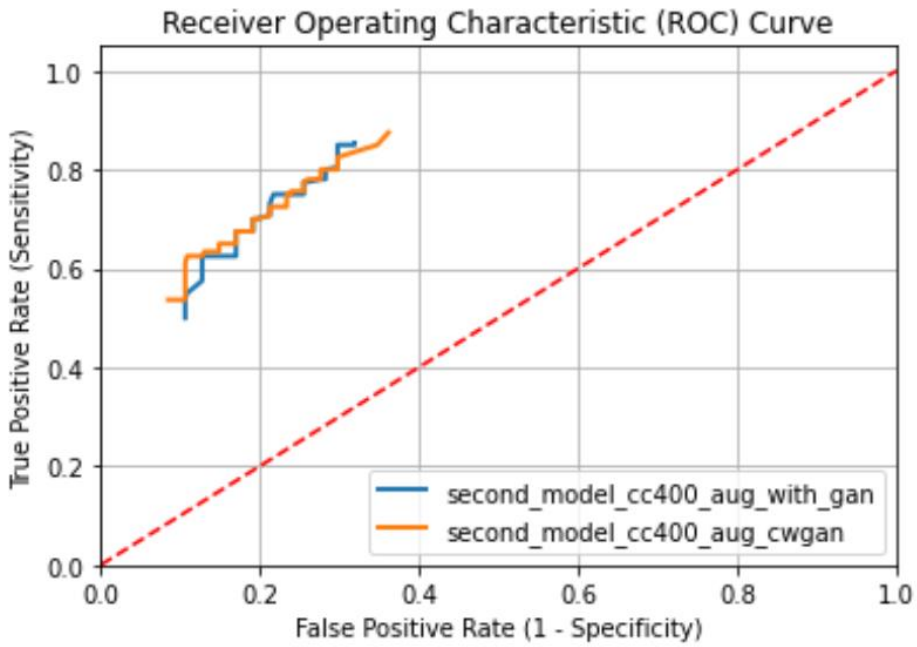
هدف مولد این است که متمایزکننده را فریب دهد پس هر چقدر متمایزکننده داده های تولید شده نویزی یا جعلی تولید شده توسط مولد را واقعی تر تشخیص دهد و احتمال خروجی بالاتری داشته باشد مقدار E بیشتر شده و خطای

مولد کمتر باید شود چون مولد در فریب دادن متمایزکننده به خوبی عمل کرده است پس این مقدار منفی می شود.

نتایج حاصل از افزایش داده با استفاده از این روش و مدل دوم به صورت زیر می باشد:

مدل ها	ACCURACY	SENSITIVITY	SPECIFICITY
SVM (test result)	۷۵.۴	۷۰.۹	**۷۹.۴**
SVM (train result)	۹۲.۷	۸۹	**۹۴**

جدول ۱۶_نتایج حاصل از اعمال مدل دوم با افزایش داده با CW_GAN روی اطلس CC_400



شکل ۱۷_منحنی ROC مقایسه بین افزایش داده با GAN ساده و CW_GAN

همانطور که از نتایج بالا مشخص است پیاده سازی با استفاده از GAN و همچنین Conditional WGAN با یکدیگر از نظر عملکردی هم پوشانی و مشابهت زیادی دارند. بنظر می رسد از ابتدا تا نیمه راه روش دوم عملکرد

بهتری داشته و از نیمه تا انتها روش اولی برتری داشته است. اما میانگین معیارهای نهایی روی روش دوم در حد یک الی دو دهم بالاتر از روش دوم بوده است.

نتیجه گیری

همانطور که بالاتر اشاره شد ابتدا از روش های مختلف پیش پردازش و استفاده از اطلس های مختلف و همچنین روش هایی برای انتخاب ویژگی های مفید مجموعه داده و استراتژی های افزایش داده استفاده کردیم و عملکرد بهترین مدل در پیاده سازی فصل اول به دست آمد که نسبت به بیس لاین هایی مانند جنگل های تصادفی و ماشین بردار پشتیبان بهتر عمل می کرد. در پیاده سازی فصل دوم که بیس لاین بهترین مدل به دست آمده از فصل اول بود روش های جدیدی برای انتخاب ویژگی و افزایش داده بکار گرفته شد و نتایج بهبود یافتند. از بیس لاین ابتدایی با دقت ۶۵٪ به دقت نهایی ۷۵.۴٪ رسیدیم و بهترین مدل نهایی به دست آمد.

نگاهی به آینده

با توجه به اینکه در بخش انتهایی از فصل اول به تاثیر مستقیم اطلاعات جانبی دیتاست روی عملکرد مدل ها پرداختیم و به نتایج خیلی خوبی رسیدیم در آینده می توان به صورت غیر مستقیم از این اطلاعات جانبی برای شناخت بهتر داده ها استفاده کرد و در نتیجه به بهبود عملکرد مدل ها پرداخت. در واقع با استفاده غیر مستقیم از اطلاعات جانبی می توان تشخیص داد نمونه هایی که در اکثریت مدل های پیشنهادی در این گزارش نتوانستند به خوبی عمل کنند چه ویژگی های مشترک جانبی دارند. مثلا می توانند همگی مربوط به یک سن خاص یا یک منطقه جغرافیایی خاص باشند. با شناسایی این نمونه ها و پیدا کردن الگوی مشترک میان آن ها می توان تغییراتی در مدل ها متناسب با نقاط ضعف این نمونه ها ایجاد کرد.

مراجع

[١] [Online]. Available: https://github.com/preprocessed-connectomes-project/abide/blob/master/Phenotypic_V١_٠b_preprocessed١.csv.

[٢] R. C. J. G. A. H. P. E. I. H. X. P. a. M. H. S. Craddock, "A whole brain fMRI atlas generated via spatially constrained spectral clustering," ٢٠١٢.

[٣] [Online]. Available: https://nilearn.github.io/dev/modules/generated/nilearn.datasets.fetch_coords_power_٢٠١١.html.

[٤] J. D. C. A. L. N. S. M. W. G. S. B. K. A. C. J. A. e. a. Power, "Functional network organization of the human brain," ٢٠١١.

[٥] C. B. Y. C. C. C. F. E. A. J. A. Craddock, "The Neuro Bureau Preprocessing Initiative: open sharing of preprocessed neuroimaging data and derivatives," ٢٠١٣.

[٦] [Online]. Available: https://nilearn.github.io/stable/auto_examples/index.html.

[٧] V. M. A. F. A. L. F. S. Taban Eslami, "ASD-DiagNet: A hybrid learning approach for detection of Autism Spectrum Disorder using fMRI data," ٢٠١٩.

[٨] [Online]. Available: https://nilearn.github.io/dev/modules/generated/nilearn.datasets.fetch_abide_pcp.html.

[٩] N. V. B. K. W. H. L. O. a. K. W. P. Chawla, "SMOTE: synthetic minority over-sampling technique," ٢٠٠٢.

[١٠] A. S. F. A. R. C. R. C. B. A. a. M. F. Heinsfeld, "Identification of autism spectrum disorder using deep learning and the ABIDE datasetv," ٢٠١٨.

[١١] P. G. ·. D. E. ·. L. Wehenkel, "Extremely randomized trees," ٢٠٠٦.

[١٢] L. X. J. L. J. Y. a. X. Y. Yaya Liu, "Attentional Connectivity-based Prediction of Autism Using Heterogeneous rs-fMRI Data from CC٢٠٠ Atlas," ٢٠٢٠.

[١٣] J. P.-A. M. M. B. X. D. W.-F. S. O. A. C. Y. B. ·. Ian J. Goodfellow, "Generative Adversarial Networks," ٢٠١٤.

[١٤] [Online]. Available: https://github.com/eriklindernoren/PyTorch-GAN/tree/master/implementations/gan.

[١٥] N. C. D. L. H. S. a. J. S. D. Jiyao Wang, "Learning Sequential Information in Task-based fMRI for Synthetic Data Augmentation," ٢٠٢٣.

[١٦] M. C. S. &. B. L. Arjovsky, "Wasserstein GAN," ٢٠١٧.

[١٧] I. A. F. A. M. D. V. &. C. A. C. Gulrajani, "Improved Training of Wasserstein GANs," ٢٠١٧.

[١٨] [Online]. Available: https://www.geeksforgeeks.org/wasserstein-generative-adversarial-networks-wgans-convergence-and-optimization/.

[١٩] M. &. O. S. Mirza, "٢٠١٤," *Conditional Generative Adversarial Nets.*

[٢٠] [Online]. Available: https://www.geeksforgeeks.org/conditional-generative-adversarial-network/.