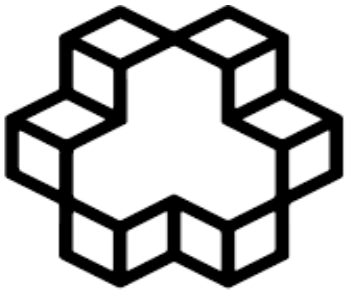


دانشکده مهندسی برق _گرایش کنترل

میان ترم

یادگیری ماشین

نگارش

فاطمه امیری

۴۰۲۰۲۴۲۴

لینک گوگل کولب

لینک گیت هاب

استاد مربوطه

جناب آقای دکتر علیاری

خرداد ماه ۱۴۰۳


بسم الله الرحمن الرحيم

فهرست مطالب

سوال یک

۱ پرسش یک

درستی یا نادرستی هریک از گزاره‌های زیر را با ذکر دلیل مشخص کنید.

۱. طبقه‌بند بیز، بهترین طبقه‌بندی است که می‌توان برای جداسازی یک مسألهٔ دوکلاسه طراحی کرد.

۲. استفاده از رویکرد بیز برای تخمین پارامترهای توزیع، می‌تواند مانع از بیش‌برازش (Overfitting) شود.

۳. استفاده از معیار Information Gain برای ساخت درخت در شرایطی که بعضی از ویژگی‌ها حالات زیادی دارند، مناسب نیست.

۴. هر شبکهٔ عصبی چندلایه با توابع فعال‌ساز خطی در لایه‌های پنهان می‌تواند به عنوان یک شبکهٔ عصبی بدون هیچ لایه پنهانی نمایش داده شود.

۱ .طبقه بند بیز، بهترین طبقه بندی است که می توان برای جداسازی یک مسألهٔ دوکلاسه طراحی کرد.

عبارت بالا نادرست است.

طبقه‌بند بیز ساده بر اساس فرض استقلال ویژگی‌ها عمل می‌کند، یعنی این مدل فرض می‌کند که همه ویژگی‌ها مستقل از یکدیگر هستند و تأثیر متقابل ندارند. این فرض در بسیاری از مسائل دنیای واقعی نادرست است و ممکن است دقت مدل را کاهش دهد. به دلیل این فرض، طبقه‌بند بیز ساده نمی‌تواند روابط پیچیده بین ویژگی‌ها را به خوبی مدل‌سازی کند، در حالی که بسیاری از مسائل طبقه‌بندی به چنین مدل‌هایی نیاز دارند. مدل‌های پیچیده‌تر مانند درخت تصمیم، ماشین بردار پشتیبان (SVM)، شبکه‌های عصبی و جنگل تصادفی معمولاً عملکرد بهتری نسبت به طبقه‌بند بیز ساده دارند، به ویژه در مسائل پیچیده و با داده‌های بزرگ، زیرا این مدل‌ها می‌توانند روابط پیچیده‌تری را در داده‌ها کشف کنند و دقت بالاتری ارائه دهند. اگرچه در برخی موارد خاص، طبقه‌بند بیز ساده ممکن است عملکرد خوبی داشته باشد، اما این عملکرد به شدت به نوع داده و ساختار آنها بستگی دارد و مدل‌های دیگر اغلب عملکرد پایدارتری نشان می‌دهند. مدل‌های دیگری که نیاز به مفروضات کمتر و انعطاف‌پذیری بیشتری دارند، معمولاً دقت بالاتری در طبقه‌بندی ارائه می‌دهند. به عنوان مثال، شبکه‌های عصبی عمیق می‌توانند روابط پیچیده را یاد بگیرند و ویژگی‌های مفیدی استخراج کنند که طبقه‌بند بیز ساده قادر به انجام آن نیست. بنابراین، طبقه‌بند بیز ساده معمولاً گزینه خوبی برای شروع و بررسی ابتدایی داده‌ها است، اما نمی‌توان گفت که

بهترین طبقه‌بند برای تمامی مسائل دوکلاسه است. انتخاب بهترین طبقه‌بند به خصوصیات داده‌ها و مسئله خاص مورد نظر بستگی دارد.

پس در کل این سوال بسیار کلی بیان شده است و پاسخ منفی دادن به آن منطقی نیست. زیرا عملکرد طبقه‌بند بیز ساده به شدت به نویز داده‌ها، فرضیات مانند استقلال یا وابستگی ویژگی‌ها و عدم تعادل طبقات بستگی دارد. بنابراین، برای پاسخ دقیق‌تر باید حوزه مساله را مشخص‌تر کنیم.

در طبقه‌بند بیز، با محاسبه احتمالات در فضاهای پیوسته و گسسته سروکار داریم که برای محاسبه آن‌ها باید از توابع چگالی احتمال متناسب استفاده کنیم. این رویکرد احتمالاتی شبیه به کلاسیفایرهای قبلی با خط تصمیم‌گیری سخت است، اما اینجا به جای طبقه‌بندی قطعی، درجه عضویت محاسبه می‌شود. قاعده بیز بهینه است اما در ابعاد بالا کارایی ندارد. عملکرد این کلاسیفایر به تغییر مرز تصمیم‌گیری و اهمیت طبقات بستگی دارد. بهترین الگوریتم طبقه‌بندی به ویژگی‌های داده‌ها و مساله مربوطه بستگی دارد. مزایای طبقه‌بند بیز شامل سادگی، سرعت و کارایی در داده‌های بزرگ و پراکنده است. اما محدودیت‌هایی مانند فرض استقلال ویژگی‌ها و مشکل در مسائل با داده‌های وابسته و عدم تعادل طبقات دارد که ممکن است عملکرد آن را کاهش دهد.

۲ .استفاده از رویکرد بیز برای تخمین پارامترهای توزیع، می تواند مانع از بیش برازش(overfitting) شود.

عبارت بالا درست است.

استفاده از رویکرد بیز برای تخمین پارامترهای توزیع می‌تواند مانع از بیش‌برازش شود، زیرا این رویکرد از توزیع‌های پیشین برای پارامترها بهره می‌برد که به عنوان نوعی تنظیم‌گر عمل کرده و از تغییرات بیش از حد پارامترها جلوگیری می‌کند. در این روش، اطلاعات پیشین با داده‌های مشاهده شده ترکیب می‌شوند تا به توزیع پسین برسیم، که باعث می‌شود مدل به جای تکیه کامل بر داده‌های آموزشی، از دانش پیشین نیز بهره‌مند شود و از تطبیق بیش از حد به داده‌های خاص جلوگیری کند. این رویکرد همچنین به کاهش واریانس مدل کمک می‌کند، زیرا مدل‌های بیزی اغلب واریانس کمتری دارند و کمتر تحت تأثیر نویز موجود در داده‌ها قرار می‌گیرند. در مدل‌های سلسله مراتبی بیزی، ساختارهای پیچیده‌تری تعریف می‌شود که به تخمین پارامترها کمک می‌کند و اطلاعات در سطوح مختلف استفاده می‌شود که می‌تواند از بیش‌برازش جلوگیری کند. علاوه بر این، رویکرد بیز با محاسبه توزیع کامل پارامترها به جای تخمین نقطه‌ای، عدم قطعیت در تخمین پارامترها را در نظر می‌گیرد و به

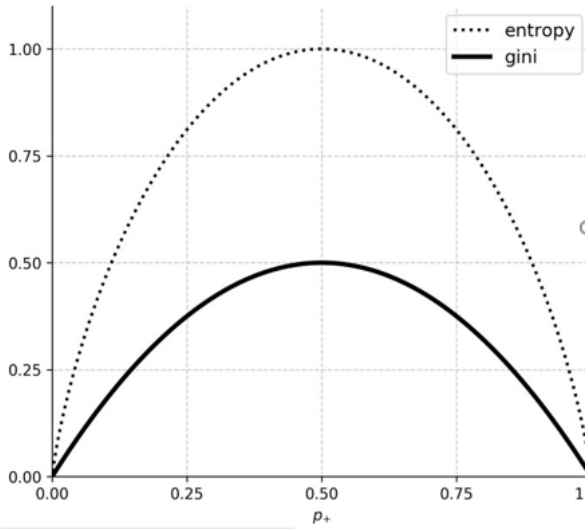
ویژه در شرایطی که داده‌ها کم هستند، از بیش‌برازش جلوگیری می‌کند. در نتیجه، این رویکرد به پایداری و دقت بالاتر مدل منجر می‌شود و از تطبیق بیش از حد به داده‌های آموزشی جلوگیری می‌کند.

۳ .استفاده از معیار information gain برای ساخت درخت در شرایطی که بعضی از ویژگی ها حالات زیادی دارند،مناسب نیست.

عبارت بالا درست است.

استفاده از معیار information gain برای ساخت درخت تصمیم در شرایطی که برخی ویژگی‌ها حالات زیادی دارند، مشکلاتی ایجاد می‌کند. این معیار به ویژگی‌هایی با تعداد حالات زیاد تمایل دارد، زیرا این ویژگی‌ها معمولاً افزایش بیشتری در اطلاعات ایجاد می‌کنند، حتی اگر این ویژگی‌ها همیشه مفید نباشند. این مسئله می‌تواند منجر به بیش‌برازش شود، به طوری که مدل بیش از حد به داده‌های آموزشی متناسب می‌شود و توانایی تعمیم به داده‌های جدید را از دست می‌دهد. ویژگی‌های با حالات زیاد، پیچیدگی مدل را افزایش داده و ممکن است مدل نهایی نتواند به خوبی بر روی داده‌های جدید عمل کند، همچنین این ویژگی‌ها می‌توانند به طور مصنوعی میزان اطلاعات را افزایش دهند و مدل را به سمت ساختاری پیچیده و غیرقابل تعمیم هدایت کنند. علاوه بر این، معیار information gain ممکن است ویژگی‌هایی را انتخاب کند که به طور واقعی کمکی به تفکیک داده‌ها نمی‌کنند، اما به دلیل تعداد حالات زیادشان، مقدار information gain بالایی دارند. این امر باعث می‌شود مدل نهایی شامل ویژگی‌های غیرمفید شود که کارایی مدل را کاهش می‌دهند. برای مقابله با این مشکل، معمولاً از معیارهای دیگری مانند Gain Ratio استفاده می‌شود که اثر تعداد حالات ویژگی‌ها را تعدیل می‌کنند. Gain Ratio به صورت خاص، معیار information gain را بر اساس تعداد حالات هر ویژگی نرمالیزه می‌کند و از ترجیح نامتناسب ویژگی‌های با حالات زیاد جلوگیری می‌کند. به طور خلاصه، استفاده از معیار information gain در این شرایط می‌تواند منجر به انتخاب ویژگی‌های غیرمفید، بیش‌برازش و کاهش تعمیم‌پذیری مدل شود، و معیارهای جایگزین مانند Gain Ratio می‌توانند به رفع این مشکلات کمک کنند.

همچنین در کلاس بحث شد که درخت‌های تصمیم از معیارهای مختلفی برای ساخت استفاده می‌کنند. مقایسه‌ای نیز انجام شد که نشان داد که معیار Gini از لحاظ سرعت و بار محاسباتی بهتر است. اما ممکن است در برخی مواقع استفاده از معیار Information Gain کافی نباشد. راه‌حل‌های مختلفی نیز برای انتخاب ویژگی‌ها مطرح شد که می‌توان از آنها استفاده کرد.



۴ .هر شبکهٔ عصبی چندلایه با توابع فعال ساز خطی در لایه های پنهان می تواند به عنوان یک شبکهٔ عصبی بدون هیچ لایه پنهانی نمایش داده شود.

متن بالا در کل درست است.

وقتی یک شبکه عصبی داریم که در لایه‌های پنهانش فقط از توابع فعال‌سازی خطی استفاده می‌کند، می‌توان گفت که این شبکه همان کارکردی را دارد که یک شبکه بدون لایه پنهان می‌تواند انجام دهد. دلیل این امر این است که با توابع فعال‌سازی خطی، تمام محاسباتی که در لایه‌ها انجام می‌شود به سادگی مجموعه‌ای از جمع و ضرب‌های ساده هستند. این بدان معناست که تمامی این لایه‌های میانی را می‌توان به یک لایه خطی فشرده کرد.

به بیان دیگر، وقتی همه توابع فعال‌سازی خطی هستند، داشتن چندین لایه ضروری نیست زیرا می‌توان همه آنها را به یک لایه ساده کاهش داد. اگرچه ممکن است این مدل در ابتدا پیچیده به نظر برسد، اما در واقع بسیار ساده‌تر است.

سوال دو

۲ پرسش دو

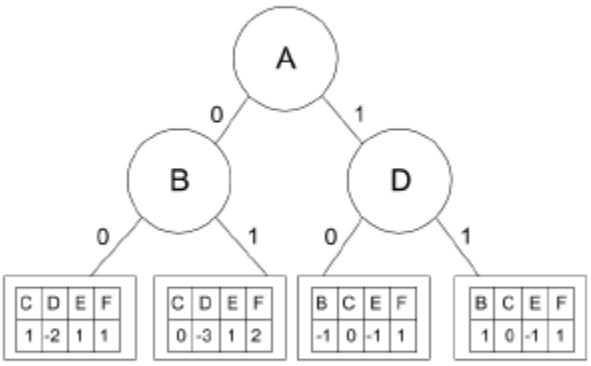
برای بهره‌برداری از ویژگی‌های مطلوب طبقه‌بندهای درخت تصمیم و پرسپترون، سارا الگوریتم جدیدی به نام «درخت پرسپترون» ایجاد کرده که ویژگی‌های هر دو را ترکیب می‌کند. درخت‌های پرسپترونی شبیه به درخت‌های تصمیم هستند؛ اما هر leaf node به‌جای مکانیزم رأی اکثریت، شامل یک پرسپترون است.

برای ایجاد یک درخت پرسپترون؛ اولین مرحله، اجرای یک الگوریتم یادگیری درخت تصمیم معمولی (مانند ID3) و انجام تقسیم‌بندی بر اساس ویژگی‌ها تا رسیدن به عمق حداکثر مشخص شده است. هنگامی که به عمق حداکثر می‌رسیم، در هر leaf node، یک پرسپترون روی ویژگی‌های باقی‌مانده که هنوز در آن شاخه استفاده نشده‌اند، آموزش داده می‌شود. دسته‌بندی یک نمونه جدید از طریق مراحل مشابهی انجام می‌شود. ابتدا نمونه از طریق درخت تصمیم بر اساس مقادیر ویژگی‌هایش گذر می‌کند. وقتی به یک leaf node می‌رسد، پیش‌بینی نهایی با اجرای پرسپترون متناظر در آن گره انجام می‌شود.

فرض کنید که دارای مجموعه داده‌ای با ۶ ویژگی دودویی $\{A, B, C, D, E, F\}$ و دو برچسب خروجی $\{-1, 1\}$ هستید. یک درخت پرسپترون با عمق ۲ روی این مجموعه داده‌ها در شکل ۱ آمده است. وزن‌های پرسپترون نیز در leaf nodeها آمده است (فرض کنید که بایاس برای هر پرسپترون $b = 1$ است).

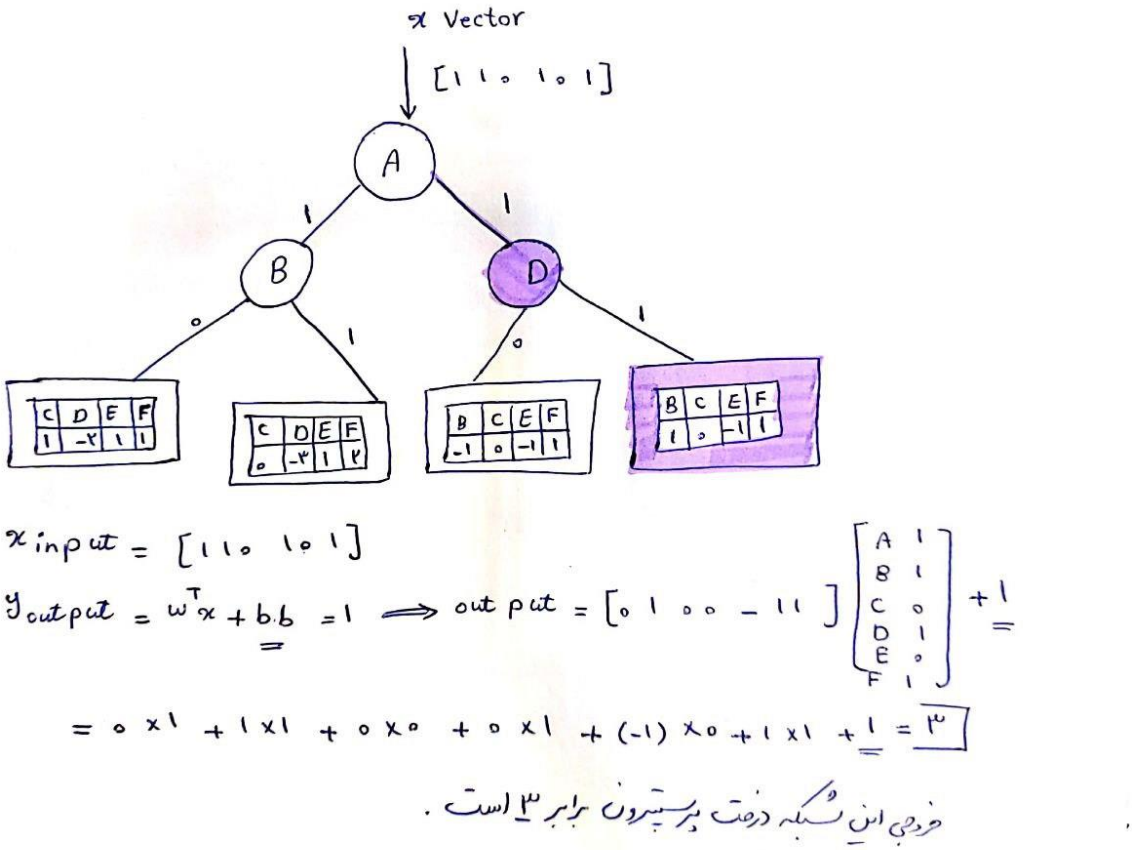
۱. برای نمونهٔ $\mathbf{x} = [1, 1, 0, 1, 0, 1]$، درخت پرسپترون داده‌شده چه برچسب خروجی‌ای را پیش‌بینی می‌کند؟

۲. آیا مرزتصمیم درخت پرسپترون همواره خطی است؟ برای مقادیر کوچک حداکثر عمق، کیفیت آموزش درخت تصمیم و درخت پرپسترون را با ذکر دلیل مقایسه کنید. آیا تفاوتی دارند؟



شکل ۱: درخت پرسپترون با عمق دو.

بخش اول)



این شبکه درخت پرسپترون با خروجی سه برای ورودی ذکر شده است. اگر به دقت توجه کنیم، در صورت سوال اشاره شده که برچسب‌های خروجی تنها شامل دو مقدار منفی و مثبت ۱ هستند. بنابراین، نمی‌توانیم این خروجی را به عنوان خروجی اصلی شبکه در نظر بگیریم و باید یک نگاشت (mapping)انجام دهیم. به عبارت دیگر، مقادیر بالای صفر را به عنوان ۱ و مقادیر زیر صفر را به عنوان -۱ در نظر بگیریم تا بتوانیم خروجی مورد نظر شبکه را بدست آوریم.

با این حال، هیچ داده‌ای در مورد این نگاشت در اطلاعات سوال ذکر نشده است. به عنوان مثال، می‌توانیم برای مواقعی که خروجی مثبت شده است، مقدار ۱ را برای خروجی در نظر بگیریم، اما برای این کار نیاز به داده‌های بیشتری داریم که در سوال ذکر نشده است.

همچنین با جایگذاری مقادیر به این پرسپترون داریم:

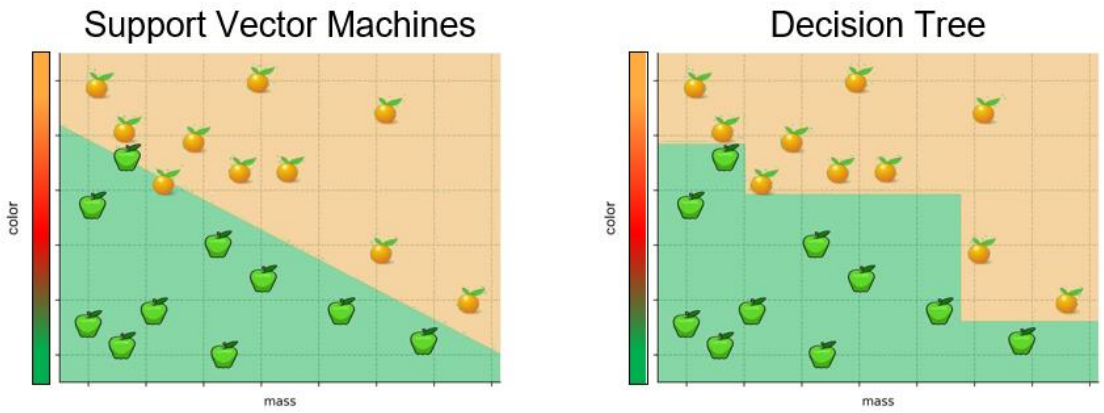
$$out = sign(1 + 0 + 0 + 1 + 1) = sign(3) = 1$$

بخش دوم)

خروجی الگوریتم‌های SVM معمولاً یک خط را برای عملیات کلاس‌بندی تولید می‌کند، به این معنا که داده‌ها را بر اساس یک خط جدا می‌کند و عملیات طبقه‌بندی را انجام می‌دهد. اما درخت تصمیم به این صورت عمل نمی‌کند. برای درخت پرسپترون، زیرا خروجی یک پرسپترون است و از آنجایی که فعال‌سازی ما یک hard limit با دو کلاس است (یعنی -۱ یا ۱) و بیشتر از دو کلاس نیست، خروجی آن غیرخطی است. اما در لایه‌های قبلی، پس از رسیدن به گره‌های پایانی (leaf node)، به دلیل انجام یک رگرسیون خطی، عملیات کاملاً خطی است.

زمانی که یک پرسپترون در لایه آخر این شبکه قرار می‌گیرد، مشابه این است که یک خط ترسیم شده باشد که داده‌ها را به دو کلاس خروجی تقسیم می‌کند. در این حالت، مانند این است که یک خط رسم شده باشد و مرز تصمیم‌گیری در نهایت به شکل یک خط خواهد بود. این نکته مهمی است که نشان می‌دهد در نهایت معمولاً در فرآیند طبقه‌بندی به یک تقسیم خطی از داده‌ها می‌پردازیم، گرچه ممکن است در مراحل میانی، فرآیند به صورت غیرخطی انجام شود.

در کلاس حل تمرین هم مرز تصمیم گیری که برای درخت تصمیم دیده بودیم به صورت های زیر است :



درخت پرسپترون (Perceptron Tree)و درخت تصمیم (Decision Tree)دو الگوریتم متفاوت برای مسائل طبقه‌بندی هستند که هر کدام ویژگی‌ها و محدودیت‌های خود را دارند.

درخت پرسپترون یک روش خطی برای طبقه‌بندی داده‌ها است. این به این معناست که درخت پرسپترون با استفاده از ترکیب خطی از ویژگی‌ها، مرزهای تصمیم را ایجاد می‌کند که معمولاً به صورت خطی است. اما اگر

داده‌ها به طور کامل قابل جداسازی خطی نباشند، درخت پرسپترون نمی‌تواند یک مرز تصمیم دقیق و بهینه ارائه دهد.

۲. برای مقادیر کوچک حداکثر عمق، کیفیت آموزش درخت تصمیم و درخت پرسپترون را با ذکر دلیل مقایسه کنید.

درخت تصمیم (Decision Tree)

کیفیت آموزش: درخت تصمیم معمولاً به طور خطی با استفاده از معیارهایی مانند میانگین مربعات خطا (Mean Squared Error)یا معیارهای دیگر، مرزهای تصمیم غیرخطی ایجاد می‌کند. این مرزهای تصمیم می‌توانند به طور غیرخطی باشند و بهینه‌سازی شده برای داده‌هایی با الگوهای مختلف عمل کنند.

مقدار کوچک حداکثر عمق: حداکثر عمق متناسب با پیچیدگی داده‌ها و الگوهای موجود است. مقادیر کوچک حداکثر عمق معمولاً به معنای استفاده از مرزهای تصمیم ساده‌تر و کمتر پیچیده است، که می‌تواند برای جلوگیری از بیش‌برازش (overfitting)در داده‌های کم و عملکرد بهتر در داده‌های جدید مفید باشد.

درخت پرسپترون (Perceptron Tree)

کیفیت آموزش: با وجود اینکه درخت پرسپترون از مرزهای تصمیم خطی استفاده می‌کند، کیفیت آموزش آن معمولاً به صورت محدودتری نسبت به درخت تصمیم است. این به دلیل محدودیت‌های خطی و عدم توانایی مدل در فراگیری الگوهای پیچیده غیرخطی است.

مقدار کوچک حداکثر عمق: مقدار کوچک حداکثر عمق ممکن است باعث ایجاد مرزهای تصمیم ساده‌تر شود که می‌تواند برای داده‌هایی با الگوهای ساده مؤثر باشد. اما این محدودیت ممکن است باعث افزایش خطا در مواردی با الگوهای پیچیده‌تر شود.

در پاسخ به این سوال که ایا این ها تفاوتی دارند یا خیر ؟ باید گفت :

بله، این دو الگوریتم با توجه به ویژگی‌ها و محدودیت‌های خود تفاوت‌های مهمی دارند که باید در انتخاب الگوریتم مورد استفاده برای هر مسئله دقت شود. درخت تصمیم به عنوان یک روش انعطاف‌پذیر و قابل تفسیر شناخته می‌شود، در حالی که درخت پرسپترون به دلیل محدودیت‌های خطی‌اش و عدم توانایی در مدل‌سازی الگوهای پیچیده‌تر، معمولاً برای مسائلی با الگوهای ساده‌تر استفاده می‌شود.

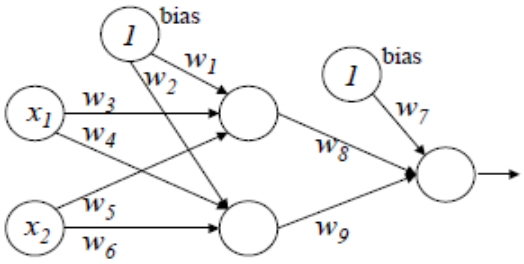
پس در کل ، اولا، ایده که مرزهای درخت تصمیم همواره خطی باشند، غلط است. در واقع، مرزهای این درخت‌ها می‌توانند به صورت غیرخطی نیز باشند.

در قسمت دوم، با کاهش عمق درخت، تصمیم‌گیری در برگ‌ها بر اساس اکثریت نمونه‌ها انجام می‌شود. این ممکن است منجر به این شود که برخی از ویژگی‌ها و حالات نادیده گرفته شوند، به دلیل اینکه تعداد زیادی ویژگی و حالت موجود است و برخی از آنها ممکن است مورد بررسی و طبقه‌بندی قرار نگیرند. اما در درخت پرسپترون، این مسئله رخ نمی‌دهد، زیرا در برگ‌های نهایی، می‌توان همه ویژگی‌های دیده‌نشده را برای پیش‌بینی کلاس مورد نظر در نظر گرفت.

سوال سه

۳ پرسش سه

شبکهٔ عصبی آورده‌شده در شکل ۲ را برای یک مسألهٔ طبقه‌بندی دوکلاسه در نظر بگیرید. فرض کنید که لایه‌های میانی از تابع فعال‌ساز خطی $(h(z) = cz)$ و لایهٔ خروجی از تابع سیگموئید $(g(z) = \frac{1}{1+e^{-z}})$ استفاده می‌کند. این شبکه می‌خواهد یک تابع برای $P(Y = 1 \mid X, w)$ که در آن $X = (x_1, x_2)$ و $W = (w_1, w_2, \ldots, w_9)$ است را یاد بگیرد.



شکل ۲: شبکهٔ عصبی سوال سوم.

۱. خروجی شبکهٔ عصبی $P(Y = 1 \mid X, w)$ را بر حسب پارامترهای شبکه (W, x) و ثابت c نوشته و مرز تصمیم نهایی را به دست آورید.

۲. آیا می‌توان یک شبکهٔ عصبی بدون لایهٔ مخفی به دست آورد که معادل شبکهٔ عصبی فوق باشد؟ در صورت وجود، شبکهٔ پیشنهادی‌تان را رسم کنید.

۱. خروجی شبکه عصبی $P(Y = 1 \mid X, w)$را بر حسب پارامترهای شبکه (W, x)ثابت c نوشته و مرز تصمیم نهایی را بدست اورید :

شبکه عصبی داده شده یک شبکه پرسپترون چندلایه (MLP)با یک لایه مخفی یا هیدن لایر و یک نرون خروجی است. تابع فعال‌سازی برای نرون‌های لایه مخفی و خروجی به صورت زیر است:

- لایه مخفی: تابع سیگموئید به صورتی که در سوال امده است.

فرض می کنیم که بردار ورودی $X = (x1, x2)$ و وزن‌ها $W = (w1, w2,\dots , w9)$ هستند.

محاسبات به صورت زیر انجام می‌شود:

وقتی می‌خواهیم بفهمیم چه خروجی‌ای از شبکه عصبی بدست می‌آید و کجا مرز تصمیم قرار می‌گیرد، باید از ورودی تا خروجی مسیر را دنبال کنیم. حالا فرض می کنیم که ما خروجی‌های لایه میانی را داریم و آنها را با h1 و h2 نشان می‌دهیم. این دلیلی دارد که از این نام‌ها استفاده می‌کنیم، زیرا در لایه میانی با تابع فعال‌سازی خطی کار می‌کنیم.

$$h1 = c(w3 * x1 + w5 * x2 + w1 * b)$$

$$h2 = c(w4 * x1 + w6 * x2 + w2 * b)$$

حالا برای خروجی نهایی، از تابع سیگموئید استفاده می‌کنیم. پس می‌شه:

$$P(Y=1|X,W) = g(w8 * h1 + w9 * h2 + w7 * b) = 1/ 1 + exp(-(w8(c(w3 * x1 + w5 * x2 + w1 * b)) * + w9 * (c(w4 * x1 + w6 * x2 + w2 * b)) + w7 * b) = 1/ 1 + exp(-(w8 * h1 + w9 * h2 + w7 * b)$$

حالا برای مرز تصمیم که برای طبقه‌بندی دودوییه، جایی که $P(Y=1|X,W)$ برابر با ۰,۵ شود، معادله آن به صورت زیر است :

$$W8 * c(w3 * x1 + w5 * x2 + w1 * 1) + w7c(w4 * x1 + w6 * x2 + w2 * 1) + w7 * 1 = 0$$

این معادله خط تصمیم نهایی را در فضای ویژگی‌ها $(x1, x2)$ نشان می‌دهد که یه خط غیرخطی است!

چون اثرات تعاملی بین x1 و x2 را نشان می دهد.

۲. آیا خروجی شبکه عصبی بدون لایه مخفی به دست آمده، معادل شبکه عصبی فوق خواهد بود؟ در صورت درست بودن، پارامترهای آن را توضیح دهید:

اگر بخواهیم یک شبکه عصبی بدون لایه مخفی برای شبکه عصبی موجود طراحی کنیم، این امر امکان‌پذیر است. دلیل این امر این است که از نظر جبری، هر شبکه‌ای که شامل لایه‌های مخفی باشد، را می‌توان با یک شبکه تک لایه جایگزین کرد، که تعداد کافی از گره‌ها را داشته باشد.

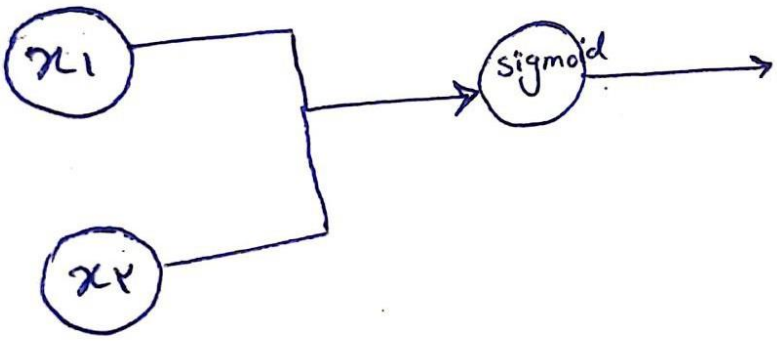
مفروض می‌کنیم که شبکه پیشنهادی بدون لایه مخفی به این صورت است که ورودی‌های x1 و x2 مستقیماً به یک گره در لایه خروجی وارد می‌شوند. تابع فعال‌سازی اینجا از تابع سیگموئید استفاده می‌شود. خروجی شبکه که با P(Y=1|X,W) نشان داده می‌شود، همان خروجی شبکه اصلی است که دارای لایه مخفی است.

حالا خروجی این شبکه تک لایه به این صورت می‌شود:

$$P(Y=1|X,W) = g(w0 + w1 * x1 + w2 * x2 + w3* x1x2 + w4 * x1^2 + w5 * x2^2)$$

در اینجا W که برابر با (w0, w1, w2, w3, w4, w5)وزن‌های شبکه هستند و باید از داده‌های آموزشی اموزش ببینن.

یکی از مزایای این شبکه تک لایه این است که به طور قابل توجهی ساده‌تر از شبکه اصلی است که دارای لایه مخفی است. با این حال، هنوز می‌تواند مرزهای تصمیم غیرخطی و پیچیده‌ای را نمایش دهد. اما باید به یاد داشت که یافتن مجموعه بهینه از وزن‌ها در این شبکه ممکن است کمی دشوارتر از شبکه اصلی باشد.



سوال چهار

ض کنید داده‌های ۹ نوامبر در اختیار شماست و می‌خواهید برای داده‌های ۱۷ نوامبر سیستم تشخیص عیب مبتنی بر شبکه‌های صبی طراحی کنید تا بتواند عیوب را تشخیص داده و تفکیک کند. این شبکه را طراحی کنید و فرآیند اجرایی و نتایجی که ارش می‌کنید حتماً شامل تمام موارد زیر باشد و به‌صورت صریح و مشخص ، پاسخ نهایی‌ای که به هر مورد مطرح‌شده اشاره رد را مشخص کنید:

۱. تعیین و انجام تقسیم‌بندی مناسب برای داده‌ها (آموزش، اعتبارسنجی و آزمون)، و تعیین و انجام روش‌های پیش‌پردازشی مناسب (از بین مواردی هم‌چون: نرمال‌سازی، استانداردسازی، حذف یا جایگزینی داده‌های خالی، حلّ مشکل عدم توازن در کلاس‌ها و... هرکدام که لازم می‌دانید را انجام دهید).
لازم است که در گزارش خود گزارشی از تعداد و توزیع درصدی داده‌ها در قسمت‌های مختلف (آموزش، اعتبارسنجی و آزمون) و در کلاس‌های مختلف را نمایش دهید.

۲. استفاده از روش‌های آموخته‌شده برای استخراج ویژگی، انجام حداقل ۵ روش آن، رسم ماتریس هم‌بستگی، تحلیل آن و حذف برخی ویژگی‌ها در صورت لزوم (با توضیح). استفاده از روش‌های فرکانسی مانند FFT نمرهٔ امتیازی دارد.

۳. رسم شبکهٔ عصبی و کل ساختار پیشنهادی از ابتدا تا انتها در قالب یک بلوک‌دیاگرام و مشخص‌کردن مواردی مانند تعداد لایه‌های پنهان، توابع فعال‌ساز لایه‌های میانی و لایهٔ خروجی، تابع اتلاف و بهینه‌ساز انتخابی و... به‌صورت آیتم‌بندی‌شده و مشخص.
استفاده از حداقل یکی از تکنیک‌های لایهٔ دورریز[1] ، برنامه‌ریز نرخ یادگیری[2] و توقف زودهنگام[3] الزامی است.

۴. رسم نمودار تابع اتلاف و دقت در فرآیند آموزش برای داده‌های آموزش و اعتبارسنجی برای حداقل دو حالت مختلف از شبکه (مثلاً تغییر در تعداد لایه‌های پنهان و یا تغییر در بهینه‌ساز).
دقت داشته باشید که فقط یک عدد نمودار خروجی برای تابع اتلاف مجاز است. یعنی تمام مقایسه‌های مربوط به توابع اتلاف روی یک نمودار واحد و با رنگ‌های مختلف انجام شود.

۵. رسم ماتریس درهم‌ریختگی به‌صورت درصدی برای داده‌های آزمون و نمایش classification_report به تفکیک هر کلاس.

۶. تکرار مرحلهٔ پیاده‌سازی با استفاده از یکی از روش‌های K-Fold Cross-validation یا Stratified K-Fold Cross-validation و نمایش ماتریس درهم‌ریختگی به ازای هر فُلد و به‌صورت کلی.

۷. اوزان مربوط به یکی از مدل‌های خود (ترجیحاً بهترین مدل) را ذخیره کنید و روی گوگل‌درایو بارگذاری کنید. سپس، در یک سلول‌کد دستوراتی بنویسید که فایل وزن را با استفاده از gdown فراخوانی کند و فرآیند تست را روی دادهٔ نمونهٔ تست که آن هم با gdown فراخوانی شده اجرا کند.

۸. پیاده‌سازی تمام موارد فوق ممکن بود؟ کدام‌یک عملی نبود و انجام ندادید؟ چرا؟ اگر به هر دلیلی نتوانستید پیاده‌سازی را با استفاده از دیتاست خواسته‌شده در این سوال انجام دهید، می‌توانید پیاده‌سازی را با استفاده از یکی از دیتاست‌هایی که در طول درس آموزش دیدید انجام دهید و حداکثر ۴۰ درصد از نمرهٔ این سوال را دریافت کنید. تشخیص میزان نمرهٔ دریافتی بر اساس دیتاست انتخابی شما و برعهدهٔ مصحح خواهد بود.

ابتدا کتابخانه های لازم را ایمپورت می کنیم :

Fatemeh Amiri

student number: 40202424

Midterm Machine Learning - Dr Aliyari

spring 2024

```
#import library
import numpy as np
import itertools
from sklearn.neural_network import MLPClassifier, MLPRegressor
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.datasets import load_diabetes, load_digits
from imblearn.under_sampling import RandomUnderSampler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
import pandas as pd
from sklearn.datasets import make_classification, make_regression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from mlxtend.plotting import plot_decision_regions
import __main__
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
from pandas_datareader.data import DataReader
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.impute import SimpleImputer
from sklearn.neural_network import MLPClassifier
import seaborn as sns
```

سپس دیتا را می خوانیم :

```
!pip install --upgrade --no-cach-dir gdown
! gdown 11nnRz-cQVv0lnldxuQrkmxhbdJ0MMiHv
```

دیتاهای این دو بخش را جدا می کنیم به صورت زیر :

```
[9]  dataset = sio.loadmat('/content/drive/MyDrive/MachineLearning/midterm/DATA.mat')
     data_NOV9 = pd.DataFrame(dataset['NOV9'])
     data_NOV17 = pd.DataFrame(dataset['NOV17'])
```

به طور کلی، ساختار داده‌ها را می‌توانیم به صورت زیر مشاهده کنیم:

```
df_NOV17 = pd.DataFrame(data_NOV17)
df_NOV17.head()
```

```
('NOV17', (86400, 17), 'double')
('NOV9', (86400, 17), 'double')
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0.0	642.0	359.0	97.4	351.3	33.9	33.1	47.6	130.5	134.6	12.0	26.6	67.2	271.6	140.1	172.0	133.6
1	1.0	635.9	356.8	97.4	352.6	35.0	32.9	47.0	130.5	134.6	12.0	26.6	67.0	271.4	140.1	172.0	133.6
2	2.0	638.8	356.0	97.4	353.4	33.9	34.0	47.4	130.5	134.6	12.0	26.6	65.9	271.5	140.1	172.0	133.6
3	3.0	637.6	350.7	97.4	354.1	34.4	33.3	46.8	130.5	134.6	12.0	26.5	66.9	271.1	140.1	171.9	133.6
4	4.0	635.7	360.4	97.4	354.3	33.7	33.3	47.4	130.5	134.6	12.0	26.5	67.1	271.3	140.1	171.9	133.6

Next steps: View recommended plots

می‌بینیم که نام ویژگی‌ها در این دیتاست مشخص نیستند. این ممکن است به دلیل جدا نشدن داده‌ها با جداکننده مناسب باشد یا اینکه دیتاست به همین شکل ذخیره شده است. همچنین، ستون اول شامل شماره‌های نمونه‌ها است که نام نمونه‌برداری برای هر ستون ذخیره شده است. می‌توانیم این ستون را حذف کنیم، زیرا در آموزش مدل ما تاثیری ندارد.

```
df_NOV9 = pd.DataFrame(data_NOV9)
df_NOV9
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0.0	699.1	300.6	97.8	296.2	42.7	42.0	46.1	129.1	133.4	12.2	26.4	53.0	242.0	136.9	164.0	132.1
1	1.0	697.7	298.9	97.8	295.4	41.8	41.6	45.3	129.1	133.3	12.2	26.4	51.6	241.4	136.9	164.1	132.1
2	2.0	696.2	301.8	97.8	295.4	42.9	41.4	46.6	129.1	133.4	12.2	26.3	50.3	241.6	136.9	164.1	132.1
3	3.0	702.8	300.9	97.7	295.1	43.1	41.5	46.0	129.1	133.4	12.2	26.2	50.2	241.6	136.9	164.2	132.1
4	4.0	701.3	292.6	97.8	294.6	42.6	42.3	47.0	129.1	133.4	12.2	26.1	51.8	241.3	136.8	163.9	132.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
86395	86395.0	598.8	373.1	97.0	379.9	31.4	30.0	46.9	128.7	132.3	13.5	26.5	69.9	276.1	140.4	172.0	133.4
86396	86396.0	601.5	365.6	97.0	378.9	30.7	30.7	46.4	128.7	132.3	13.5	26.5	69.5	276.1	140.4	172.0	133.4
86397	86397.0	597.8	374.8	97.0	378.5	30.1	30.5	46.4	128.7	132.3	13.5	26.5	70.4	276.2	140.4	172.0	133.4
86398	86398.0	593.7	379.7	97.0	378.8	30.3	30.3	46.1	128.7	132.3	13.5	26.5	70.5	276.2	140.4	172.0	133.4
86399	86399.0	582.7	377.8	97.0	380.3	29.8	30.5	46.2	128.7	132.3	13.5	26.5	71.7	276.3	140.4	171.8	133.4

86400 rows × 17 columns

از دیتاست دید کلی به دست می اوریم :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86400 entries, 0 to 86399
Data columns (total 17 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       86400 non-null  float64
 1   1       86400 non-null  float64
 2   2       86400 non-null  float64
 3   3       86400 non-null  float64
 4   4       86400 non-null  float64
 5   5       86400 non-null  float64
 6   6       86400 non-null  float64
 7   7       86400 non-null  float64
 8   8       86400 non-null  float64
 9   9       86400 non-null  float64
 10  10      86400 non-null  float64
 11  11      86400 non-null  float64
 12  12      86400 non-null  float64
 13  13      86400 non-null  float64
 14  14      86400 non-null  float64
 15  15      86400 non-null  float64
 16  16      86400 non-null  float64
dtypes: float64(17)
memory usage: 11.2 MB
```

داده های دیتاست پس از حذف ستون اول به شکل زیر اند :

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	642.0	359.0	97.4	351.3	33.9	33.1	47.6	130.5	134.6	12.0	26.6	67.2	271.6	140.1	172.0	133.6
1	635.9	356.8	97.4	352.6	35.0	32.9	47.0	130.5	134.6	12.0	26.6	67.0	271.4	140.1	172.0	133.6
2	638.8	356.0	97.4	353.4	33.9	34.0	47.4	130.5	134.6	12.0	26.6	65.9	271.5	140.1	172.0	133.6
3	637.6	350.7	97.4	354.1	34.4	33.3	46.8	130.5	134.6	12.0	26.5	66.9	271.1	140.1	171.9	133.6
4	635.7	360.4	97.4	354.3	33.7	33.3	47.4	130.5	134.6	12.0	26.5	67.1	271.3	140.1	171.9	133.6

طبق دستورالعمل سوال، ابتدا باید پیش‌پردازش‌های لازم را روی دیتاست انجام دهیم. این شامل نرمال‌سازی داده‌ها و بررسی توازن در داده‌های هدف می‌شود تا اطمینان حاصل کنیم که هر کلاس تعداد یکسانی نمونه دارد یا خیر. همچنین، می‌توانیم این دیتاست را از لحاظ آماری نیز مورد بررسی قرار دهیم.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
count	86400.000000	86400.000000	86400.000000	86400.000000	86400.000000	86400.000000	86400.000000	86400.000000	86400.000000	86400.000000	86400.000000	86400.000000	86400.000000	86400.000000	86400.000000	86400.000000
mean	665.175924	339.236698	95.670477	340.357216	37.363172	36.485388	46.747105	128.365163	132.272375	12.975722	23.130479	64.120027	257.463932	138.445428	164.330174	132.180851
std	48.113068	34.694585	3.079839	39.560152	6.474250	5.742347	2.183006	0.704929	0.933178	0.621999	2.673218	7.015711	18.795637	1.976538	8.323212	1.228520
min	491.600000	205.600000	69.600000	198.500000	26.000000	1.800000	37.000000	125.600000	127.800000	10.300000	17.600000	41.100000	184.200000	130.400000	126.000000	126.600000
25%	630.300000	307.900000	95.900000	301.800000	33.100000	32.500000	46.100000	127.900000	131.700000	12.800000	20.800000	59.200000	241.500000	136.800000	160.300000	131.500000
50%	652.000000	353.600000	96.300000	359.000000	34.700000	34.100000	46.600000	128.300000	132.300000	13.100000	23.300000	65.900000	266.700000	139.400000	165.900000	132.500000
75%	706.700000	365.600000	96.600000	370.800000	41.900000	41.100000	47.200000	128.800000	132.900000	13.300000	24.700000	69.900000	271.200000	139.900000	169.700000	133.000000
max	812.900000	432.200000	97.800000	428.200000	100.000000	67.600000	94.700000	130.700000	134.600000	14.200000	29.700000	77.100000	280.300000	140.700000	181.100000	134.400000

حالا داده های تست را برسی می کنیم :

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0.0	642.0	359.0	97.4	351.3	33.9	33.1	47.6	130.5	134.6	12.0	26.6	67.2	271.6	140.1	172.0	133.6
1	1.0	635.9	356.8	97.4	352.6	35.0	32.9	47.0	130.5	134.6	12.0	26.6	67.0	271.4	140.1	172.0	133.6
2	2.0	638.8	356.0	97.4	353.4	33.9	34.0	47.4	130.5	134.6	12.0	26.6	65.9	271.5	140.1	172.0	133.6
3	3.0	637.6	350.7	97.4	354.1	34.4	33.3	46.8	130.5	134.6	12.0	26.5	66.9	271.1	140.1	171.9	133.6
4	4.0	635.7	360.4	97.4	354.3	33.7	33.3	47.4	130.5	134.6	12.0	26.5	67.1	271.3	140.1	171.9	133.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
86395	86395.0	661.1	383.6	96.0	367.9	34.0	32.9	48.3	130.6	134.5	13.5	21.1	77.2	275.1	140.4	173.0	133.7
86396	86396.0	669.6	363.4	96.0	366.1	33.8	32.7	48.0	130.6	134.5	13.5	21.1	76.3	274.6	140.4	172.9	133.7
86397	86397.0	664.2	372.6	96.0	363.0	34.4	32.9	48.5	130.6	134.5	13.5	21.0	75.6	274.6	140.4	173.1	133.7
86398	86398.0	670.3	362.9	96.0	361.5	33.9	32.9	47.9	130.6	134.5	13.5	21.0	75.4	275.0	140.4	173.0	133.7
86399	86399.0	665.9	363.1	96.0	361.1	34.2	34.4	48.2	130.6	134.5	13.5	21.0	76.7	275.0	140.4	173.1	133.8

86400 rows × 17 columns

لیبل میزنیم.. فالت و نرمال را عددصفر و یک نسبت می دهیم :

```
b
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	label
0	54600.0	503.5	413.2	97.4	404.4	32.1	31.9	46.4	129.5	133.6	12.7	22.2	71.3	258.0	138.8	161.7	132.3	1
1	54601.0	498.4	410.3	97.5	404.9	32.6	30.1	46.7	129.5	133.6	12.7	22.2	70.6	258.5	138.8	161.7	132.4	1
2	54602.0	501.8	409.8	97.5	405.3	33.1	31.7	47.2	129.6	133.6	12.7	22.3	71.9	258.5	138.8	161.6	132.4	1
3	54603.0	504.3	411.7	97.5	405.4	33.8	31.8	47.4	129.5	133.6	12.7	22.3	70.2	258.4	138.8	161.6	132.3	1
4	54604.0	506.2	405.6	97.5	405.0	33.9	32.9	47.8	129.5	133.5	12.7	22.2	71.2	258.0	138.8	161.7	132.3	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
197	56766.0	780.7	266.2	97.1	238.2	63.3	52.9	58.1	130.7	134.8	12.9	20.6	67.3	264.8	139.4	169.6	133.2	0
198	56767.0	784.9	266.7	97.1	234.2	63.2	53.8	57.9	130.7	134.8	12.9	20.6	68.3	264.8	139.4	169.5	133.2	0
199	56768.0	785.8	261.1	97.1	230.5	62.7	55.6	57.5	130.7	134.8	12.9	20.7	68.0	265.1	139.4	169.7	133.2	0
200	56769.0	789.7	260.1	97.1	227.8	62.5	55.2	57.1	130.7	134.8	12.9	20.7	68.9	265.2	139.4	170.0	133.2	0
201	56770.0	788.8	265.2	97.2	224.7	61.5	54.1	56.4	130.7	134.8	12.9	20.9	68.1	265.6	139.4	170.0	133.2	0

202 rows × 18 columns

```
label
0    202
1    483
dtype: int64
```

نا متوازن هستند !

مشاهده می‌شود که عدم توازن در داده‌های دیتاست وجود دارد. با این حال، پیش از اقدام به متعادل‌سازی، بهتر است عملکرد شبکه را ارزیابی کنیم. سپس، در صورت نیاز به بهبود عملکرد، می‌توانیم به بررسی و اعمال روش‌های متعادل‌سازی داده‌ها بپردازیم.

فیچر اکسترکشن ! مثل سوال ۲ تمرین

```python
# Define feature extraction functions
def calculate_standard_deviation(data):
    return np.std(data, axis=0)

def find_peak(data):
    return np.max(data, axis=0)

def calculate_crest_factor(data):
    peak_value = find_peak(data)
    rms_value = np.sqrt(np.mean(data ** 2, axis=0))
    return peak_value / rms_value

def calculate_skewness(data):
    return skew(data, axis=0)

def calculate_clearance_factor(data):
    peak_to_peak = np.ptp(data, axis=0)
    rms_value = np.sqrt(np.mean(data ** 2, axis=0))
    return peak_to_peak / rms_value

def calculate_peak_to_peak(data):
    return np.ptp(data, axis=0)

def calculate_shape_factor(data):
    mean_value = np.mean(data, axis=0)
    std_value = np.std(data, axis=0)
    return std_value / mean_value

def calculate_impact_factor(data):
    peak_value = find_peak(data)
    mean_value = np.mean(data, axis=0)
    return peak_value / mean_value
```

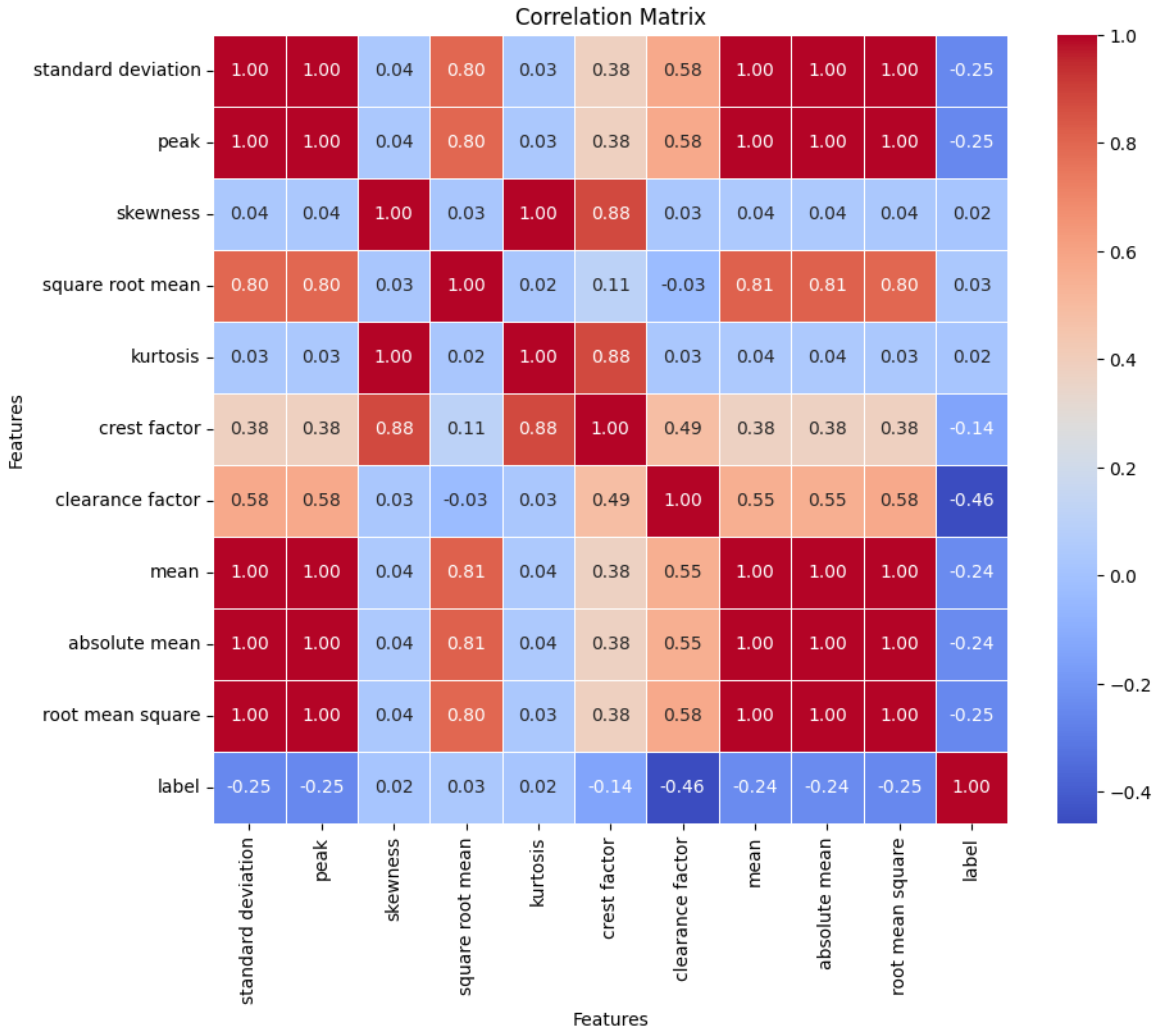
به همین ترتیب، می‌توانیم دیتاست تولید شده با ویژگی‌هایی که خودمان استخراج کرده‌ایم و برچسب‌هایی که اضافه کرده‌ایم را به عنوان یک DataFrame جدید ذخیره کنیم و ادامه مراحل را با استفاده از آن پیش ببریم.

	standard deviation	peak	skewness	square root mean	kurtosis	crest factor	clearance factor	mean	absolute mean	root mean square	label
0	13463.974892	57275.0	3.879598	555.379531	13.054089	4.242075	103.127675	3329.700000	3329.700000	13501.647587	1
1	13464.322053	57276.0	3.879601	554.656283	13.054106	4.242078	103.263952	3329.294444	3329.294444	13501.874539	1
2	13464.387318	57277.0	3.879597	555.933208	13.054086	4.242074	103.028564	3330.050000	3330.050000	13502.122330	1
3	13464.685069	57278.0	3.879603	555.585177	13.054114	4.242077	103.094903	3329.833333	3329.833333	13502.349320	1
4	13464.937010	57279.0	3.879610	555.509884	13.054149	4.242079	103.110677	3329.794444	3329.794444	13502.577010	1
...	...	...	...	...	...	...	...	...	...	...	...
680	13344.572834	56766.0	3.879438	551.395728	13.053296	4.242027	102.949655	3299.766667	3299.766667	13381.811642	0
681	13344.787786	56767.0	3.879428	551.555971	13.053250	4.242023	102.921558	3299.944444	3299.944444	13382.057926	0
682	13345.121617	56768.0	3.879430	551.028998	13.053258	4.242025	103.021801	3299.600000	3299.600000	13382.287401	0
683	13345.360052	56769.0	3.879421	550.910229	13.053212	4.242022	103.045827	3299.683333	3299.683333	13382.532511	0
684	13345.621169	56770.0	3.879420	550.511191	13.053210	4.242022	103.122336	3299.638889	3299.638889	13382.767478	0

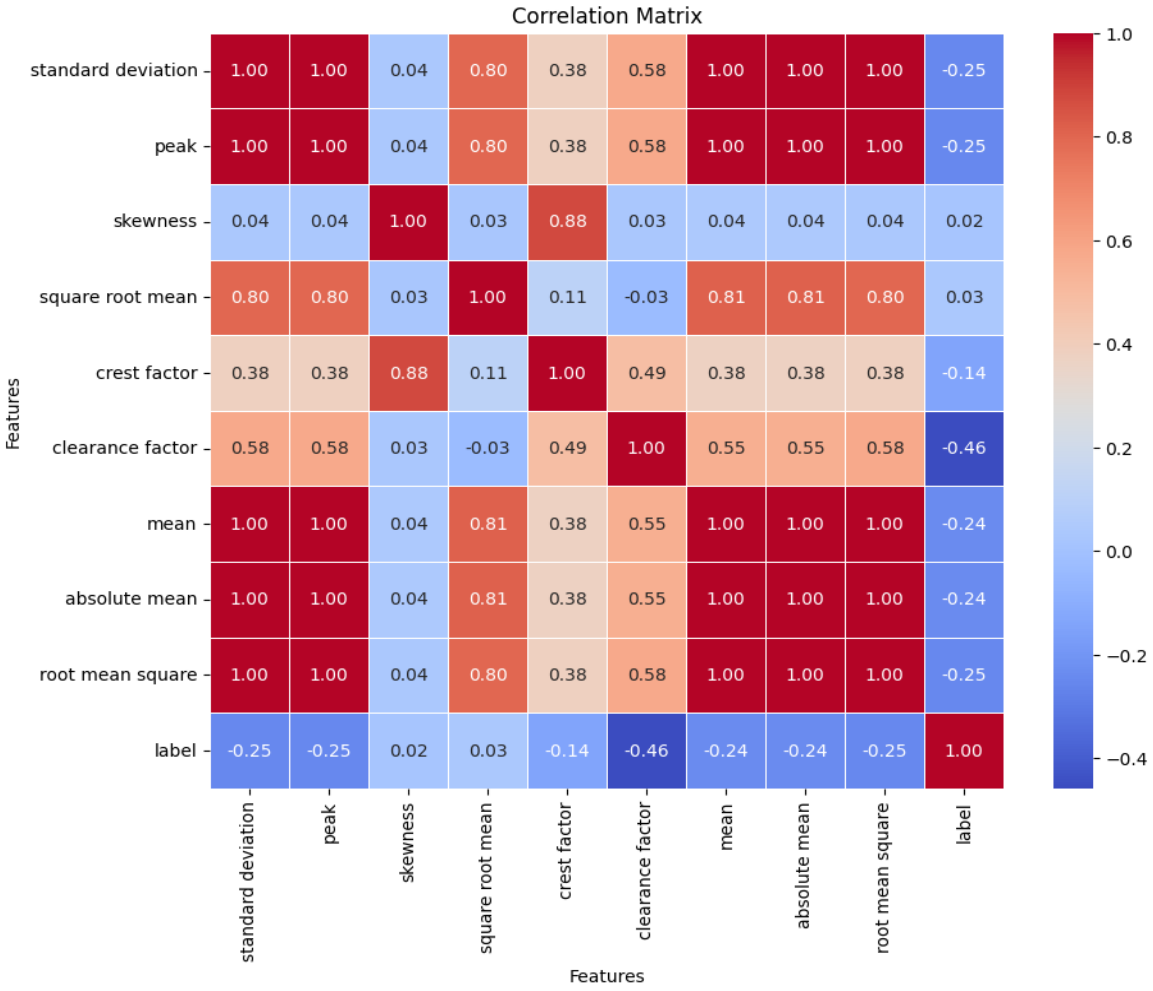
میتوانیم برای دیتاست های بدست امده فالت و نرمال بصورت جداگانه ماتریس Confusion را رسم کنیم:

اگر دقت کنید، سطر آخر لیبل‌ها است که ما قصد داریم خرابی را از آن تشخیص دهیم. مشاهده می‌کنیم که ویژگی " "clearance factorنسبت به سایر ویژگی‌ها وابستگی خطی بیشتری دارد و می‌توانیم از آن برای آموزش مدل استفاده کنیم. تحلیل این ماتریس به این صورت است که هر چقدر مقادیر این سطر نسبت به ویژگی‌ها بیشتر باشد، وابستگی خطی بیشتری وجود دارد و در نتیجه احتمالاً می‌توانیم خرابی را بهتر تشخیص دهیم.

حتی ویژگی‌هایی که مقادیر کمی دارند، نبودنشان لزوماً عملکرد مدل را کاهش نمی‌دهد، زیرا این ماتریس تنها وابستگی‌های خطی را بررسی می‌کند. در صورتی که یک ویژگی به صورت غیر خطی به هدف وابسته باشد، ممکن است در نظر نگرفتن آن منجر به کاهش عملکرد و دقت شبکه شود.

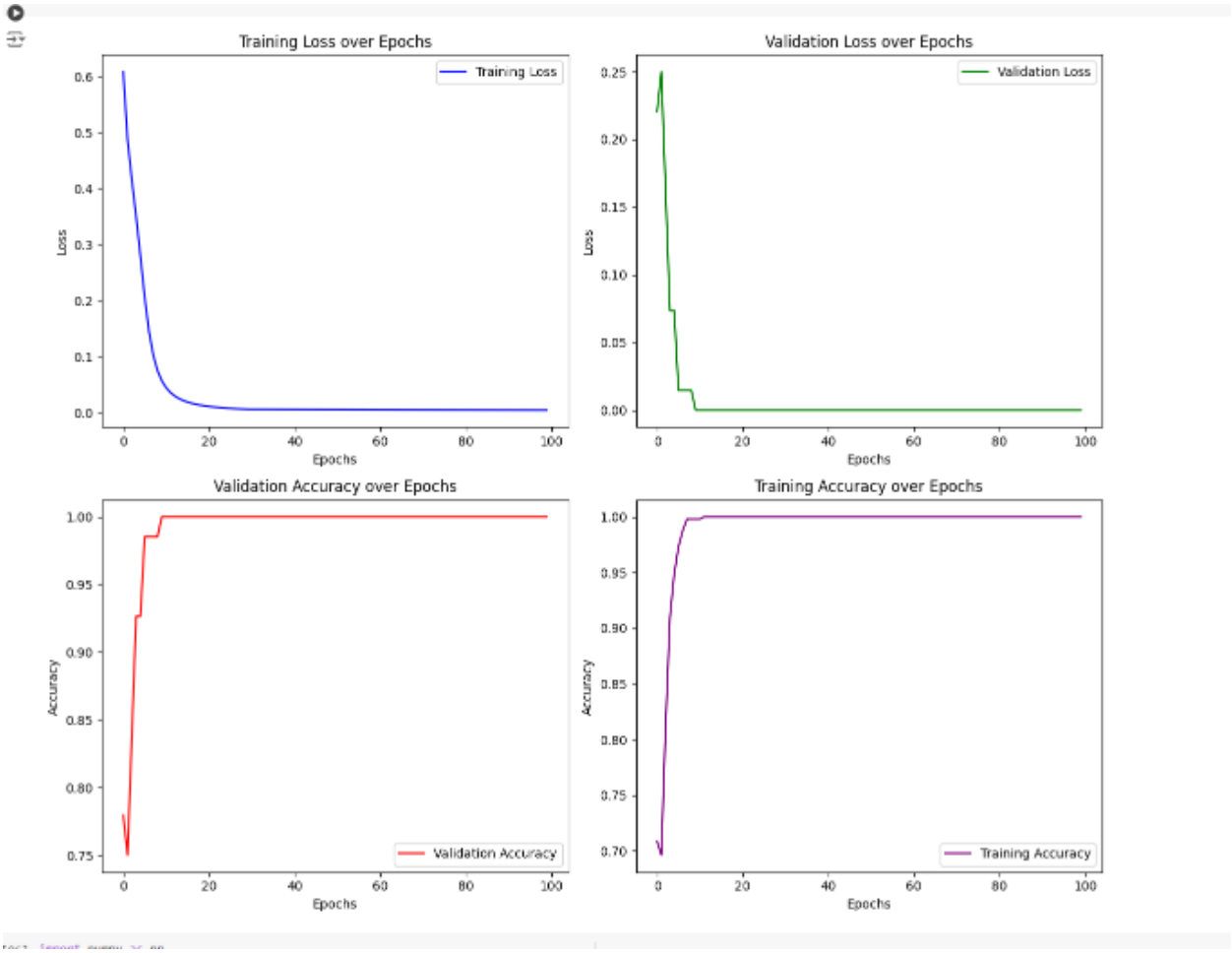

Correlation Matrix
standard deviation 1.00 1.00 0.04 0.80 0.03 0.38 0.58 1.00 1.00 1.00 -0.25
peak 1.00 1.00 0.04 0.80 0.03 0.38 0.58 1.00 1.00 1.00 -0.25
skewness 0.04 0.04 1.00 0.03 1.00 0.88 0.03 0.04 0.04 0.04 0.02
square root mean 0.80 0.80 0.03 1.00 0.02 0.11 -0.03 0.81 0.81 0.80 0.03
kurtosis 0.03 0.03 1.00 0.02 1.00 0.88 0.03 0.04 0.04 0.03 0.02
crest factor 0.38 0.38 0.88 0.11 0.88 1.00 0.49 0.38 0.38 0.38 -0.14
clearance factor 0.58 0.58 0.03 -0.03 0.03 0.49 1.00 0.55 0.55 0.58 -0.46
mean 1.00 1.00 0.04 0.81 0.04 0.38 0.55 1.00 1.00 1.00 -0.24
absolute mean 1.00 1.00 0.04 0.81 0.04 0.38 0.55 1.00 1.00 1.00 -0.24
root mean square 1.00 1.00 0.04 0.80 0.03 0.38 0.58 1.00 1.00 1.00 -0.25
label -0.25 -0.25 0.02 0.03 0.02 -0.14 -0.46 -0.24 -0.24 -0.25 1.00
Features
standard deviation
peak
skewness
square root mean
kurtosis
crest factor
clearance factor
mean
absolute mean
root mean square
label
Features
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

اکنون می‌بینید که ویژگی‌هایی با مقادیر کم را در DataFrame قرار ندادیم و قصد داریم با استفاده از این ویژگی‌های انتخابی، خرابی را تشخیص دهیم. به نظر می‌رسد با این ویژگی‌های انتخاب‌شده بتوانیم عملکرد بهتری از مدل به دست آوریم.




```
[93] mlp_model1 = MLPClassifier(hidden_layer_sizes=(15,5), activation='relu', solver='sgd', learning_rate_init=0.01, max_iter=5, warm_start=True, random_state=24, momentum=0.95)
```

همانطور که مشاهده میکنید تابع فعالساز رلو انتخاب شده و برای حل گر نیز از sgd استفاده شده است.



مشاهده می‌شود که شبکه به خوبی همگرا شده است و توانایی کاهش خطا را تا حد زیادی دارد. همچنین دقت آن به یک همگرا شده است که این یک ویژگی جالب در مهندسی نیست. اگر دقت در مرحله‌ی آزمون ۱۰۰ درصد باشد، باید اطمینان حاصل کنیم که وزن‌ها به درستی به‌روزرسانی شده‌اند و عملیات آموزش به خوبی انجام شده است. به این ترتیب، می‌توانیم بگوییم که عملکرد شبکه به‌طور قابل قبولی مناسب است.

دیده می‌شود که شبکه به خوبی همگرا شده است و توانایی کاهش خطا را تا جای ممکن دارد. همچنین دقت آن به یک همگرا شده است که این ویژگی به طور خاص در مهندسی جالب نیست. اگر در آزمون دقت ۱۰۰ درصد بود، باید حتماً وزن‌ها را بررسی کنیم و اطمینان حاصل کنیم که گرادیان به درستی وزن‌ها را به‌روزرسانی کرده و فرآیند آموزش به خوبی انجام شده است. در این صورت، عملکرد به طور قابل قبولی است. همچنین برای

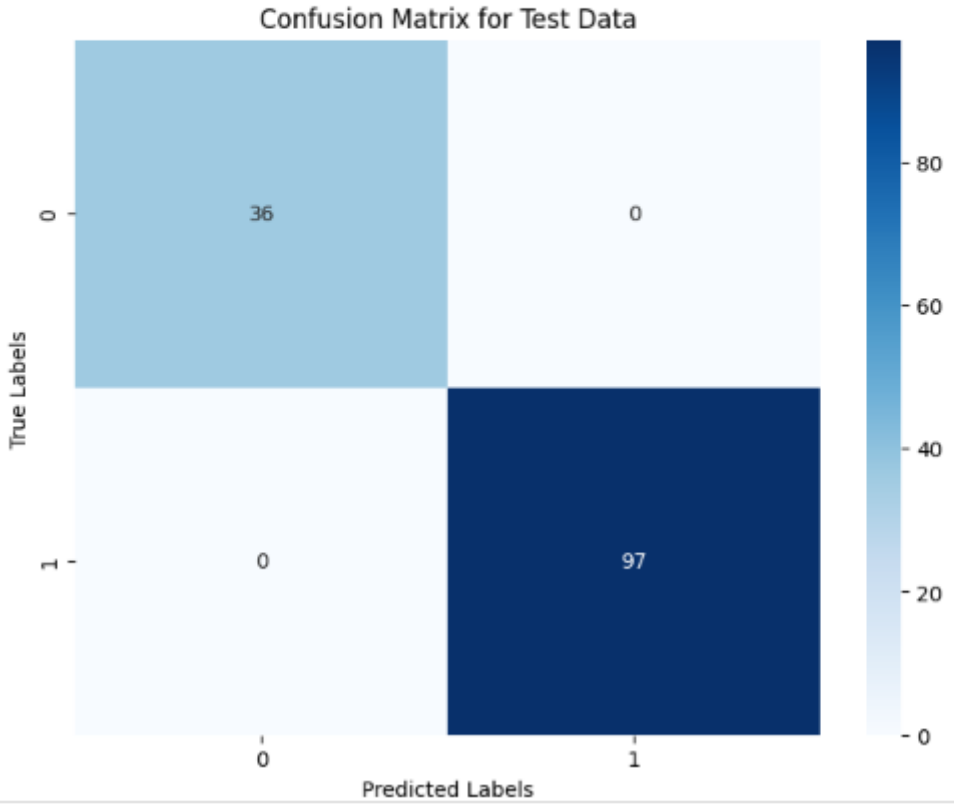
انتخاب داده‌های آموزش و آزمون از روش k-fold نیز استفاده کرده‌ایم که می‌توانید نتایج هر حالت را در زیر مشاهده کنید.

```python
from sklearn.metrics import confusion_matrix, classification_report

# Classification report for training data
train_report = classification_report(y_train, train_preds)
print("\nClassification Report for Training Data:")
print(train_report)
```

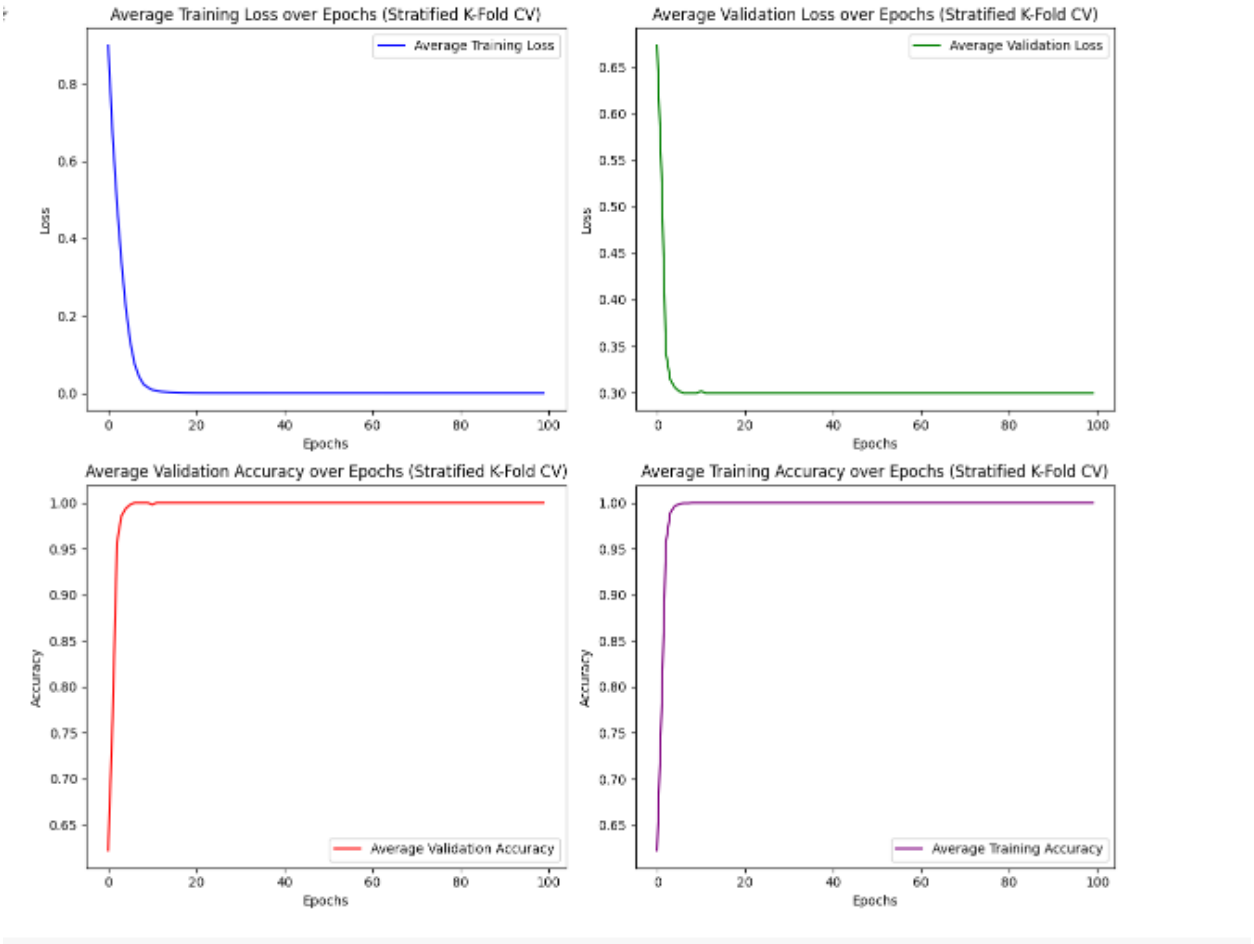
```
Classification Report for Training Data:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       145
           1       1.00      1.00      1.00       339

    accuracy                           1.00       484
   macro avg       1.00      1.00      1.00       484
weighted avg       1.00      1.00      1.00       484
```

```python
plt.ylabel('True Labels')
plt.title('Confusion Matrix for Training Data')
plt.show()
```

```
Classification Report for Test Data:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        36
           1       1.00      1.00      1.00        97

    accuracy                           1.00       133
   macro avg       1.00      1.00      1.00       133
weighted avg       1.00      1.00      1.00       133
```

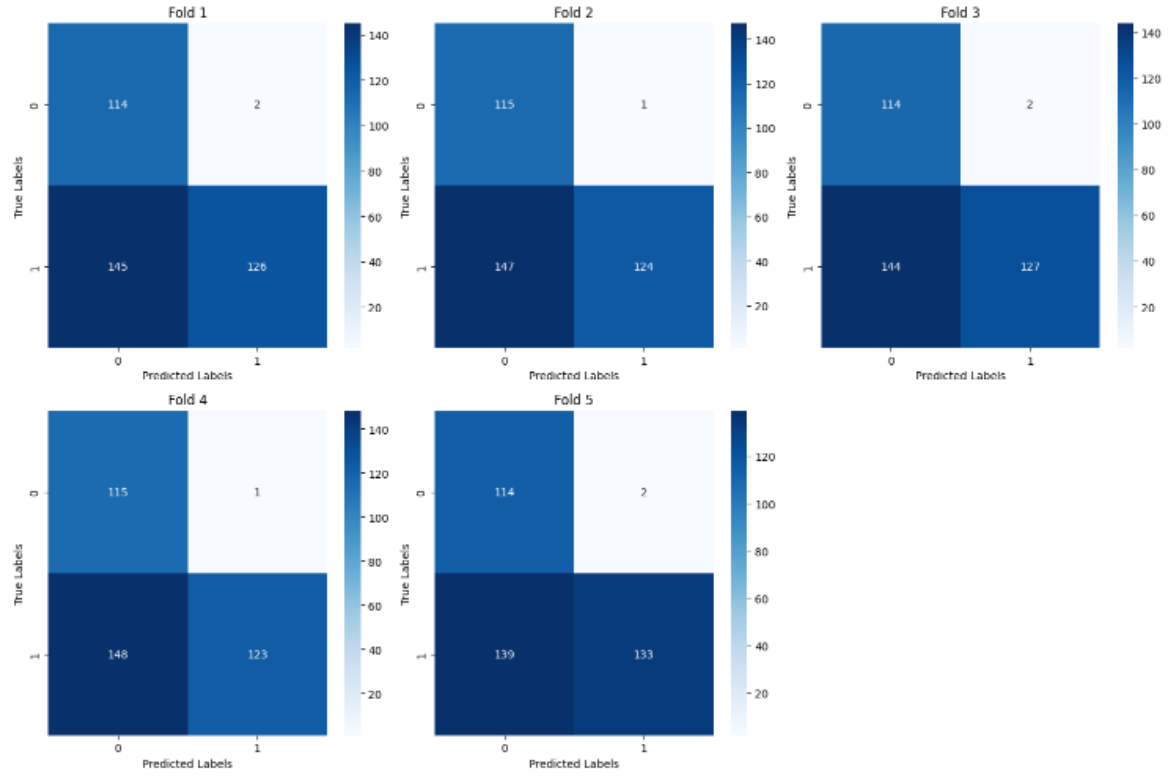


```
plt.tight_layout()
plt.show()
```

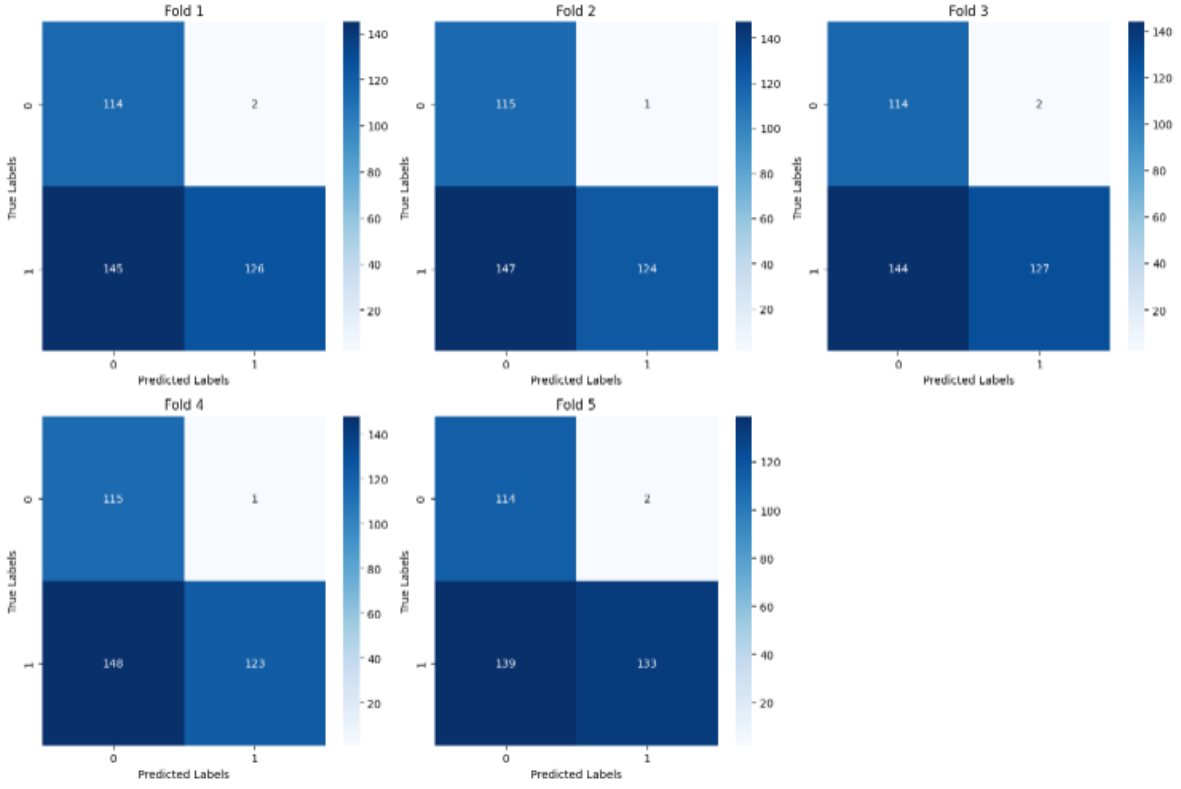


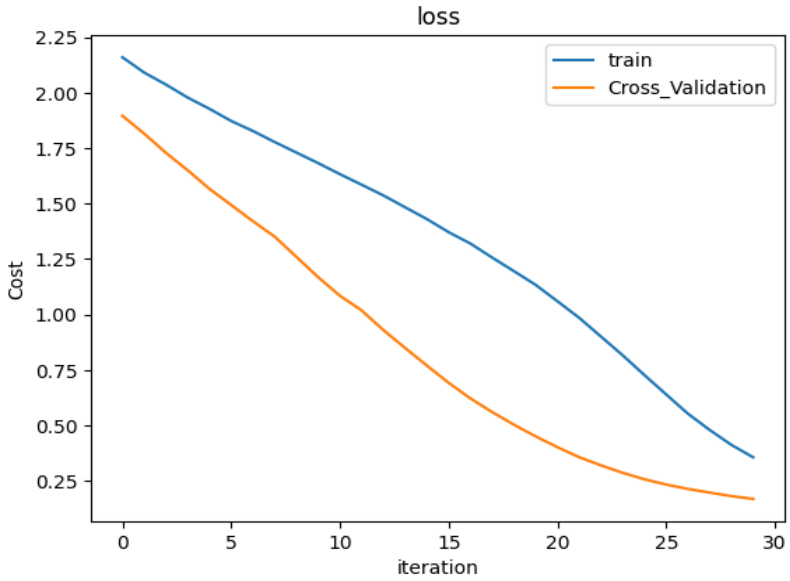
```
plt.figure(figsize=(15, 10))

for i, (train_index, val_index) in enumerate(skf.split(x_train_scaled_imputed, y_train)):
    x_train_fold, x_val_fold = x_train_scaled_imputed[train_index], x_train_scaled_imputed[val_index]
    y_train_fold, y_val_fold = y_train[train_index], y_train[val_index]

    mlp_model_fold = MLPClassifier(hidden_layer_sizes=(20, 15), activation='tanh', solver='adam', learning_rate_init=0.01,
                                   max_iter=1, warm_start=True, random_state=64, momentum=0.95)

    mlp_model_fold.fit(x_train_fold, y_train_fold)
    train_preds_fold = mlp_model_fold.predict(x_train_fold)

    train_cm_fold = confusion_matrix(y_train_fold, train_preds_fold)

    ax = plt.subplot(2, 3, i+1)
    sns.heatmap(train_cm_fold, annot=True, fmt='d', cmap='Blues', xticklabels=mlp_model_fold.classes_, yticklabels=mlp_model_fold.classes_)
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.title(f'Fold {i+1}')
    plt.tight_layout()

plt.show()
```

```
plt.show()
```



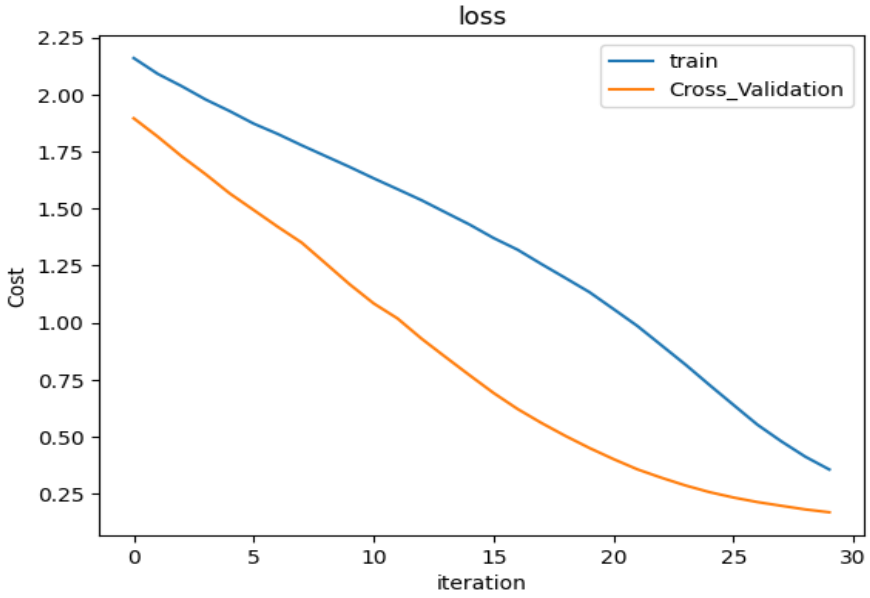
```
plt.figure(figsize=(15, 10))

for i, (train_index, val_index) in enumerate(skf.split(x_train_scaled_imputed, y_train)):
    x_train_fold, x_val_fold = x_train_scaled_imputed[train_index], x_train_scaled_imputed[val_index]
    y_train_fold, y_val_fold = y_train[train_index], y_train[val_index]

    mlp_model_fold = MLPClassifier(hidden_layer_sizes=(20, 15), activation='tanh', solver='adam', learning_rate_init=0.01,
                                   max_iter=1, warm_start=True, random_state=64, momentum=0.95)

    mlp_model_fold.fit(x_train_fold, y_train_fold)
    train_preds_fold = mlp_model_fold.predict(x_train_fold)

    train_cm_fold = confusion_matrix(y_train_fold, train_preds_fold)

    ax = plt.subplot(2, 3, i+1)
    sns.heatmap(train_cm_fold, annot=True, fmt='d', cmap='Blues', xticklabels=mlp_model_fold.classes_, yticklabels=mlp_model_fold.classes_)
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.title(f'Fold {i+1}')
    plt.tight_layout()

plt.show()
```






```
The shape of Training set is = (63988, 17)
The shape of Cross-Validation set is = (25920, 17)
The shape of Test set is = (25920, 17)
The ratio of examples in Cross-Validation set to Data set  is = 28.811844869557486
There Are  1165 instances of system fault in training set
There Are  478 instances of system fault in Cross_Validation set
There Are  6489 instances of system fault in test set
 The Mean and Variance of Data set Used in Normalization are as blow

 [4.31995000e+04 6.65175924e+02 3.39236698e+02 9.56704769e+01
 3.40357216e+02 3.73631725e+01 3.64853877e+01 4.67471053e+01
 1.28365163e+02 1.32272375e+02 1.29757222e+01 2.31304792e+01
 6.41200266e+01 2.57463932e+02 1.38445428e+02 1.64330174e+02
 1.32180851e+02]

 Variances
 [2.49415316e+04 4.81127894e+01 3.46943841e+01 3.07982167e+00
 3.95599227e+01 6.47421299e+00 5.74231356e+00 2.18299297e+00
 7.04924598e-01 9.33172882e-01 6.21994951e-01 2.67320253e+00
 7.01567065e+00 1.87955279e+01 1.97652670e+00 8.32316365e+00
 1.22851276e+00]
Epoch 1/30
125/125 [==============================] - 10s 27ms/step - loss: 2.1597 - val_loss: 1.8958
Epoch 2/30
125/125 [==============================] - 4s 30ms/step - loss: 2.0907 - val_loss: 1.8154
Epoch 3/30
125/125 [==============================] - 3s 20ms/step - loss: 2.0366 - val_loss: 1.7289
Epoch 4/30
125/125 [==============================] - 3s 20ms/step - loss: 1.9780 - val_loss: 1.6503
Epoch 5/30
125/125 [==============================] - 3s 21ms/step - loss: 1.9273 - val_loss: 1.5666
Epoch 6/30
125/125 [==============================] - 3s 25ms/step - loss: 1.8725 - val_loss: 1.4938
Epoch 7/30
125/125 [==============================] - 3s 28ms/step - loss: 1.8273 - val_loss: 1.4206
Epoch 8/30
125/125 [==============================] - 3s 20ms/step - loss: 1.7776 - val_loss: 1.3509
Epoch 9/30
125/125 [==============================] - 3s 21ms/step - loss: 1.7300 - val loss: 1.2599
Epoch 10/30
125/125 [==============================] - 3s 21ms/step - loss: 1.6826 - val_loss: 1.1682
Epoch 11/30
125/125 [==============================] - 4s 31ms/step - loss: 1.6328 - val loss: 1.0845
Epoch 12/30
125/125 [==============================] - 3s 23ms/step - loss: 1.5851 - val_loss: 1.0183
```

```
Epoch 13/30
125/125 [==============================] - 2s 20ms/step - loss: 1.5368 - val loss: 0.9297
Epoch 14/30
125/125 [==============================] - 3s 20ms/step - loss: 1.4833 - val loss: 0.8492
Epoch 15/30
125/125 [==============================] - 2s 20ms/step - loss: 1.4302 - val_loss: 0.7697
Epoch 16/30
125/125 [==============================] - 4s 35ms/step - loss: 1.3711 - val loss: 0.6920
Epoch 17/30
125/125 [==============================] - 3s 21ms/step - loss: 1.3200 - val loss: 0.6218
Epoch 18/30
125/125 [==============================] - 2s 20ms/step - loss: 1.2568 - val_loss: 0.5603
Epoch 19/30
125/125 [==============================] - 2s 20ms/step - loss: 1.1956 - val loss: 0.5034
Epoch 20/30
125/125 [==============================] - 3s 22ms/step - loss: 1.1337 - val loss: 0.4506
Epoch 21/30
125/125 [==============================] - 4s 32ms/step - loss: 1.0599 - val_loss: 0.4020
Epoch 22/30
125/125 [==============================] - 2s 20ms/step - loss: 0.9853 - val loss: 0.3570
Epoch 23/30
125/125 [==============================] - 3s 20ms/step - loss: 0.9010 - val loss: 0.3202
Epoch 24/30
125/125 [==============================] - 2s 19ms/step - loss: 0.8164 - val_loss: 0.2865
Epoch 25/30
125/125 [==============================] - 3s 24ms/step - loss: 0.7273 - val_loss: 0.2573
Epoch 26/30
125/125 [==============================] - 4s 29ms/step - loss: 0.6400 - val_loss: 0.2337
Epoch 27/30
125/125 [==============================] - 3s 20ms/step - loss: 0.5530 - val_loss: 0.2137
Epoch 28/30
125/125 [==============================] - 3s 20ms/step - loss: 0.4798 - val_loss: 0.1973
Epoch 29/30
125/125 [==============================] - 2s 20ms/step - loss: 0.4125 - val loss: 0.1813
Epoch 30/30
125/125 [==============================] - 4s 36ms/step - loss: 0.3563 - val_loss: 0.1689
```




```
810/810 [==============================] - 2s 2ms/step - loss: 1.1612
1.1611701250076294
```