

الْخَلَقُ



دانشکده مهندسی برق و کامپیو تر

سامانه پیش‌بینی ابتلا به دیابت
(مبتنی بر الگوریتم‌های یادگیری ماشین)

نام و نام خانوادگی دانشجو: قوام الدین سلیمانی

استاد راهنما: سرکار خانم دکتر فرشته دهقانی

تقدیر و تشکر

تقدیم به همه کسانی که با پاکدلی، پرتوی دانایی و آگاهی را بر تاریکی‌های کاناپی می‌افشانند.

در این زمان که به لطف پروردگار مهربان، بار دیگر فرصت آموختن به من عطا شد، و موفق به اتمام این دوره شدم؛ بر خود لازم می‌دانم از تمام بزرگوارانی که مرا در این دوره یاری کردند، قدردانی کنم.

از استاد راهنمای محترم، سرکار خانم دکتر دهقانی که راهنمایی و بزرگواری ایشان در شرایط مختلف، همواره شامل حال من بوده و انجام این پروژه بدون همیاری ایشان ممکن نبود، بی‌نهایت سپاسگزارم و همواره برای ایشان، آرزوی موفقیت و سلامتی دارم.

از خانواده عزیزم که در این دوره، بزرگترین پشتیبان و یاری‌گر من بودند، تشکر می‌کنم و امیدوارم چون همیشه، در تندرنستی و خوشبختی، مایه دلگرمی ام باشند.

همچنین از سایر اساتید محترم گروه مهندسی کامپیوتر دانشگاه کاشان و دوستانم که توجهات و محبت ایشان در این دوره مایه پیشرفت من بوده نیز کمال تشکر را دارم و برای ایشان بهترین‌ها را از خداوند منان خواستارم.

فهرست مطالب

۱	تقدیر و تشکر
چهار	فهرست تصاویر
۵	چکیده
۶	۱ مقدمه
۶	۱-۱ تاریخچه
۷	۱-۱-۱ علل ابتلا
۷	۲-۱-۱ راهکارهای پیشگیری و استعدادسنجی
۸	۲-۱-۲ انگیزه و اهداف انجام این پژوهش
۸	۳-۱ موارد انجام شده
۱۰	۲ ادبیات پژوهش
۱۰	۱-۲ مقدمه
۱۰	۲-۲ روش‌های داده‌کاوی
۱۰	۱-۲-۲ طبقه‌بندی و خوشه‌بندی
۱۲	۲-۲-۲ بیش‌بازش و کم‌بازش، افزایش داده‌ها
۱۵	۳-۲ معرفی الگوریتم‌های طبقه‌بندی
۱۵	۱-۳-۲ رگرسیون لجستیک
۱۸	۲-۳-۲ درخت تصمیم
۲۲	۳-۳-۲ جنگل درختان تصادفی
۲۳	۴-۳-۲ AdaBoost
۲۴	۵-۳-۲ Naïve Bayes
۲۵	۴-۲ دادگان، پیش‌پردازش و مصورسازی داده‌ها
۲۵	۱-۴-۲ دادگان
۲۶	۲-۴-۲ پیش‌پردازش داده‌ها
۳۰	۳-۴-۲ انتخاب ویژگی
۳۰	۴-۴-۲ مصورسازی
۳۰	۵-۴-۲ انواع نمودارها و داده‌های اجمالی
۳۲	۶-۴-۲ اندازه‌گیری میزان خطای دقت
۳۴	۷-۴-۲ Precision ، ACC ، AUC و Recall

۳۵	۵-۲ نتیجه‌گیری
۳۶	۳ کارهای پیشین
۳۶	۱-۳ مقدمه
۳۶	۲-۳ مقاله ۱
۳۸	۳-۳ مقاله ۲
۳۹	۴-۳ نتیجه‌گیری
۴۰	۴ روش‌ها و نتایج
۴۰	۱-۴ مقدمه
۴۰	۲-۴ روش پیشنهادی
۴۰	۱-۲-۴ پیاده‌سازی
۶۳	۲-۲-۴ Arduino
۶۷	۳-۲-۴ ابزار Node Red
۶۹	۴-۲-۴ Flask
۷۰	۳-۴ نتیجه‌گیری
۷۲	۵ جمع‌بندی و کارهای آتی
۷۲	۱-۵ جمع‌بندی
۷۲	۲-۵ کارهای آتی
۷۴	پیوست
۷۴	کد استاندارد سازی متون فارسی آمیخته به عبارات انگلیسی	۱-
۷۵	واژه‌نامه
۷۷	مراجع

فهرست تصاویر

۱-۱	روندنمای پروژه	۹
۱-۲	مقایسه خوشبندی و طبقه‌بندی	۱۲
۲-۲	بیشبرازش و کم‌برازش	۱۲
۳-۲	روش K-Fold	۱۴
۴-۲	روش K-Fold تکرار شونده	۱۴
۵-۲	نمودار پراکندگی	۱۵
۶-۲	نمودار رگرسیون خطی چندگانه در فضا	۱۶
۷-۲	مقایسه رگرسیون لجستیک و خطی	۱۷
۸-۲	نمودار تابع لجستیک	۱۷
۹-۲	مدل Logist Regression	۱۸
۱۰-۲	مثال دادگان مورد استفاده در درخت تصمیم	۱۹
۱۱-۲	مثال درخت تصمیم	۱۹
۱۲-۲	انواع شاخه‌ها در درخت تصمیم	۲۰
۱۳-۲	نمونه‌ای از آنتروپی	۲۰
۱۴-۲	روش Bootstrap در جنگل تصادفی	۲۲
۱۵-۲	مدل جنگل درختان تصادفی	۲۳
۱۶-۲	الگوریتم AdaBoost: در اینجا مراحل ذکر شده به صورت متناوب تکرار می‌شود [۳۰].	۲۴
۱۷-۲	نمونه‌ای از داده‌های گم شده	۲۶
۱۸-۲	تعداد داده‌های گم شده در این پروژه به ازای هر ستون	۲۷
۱۹-۲	نمونه‌ای از متغیرهای رقمهی	۲۸
۲۰-۲	نمونه‌برداری مجدد	۲۹
۲۱-۲	نمودار Pairplot قبل از نمونه‌برداری مجدد	۳۱
۲۲-۲	نقشه حرارتی	۳۲
۲۳-۲	ساختار ماتریس آشفتگی: پارامترهای مذکور در این بخش در این ماتریس قرار گرفته	۳۳
۲۴-۲	ساختار ماتریس آشفتگی برای الگوریتم جنگل درختان تصادفی	۳۴
۲۵-۲	نمای کلی نمودار ROC	۳۵
۱-۳	مدل SVM	۳۷
۲-۳	دقتهای اندازه‌گیری در مقاله اول	۳۷
۳-۳	منحنی ROC در مقاله اول	۳۸

۴-۳	دقت‌های اندازه‌گیری در مقاله دوم	۳۸
۵-۳	منحنی ROC در مقاله دوم	۳۹
۱-۴	ردیف‌های دادگان	۴۱
۲-۴	جایگزینی مقادیر نامشخص با متoste‌های آماری	۴۲
۳-۴	ایجاد مدل RFE	۴۲
۴-۴	ویژگی‌های انتخاب شده	۴۴
۵-۴	نمودار مقایسه ابتلا به دیابت در نژادهای مختلف	۴۵
۶-۴	ماتریس Scatter	۴۶
۷-۴	نمودار عوامل موثر در ابتلا دیابت (۱)	۴۶
۸-۴	نمودار عوامل موثر در ابتلا دیابت (۲)	۴۷
۹-۴	نقشه حرارتی	۴۸
۱۰-۴	نمودار نمونه‌برداری شده عوامل موثر در ابتلا دیابت (۱)	۴۹
۱۱-۴	نمودار نمونه‌برداری شده عوامل موثر در ابتلا دیابت (۲)	۵۰
۱۲-۴	نمودار Pairplot بعد از نمونه‌برداری مجدد	۵۱
۱۳-۴	نمودار ACC	۵۸
۱۴-۴	نمودار AUC	۵۹
۱۵-۴	نمودار Precision	۵۹
۱۶-۴	نمودار Recall	۵۹
۱۷-۴	نمودار ROC برای الگوریتم AdaBoost	۶۰
۱۸-۴	نمودار ROC برای الگوریتم جنگل درختان تصادفی	۶۰
۱۹-۴	نمودار ROC برای الگوریتم Naïve Bayes	۶۰
۲۰-۴	نمودار ROC برای الگوریتم درخت تصمیم	۶۱
۲۱-۴	ساختار ماتریس آشفتگی برای الگوریتم جنگل درختان تصادفی	۶۳
۲۲-۴	محیط توسعه Arduino در حال دریافت اعداد تولید شده توسط بورد	۶۷
۲۳-۴	صفحه داشبورد پیش‌بینی کاربر و بهروزرسانی دادگان	۶۸
۲۴-۴	طراحی بخش بهروزرسانی در NodeRed	۶۹
۲۵-۴	طراحی بخش پیش‌بینی در NodeRed	۶۹
۲۶-۴	بخشی از کد Flask	۷۰

چکیده

دیابت یک بیماری مزمن است که فرد مبتلا، قند خون بالاتر از حد مجاز را داراست و این مسئله موجب عوارض و مشکلات جدی در سلامت وی (از جمله برخی نارسایی‌ها، سکته‌ها، آسیب و از کار افتادن اندام‌ها) می‌شود. لازمه ابتلا و بروز این بیماری، عوامل ژنتیکی و محیطی می‌باشد. لذا در صورت وجود احتمال ابتلا به این بیماری در افراد، می‌توان با تغییر سبک زندگی و کنترل‌های پزشکی، تا حدی از ابتلای به این بیماری در افراد مستعد و محتمل، جلوگیری کرد.

روش‌هایی که بتواند کمک کند تا با دقت مناسبی بتوان ابتلای افراد مختلف به بیماری را پیش‌بینی کرد، بسیار حائز اهمیت هستند. یادگیری ماشین و داده‌کاوی با استفاده از داده‌های مختلف که در گذشته جمع آوری شده‌اند می‌توانند فایده زیادی در این امر داشته باشند. پس از الگوریتم‌های مختلف یادگیری ماشین از جمله درخت تصمیم، جنگل درختان تصادفی، بیزساده و AdaBoost استفاده شد تا دقت هر یک اندازه‌گیری شود.

داده‌هایی که برای آموزش مدل‌ستفاده شد، از مجموعه دادگان بیماران آمریکایی^۱ است که در سال ۲۰۰۹ تا ۲۰۱۲ جمع آوری شده بودند. این داده‌ها را با استفاده از روش‌های داده‌کاوی گوناگون، پردازش با روش رگرسیون لجستیک ویژگی‌های موثر را انتخاب و سپس مصورسازی و نمونه‌برداری شد و سپس با الگوریتم‌های مذکور و معیارهای ارزیابی مربوط به آن‌ها، سعی در شناسایی بهترین مدل پیش‌بینی‌کننده شد.

مدل جنگل درختان تصادفی حاوی بهترین نتایج نسبت به سایر مدل‌ها بود.

کلید واژه‌ها: داده کاوی، دیابت، درخت تصمیم، الگوریتم درخت تصادفی، قندخون، رگرسیون لجستیک، طبقه‌بندی، پیش‌بینی، یادگیری ماشین

^۱NHANES

فصل ۱

مقدمه

۱ - ۱ تاریخچه

دیابت چیست؟

دیابت، یک بیماری مزمن است و زمانی رخ می‌دهد که بدن انسولین کافی تولید نمی‌کند یا نمی‌تواند به طور موثر از انسولین تولید شده استفاده کند [۵]. انسولین هورمونی است که به تنظیم سطح قند خون کمک می‌کند. هنگامی که دیابت به درستی مدیریت نشود، می‌تواند منجر به عوارض جدی سلامتی مانند بیماری‌های قلبی، انواع سکته مغزی و قلبی، نارسایی کلیه، کوری و آسیب عصبی شود [۱۸]. آمار ابتلا و مرگ و میر بسیار بالایی از این بیماری در جهان وجود دارد و متاسفانه روز به روز این آمار افزایش می‌یابد.

طبق آمارها، از هر ۱۰ نفر که به دیابت مبتلا هستند، بیش از ۸ نفر آن‌ها از این مسئله آگاهی ندارند و عده زیادی از افراد هم به پیش‌دیابت مبتلا هستند [۶]. در پیش‌دیابت، سطح قند خون بالاتر از حد طبیعی است، اما به اندازه کافی برای تشخیص دیابت بالا نیست. پیش‌دیابت خطر ابتلا به دیابت، بیماری قلبی و سکته را افزایش می‌دهد [۵]. اگر پیش‌دیابت در افراد وجود داشته باشد، یک برنامه برای تغییر سبک زندگی، می‌تواند به افراد در جلوگیری از این بیماری کمک کند [۱۸].

این بیماری سه نوع دارد [۵]:

۱. دیابت نوع اول که معروف به دیابت جوانی است چون افراد با سن کمتر از ۳۰ سال معمولاً مبتلا می‌شوند. در این نوع به طور ساده می‌توان گفت میزان انسولین مورد نیاز که توسط پانکراس باقیستی ساخته شود و در خون وجود داشته باشد کافی نیست.
۲. دیابت نوع دوم که به بزرگسالی معروف است و در افراد میانسال و مسن رایج‌تر است در اثر عدم جذب انسولین موجود در خون توسط سلول‌ها می‌باشد.
۳. دیابت نوع سوم دیابت بارداری است که در خانم‌های باردار به طور موقت اتفاق می‌افتد.

۱-۱-۱ علل ابتلا

در ابتلا به این بیماری بنا به نوع آن و همچنین شرایط ژنتیکی و محیطی افراد مختلف، فاکتورهای متنوعی مطرح است [۵]:

- **فاکتورهای محیطی** مطابق تحقیقات و بررسی‌های انجام شده از سال‌ها پیش تا کنون، عوامل سبک زندگی چون رژیم غذایی نامناسب، عدم فعالیت بدنی و اضافه وزن (مخصوصاً میزان توده بدنی) می‌تواند خطر ابتلا به دیابت را افزایش دهد [۱۸]. همچنین وجود بیماری‌های زمینه‌ای مثلاً در پانکراس بین افراد می‌تواند در مبتلا شدن به این بیماری موثر باشد که بنا به تعریف دیابت نوع یک، این عامل مربوط به همین نوع می‌شود [۵].

- **عوامل ژنتیکی و غیرمحیطی** برخی از افراد استعداد ژنتیکی برای دیابت دارند، به این معنی که بدن آن‌ها بیشتر در معرض ابتلا به این بیماری است. عواملی مثل جنسیت، نژاد و شاخص‌هایی خونی مختلف که می‌تواند در اثر بیماری‌های خانوادگی و ارثی دیگری در افراد وجود داشته باشد. مثل برخی ویروس‌ها، وجود کلسترول، چربی و فشار خون و ... [۱۸] [۳] [۴].

۲-۱-۱ راهکارهای پیش‌گیری و استعدادسنجی

مطابق توصیه متخصصین اگر بتوان افرادی را که استعداد ابتلا به این بیماری را دارند، شناسایی کرد و این افراد سبک زندگی و روش‌هایی خاصی را در پیش‌بگیرند، می‌توانند از ابتلا به این بیماری پیش‌گیری کنند [۳].

- **مبتلایان به پیش‌دیابت** مطابق توصیه پزشکان، در افرادی که به پیش‌دیابت مبتلا باشند یا سابقه این بیماری در خانواده آن‌ها وجود داشته باشد، به طور پیش‌فرض باید بر یک سبک زندگی سالم، اهتمام ورزند. در این راستا می‌توان به موارد ذیل اشاره کرد [۴]:

- حفظ رژیم غذایی غنی از فیبر مثل انواع میوه‌ها و سبزیجات و کاهش مصرف غذاهای شور، چرب و شیرین
- ورزش منظم
- استفاده از برخی داروها مطابق تجویز پزشک

- **سایر افراد جامعه** مطابق آمارها، سالانه بخش دیگری از افراد جامعه که از دسته قبلی سوا بوده اند، به بیماری دیابت مبتلا می‌شوند [۳]. در اینجا با تحلیل برخی فاکتورهای سلامتی می‌توان پیش‌بینی کرد که آیا این افراد ممکن است با ادامه سبک زندگی کنونی، آیا ممکن است دچار عوارض این بیماری شوند و آیا بهتر است با تغییر سبک زندگی خود از شدیدشدن این بیماری جلوگیری کنند یا نه؟

در این زمینه تحقیقات آماری و بررسی‌های مختلفی انجام شده تا بتوان با اندازه‌گیری برخی فاکتورهای کمی و کیفی در افراد، مسئله استعداد در ابتلا به این بیماری را در آن‌ها بررسی کرد. چالشی که در این زمینه وجود دارد این است که بسیاری از داده‌های غیرخطی و غیراستاندارد پزشکی با ارتباطات و ساختارهای پیچیده وجود دارند که این بررسی‌ها را دشوار می‌سازد [۶].

۱-۲ انجیزه و اهداف انجام این پژوهش

در راستای همه موارد مطرح شده در بخش‌های قبلی، بررسی شد که با توجه به امکانات و امروزی و دسترسی به مقالات و منابع گوناگون و همچنین توسعه ابزارهای مبتنی بر یادگیری ماشین و هوش مصنوعی، آیا می‌توان در صدد یافتن بهترین راهکارها برای نجات جان انسان‌های بیشتر با درنظرگیری مناسب‌ترین الگوریتم‌ها بود؟

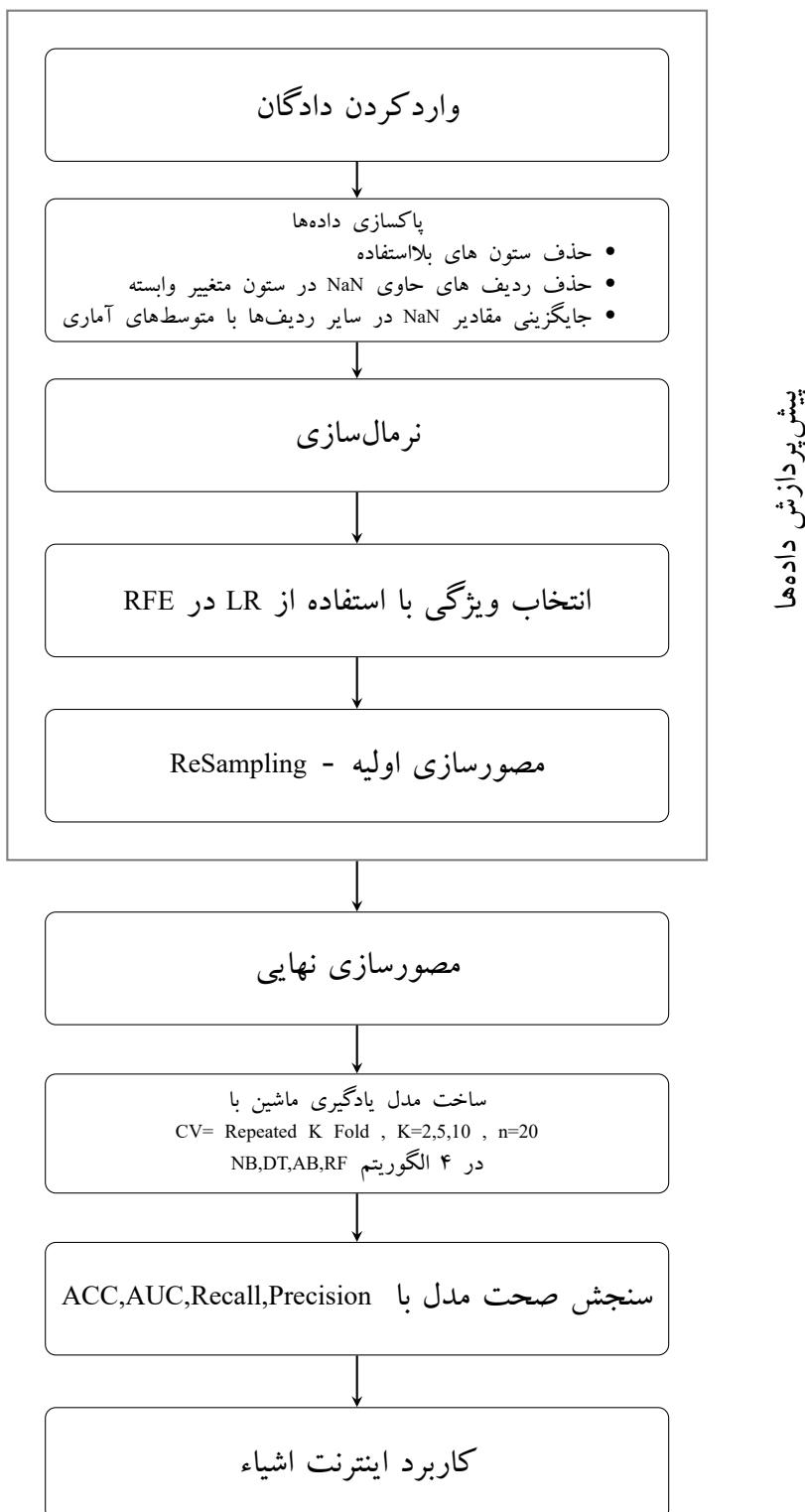
اگر بتوان افرادی را که احتمال ابتدا به دیابت در آینده برای آنان زیاد است را شناسایی کرد، می‌توان با ارائه برنامه‌های پزشکی مناسب از ابتلای افراد به بیماری مذکور جلوگیری کرد.

در نهایت نیز جهت ادامه پژوهه، یک سیستم پیش‌بینی آنلاین با کمک اینترنت اشیاء و دو چارچوب Flask و NodeRed طراحی و معرفی شد که افراد گوناگون بتوانند از آن بهره ببرند.

۱-۳ موارد انجام شده

در ابتدا مقالات مختلفی را درمورد الگوریتم‌های رگرسیون لجستیک، درخت تصمیم، جنگل درختان تصادفی، بیز ساده و AdaBoost مطالعه شد و مورد بررسی قرار گرفت.

پس از یافتن مجموعه دادگان نمونه که مربوط به اطلاعات بیماران آمریکایی در سال‌های ۲۰۰۹ تا ۲۰۱۲ بوده است، با به کارگیری کتابخانه‌های مختلف پایتون از جمله `seaborn`, `Pandas`, `scikit learn`, `numpy` و ... عملیات‌های گوناگونی بر روی داده‌ها انجام شد. از جمله: تمیز کردن دادگان، انتخاب ویژگی، استانداردسازی داده‌ها، مصورسازی و در نهایت مدل‌سازی و دستیابی به نتایج نهایی که در نهایت الگوریتم جنگل درختان تصادفی با دقت ۹۶٪ بهترین عملکرد را بین باقی الگوریتم‌ها در این پیش‌بینی، کسب کرد. در شکل ۱-۱ روند نمای پژوهه نشان داده شده است.



شکل ۱-۱: روند نمای پروژه

فصل ۲

ادبیات پژوهش

۱-۲ مقدمه

روش‌های نظارت شده‌ای مانند طبقه‌بندی و تخمین تلاش می‌کنند تا رابطه‌ای میان صفات خاصه ورودی (که گاه متغیرهای مستقل نامیده می‌شوند) را با یک یا چند صفت خاصه هدف (که گاه متغیر وابسته نامیده می‌شود) کشف کنند. در نهایت این رابطه با یک ساختار به عنوان مدل نمایش داده می‌شود [۱].

به بیان دیگر، در یادگیری تحت نظارت همان‌طور که از نام آن پیداست، یک ناظر به عنوان یاددهنده در این نوع الگوریتم یادگیری ماشین حضور خواهد داشت. مدل را با داده‌های برچسب‌گذاری شده مناسب آموختش دادم. الگوریتم‌های یادگیری نظارت شده سعی می‌کنند روابط و وابستگی‌هایی بین متغیرها ایجاد کنند که به ترتیب «ویژگی‌ها» و «برچسب‌ها» نامیده می‌شوند. سپس الگوریتم‌ها از داده‌ها، (با استفاده از روابط بین ویژگی‌ها) یاد می‌گیرند و خروجی را پیش‌بینی می‌کنند [۱۴].

اما در یادگیری بدون نظارت هیچ ناظر یا یاد دهنده‌ای وجود نخواهد داشت، بنابراین هیچ آموختشی یا آموختشی به ماشین ارائه نخواهد شد. یادگیری بدون نظارت با داده‌های بدون برچسب سروکار دارد که در آن نمی‌توان روابط و وابستگی‌ها را در داده‌ها اندازه‌گیری کرد. در این پژوهش، مدل‌ها سعی می‌کنند داده‌های مرتب نشده را بر اساس الگوها و شباهت‌ها در داده‌ها به صورت خوش‌بندی گروه‌بندی کنند [۱۴].

۲-۲ روش‌های داده‌کاوی

۱-۲-۲ طبقه‌بندی و خوش‌بندی

در این بخش به طور واضح‌تر درمورد تفاوت الگوریتم‌های بانظارت (طبقه‌بندی‌کننده و پیش‌بینی) و بدون نظارت (خوش‌بندی) در مدل‌های رایج توضیح داده خواهد شد.

• طبقه‌بندی

در این مدل، گیرنده پیشنهاد می‌تواند پاسخ دهد یا پاسخ ندهد. متقاضی وام می‌تواند به موقع بازپرداخت کند، دیر بازپرداخت کند یا اعلام ورشکستگی کند. تراکنش کارت اعتباری می‌تواند

عادی یا تقلیبی باشد. بسته‌ای از داده‌هایی که در یک شبکه حرکت می‌کنند می‌تواند عادی یا تهدیدکننده باشد. یک اتوبوس در سیستم حمل و نقل می‌تواند در دسترس باشد یا در دسترس نباشد. قربانی یک بیماری می‌تواند بهبود یابد، یا خیر [۱۱].

یک کار رایج در داده کاوی، بررسی داده‌هایی است که در آن طبقه‌بندی ناشناخته است یا در آینده رخ خواهد داد، با هدف پیش‌بینی اینکه طبقه‌بندی چیست یا چه خواهد بود. داده‌های مشابهی که طبقه‌بندی شناخته است، برای توسعه قوانین استفاده می‌شوند که بعداً روی داده‌های دارای طبقه‌بندی ناشناخته اعمال شوند [۱۱].

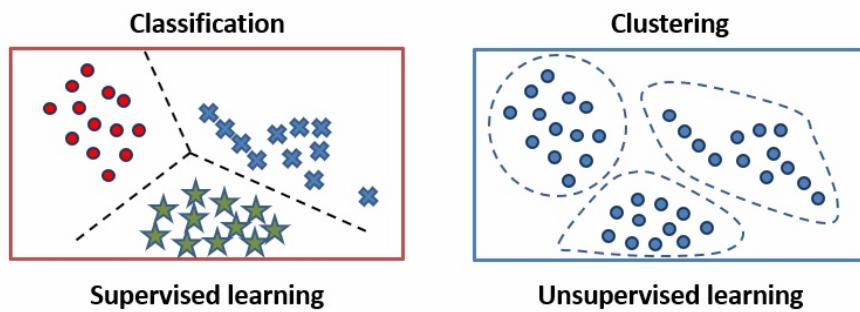
به بیان دیگر، پیش‌بینی مشابه طبقه‌بندی است، با این تفاوت که سعی می‌شود ارزش یک متغیر عددی (مثلاً مقدار خرید) را به جای یک طبقه (مثلاً خریدار یا غیرخریدار) پیش‌بینی کرد. البته در طبقه‌بندی سعی می‌شود یک طبقه پیش‌بینی شود، اما گاهی در ادبیات داده کاوی، از اصطلاحات تخمین و رگرسیون برای اشاره به پیش‌بینی مقدار یک متغیر پیوسته استفاده می‌شود، و پیش‌بینی ممکن است هم برای داده‌های پیوسته و هم برای داده‌های طبقه‌ای استفاده شود [۱۱].

ضمانت در فرایند ساخت مدل طبقه‌بندی، داده‌های اولیه به دو بخش تقسیم می‌شوند که در ادامه توضیح داده خواهد شد؛ داده‌های آموزشی مدل را ساخته و داده‌های اعتبارسنجی که زیر مجموعه‌ای از داده‌های اصلی هستند و مستقل از داده‌های آموزشی‌اند، عملکرد مدل را ارزیابی می‌کنند [۱۱].

- **خوش‌بندی تجزیه و تحلیل خوش‌هایی** از رکوردهای مشابه بر اساس چندین اندازه‌گیری انجام شده بر روی آن‌ها انجام می‌شود. ایده کلیدی خوش‌بندی به این صورت است که خوش‌های روش‌هایی تو صیف شود که برای اهداف تحلیل مفید باشد. این ایده، در بسیاری از زمینه‌ها از جمله نجوم، باستان‌شناسی، پزشکی، شیمی، آموزش، روانشناسی، زبان‌شناسی و جامعه‌شناسی به کار گرفته شده است. برای مثال، زیست‌شناسان از طبقات اصلی و طبقات فرعی برای سازماندهی گونه‌ها استفاده گسترده‌ای کرده‌اند [۱۱]. بدیهی است که ممکن است مشخص نباشد چه تعداد دسته یا خوش‌های در پایان این فرایند ایجاد می‌شود.

به زبان دیگر، می‌توان گفت زمانی که پیش‌فرض خاصی در مورد کلاس‌ها نباشد و صرفاً سنجش وجود داده‌ها در گروه‌ها ملاک باشد، از خوش‌بندی استفاده می‌شود [۱۱] (مثلاً چندین متن باشد و هدف یافتن متن‌های مشابه در یک دسته باشد) اما اگر هدف این باشد که هر رکورد از داده‌ها، الزاماً در یک طبقه یا کلاس خاص قرار گیرد، از طبقه‌بندی استفاده می‌شود که روش باظارت است [۱۱] (مانند همین پروژه).

از دید دیگر می‌توان گفت هرگاه حسایت برای تشخیص‌های گوناگون زیاد باشد، (یعنی لزوماً برچسبی بر داده‌ای زده شود که اهمیت زیادی داشته باشد مثل تشخیص دیابت در افراد) در اینجا از مدل‌های طبقه‌بندی استفاده می‌شود و اگر تنها خواسته شود داده‌ها در دسته‌های گوناگون قرار گیرند و نام کلاس‌های دسته‌بندی مشخص نباشد، می‌توان خوش‌بندی را اعمال کرد. (مثلاً یک فروشگاه اینترنتی که به گروه‌هایی از خریداران مختلف، پیشنهادهای گوناگونی در ازای خریدهای قبلی آن‌ها می‌دهد.)



شکل ۱-۲: مقایسه خوشبندی و طبقه‌بندی

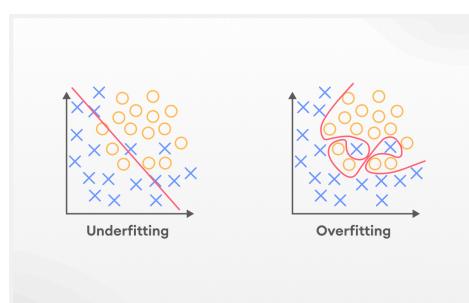
۲-۲-۲ بیش‌برازش و کم‌برازش، افزایش داده‌ها

اگر خیلی ساده در مورد این دو مفهوم توضیح داده شود، می‌توان گفت در یک مدل یادگیری ماشین، کم‌برازش در خروجی زمانی اتفاق می‌افتد که مدل خیلی ساده است و نمی‌تواند رابطه و الگوی بین داده‌ها را با دقت مشخص کند و از این جهت در پیش‌بینی و سنجش ناتوان خواهد بود، در حالی که بیش‌برازش زمانی اتفاق می‌افتد که مدل بیش از حد پیچیده باشد و داده‌های آموزشی را به خاطر بسپارد. بنابر این در مقابل داده‌های جدید تعمیم و انطباق پیدا نمی‌کند. [۲۲]

راه حل جلوگیری از این مشکل، تقسیم داده‌ها و اعتبارسنجی متقابل است [۲۲] که در ادامه توضیح داده می‌شود.

یک مثال خوب که می‌توان برای این مسئله بیان کرد، نمونه لیوان است. اگر فرض شود مدلی که ساخته شده، داشتن دسته را برای یک شیء به نام "لیوان" را ضروری بداند، پس به نظر می‌رسد شرط اضافه‌ای برای تشخیص لیوان بودن یا نبودن اشیاء منظور کرده است؛ و در این حالت اصطلاحاً مدل دچار بیش‌برازش شده است.

بر عکس در حالتی که مدل اهمیتی برای داشتن ارتفاع نسبت به سطح را در نظر نگیرد، ممکن است یک بشقاب را به عنوان یک لیوان در نظر بگیرد و در این حالت اصطلاحاً مدل دچار کم‌برازش شده است.



شکل ۲-۲: بیش‌برازش و کم‌برازش

بخش کردن داده‌ها یا افزایش^۱ برای ارزیابی مدل به طور مؤثر استفاده می‌شود. این تکنیک، برای ارزیابی عملکرد مدل یادگیری ماشین به این صورت عمل می‌کند که یک مجموعه داده داده شده را می‌گیرد و آن را به دو یا سه زیر مجموعه تقسیم می‌کند. [۲۳]

- داده‌های آموزشی یا (Train) :

مجموعه‌ای از داده‌های مورد استفاده برای ساخت و یادگیری جهت تناسب پارامترها با مدل یادگیری ماشین و ایجاد روابط در مدل مد نظر است.

- داده‌های اعتبارسنجی (استفاده از این مورد اختیاری است). :

مجموعه‌ای از داده‌ها برای ارائه یک ارزیابی بی‌طرفانه از یک مدل برآذش شده در مجموعه داده آموزشی در حین تنظیم روابط درون مدل استفاده می‌شود.

- داده‌های سنجش (Test) :

مجموعه‌ای از داده‌ها برای ارائه یک ارزیابی بی‌طرفانه از یک مدل نهایی برآذش شده در مجموعه داده آموزشی استفاده می‌شود. در واقع این قسمت تضمین می‌کند که مدل بر اساس داده‌های دیده نشده ارزیابی می‌شود. در واقع هدف از استفاده از داده‌های آزمایشی، تعیین میزان تعمیم مدل به داده‌های جدید و دیده نشده است.

برای دستیابی به نتایج دقیق، بسیار مهم است که اطمینان حاصل کرد که داده‌ها به صورت تصادفی انتخاب شده اند و شامل طیف متنوعی از نمونه‌ها هستند. این مسئله برای به حداقل رساندن سوگیری و بهبود قابلیت‌های تعمیم مدل کمک می‌کند [۲۳].

با این حال، با تقسیم داده‌های موجود به سه مجموعه، تعداد نمونه‌هایی را که می‌توان برای یادگیری مدل استفاده کرد، به شدت کاهش می‌دهد که راه حل آن استفاده از اعتبارسنجی متقابل است که در آن نیازی داده‌های اعتبارسنجی نیست [۲۵] و در ادامه توضیح داده می‌شود.

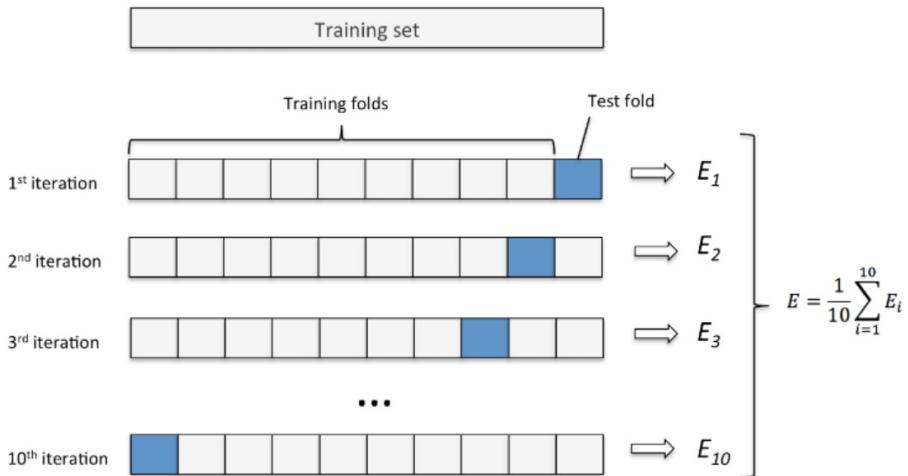
اعتبارسنجی متقابل^۲

برای جلوگیری از برآذش بیش از حد، می‌توان از تکنیک‌هایی استفاده کرد که داده‌های آموزشی و سنجش را به قسمت‌های بیشتری تقسیم کنند و آن‌ها را به طرق مختلف در مقابل هم مورد ارزیابی قرار دهد تا مدل دقت بیشتری پیدا کند.

این روش، شامل تقسیم داده‌های آموزشی به k مجموعه‌های کوچکتر و آموزش یک مدل با استفاده از $k-1$ عدد از این مجموعه‌ها است، در حالی که از مجموعه باقی مانده، به عنوان مجموعه آزمایشی، برای محاسبه میزان عملکرد مدل استفاده می‌شود. این فرآیند k بار تکرار می‌شود و هر مجموعه یک بار به عنوان مجموعه تست عمل می‌کند. سپس میانگین معیارهای عملکرد به دست آمده (که در فصل‌های آینده ذکر می‌شود) در هر تکرار، به عنوان معیار عملکرد کلی در نظر گرفته می‌شود. این روش از نظر محاسباتی گران است اما از هدر رفتن بیش از حد داده‌ها جلوگیری می‌کند [۲۵].

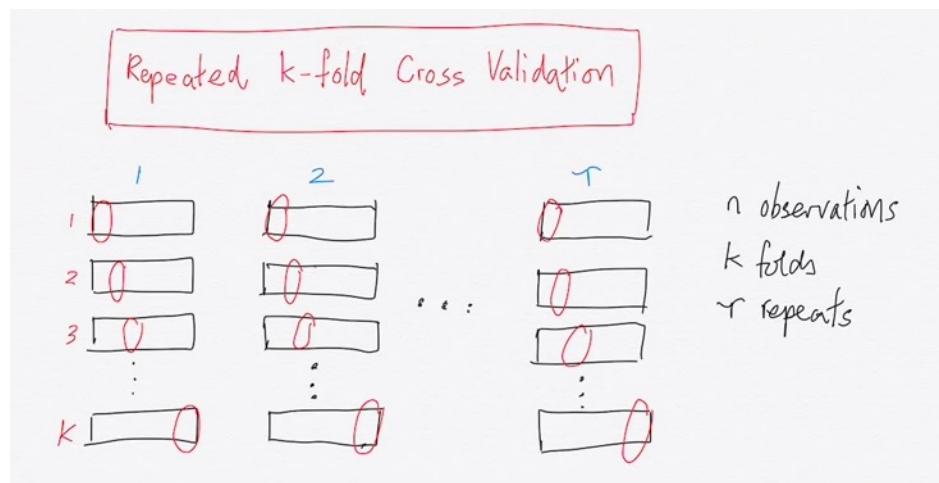
¹Data Splitting

²Cross validation



شکل ۲-۳: روش K-Fold

روش دیگری که ذکر شد، اعتبارسنجی K-Fold مکرر است که فرآیند اعتبارسنجی K-Fold را n بار با تقسیم‌های مختلف هر بار تکرار می‌کند. این روش زمانی مفید است که چندین بار لازم است فرآیند اعتبارسنجی تکرار شود [۲۵]. در واقع این روش علاوه بر تقسیم داده‌ها به K -Fold، فرآیند را چندین بار تکرار می‌کند و قبل از هر تکرار به طور مستقل داده‌ها را به هم می‌زنند. این روش با میانگین گرفتن مکرر اعتماد به مدل را افزایش می‌دهد [۱۵].



شکل ۲-۴: روش K-Fold تکرار شونده

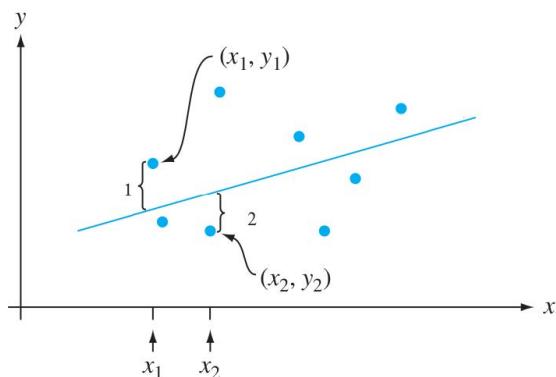
۳-۲ معرفی الگوریتم‌های طبقه‌بندی

۳-۲-۱ رگرسیون لجستیک^۳

خوب است در ابتدا مروری بر رگرسیون خطی^۴ داشت تا در ادامه بتوان رگرسیون لجستیک را بهتر درک کرد.

رگرسیون خطی:

در بسیاری از بررسی‌های آماری، لازم است یک متغیر وابسته، از روی یک یا چند متغیر مستقل پیش‌بینی شود که اصطلاحاً به آن رگرسیون یا برگشت می‌گویند [۲]. برای مثلاً میزان ساعت مطالعه یک متغیر مستقل است و نمره اخذ شده در درسی متغیری وابسته است و بین این دو رابطه وجود دارد. سپس نمونه‌ای از جمعیت را در نظر گرفته و در آن مقادارهای X_1 تا X_n در متغیر مستقل خود مقابله مقادیر نظیر در متغیر وابسته از Y_1 تا Y_n قرار داده [۲]. سپس آن‌ها را مثل یک نمودار در صفحه مختصات به یکدیگر متصل کرده که به آن نمودار پراکندگی گویند [۲].



شکل ۲-۵: نمودار پراکندگی

حالا می‌توان خطی را در این صفحه مختصات درنظر گرفت که تا حد زیادی منطبق بر نقاط باشد که در واقع یک نمودار پیش‌بینی‌کننده Y بر مبنای X است که به آن معادله رگرسیون Y بر روی X گویند [۲]. حالا رابطه این نقاط و منحنی را با $\mu_{Y|x} = E(Y|x) = \alpha + \beta x$ مشخص و α و β پارامترهایی هستند که باید مقدار دهی شوند تا خط بر نقاط منطبق باشد [۲]. مسئله‌ای که در اینجا مطرح است این است که ممکن است خط بر نقاط مختلف منطبق نشود. لذا اینجا باید حالت بهینه‌ای را در نظر گرفت حداقل مقدار خطا را (یا به بیان بهتر اختلاف) در مجموع داشته که روش حداقل مربعات برای یافتن میزان بهینه α و β که دارای بیشترین انطباق و کمترین خطا باشند، کمک‌کننده است [۲]. بر اساس همین روش، با معادلات زیر به مقادیر بهینه α و β می‌توان دست یافت [۲].

$$\alpha = \bar{y} - \beta \bar{x} \quad (1-2)$$

$$\beta = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} \quad (2-2)$$

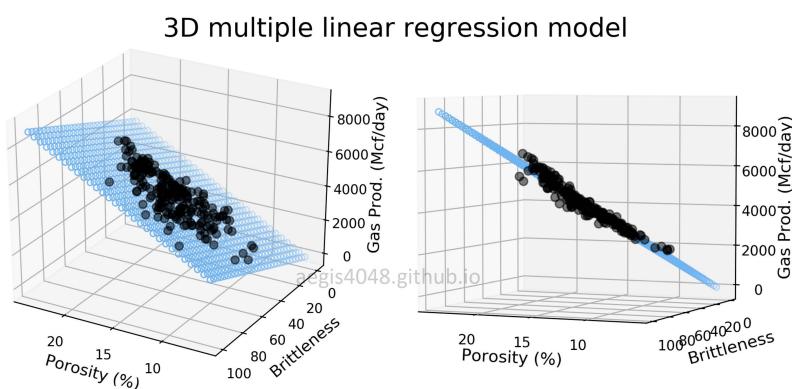
^۳Logistic Regression

^۴Linear Regression

که در روابط فوق \bar{x} و \bar{y} میانگین y و x هستند. با تعمیم این روابط و اصول بیان شده می‌توان حالتی را در نظر گرفت که چندین متغیر مستقل داشت. (مثلاً ۲ تا) که این مدل به رگرسیون خطی چندگانه معروف است و در آنجا نمودار حالت فضایی پیدا خواهد کرد و با رابطه زیر می‌توان آن را بیان کرد.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} + \epsilon \quad i = 1, \dots, n \quad (3-2)$$

که در این رابطه، p ابعاد مورد نظر می‌باشد [۳۴].



شکل ۲-۶: نمودار رگرسیون خطی چندگانه در فضا

در کل می‌توان گفت روش‌های رگرسیون زمانی مناسب است که مقادیر مستقل در مجموعه داده‌ها به کلاس‌ها (به بیان دیگر طبقه‌بندی‌ها) وابستگی داشته باشند. ضریب همبستگی خطی که با رابطه ۲ می‌شود میزان این وابستگی را نشان می‌دهد.

$$p(X, Y) = \text{corr}(X, Y) = \frac{\text{Cov}(x, y)}{(\text{Var}(x)\text{Var}(y))^{\frac{1}{2}}} \quad (4-2)$$

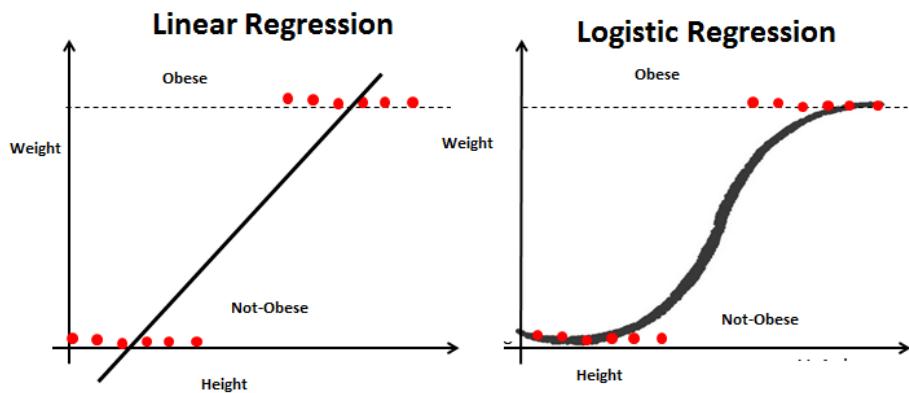
صورت کسر کواریانس x و y است و مخرج واریانس X و واریانس y می‌باشد. حاصل مقداری است از ۱ - تا ۱ که میزان وابستگی مستقیم یا معکوس را نشان می‌دهد [۱] و در صورت نبود میزان وابستگی مقدار ۰ است [۲].

رگرسیون لجستیک:

حالا که با مفاهیم ابتدایی رگرسیون آشنایی پیدا شد به بیان نوع دیگری از آن به نام رگرسیون لجستیک پرداخته می‌شود.

در این مدل به جای اینکه مقدار عددی برای متغیر وابسته تعریف شود، بر اساس احتمال متغیرهای رقیمی را برای پیشگویی در نظر گرفته [۱]. مثلاً در پروژه، فاکتور BMI در افراد یک متغیر مستقل است و بر ابتلا به دیابت در افراد موثر بوده؛ همچنین در اینجا متغیر وابسته، همان دیابت گرفتن یا نگرفتن فرد می‌باشد که با ۰ و ۱ در نظر گرفته شده است. پس این مدل برای مواردی استفاده می‌شود که حالت کلاس‌بندی برای متغیرها وجود داشته باشد [۸] [۹]. ضمناً در حالت کلاس بندی، نمی‌توان حالت ۰ یا ۱ را به عنوان اعدادی در حالت رگرسیون خطی منظور کرد؛ چرا که ۰ و ۱ را به عنوان مقادیر عدد

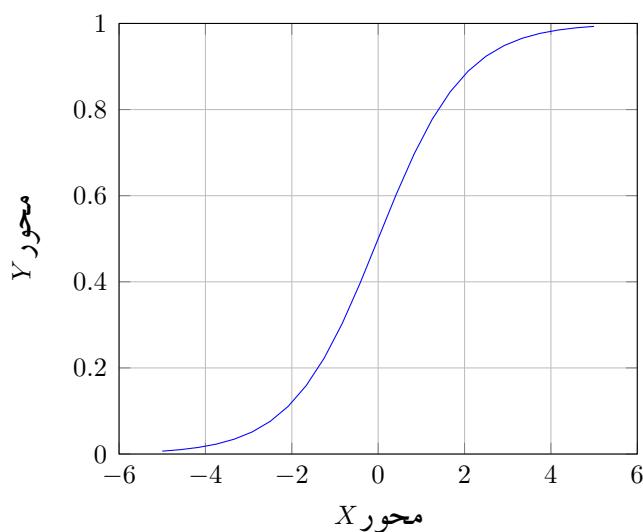
درنظر گرفته شده و امکان نمایش کلاس‌های گوناگون در یک نمودار خطی مشخص وجود ندارد [۹]. اگر به شکل ۷-۲ دقت شود، می‌توان متوجه شد در حالت رگرسیون خطی برخی از نمونه‌ها در کلاس مربوطه قرار نگرفته‌اند.



شکل ۷-۲: مقایسه رگرسیون لجستیک و خطی

علت این امر مشخص است. زیرا زمانی که کلاس‌های گوناگون باشند، ساختار استدلالی که در الگوریتم رگرسیون خطی مطرح است نمی‌تواند طبقه‌بندی صحیحی انجام دهد. نهایتاً چاره این است که از یک نمودار منحنی شکل برای فشرده کردن نتایج بین ° تا ۱ استفاده شود [۹]: رابطه ۵-۲ و شکل ۸-۲

$$\text{logistic}(x) = \frac{1}{1 + \exp^{-x}} \quad (5-2)$$



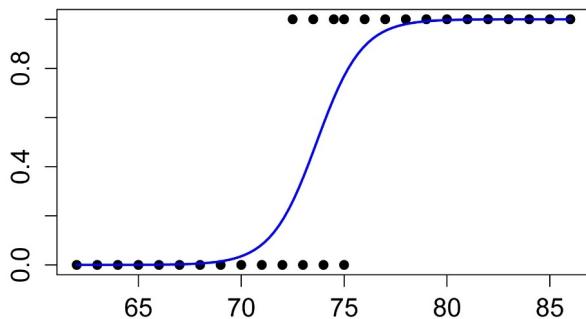
شکل ۸-۲: نمودار تابع لجستیک

سپس اگر سمت راست رابطه ۳-۲ در جای X در معادله ۵-۲ (که آن را تابع سیگماوار^۵ می‌نامند) قرار داده شود، فقط مقادیر ۰ و ۱ را خروجی می‌دهد.

حال در این مدل، چون قرار است هر نمونه به یک کلاس تعلق یابد، بر اساس روابط مطرح شده در این بخش و تعمیم خواص آماری (واریانس)، با دانش بر این که وزن ویژگی‌ها عاملی اثر گذار تعیین مرز جداسازی در نمودارها خواهد بود؛ لذا نهایتاً رابطه زیر از آن حاصل شده که بر اساس احتمال قرارگیری هر نمونه در هر کلاس، کلاس‌بندی را انجام می‌دهد [۸][۹]:

$$g(x) = \ln\left(\frac{p(x)}{1 - P(x)}\right) = \frac{e^{b_0 + b_1 x}}{1 + e^{b_0 + b_1 x}} = \ln(e^{b_0 + b_1 x}) = b_0 + b_1 x \quad (6-2)$$

در رابطه فوق، کاری که انجام می‌شود به این صورت است که احتمال قرارگیری یک نمونه در یک کلاس نسبت به حالت دیگر سنجیده می‌شود و در صورتی که مثلاً با احتمال بالای ۵۰ درصد در طبقه ۱ قرار می‌گیرد، این کار صورت می‌پذیرد و در غیر این صورت به کلاس ۰ متعلق است. (مشخص است احتمال دقیقاً ۵۰ درصد روی نقطه (۰،۰) قرار دارد.) [۸][۹] به بیان دیگر می‌توان گفت، Logit پاسخ Y، ترکیب خطی پیش‌بینی‌کننده‌ها (یا همان X) است. [۶]



شکل ۹-۲: مدل Logist Regression

۲-۳-۲ درخت تصمیم

در یادگیری ماشین، درخت تصمیم یک مدل پیش‌بینی‌کننده است که مجموعه‌ای از تصمیم‌ها و پیامدهای احتمالی را در ساختاری درخت‌مانند نشان می‌دهد و در واقع تصمیم‌گیری انسانی را تقليد می‌کند، جایی که تصمیمات منجر به تصمیمات بیشتر یا نتایج نهایی می‌شود [۱۲].

یک درخت تصمیم، می‌تواند ساختاری مشابه شکل ۱۱-۲ با توجه به یک دادگان نمونه که در شکل ۱۰-۲ داشته باشد [۱۱]. در این نمونه، چیزی که برآورد شده این است که یک مشتری که به فروشگاه کامپیوتر مراجعه می‌کند، آیا ممکن است در نهایت یک کامپیوتر از آن فروشگاه بخرد؟ لذا در این ساختار درختی، عناصر مختلف از آن ملاحظه می‌شود مثل ریشه^۶، برگ‌ها^۷ و گره‌های میانی و همچین شاخه‌ها^۸. مثلاً در اینجا، ابتدا سن بررسی می‌شود اگر میانسال باشد خرید را انجام می‌دهد. اما اگر

^۶Sigmoid/ Logistic Function

^۷Root node

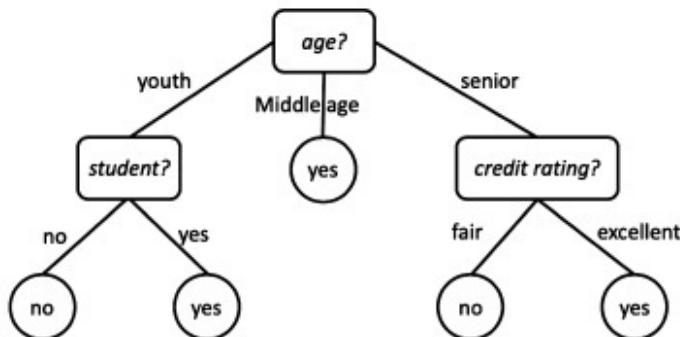
^۸Leaf node

^۹Branch

جوان باشد، تنها در صورتی که دانشجو باشد خرید را انجام می‌دهد. همچنین در صورت اینکه سن بالا داشته باشد، در صورت داشتن اعتبار مالی مناسب این خرید انجام می‌شود.

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

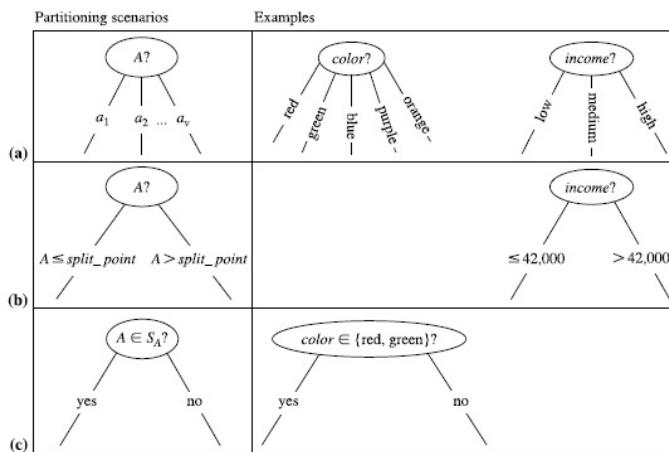
شکل ۲-۱۰: مثال دادگان مورد استفاده در درخت تصمیم



شکل ۲-۱۱: مثال درخت تصمیم

همچنین انواعی از حالت‌های شاخه‌ها را می‌توان داشت، که به ۳ صورت می‌توانند قرار گیرند.
(شکل ۲-۱۲)

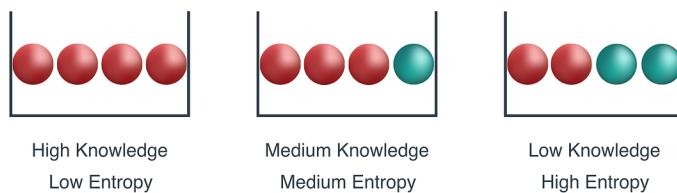
۱. ویژگی‌های گستته چندگانه مثل رنگ یا گروه‌های خونی
۲. ویژگی‌های مقداری پیوسته مثل درآمد یا سن (به صورت ویژگی شرطی که از مقداری کمتر یا بیشتر باشند).
۳. ویژگی‌های باینری مثل داشتن فعالیت فیزیکی یا نداشتن آن



شکل ۱۲-۲: انواع شاخه‌ها در درخت تصمیم

انتخاب بهترین ویژگی یکی از مسائلی که اینجا مطرح می‌شود، این است که در گره ریشه، و گره‌های بعدی، کدام ویژگی قرار داده شود که بهترین پیش‌بینی انجام شود؟ برای این کار چندین روش وجود دارد که دو روش از آن‌ها در اینجا معرفی شده است:

- آنتروپی اطلاعات^۹ در شکل ۱۳-۲ به سادگی مفهوم آنتروپی به تصویر کشیده شده است (که در سطحی که نوع خاصی از توابع‌ها به طور قابل توجهی بیشتر از انواع دیگر باشد، آنتروپی کمتر است.). [۲۹] لذا اگر یک نمونه از سطحی که دارای آنتروپی کمتری است برداشته شود، می‌توان گفت با احتمال بالایی نوع آن توب به درستی پیش‌بینی شده است.



شکل ۱۳-۲: نمونه‌ای از آنتروپی

همین ایده، در انتخاب بهترین ویژگی در گره‌های درخت تصمیم می‌تواند مورد استفاده قرار گیرد. یعنی باید ویژگی‌هایی را برای هر گره در نظر گرفت که کمترین آنتروپی را داشته باشد. در واقع آنتروپی معیاری از ناچالصی یا تصادفی بودن در یک مجموعه داده است. از رابطه ۷-۲، برای محاسبه آنتروپی ویژگی‌ها استفاده می‌شود [۱۲].

$$H(X) = - \sum_{i=1}^n p_x \log_2 p_x \quad (7-2)$$

در این رابطه :

- آنتروپی $H(X)$ آنتروپی مجموعه داده X است.

^۹Information entropy (Information gain ratio)

- p احتمال وقوع هر کلاس ممکن در مجموعه داده است.
- نهایتاً مجموع بر روی تمام مقادیر کلاس ممکن گرفته می‌شود.

• ضریب Gini

فرمول Gini نیز برای محاسبه ناخالصی یا ناهمگنی یک گره استفاده می‌شود و احتمال طبقه‌بندی نادرست یک عنصر تصادفی انتخاب شده در یک مجموعه داده را اندازه‌گیری می‌کند [۱۲].

این شاخص از ۰ تا ۱ متغیر است. مقدار ۰ نشان می‌دهد که گره خالص است، به این معنی که همه عناصر در گره متعلق به یک کلاس هستند. بر عکس، مقدار ۱ نشان دهنده ناخالصی کامل است، جایی که عناصر از کلاس‌های مختلف به طور مساوی در گره توزیع می‌شوند [۱۲].

شاخص جینی با جمع کردن مجدد احتمالات هر کلاس در گره و کم کردن آن از یک محاسبه می‌شود. می‌توان آن را با فرمول زیر نشان داد:

$$Gini = 1 - \sum_j (p_j)^2 \quad (۸-۲)$$

این روش نیز مشابه روش قبلی (که در الگوریتم CART) نیز استفاده می‌شود، p_j نشان دهنده احتمال انتخاب تصادفی یک عنصر متعلق به کلاس زاست [۱۲].

الگوریتم‌های گوناگونی برای ساخت درخت تصمیم استفاده می‌شود مثل (... ID3, C4.5, CART) که در کتابخانه مورد استفاده پروره یعنی scikit-learn از الگوریتم CART برای ساخت درخت تصمیم استفاده می‌شود [۲۸] (این الگوریتم مانند ID3 و C4.5 حریصانه است) و در اینجا عملکرد آن بیان می‌شود.

الگوریتم CART ^۱: الگوریتم درخت طبقه‌بندی و رگرسیون، یک الگوریتم درخت تصمیم محبوب است و با بخش‌بندی بازگشتی مجموعه داده بر اساس مقادیر ویژگی‌های مختلف، برای رسیدن به یک تصمیم یا پیش‌بینی نهایی عمل می‌کند. این یک الگوریتم حریصانه است (یعنی درخت را از بالا به پایین بر اساس بهترین ویژگی در گره‌ها، می‌سازد [۱۱]). مرحله کلی این الگوریتم‌ها تقریباً مشابه همدیگر هستند و به این ترتیب است [۱۱].

۱. ابتدا همه ویژگی‌ها را در ریشه قرار داده.
۲. بهترین آن‌ها را بر اساس یک ویژگی مثل آنتروپی یا ضریب Gini در آن گره انتخاب شود.
۳. شاخه‌ها ایجاد شود.

۴. این روند در هر گره ادامه داده شود تا به نتایج (برگ‌های نهایی) دست یافته شود.
همچنین لازم به ذکر است که درختان تصمیم‌گیری تمایل دارند که داده‌های آموزشی را بیش از حد برآراش دهند، بسیار خاص می‌شوند و قابلیت تعمیم را از دست می‌دهند. هر س درخت، فرآیند کاهش اندازه درخت تصمیم برای جلوگیری از برآراش بیش از حد است که شاخه‌های غیر ضروری را برای افزایش سادگی و دقت با روش‌های مختلف کوتاه می‌کند [۱۱].

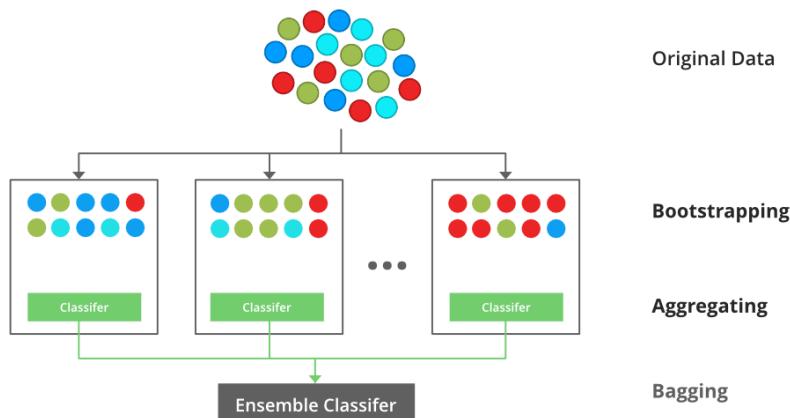
- پیش‌حرس: پیش روی ساخت گره را در عمق خاصی را محدود کرد.

^۱ Classification and Regression Trees

- پس حرس: برخی از گره‌ها با روش‌هایی انتخاب و به برگ تبدیل می‌شوند. سپس مجدداً معیارهای ارزیابی خاصی را در نظر گرفته که مشخص کند این حرس دقت مدل را افزایش می‌دهد یا نه. مشخصاً این روش کنتر است.

۳-۳-۲ جنگل درختان تصادفی

این الگوریتم از ساختار درختان تصادفی، مجموعه‌ای از درختان تصمیم را با انتخاب تصادفی چندین زیر مجموعه، از مجموعه مرجع داده‌های آموزشی ایجاد (که این را روش Bootstrap می‌نامند) و مجدداً انتخاب زیر مجموعه‌ای از برخی ویژگی‌های دادگان را در هر تقسیم ایجاد می‌کند. هر درخت تصمیم به طور مستقل بر روی نمونه‌های مختلف Bootstrap و زیر مجموعه‌های ویژگی آموزش داده می‌شود (انتخاب ویژگی مطابق مباحث قبلی در درخت تصمیم یعنی با ضریب جینی یا آنتروپی است [۱۲]) و با درختان دیگر متفاوت است. هنگام انجام یک پیش‌بینی، تمام درختان جنگل تصادفی به نتیجهٔ نهایی رأی می‌دهند و طبقه‌بندی‌ای که اکثریت آرا را داشته باشد، به عنوان کلاس یا طبقه‌بندی پیش‌بینی شده نهایی، انتخاب می‌شود [۱۲].



شکل ۲: روش Bootstrap در جنگل تصادفی

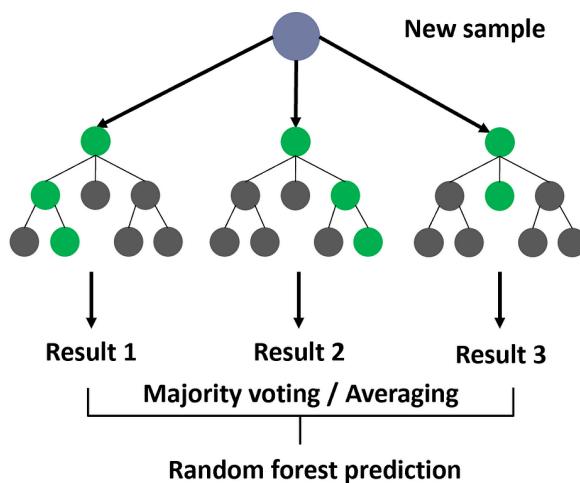
با ترکیب پیش‌بینی‌های درخت‌های تصمیم چندگانه، الگوریتم جنگل تصادفی به کاهش بیش‌برازش و بهبود عملکرد تعمیم کمک می‌کند. همچنین پایداری بیشتر در ساخت مدل در اثر به وجود آمدن مواردی چون نقاط دورافتاده و داده‌های گم شده را فراهم می‌کند [۱۲].

مثالی که می‌توان برای یک نمونه از این مدل زد، این است در یک جنگل تصادفی ۳ درخت حکم می‌کنند که فرد به دیابت مبتلا خواهد شد و ۱ درخت حکم می‌کند که این فرد به دیابت مبتلا نخواهد شد. لذا اکثریت آراء بر مبتلا شدن فرد اتفاق نظر دارند. پس نتیجهٔ آن پیش‌بینی، مبتلا شدن فرد است.

حال اگر فرض شود قصد ساخت یک مدل با این روش باشد، این مراحل به ترتیب انجام می‌شود [۶]:

۱. مجموعه داده را به دو قسمت آموزشی و آزمایشی تقسیم کرده. از مجموعه آموزشی داده شده، یک مجموعه داده جدید با استفاده از روش Bootstrap ایجاد شود.
۲. بر اساس نتایج مرحله ۱ یک درخت تصمیم ساخته شود.

۳. مرحله ۱ و ۲ تکرار شود و درختان زیادی را تولید شود که از یک جنگل تشکیل شده اند.
۴. از هر درخت در جنگل برای رأی دادن به ورودی (یک ردیف داده که نتیجه آن پیش‌بینی مطلوب است) استفاده شود.
۵. میانگین آرا برای هر کلاس محاسبه شود. کلاسی که بیشترین رای را می‌دهد متعلق به برچسب طبقه‌بندی برای ورودی داده است.
۶. در نهایت دقت طبقه‌بندی کننده مبتنی بر جنگل تصادفی را محاسبه کنید.



شکل ۲: مدل جنگل درختان تصادفی

AdaBoost ۴-۳-۲

این الگوریتم یک فرض ساده در نظر دارد و آن یادگیری گروهی است [۱۳]. فرض شود کسانی که در مجموعه داده‌های آموزشی دچار اشتباه در تشخیص ابتلا به بیماری دیابت شدند، از مجموعه بقیه داده‌های آموزشی جدا می‌شوند و در یک طبقه‌بندی جدید مجدداً مورد ارزیابی قرار می‌گیرند؛ همان‌طور که در واقعیت ممکن است هرپزشکی معیارهای مختلفی را برای تشخیص بیماری مراجعه کننده‌اش داشته باشد و هر پزشکی نمی‌تواند همه افراد را درست تشخیص دهد پس اگر یک تیم پزشکی داشته، نظرات جمعی می‌توانند بیماران بیشتری را به درستی شناسایی کنند [۱۱].

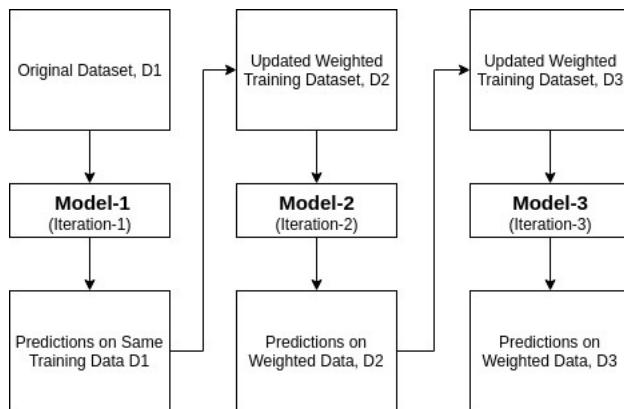
این روند جداسازی داده‌های آموزشی و امتیاز دهی در تشخیص توسط طبقه‌بندی کننده‌های مختلف درون درختان تصمیم ضعیف، مکرراً تکرار می‌شود تا درنهایت طبقه‌بندی‌های متعددی داشته که هر کدام بر اساس میزان سرآمدی در تست‌های مختلف، امتیازات مختلفی دریافت کنند [۱۳]. بعد از ساخته شدن مدل، حالا می‌توان به نسبت امتیاز هر طبقه‌بندی کننده داده‌های دریافته را به صورت شناسی بین هر کدام تقسیم کرد تا پیش‌بینی صورت گیرد و این مسئله موجب تقویت سنجش می‌گردد. به همین دلیل به آن الگوریتم Adaptive Boosting یعنی تقویت کننده تطبیقی گویند [۱۱] [۱۳].

اگر از دید ریاضی الگوریتم بررسی شود، برای محاسبه میزان خطای مدل، M_i وزن هر یک از تاپل‌های D_i را که اشتباه طبقه‌بندی کرد، جمع می‌کنند.

$$\text{error}(M_i) = \sum_{j=1}^d w_i \times \text{err}(X_j) \quad (9-2)$$

حالا اگر طبقه‌بندی‌کننده X_j اشتباه کند، میزان ارور برای آن ۱ اندازه‌گیری شده و در غیر این صورت ۰ است. اگر یک طبقه‌بندی‌کننده آن قدر ضعیف باشد که خطایش از ۰.۵ بیشتر شود آن را دیگر در نظر نمی‌گیرند [۱۱]. سپس یک مجموعه داده جدید به نام D_i تولید و از آن یک M_i جدید استخراج و دوباره این روند ادامه داده می‌شود. هر چه میزان خطای طبقه‌بندی‌کننده کمتر باشد، دقیق‌تر است و بنابراین، وزن آن برای انتخاب شدن باید بیشتر باشد. از رابطه ۱۰-۲ برای محاسبه وزن هر طبقه‌بندی استفاده می‌شود [۱۱].

$$\log\left(\frac{1 - \text{error}(M_i)}{\text{error}(M_i)}\right) \quad (10-2)$$



شکل ۲-۱۶: الگوریتم AdaBoost: در اینجا مراحل ذکر شده به صورت متناوب تکرار می‌شود [۳۰].

Naïve Bayes ۵-۳-۲

طبقه‌بندی‌کننده بیز ساده بر روی مفهوم احتمال شرطی کار می‌کند و به این به این سؤال پاسخ می‌دهد که احتمال اینکه یک تاپل داده شده از تعلق یک مجموعه داده به یک کلاس خاص، چقدر است [۱۲].

$$P(H|x) = \frac{P(x|H)P(H)}{P(x)} \quad (11-2)$$

- $P(H|x)$: احتمال وقوع رویداد H با توجه به اینکه رخداد X به وقوع پیوسته است. (احتمال پسین) (مثلاً احتمال این که فردی با ویژگی خاصی به دیابت مبتلا شود.)
- $P(X|H)$: احتمال وقوع رویداد X با توجه به اینکه رویداد H به وقوع پیوسته است. (برعکس مورد قبل است، یعنی چند درصد از افراد دارای دیابت، ویژگی مورد نظر را مثلاً چاقی را دارا می‌باشند.)

^{۱۱}Likelihood

- $P(X)$: احتمال رویداد X (درصد افرادی که ویژگی خاصی را در دادگان دارند، مثلاً در کل فلان درصد از افراد چاق هستند.)

- $P(H)$: احتمال رویداد H (احتمال پیشین) (مثلاً مشخص است که احتمال مبتلا شدن افراد چاق به دیابت درصد خاصی دارد.)

حال به عنوان مثال، برای مبتلا شدن به دیابت، این احتمال را برای افراد محاسبه و احتمال وقوع هر کدام از کلاس‌ها که بیشتر شد (رابطه ۱۲-۲)، آن داده متعلق به همان طبقه است [۱۲]. (رابطه ۱۳-۲)

$$P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)} \quad (12-2)$$

$$\text{argmax} P(C_i|x) = \text{argmax} P(x|C_i)P(C_i) \quad (13-2)$$

در نهایت، زمانی که این احتمال برای تک تک ویژگی‌ها برای هر کلاس به صورت جداگانه محاسبه شد، نتیجه را برای هر ویژگی در همان کلاس ضرب کرده تا در کل مشخص شود برای این ویژگی‌ها، وقوع کدام کلاس متحمل‌تر است [۱۲]: (رابطه ۱۴-۲)

نکته‌ای که قابل ذکر است این است که چون ممکن است احتمال موردی صفر شود و در عملیات ضرب نتیجه کل حاصل شده صفر شود، به هر احتمال عدد ۱ اضافه می‌شود که این روش را Laplacian correction می‌نامند [۲۷].

$$P(c|x) = P(x_1|x) \times P(x_2|x) \times P(x_3|x) \times \dots \times P(x_n|x) \times P(c) \quad (14-2)$$

یکی از مزایایی که این روش دارد این است که روش سریعی است، به این دلیل که پس از افزودن داده‌های جدید، نیاز نیست مدل مجدداً باز سازی شود [۱۲].

۴-۲ دادگان، پیش‌پردازش و مصوّرسازی داده‌ها

۱-۴-۲ دادگان

دادگانی^{۱۲} که از آن استفاده شد، برگرفته از یک برنامه مطالعاتی به نام NHANES بوده که جهت بررسی سلامتی کودکان و بزرگسالان از سوی CDC^{۱۳} تدوین شده است.

برنامه NHANES در اوایل دهه ۱۹۶۰ آغاز شد و به صورت مجموعه‌ای از نظرسنجی‌ها با تمرکز بر گروه‌های مختلف جمعیتی یا موضوعات بهداشتی انجام شده است. در سال ۱۹۹۹، این نظرسنجی به

^{۱۲}Dataset

^{۱۳}US Centers for Disease Control and Prevention

یک برنامه مستمر تبدیل شد که تمرکز در حال تغییری بر روی انواع اندازه گیری‌های سلامت و تغذیه برای رفع نیازهای نوظهور دارد [۱۹].

صاحبہ NHANES شامل سوالات جمعیت شناختی، اجتماعی-اقتصادی، رژیم غذایی و سلامتی است. جزء معاینه شامل اندازه گیری‌های پزشکی، دندانی و فیزیولوژیکی و همچنین تست‌های آزمایشگاهی است که توسط پرسنل پزشکی بسیار آموزش دیده انجام می‌شود [۱۹].

۲-۴-۲ پیش‌پردازش داده‌ها

پیش‌پردازش داده‌ها چیست؟

داده‌هایی که جمع آوری می‌شود، انواع مختلفی دارند. مانند رشته‌ها، انواع اعداد، مقادیر نامشخص و همچنین منابع داده‌های نیز ممکن است طبقه‌بندی‌های مختلفی از داده‌ها را با فرمتهای گوناگون ارائه کنند که کار را برای سیستم یادگیری ماشین دشوار می‌سازد.

داده‌های گم‌شده چیست؟

وقتی دادگان مورد بررسی قرار گرفت، مشخص شد در برخی از سلول‌ها، جای برخی مقادیر خالی هستند (در پروسه جمع آوری داده‌ها این اطلاعات به هر دلیلی ثبت نشده‌اند) یا در اثر عملیات‌های مربوط به شناسایی داده‌های پرت، برخی از داده‌ها را حذف کرده و جای آن‌ها خالی می‌ماند. در این موقع باید با استفاده از تدابیری، داده‌های گم‌شده را با مقادیری جایگزین کرد یا کلا آن رکوردها را حذف کرد تا مدل‌های دقیق‌تری پدید آید. (شکل ۲-۱۷)

	Gender	Age	Race1	Education	MaritalStatus	Work	Weight	Height	BMI	BPSysAve	BPDiaAve	DirectChol
0	male	34	White	High School	Married	NotWorking	87.4	164.7	32.22	113.0	85.0	1.29
1	male	34	White	High School	Married	NotWorking	87.4	164.7	32.22	113.0	85.0	1.29
2	male	34	White	High School	Married	NotWorking	87.4	164.7	32.22	113.0	85.0	1.29
3	male	4	Other	Nan	Nan	Nan	17.0	105.4	15.30	NaN	NaN	NaN
4	female	49	White	Some College	LivePartner	NotWorking	86.7	168.4	30.57	112.0	75.0	1.16

شکل ۲-۱۷: نمونه‌ای از داده‌های گم‌شده

Data columns (total 15 columns):			
#	Column	Non-Null Count	Dtype
0	AgeDecade	9858	non-null object
1	Race1	9858	non-null object
2	Poverty	9858	non-null float64
3	Work	9858	non-null object
4	DirectChol	9858	non-null float64
5	TotChol	9858	non-null float64
6	HealthGen	9858	non-null object
7	Depressed	9858	non-null object
8	SleepTrouble	9858	non-null object
9	SmokeNow	9858	non-null object
10	HardDrugs	9858	non-null object
11	SameSex	9858	non-null object
12	SexOrientation	9858	non-null object
13	PregnantNow	9858	non-null object
14	Diabetes	9858	non-null object
dtypes: float64(3), object(12)			

شکل ۱۸-۲: تعداد داده‌های گم شده در این پروژه به ازای هر ستون

راه حل‌های داده‌های گم شده

• حذف ردیف‌های حامل داده‌های گم شده

یکی از راهکارهایی که در هنگام کار با داده‌های گم شده انجام می‌شود، حذف کل رکوردهایی است که دارای این مقادیر هستند. این مسئله یک بدی دارد و بدی آن این است که داده‌های کمتری برای آموزش مدل در اختیار داشته و مدل ضعیفتر خواهد بود. اما اگر به هر دلیلی نتوان این مقادیر را با مقادیری دارای تقریب خوب پر کرد، چاره دیگری نیست. [۱۱]

• جایگزینی با متغیرهای آماری

یکی از راهکارهای دیگر برای مدیریت داده‌های گم شده، جایگزین کردن مقادیر گم شده با جایگزینی با متغیرهای آماری است. برخی از ستون‌ها را که زیاد اهمیت نداشته باشند را می‌توان با مقادیر میانگین پر کرد. مثلاً اگر دادگانی که حاوی اطلاعات مشتریان یک بانک وجود داشته باشد که خواسته شود از آن برای وام دادن به آن‌ها بهره برد، می‌توان در ستونی که مربوط به میزان حقوق ماهیانه هر فرد است، در صورت مشاهده مقادیر NaN، آن‌ها را با میانگین حقوق مشتریان جایگزین کرد. این راهکار، صرفاً به افراد امکان می‌دهد تا تحلیل را ادامه داده و اطلاعات موجود در این رکورد را برای متغیرهای دیگر از دست ندهند [۱۲].

استاندارد سازی داده‌ها

برای ادامه مراحل، لازم بود کارهای بیشتری را روی داده‌های انجام می‌گرفت. از جمله ایجاد متغیرهای ساختگی (رقمی) و مقیاس‌بندی متغیرهای ساختگی (رقمی)

در مدل‌های مختلف یادگیری ماشین به عنوان مثال در همین رگرسیون لجستیک، لازم است تا تنها متغیرهای عددی به عنوان ورودی جهت مدل سازی ارائه شود و این مدل نمی‌تواند متغیرهای رشته‌ای را تشخیص دهد. پس از یک راهکار استفاده می‌شود. مقادیر درون ستون‌های رشته‌ای را با یک مقدار

عددی از ۱ تا n جایگزین می‌کنند. به عنوان مثال در ستون مربوط به نژاد فرد، به هر کدام از نژادهای گوناگون عددی اختصاص داده می‌شود. (شکل ۱۹-۲)

Decade	Race1	Poverty	Work	DirectChol	TotChol	HealthGen	Depressed	SleepTrouble	SmokeNow	HardDrugs	SameSex	SexOrientation	PregnantNow	Dial
3	4	1.36	1	1.290000	3.490000	2	2	1	0	1	0	1	0	
3	4	1.36	1	1.290000	3.490000	2	2	1	0	1	0	1	0	
3	4	1.36	1	1.290000	3.490000	2	2	1	0	1	0	1	0	
0	3	1.07	2	1.365029	4.878875	2	1	0	0	0	0	1	0	
4	4	1.91	1	1.160000	6.700000	2	2	1	1	1	1	1	0	

شکل ۱۹-۲: نمونه‌ای از متغیرهای رقمی

مقیاس‌بندی در نرمال‌سازی

برخی از الگوریتم‌ها نیاز دارند که داده‌ها قبل از پیاده سازی مؤثر الگوریتم نرمال‌سازی شوند. برای نرمال‌سازی یک متغیر، میانگین را از هر مقدار کم کرده و سپس بر انحراف استاندارد تقسیم می‌شود. این عملیات گاهی اوقات استانداردسازی نیز نامیده می‌شود.^[۱۲]

علت این امر این است که واحد اندازه‌گیری مورد استفاده می‌تواند بر تجزیه و تحلیل داده‌ها تأثیر بگذارد. به عنوان مثال، تغییر واحدهای اندازه‌گیری از متر به اینچ برای قد، یا از کیلوگرم به پوند برای وزن، ممکن است به نتایج بسیار متفاوتی منجر شود.^[۱۲] به طور کلی، درنظرگرفتن یک ویژگی در واحدهای کوچک‌تر، به محدوده بزرگتری برای آن منجر می‌شود و بر وزن آن ویژگی اثرگذار است. برای جلوگیری از به وجود آمدن این مشکل، داده‌ها باید نرمال یا استاندارد شوند. یعنی داده‌ها در محدوده‌های کوچک‌تر یا نرمال‌تر قرار بگیرند. این مسئله باعث می‌شود تا به همه ویژگی‌ها وزن یکسانی داده شود. [۱۲] لذا قبل از انجام این عملیات، بررسی داده‌ها و ساختارشان دارای اهمیت است. [۲۶]

برای روش‌های مبتنی بر فاصله، نرمال‌سازی به جلوگیری از برتری ویژگی‌هایی با دامنه‌های اولیه بزرگ (مانند درآمد) کمک می‌کند تا ویژگی‌هایی با دامنه‌های اولیه کوچک‌تر (مانند ویژگی‌های باينری) سبقت بگیرد. [۱۲]

روش مورد استفاده در این پژوهه از بین روش‌های مختلف StandardScaler است.

از رابطه زیر مقدار جدید ویژگی مقیاس‌بندی شده به دست می‌آید [۲۶]:

$$x_{scaled} = (x - mean(x)) \div StandardDeviation(x) \quad (15-2)$$

در واقع استانداردسازی فرآیندی است که متغیر را روی ۰ (میانگین صفر) قرار می‌دهد و واریانس را به ۱ (واریانس واحد) استاندارد می‌کند. پس از مقیاس‌بندی استاندارد، انحراف معیار نیز ۱ (انحراف معیار استاندارد) خواهد بود. در واقع برای استاندارد کردن ویژگی ها، میانگین را از هر مشاهده کم کردم و سپس نتیجه را بر انحراف استاندارد تقسیم کردم. خاصیت دیگر این کار، سرعت بخشیدن به روند ساخته شدن مدل‌های یادگیری ماست.^[۲۶]

نتیجه این تبدیل، z-score نامیده می‌شود که نشان می‌دهد یک مشاهده معین از میانگین چند انحراف استاندارد انحراف دارد. از این رو، استانداردسازی، نرمال‌سازی امتیاز z نیز نامیده می‌شود.^[۲۶]

در ایجاد بسیاری از مدل‌های یادگیری ماشین انجام این فرایند ضروری است. [۱۲]

نمونهبرداری مجدد

نمونهبرداری مجدد^{۱۴} به تکنیک‌هایی اشاره دارد که توزیع مجموعه داده را برای مشکل نامتعادل بودن کلاس خاصی، انجام می‌دهد. دو روش رایج نمونهبرداری مجدد عبارتند از نمونهبرداری کم و نمونهبرداری بیشازحد [۳۳] [۱۲].

نمونهبرداری کم^{۱۵} شامل کاهش اندازه طبقه اکثريت با حذف برخی از نمونهها است. اين کار می‌تواند به متعادل شدن توزیع آماری نمونهها کمک کند، اما ممکن است منجر به از دست رفتن اطلاعات مهم شود. [۳۳].

از طرف ديگر، نمونهبرداری بیشازحد^{۱۶} شامل روش‌های افزایش اندازه طبقه اقلیت با افزودن نمونه‌های مصنوعی یا تکراری است. هدف این تکنیک، ارائه نمونه‌های آموزشی بیشتر برای طبقه اقلیت و بهبود توانایی مدل در یادگیری الگوهای داده‌ها است [۳۳].

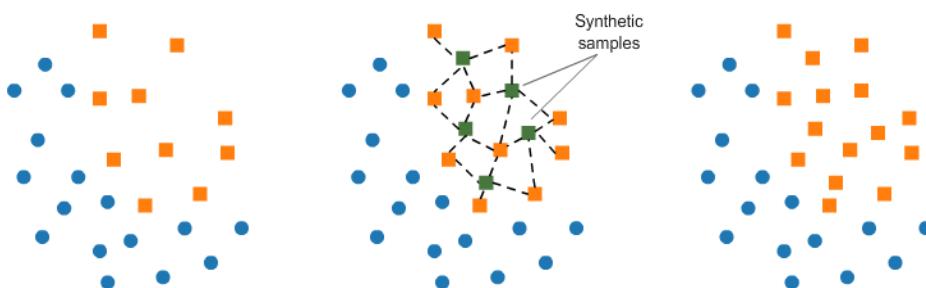
SMOTE^{۱۷} یک روش محبوب برای نمونهبرداری بیشازحد از کلاس اقلیت است که برای تولید نمونه‌های مصنوعی مابین نمونه‌های کلاس اقلیت موجود طراحی شده است [۱۶] [۳۳].

نحوه عملکرد SMOTE:

۱. برای هر نمونه کلاس اقلیت، k همسایه نزدیک خودش (معمولاً $k=5$) بر اساس متريک فاصله انتخاب می‌شود.

۲. یک نمونه مصنوعی جدید با انتخاب تصادفي یکی از k نزدیکترین همسایه‌هایش تولید می‌شود. تولید اين نمونه، در امتداد خطی که نمونه اصلی و همسایه انتخابی را به هم متصل می‌کند، انجام می‌شود.

۳. اين فرآيند برای تولید تعداد مورد نظر نمونه مصنوعی برای کلاس اقلیت تکرار می‌شود.



شکل ۲-۲۰: نمونهبرداری مجدد

SMOTE با تولید نمونه‌های مصنوعی، به طور موثر فضای ویژگی‌های کلاس اقلیت را گسترش می‌دهد. اين کار به مدل کمک می‌کند تا الگوهای مبنا را بیاموزد و عملکرد خود را در طبقه اقلیت بهبود بخشد [۳۳].

^{۱۴}Resampling

^{۱۵}Underfitting

^{۱۶}Overfitting

^{۱۷}Synthetic Minority Oversampling Technique

۳-۴-۲ انتخاب ویژگی

انتخاب ویژگی^{۱۸}، فرآیند انتخاب مرتبط ترین ویژگی‌های یک مجموعه داده برای ساخت یک مدل یادگیری ماشین است. یعنی زیرمجموعه‌ای از ویژگی‌ها را از مجموعه داده بتوان به دست آورد که دقت پیش‌بینی را به حد اکثر برساند [۲۰]. انجام درست این مرحله، به بهبود عملکرد مدل، کاهش بیش‌برازش و افزایش اعتبار مدل، کمک می‌کند [۱۲].

یک از راه‌هایی که برای انتخاب ویژگی وجود دارد، روش حذف ویژگی بازگشتی (RFE)^{۱۹} است که معمولاً با رگرسیون لجستیک انجام می‌شود. RFE با حذف مکرر ویژگی‌های کم اهمیت از مجموعه داده، تا رسیدن به تعداد مطلوبی از ویژگی‌ها کار می‌کند [۲۰].

از مزایای RFE می‌توان به توانایی آن در مدیریت مجموعه داده‌های بزرگ و کار با هر الگوریتم یادگیری نظارت شده اشاره کرد [۲۱].

نحوه کار این الگوریتم به شرح زیر است [۲۱]:

۱. اهمیت همه ویژگی‌ها با استفاده از الگوریتم یادگیری ماشین داده شده به RFE رتبه بندی می‌شود.
 ۲. کم اهمیت ترین ویژگی حذف می‌شود.
 ۳. با استفاده از ویژگی‌های باقی‌مانده یک مدل ساخته می‌شود
 ۴. مرحله ۳-۱ تکرار شده تا به تعداد مورد نظر ویژگی‌ها برسیم.
- ضممنا برای استفاده از RFE، داده‌ها باید مقیاس‌بندی و نرمال‌سازی شده باشند [۲۱].

۴-۴-۲ مصورسازی

تعريف: به طور ساده می‌توان گفت زمانی که داده‌هایمان را به صورت انواع نمودارها، نقشه‌ها و شکل‌های مختلف بصری درآورده شود تا نتیجه‌گیری و تحلیل آن‌ها توسط مغز جهت شناسایی الگوها و نقاط پرت در داده آسان‌تر شود، این کار انجام می‌گیرد [۲۴].

اهمیت: یک تصویر هزاران برابر بیشتر از کلمات ارزش دارد.

جمله فوق، به نوعی اهمیت مصورسازی را نمایان می‌کند. درواقع با انجام این‌کار، درک بهتر و سریع‌تری نسبت به داده‌های عظیم وجود خواهد داشت. قابل ذکر است که Dataset مورد استفاده در اینجا، حدود ۱۰۰۰۰ رکورد را در خود جای داده است.

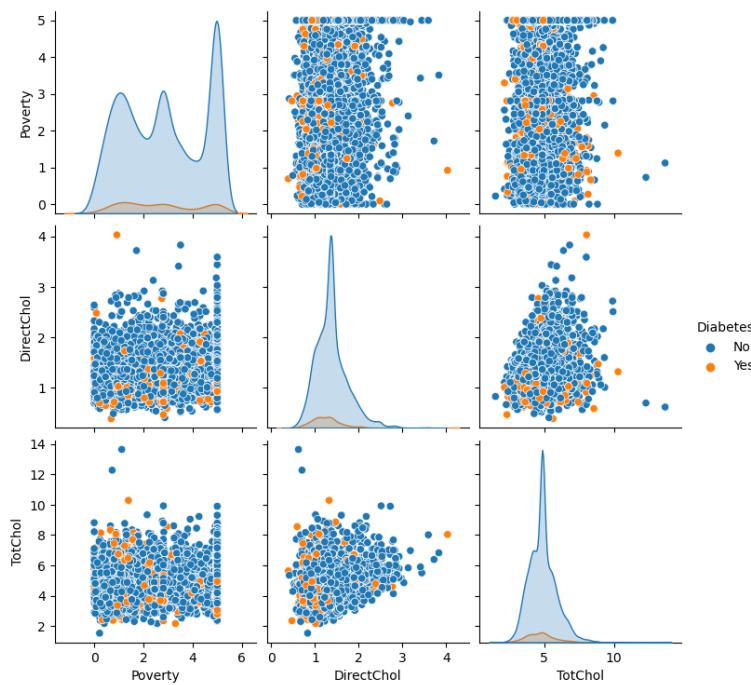
ضممنا در کنار این موارد، با مصورسازی داده‌ها دیگر نیازی به توضیحات اضافه نخواهد بود و بسیاری از افراد عادی قادر به درک موضوع مطرح شده خواهد بود [۱۰]. مواردی چون استاندارد نبود و لزوم استفاده از روش‌های نمونه‌برداری نیز در اینجا مشخص می‌شود. [۱۲]

۵-۴-۲ انواع نمودارها و داده‌های اجمالی

نمودار Scatter یا Pairplot

^{۱۸}Feature selection

^{۱۹}Recursive Feature Elimination



شکل ۲۱-۲: نمودار Pairplot قبل از نمونه‌برداری مجدد

در شکل ۲۱-۲ نمودار Pairplot مشاهده می‌شود. این نمودار برای تعجیم رابطه بین دو متغیر استفاده می‌شود. هر نقطه داده با یک نقطه در نمودار نشان داده می‌شود که محور x و y متغیرهای مختلف را نشان می‌دهد. نمودارهای پراکنده‌گی، به شناسایی خوشها، الگوها یا همبستگی‌های بین متغیرها کمک می‌کند و محققان را قادر می‌سازد تا در انتخاب شاخصه‌های کلیدی، تصمیمات آگاهانه بگیرند [۱۰].

در این نمودار، نقاط نارنجی رنگ نشان دهنده افراد مبتلا به دیابت است و نقاط آبی رنگ نشان دهنده افرادی بدون ابتلا به این بیماری است و این نمودار از دادگان همین پروژه استخراج شده است. به راحتی می‌توان تشخیص داد که افرادی که به دیابت مبتلا هستند در هر ویژگی فیزیولوژی در چه سمتی تمرکز دارند.

نقشه‌های حرارتی

نقشه‌های حرارتی در تعجیم مقادیر زیادی از داده‌ها در قالب ماتریس مانند موثر هستند. آنها از گرادیان رنگ برای نشان دادن مقدار مقادیر در دو متغیر استفاده می‌کنند. نقشه‌های حرارتی به ویژه در شناسایی همبستگی‌ها و کشف الگوهای پنهان در مجموعه داده‌های چند بعدی مفید هستند، و به انتخاب ویژگی و تشخیص نقاط پرت کمک می‌کنند. سایه‌های تیره‌تر با همبستگی قوی‌تر (ثبت یا منفی) مطابقت دارد [۱۲].

نمونه‌ای از این نمودار که در پروژه استفاده شد، در شکل ۲۲-۲ نمایش داده شده است.

	Poverty	DirectChol	TotChol
Poverty	1.000000	0.114370	0.078125
DirectChol	0.114370	1.000000	0.221467
TotChol	0.078125	0.221467	1.000000

شکل ۲۲-۲: نقشه حرارتی

همان‌طور که ملاحظه می‌شود، در نقاط سردتر، ارتباط بین دو متغیر ضعیف‌تر است.

Histogram نمودارهای هیستوگرام (میله‌ای) برای نمایش داده‌های دسته‌بندی یا عددی در قالب گرافیکی استفاده می‌شود. نمودارهای میله‌ای داده‌ها را از طریق میله‌های عمودی نشان می‌دهند، در حالی که هیستوگرام‌ها توزیع متغیرهای پیوسته را نشان می‌دهند. این تجسم‌ها برای درک توزیع هر ویژگی و شناسایی عدم تعادل داده‌ها یا نقاط پرت مفید هستند [۱۲]. نمونه‌ای از آن‌ها در شکل ۴-۸ و ۵-۴ در بخش پیاده‌سازی نمایش داده شده‌اند.

۶-۴-۲ اندازه‌گیری میزان خطای دقت

به دلیل پدید آمدن مشکلاتی از قبیل بیش‌برازش و کم‌برازش داده‌ها که در بخش‌های قبلی به آن اشاره شد مدل‌ها دچار اشتباهاتی در پیش‌بینی‌ها می‌شود. در این بخش به بیان انواع خطاهای و روابطی که برای سنجش آن‌ها به کار گرفته شده بیان می‌شود [۱۱].

ابتدا به معروفی ۴ تاپل مختلف پرداخته که بعدا از آن‌ها در محاسبه میزان خطای استفاده می‌شود [۱۱]:

- TP° : این تاپل موارد مثبت واقعی را علامت‌گذاری می‌کند. مثلا در این پروژه مواردی که احتمال ابتلا به دیابت در آن‌ها مثبت پیش‌بینی شده و در دادگان هم مثبت بوده در این دسته قرار می‌گیرد.
- $TN^{†}$: این تاپل موارد منفی واقعی را علامت‌گذاری می‌کند. مثلا در این پروژه مواردی که احتمال ابتلا به دیابت در آن‌ها منفی پیش‌بینی شده و در دادگان هم منفی بوده در این دسته قرار می‌گیرد.
- $FP^{‡}$: این تاپل موارد مثبت کاذب را علامت‌گذاری می‌کند. مثلا در این پروژه مواردی که احتمال ابتلا به دیابت در آن‌ها مثبت پیش‌بینی شده اما در دادگان هم منفی بوده در این دسته قرار می‌گیرد.
- $FN^{§}$: این تاپل موارد منفی کاذب را علامت‌گذاری می‌کند. مثلا در این پروژه مواردی که احتمال ابتلا به دیابت در آن‌ها منفی پیش‌بینی شده اما در دادگان هم مثبت بوده در این دسته قرار می‌گیرد.

• $Precision^{¶}$

در این معیار، با استفاده از موارد مطرح شده در بخش قبلی، نسبت نمونه‌های مثبت واقعی را از بین تمام نمونه‌های مثبت پیش‌بینی شده (هم مثبت واقعی و هم مثبت کاذب) اندازه‌گیری می‌کند. این نسبت مثبت‌های واقعی به مجموع موارد مثبت واقعی و مثبت کاذب است. [۱۱] به عنوان

^{۱۰}True positives

^{۱۱}True negatives

^{۱۲}False positives

^{۱۳}False negatives

^{۱۴}صحت

مثال در تشخیص دیابت، صحبت بالا نشان دهنده تعداد کم مثبت کاذب است؛ به این معنی که مدل به درستی افراد بدون دیابت را شناسایی می‌کند. اگر میزان صحبت مدل پایین باشد، افراد بسیاری بیمار تشخیص داده می‌شوند. [۱۱]

$$Precision = \frac{TP}{TP + FP} \quad (16-2)$$

: $(1 - Specificity)$ یا **FPR** •

رابطه ۱۷-۲ نشان دهنده نسبت افراد سالم شناسایی شده واقعی به کل افراد سالم (افراد اشتباهها بیمار تشخیص داده شده به علاوه افراد سالم شناسایی شده واقعی) است.

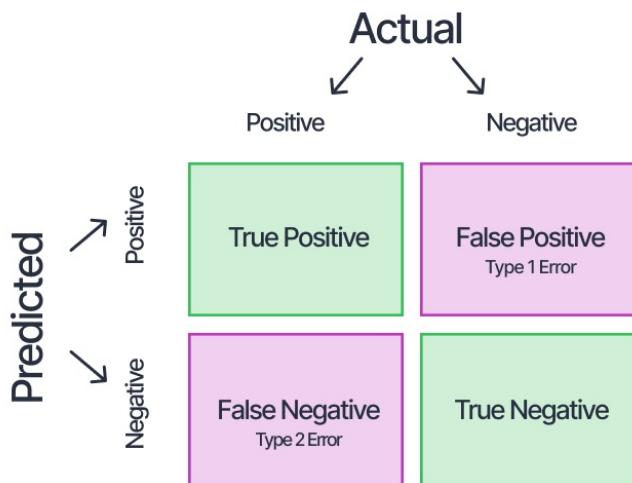
$$1 - Specificity = FPR = \frac{FP}{FP + TN} \quad (17-2)$$

: $Sensitivity$ یا **TPR** •

رابطه ۱۸-۲ نشان دهنده نسبت بیماران شناسایی شده واقعی به کل بیماران (افراد اشتباهها سالم تشخیص داده شده به علاوه افراد بیمار شناسایی شده واقعی) است.

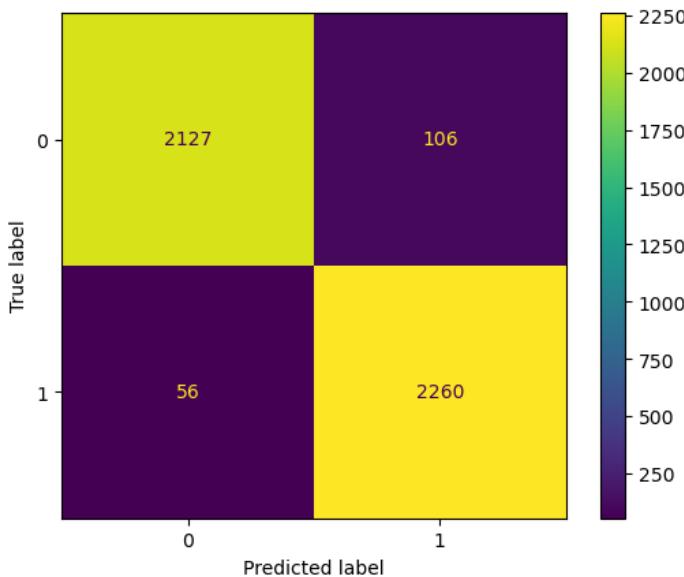
$$Sensitivity(Recall) = TPR = \frac{TP}{TP + FN} \quad (18-2)$$

تعدادی از این مقادیر را در ماتریس آشفتگی^{۲۵} نشان می‌دهند [۳۲] و ساختار آن در شکل ۲۳-۲ است. همچنین این ماتریس را برای مدل جنگل درختان تصادفی رسم کردم (شکل ۲۴-۲) :



شکل ۲۳-۲: ساختار ماتریس آشفتگی: پارامترهای مذکور در این بخش در این ماتریس قرار گرفته اند.

^{۲۵}Confusion



شکل ۲-۲: ساختار ماتریس آشفتگی برای الگوریتم جنگل درختان تصادفی

به طور کلی این ماتریس، خلاصه‌ای از درستی نتایج پیش‌بینی هایمان را نشان می‌دهد [۳۲]. حال میزان میزان دقت با رابطه ۱۹-۲ محاسبه می‌شود.

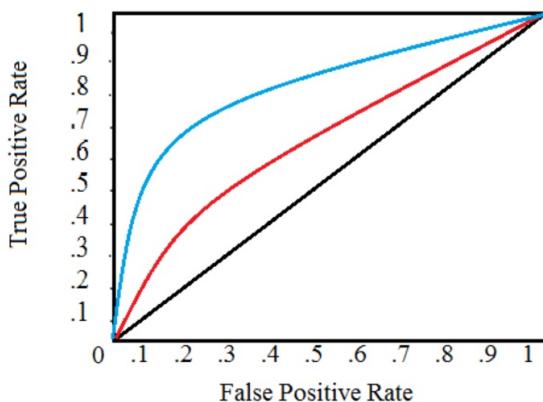
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (19-2)$$

Precision و Recall ، ACC ، AUC ۷-۴-۲

برای اینکه مقادیر پیش‌بینی‌های صحیح و غلط به صورت یک نمودار نمایش داده شود، از دو شاخصه TPR و FPR که در بخش قبلی معرفی شد، استفاده شده است [۱۱]. در اینجا مقادیر مختلف برای هر مرز (اشاره به تابع Sigmoid که در بخش‌های قبلی مطرح شد) پیش‌بینی را از ۰ تا ۱ اندازه‌گیری و در نمودار علامت زده [۱۱]. اگر مقدار TPR برای یک مرز برابر با ۱ باشد یعنی در آن مرز مدل خیلی خوب عمل کرده و اگر برابر ۵٪ باشد یعنی تصادفی عمل شده است [۱۱]. مثلاً در یک مرز در نظر گرفته اگر مقدار تابع Sigmoid از ۷٪ بیشتر بود، آن مورد را یک مورد مثبت شناسایی کن. در نمودار ROC^{۲۶} مولفه طول‌ها برابر با مقدار FPR است و مولفه عرض‌ها برابر TPR در آستانه‌های گوناگون است [۱۱]. سطح زیر این نمودار را با AUC^{۲۷} نمایش داده که بالا بودن این مساحت بیانگر این است که پیش‌بینی‌های بیشتری در این مدل درست بوده است [۱۱].

^{۲۶}Receiving Operating Characteristic

^{۲۷}Area Under the Curve



شکل ۲-۲۵: نمای کلی نمودار ROC

۵-۲ نتیجه‌گیری

در این فصل سعی شد تا ادبیات و مفاهیم مهمی که در انجام این پژوهه با آن‌ها درگیر بوده بیان شود از جمله بیان مفاهیم ابتدایی روش‌های داده کاوی شامل خوشه‌بندی و طبقه‌بندی (یادگیری بی‌نظرات و بانظرات) و همچنین، معرفی و بیان روش عملکرد الگوریتم‌های مورد استفاده از جمله رگرسیون لجستیک، درخت تصمیم، جنگل درختان تصادفی، AdaBoost و بیز ساده.

در دامه نیز، درمورد داده‌ها و عملیات آنها از جمله روش‌های پیش‌پردازش داده و انواع مصورسازی‌ها و درنهایت روش‌هایی برای سنجش و مقایسه کارایی الگوریتم‌هاییمان نیز مفاهیمی مطرح شد.

یکی از نکاتی که از بخش روش‌های سنجش کارایی الگوریتم‌ها می‌توان برداشت کرد، این است که AUC، اندازه‌گیری عملکرد کلی یک الگوریتم در تمام آستانه‌های طبقه‌بندی است و به نوعی می‌توان گفت AUC بالاتر، نشان دهنده عملکرد بهتر در تشخیص نمونه‌های مثبت و منفی است.

از سوی دیگر، ACC، اندازه‌گیری نسبت نمونه‌های طبقه‌بندی صحیح (یعنی TN و TP) از تعداد کل نمونه‌ها است. این معیار، نشان دهنده صحت کلی پیش‌بینی‌های طبقه‌بندی‌کننده است.

در فصل آینده در مورد پیش زمینه‌های انجام این تحقیق مطالبی بیان می‌شود.

فصل ۳

کارهای پیشین

۱-۳ مقدمه

پیرو مباحث قبلی، در انجام پژوهش‌های گوناگون بررسی و مقایسه روش‌های مختلف در روند توسعه و تحقیق مجدد، نکته بسیار مهمی است و می‌تواند اشکالات کارهای پیشین را برطرف نمود و در زمینه موارد به روز آن‌ها را به کار گرفت. همچنین ایده‌ها و نکاتی در هر مقاله ذکر شده است که می‌تواند رهنماوهای مفید برای کارهای آتی طلقی شوند. از جمله الگوریتم‌های مختلف و بیان روش‌های داده کاوی مربوط به آن.

۲-۳ مقاله ۱

مقاله [۷] در سال ۲۰۱۸ با استفاده از داده‌های بیماران هندی^۱ روند بررسی را بر روی سه الگوریتم درخت تصمیم، SVM^۲ و بیز ساده انجام داده و نتایج بیانگر این بوده که الگوریتم بیز ساده ۷۶٪ به عنوان موثر ترین الگوریتم در این بررسی درنظر گرفته شده است.

نکته قابل ملاحظه در این مقاله این است که از الگوریتم SVM هم برای پیش‌بینی استفاده شده است که معمولاً برای دادگان با مقادیر زیاد روش مناسبی است [۳۱].

به طور خلاصه می‌توان گفت الگوریتم SVM یا ماشین بردار پشتیبان، با تقسیم داده‌ها به دسته‌های مختلف بر اساس داده‌های آموزشی داده شده کار می‌کند که سعی می‌کند خطی را ترسیم کند که به عنوان ابرصفحه (یک فضای V شکل n بعدی زیرمجموعه ابعاد $n-1$ یا معادل آن که ۱بعده‌مند در V می‌باشد). شناخته می‌شود که به بهترین وجه دسته‌های مختلف داده‌ها را از هم جدا می‌کند. این کار را با یافتن ابرصفحه بهینه که فاصله بین نزدیکترین نقاط داده هر دسته را به حداقل می‌رساند، انجام می‌دهد [۳۱].

به عبارت ساده‌تر، داده‌ها را به صورت نقطه‌هایی روی یک تکه کاغذ در نظر بگیرید و هر نقطه به دسته خاصی تعلق دارد. الگوریتم SVM سعی می‌کند خطی را بر روی کاغذ بکشد که نقاط را به بهترین

^۱PIDD

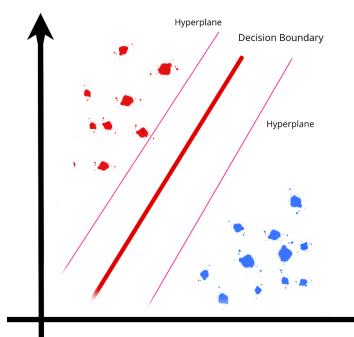
^۲Support Vector Machine

نحو از هم جدا کند، به طوری که نقاط متعلق به یک دسته در یک طرف خط و نقاط متعلق به دسته دیگر در سمت دیگر قرار گیرند.

با این حال، گاهی اوقات نقاط داده را نمی‌توان به طور کامل با یک خط از هم جدا کرد. در چنین مواردی، الگوریتم از تکنیکی به نام "ترفند هسته" برای تبدیل داده‌ها به فضایی با ابعاد بالاتر استفاده می‌کند، جایی که یافتن یک خط جداکننده آسان‌تر می‌شود [۳۱].

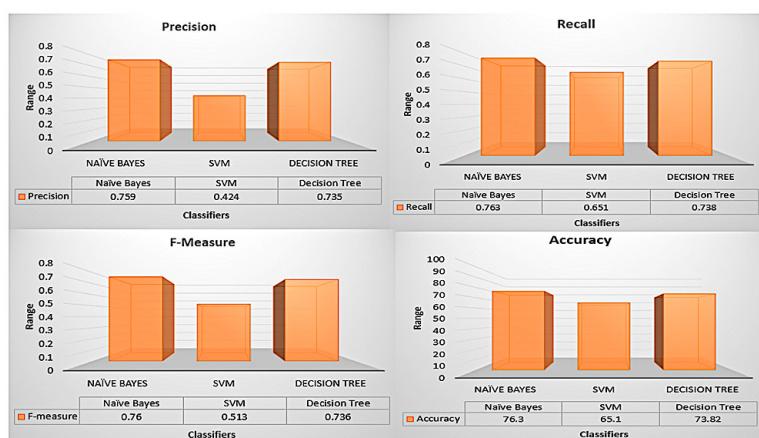
هنگامی که خط رسم شد، نقاط داده جدید و نادیده را می‌توان با تعیین اینکه در کدام سمت خط قرار می‌گیرند، به یک دسته اختصاص داد.

هدف SVM یافتن بهترین ابرصفحه است که حاشیه بین نقاط داده دسته‌های مختلف را به حداقل می‌رساند، که به پیش‌بینی دقیق داده‌های جدید کمک می‌کند [۳۱].

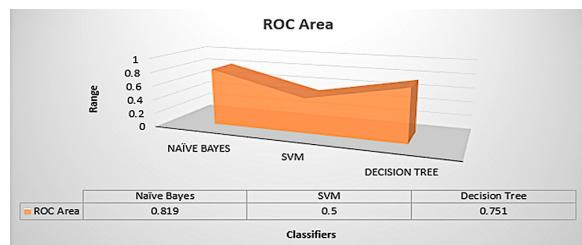


شکل ۱-۳: مدل SVM

خلاصه معیارهای دقت طبقه‌بندی در این مقاله به شرح زیر است:



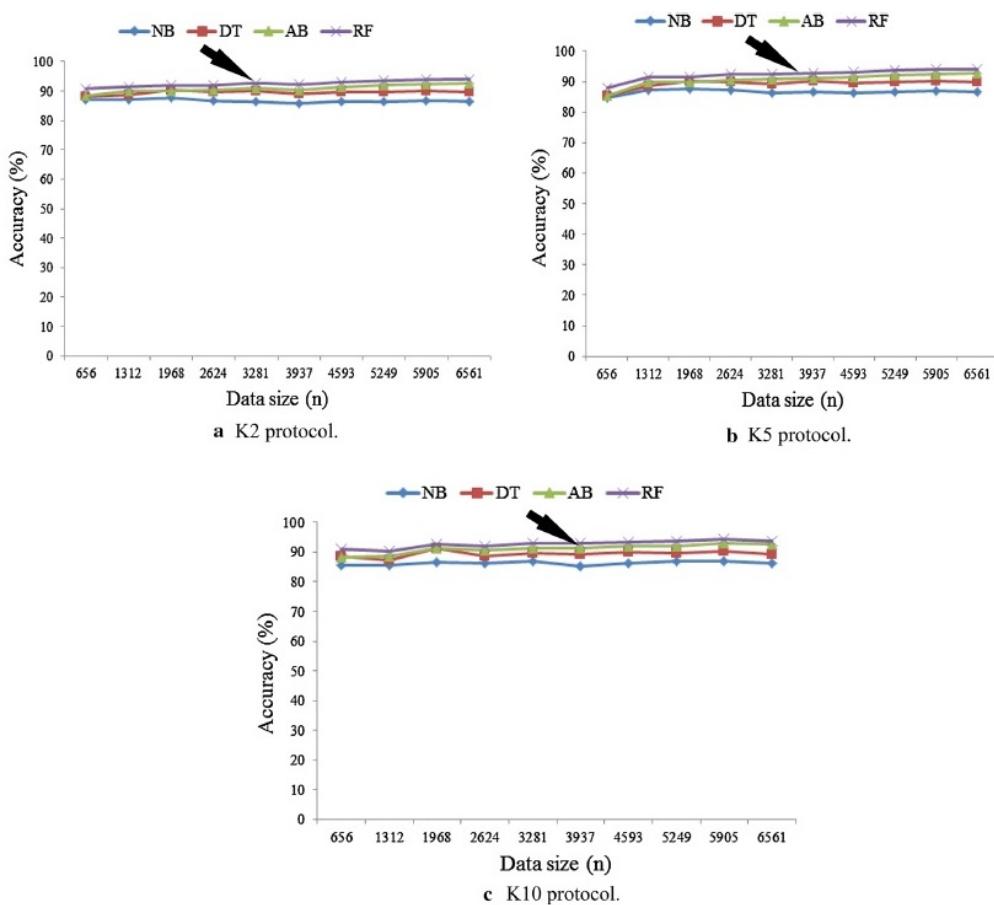
شکل ۲-۳: دقت‌های اندازه‌گیری در مقاله اول



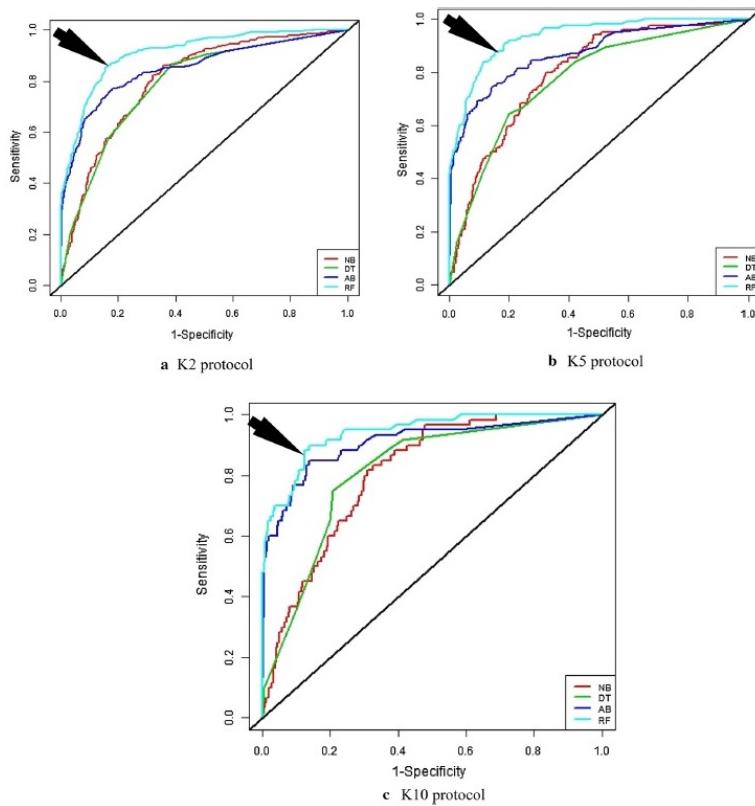
شکل ۳-۳: منحنی ROC در مقاله اول

۳-۳ مقاله ۲

در مقاله اشاره شده [۶] در سال ۲۰۲۰ روش‌های مختلفی برای پیش‌بینی ابتلا به دیابت انجام شده و جنگل درختان تصادفی در اعتباری سنجی متقابل K-Fold "مکرر" با مقدار $K=10$ بهترین نتیجه طبقه‌بندی را با دقت ۹۴٪ حاصل کرده است. در این مقاله روش‌های بیز ساده و AdaBoost هم مورد استفاده قرار گرفته‌اند که نتایج آن‌ها داری دقت مناسبی نبوده است.



شکل ۳-۴: دقت‌های اندازه‌گیری در مقاله دوم



شکل ۳-۵: منحنی ROC در مقاله دوم

ضمیما در این مقاله، از همان مجموعه دادگانی استفاده شده که در همین پایان نامه مورد استفاده قرار گرفته است.

۴-۳ نتیجه‌گیری

در این بخش، تلاش شد تا برخی از مقالاتی که این موضوع را مورد بحث قرار دادند بررسی و متوجه تفاوت‌ها در زمینه یافتن بهترین الگوریتم برای تشخیص ابتلا به دیابت، شده.

در نتیجه، حاصل مطالعه مقالات اساسی که در گذشته تالیف شده بودند، این شد که الگوریتم‌های مختلفی در روند توسعه استفاده پژوهه مورد بررسی واقع شوند که از جمله می‌توان به الگوریتم‌های بیز ساده، جنگل درختان تصادفی، الگوریتم درخت تصمیم و AdaBoost به همراه روش رگرسیون لجستیک برای انتخاب ویژگی اشاره کرد.

فصل ۴

روش‌ها و نتایج

۱-۴ مقدمه

در این بخش بررسی می‌شود که الگوریتم‌های گوناگونی که برای ساخت مدل‌های گوناگون استفاده شد تا چه حد می‌تواند قابل اعتماد واقع شود و میزان خطا در هرکدام چقدر است. چرا که در این پژوهه، کار پیش‌بینی انجام شده و داده‌های آزمون نشان داده که برخی از ردیف‌های به صورت اشتباه پیش‌بینی شده اند. برای انتخاب بهترین مدل روشی که انجام می‌شود این است که میزان خطاهای را با روش‌های گوناگون به دست آورده و سپس الگوریتمی که بیشترین صحت را ارائه داده به عنوان بهترین مدل برگزیده شود [۱۱].

همچنین برای بخش دیگری از این سنجش می‌توان نمودارهای گوناگونی را رسم کرد از جمله ROC که می‌تواند این میزان دقیقت را به صورت بصری نمایش دهد.

و در نهایت به بررسی روش طراحی یک مدل ساده برای سامانه آنلاین پیش‌بینی دیابت با ابزارهای اینترنت اشیاء و دو چارچوب NodeRed و Flask پرداخته شده است.

۲-۴ روش پیشنهادی

۱-۲-۴ پیاده‌سازی

پس از وارد کردن کتابخانه‌های مورد نیاز، به عملیات وارد کردن داده‌ها و پیش‌پردازش آن‌ها پرداخته شده است:

```
FirstDS = pd.read_csv('NHANES.csv')
FirstDS.info()
```

تعداد ۷۴ ستون داده وجود داشته که باید عملیات انتخاب ویژگی را برای آن انجام شود.

#	Column	Non-Null Count	Dtype	#	Column	Non-Null Count	Dtype		
0	Unnamed: 0	10000	non-null	int64	26	BPDiaAve	8551	non-null	float64
1	ID	10000	non-null	int64	27	BPSys1	8237	non-null	float64
2	SurveyYr	10000	non-null	object	28	BPDia1	8237	non-null	float64
3	Gender	10000	non-null	object	29	BPSys2	8353	non-null	float64
4	Age	10000	non-null	int64	30	BPDia2	8353	non-null	float64
5	AgeDecade	9667	non-null	object	31	BPSys3	8365	non-null	float64
6	AgeMonths	4962	non-null	float64	32	BPDia3	8365	non-null	float64
7	Race1	10000	non-null	object	33	Testosterone	4126	non-null	float64
8	Race3	5000	non-null	object	34	DirectChol	8474	non-null	float64
9	Education	7221	non-null	object	35	TotChol	8474	non-null	float64
10	MaritalStatus	7231	non-null	object	36	UrineVol1	9013	non-null	float64
11	HHIncome	9189	non-null	object	37	UrineFlow1	8397	non-null	float64
12	HHIncomeMid	9189	non-null	float64	38	UrineVol2	1478	non-null	float64
13	Poverty	9274	non-null	float64	39	UrineFlow2	1476	non-null	float64
14	HomeRooms	9931	non-null	float64	40	Diabetes	9858	non-null	object
15	HomeOwn	9937	non-null	object	41	DiabetesAge	629	non-null	float64
16	Work	7771	non-null	object	42	HealthGen	7539	non-null	object
17	Weight	9922	non-null	float64	43	DaysPhysHlthBad	7532	non-null	float64
18	Length	543	non-null	float64	44	DaysMentHlthBad	7534	non-null	float64
19	HeadCirc	88	non-null	float64	45	LittleInterest	6667	non-null	object
20	Height	9647	non-null	float64	46	Depressed	6673	non-null	object
21	BMI	9634	non-null	float64	47	nPregnancies	2604	non-null	float64
22	BMICatUnder20yrs	1274	non-null	object	48	nBabies	2416	non-null	float64
23	BMI_WHO	9603	non-null	object	49	Age1stBaby	1884	non-null	float64
24	Pulse	8563	non-null	float64	...				
25	BPSysAve	8551	non-null	float64	74	SexOrientation	4842	non-null	object

شکل ۱-۴: ردیف‌های دادگان

با توجه به اینکه در برخی ستون‌ها مقدار NaN وجود دارد، باید از راهکارهای مطرح شده در بخش های قبلی استفاده کرد. پس با استفاده از کد زیر، شروع به پیدا کردن انواع مقادیر نامشخص در ستون‌ها شد:

```
FirstDS.isnull().sum()
.
.
.
FirstDS = FirstDS.dropna(subset=['Diabetes'])
```

در قسمت دوم کد بالا، عملیات حذف داده‌های گم شده در ستون متغیر وابسته انجام شد. در واقع ردیف‌هایی که ستون آخرشان (یعنی Diabetes) دارای مقادیر نامشخص بودند حذف شد. زیرا تمام مدل سازی‌ها وابسته به آخرین ستون است و اگر این ستون مقادیر نادرستی داشته باشد، مدل‌های ساخته شده دقت پایینی خواهند داشت.

سپس برخی از ویژگی‌ها که موجب بیش‌برازش مدل می‌شوند، به صورت دستی حذف شد تا الگوریتم انتخاب ویژگی بتواند عملکرد بهتری داشته باشد. مثل SurveyYr سال سنجش، ID، شاخص BMI برای افراد زیر ۲۰ سال به علت وابستگی این ستون به ستون سن، ویژگی HeadCirc که تماماً دارای مقادیر نامشخص بود و در مدل اشکال ایجاد می‌کرد.

```
FirstDS = FirstDS.drop('HeadCirc', axis=1)
FirstDS = FirstDS.drop('BMICatUnder20yrs', axis=1)
FirstDS = FirstDS.drop('SurveyYr', axis=1)
```

```
FirstDS = FirstDS.drop('ID', axis=1)
```

پس از شمارش تعداد ردیف‌هایی با مقادیر گم شده، عملیات مربوط به جایگزینی آن‌ها را با روش زیر انجام شد. یعنی مقادیر ستون‌هایی که مقدار پیوسته داشتند را با میانگین ستون جایگزین و در مقادیر گسته، از مُد (نمونه‌ای با بیشترین تکرار) استفاده شد.

```

We also use their mode for discrete values.

# Impute NaN values with mode
Categorical_cols = FirstDS.select_dtypes(include=['object']).columns
for col in Categorical_cols:
    mode = FirstDS[col].mode()[0]
    FirstDS[col] = FirstDS[col].fillna(mode)

Now we replace other columns with undefined values with the average values of those columns.

# Impute NaN values with mean
continuous_cols = FirstDS.select_dtypes(include=np.number).columns
FirstDS[continuous_cols] = FirstDS[continuous_cols].fillna(FirstDS[continuous_cols].mean())

We check again whether we have a column with an undefined value or not:

nan_mask = FirstDS.isna()
columns_with_nan = nan_mask.any()
columns_names = columns_with_nan[columns_with_nan].index.tolist()
#New_dataset=FirstDS

print(columns_names)

```

شکل ۲-۴: جایگزینی مقادیر نامشخص با متوسط‌های آماری

سپس متغیر‌های وابسته و مستقل را مشخص و مدل RFE فراخوانی شد. همچنین عملیات مربوط به ایجاد متغیر‌های رقمی را نیز در آن انجام شد.

```

from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

X_val = FirstDS.drop('Diabetes', axis=1)
Y_val = FirstDS['Diabetes']

from sklearn.preprocessing import LabelEncoder

model = LogisticRegression()
rfe = RFE(model, n_features_to_select=18)

original_cols1 = X_val.select_dtypes(include=['object']).columns
XX_val=X_val
YY_val=Y_val

XX_val = pd.get_dummies(XX_val, columns=original_cols1, drop_first=True)

le = LabelEncoder()
categorical_features2 = XX_val.select_dtypes(include=[np.object]).columns.tolist()
XX_val[categorical_features2] = XX_val[categorical_features2].apply(lambda x: le.fit_transform(x))
XX_val=X_val
YY_val = le.fit_transform(YY_val)

XX_val.head()

Unamed:0  Gender  Age  AgeDecade  AgeMonths  Race1  Race3  Education  MaritalStatus  HHIncome  ...  AgeFirstMarij  RegularMarij  AgeRegMarij  HardDrugs  SexEver  SexAge  SexNumPartialife  SameSex  SexOrientation  PregnantNow
0         0      1     34        3     409.0     4     5       3      2     5   ...          0    17.691069      1     1    16.000000     8.000000      0         1         0
1         1      1     34        3     409.0     4     5       3      2     5   ...          0    17.691069      1     1    16.000000     8.000000      0         1         0
2         2      1     34        3     409.0     4     5       3      2     5   ...          0    17.691069      1     1    16.000000     8.000000      0         1         0

```

شکل ۳-۴: ایجاد مدل RFE

درنهایت به علت محدودیت منابع سخت‌افزاری، ۱۶ متغیر به عنوان حداقل متغیر‌های لازم، از مدل درخواست شد.

```
# Use RFE to select the top 16 features
```

```
rfe = RFE(model, n_features_to_select=16)
rfe.fit(XX_val, Y_val)

# Print the selected features
#print(rfe.support_)
count = rfe.support_.tolist().count(True)
print(count)
list1=rfe.support_.tolist()
list2=XX_val.columns.tolist()
result = [list2[i] for i in range(len(list1)) if list1[i]]

print(result)
```

متغیر‌های زیر برگزیده شدند:

```
['AgeDecade', 'Race1', 'Poverty', 'Work', 'DirectChol', 'TotChol', 'HealthGen',
'Depressed', 'nPregnancies', 'SleepTrouble', 'SmokeNow', 'Smoke100n', 'HardDrugs',
'SameSex', 'SexOrientation', 'PregnantNow']
```

به ترتیب معادل: رده‌بندی سنی، نژاد، شاخص فقر، وضعیت شغلی، کلسترول مستقیم، کلسترول کل، وضعیت سلامت عمومی، داشتن افسردگی، دفعات بارداری، مشکل خواب، سیگار کشیدن مداوم، کشیدن ۱۰۰ نخ سیگار در عمر، مواد مخدر، داشتن رابطه جنسی، گرایش جنسی، باردار بودن هستند [۱۹].

برای اینکه مدل دقیق‌تر باشد و بیش‌برازش به وجود نیاید، می‌توان برخی متغیرها که استقلال کمتری دارند را حذف کرد [۱۲]، به عنوان مثال طبق مطالب مطرح شده در بخش‌های قبلی، مشخص است باردار بودن عاملی موثر در داشتن دیابت است، پس دفعات آن کمتر اهمیت دارد تا اینکه شخص چندبار باردار بوده، لذا این ستون حذف شد. یا سیگار کشیدن مکرر، عامل موثرتری نسبت به کشیدن ۱۰۰ نخ سیگار در طول عمر است و کسی که مکررا سیگار می‌کشد، قطعاً این عامل برایش مثبت ارزیابی می‌شود.

گزینه‌های هر ویژگی را در ادامه می‌بینیم:

```
a=non_numerical_columns = dataset.select_dtypes(exclude=['number']).columns

for column in a:
    unique_values = dataset[column].unique() # Get unique values of the column
    num_unique_values = len(unique_values) # Count the number of unique values

    print(f"Column '{column}':")
    print(f"Number of unique values: {num_unique_values}")
    print("Unique values:", unique_values)
    print()
```

```

Column 'AgeDecade':
Number of unique values: 8
Unique values: ['30-39' '0-9' '40-49' '60-69' '50-59' '10-19' '20-29' '70+']

Column 'Race1':
Number of unique values: 5
Unique values: ['White' 'Other' 'Mexican' 'Black' 'Hispanic']

Column 'Work':
Number of unique values: 3
Unique values: ['NotWorking' 'Working' 'Looking']

Column 'HealthGen':
Number of unique values: 5
Unique values: ['Good' 'Vgood' 'Fair' 'Excellent' 'Poor']

Column 'Depressed':
Number of unique values: 3
Unique values: ['Several' 'None' 'Most']

Column 'SleepTrouble':
Number of unique values: 2
Unique values: ['Yes' 'No']

Column 'SmokeNow':
Number of unique values: 2
Unique values: ['No' 'Yes']

Column 'HardDrugs':
Number of unique values: 2
Unique values: ['Yes' 'No']

Column 'SameSex':
Number of unique values: 2
Unique values: ['No' 'Yes']

Column 'SexOrientation':
Number of unique values: 3
Unique values: ['Heterosexual' 'Bisexual' 'Homosexual']

Column 'PregnantNow':
Number of unique values: 3
Unique values: ['No' 'Unknown' 'Yes']

```

شکل ۴-۴: ویژگی‌های انتخاب شده

سپس عملیات مربوط ایجاد مصورسازی داده‌ها صورت گرفت:

```

YesDia = New_dataset['Diabetes'].values == 'Yes'
NoDia = New_dataset['Diabetes'].values == 'No'
YesDia=New_dataset[YesDia]
NoDia=New_dataset[NoDia]

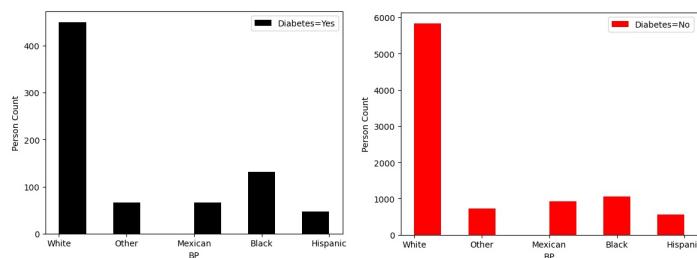
Race1 = YesDia['Race1'].tolist()
Race0 = NoDia['Race1'].tolist()

```

```
plt.hist([Race1], color=[  
    'Black'], label=['Diabetes=Yes'])  
plt.xlabel('BP')  
plt.ylabel('Person Count')  
plt.legend()  
plt.show()
```

```
plt.hist([Race0], color=[  
    'Red'], label=['Diabetes=No'])  
plt.xlabel('BP')  
plt.ylabel('Person Count')  
plt.legend()  
plt.show()
```

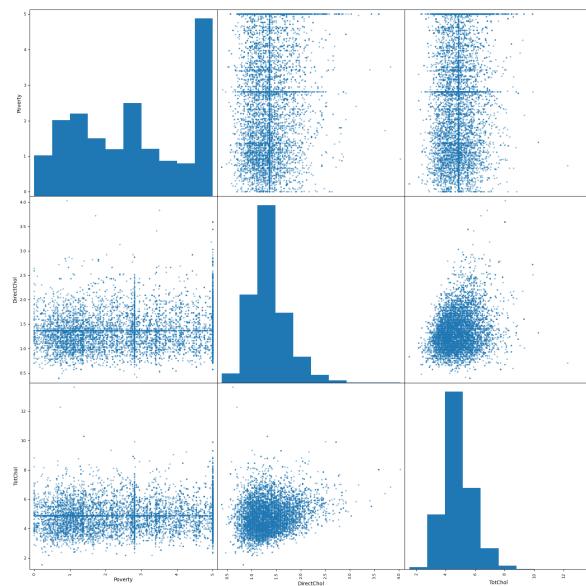
نتیجهٔ عملیات فوق، نمودارهای زیر می‌باشد (در این نمودار، تاثیر نژاد در ابتلا به دیابت مشخص است):



شکل ۴-۵: نمودار مقایسهٔ ابتلا به دیابت در نژادهای مختلف

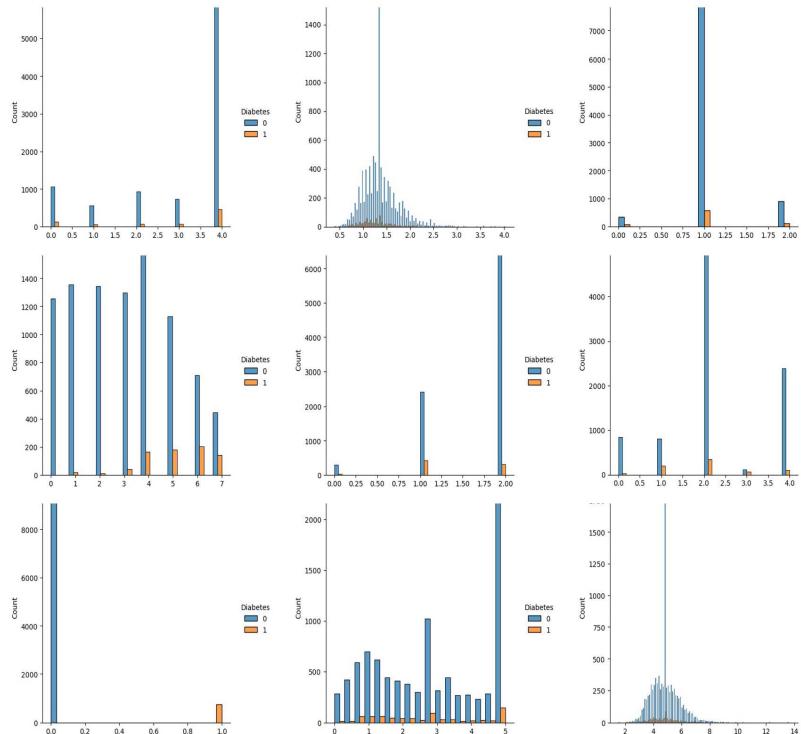
سپس با کد زیر، ماتریس scatter برای نمایش همبستگی میان داده‌ها رسم کرده:

```
from pandas.plotting import scatter_matrix  
scatter_matrix(dataset , figsize=(20, 20))  
plt.show()
```

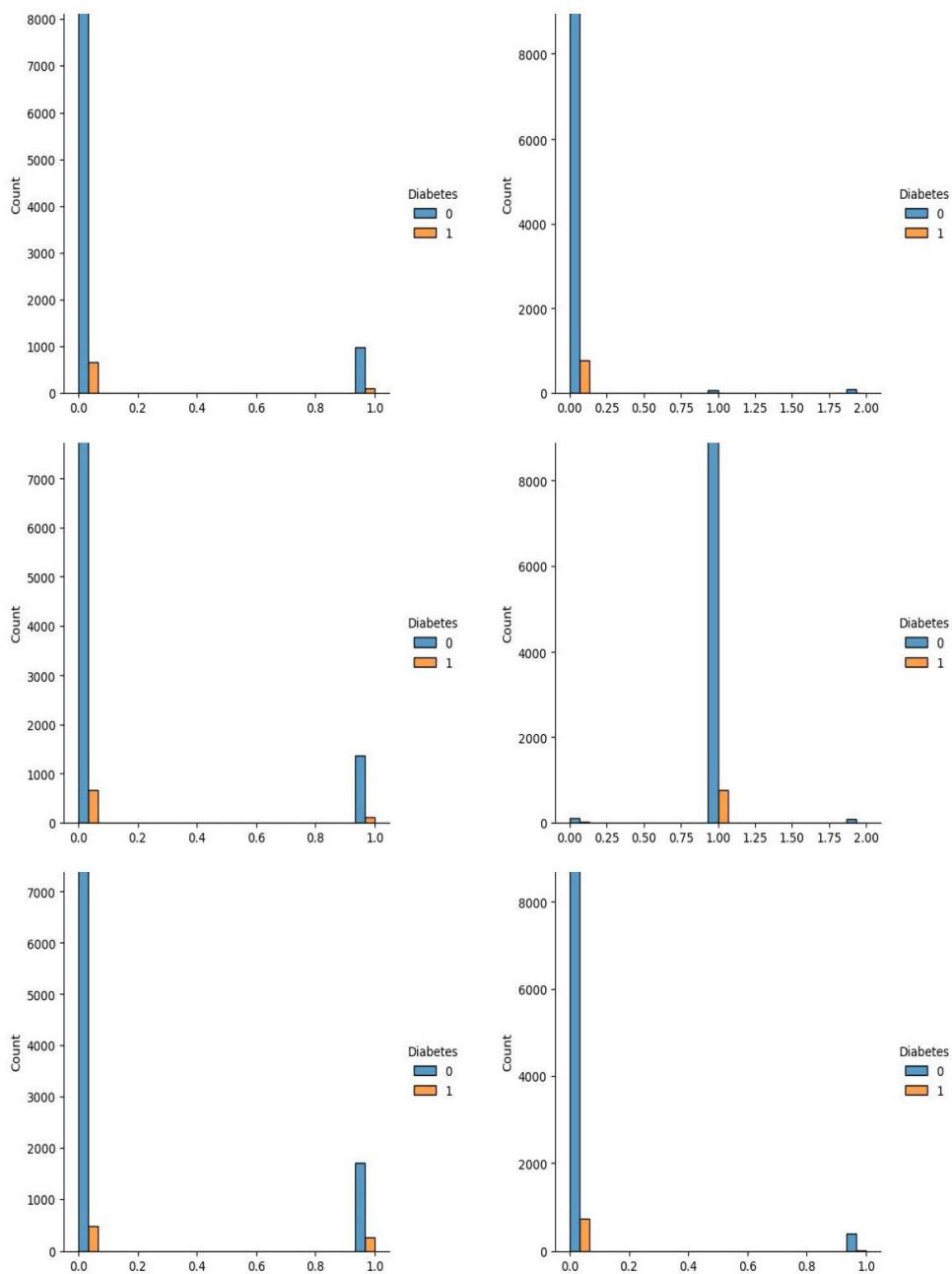


شکل ۶-۴: ماتریس Scatter

در نمودارهایی که در ادامه وجود دارد، تاثیر پارامترهای گوناگون در ابتلای افراد به دیابت مورد بررسی قرار گرفته است:



شکل ۷-۴: نمودار عوامل موثر در ابتلا دیابت (۱)



شکل ۴-۸: نمودار عوامل موثر در ابتلا دیابت (۲)

همچنین از کد زیر برای رسم نمودارهای فوق استفاده شد.

```
for i in Dataframe1.columns:
    sns.displot(Dataframe1, x=i,multiple="dodge",hue="Diabetes")

for i in dataset_new.columns:
    sns.displot(dataset, x=i,multiple="dodge",hue="Diabetes")
```

نمودار نقشه حرارتی نیز با این کد رسم شد:

```
#Converting New_dataset into a panda dataframe
Dataframe1 = New_dataset
corr = Dataframe1.corr()
```

#Displaying dataframe of correlation values

```
corr.style.background_gradient(cmap = 'coolwarm')
```

پارامترهایی که با هم ارتباط ندارند، با رنگ‌های سردتری مشخص اند.



شکل ۴-۹: نقشه حرارتی

به نظر می‌رسد که داده‌ها توزیع مناسبی ندارد. تعداد افرادی که دیابت ندارند خیلی بیشتر از افرادی است که دیابت دارند. پس باید با روش‌های نمونه‌برداری مجدد مثل SMOTE داده‌های مصنوعی ایجاد شود (Over sampling) تا مدل دقیق‌تری داشته باشد. دادگان جدید را مجدداً نرم‌افزاری و سپس دوباره نمودارها را ایجاد کرده:

```
# Selecting X & Y
X_val = New_dataset.drop('Diabetes', axis=1)
Y1 = New_dataset['Diabetes']

from imblearn.over_sampling import SMOTE
smote = SMOTE(sampling_strategy='auto')
X_resampled, y_resampled = smote.fit_resample(X_val, Y1)

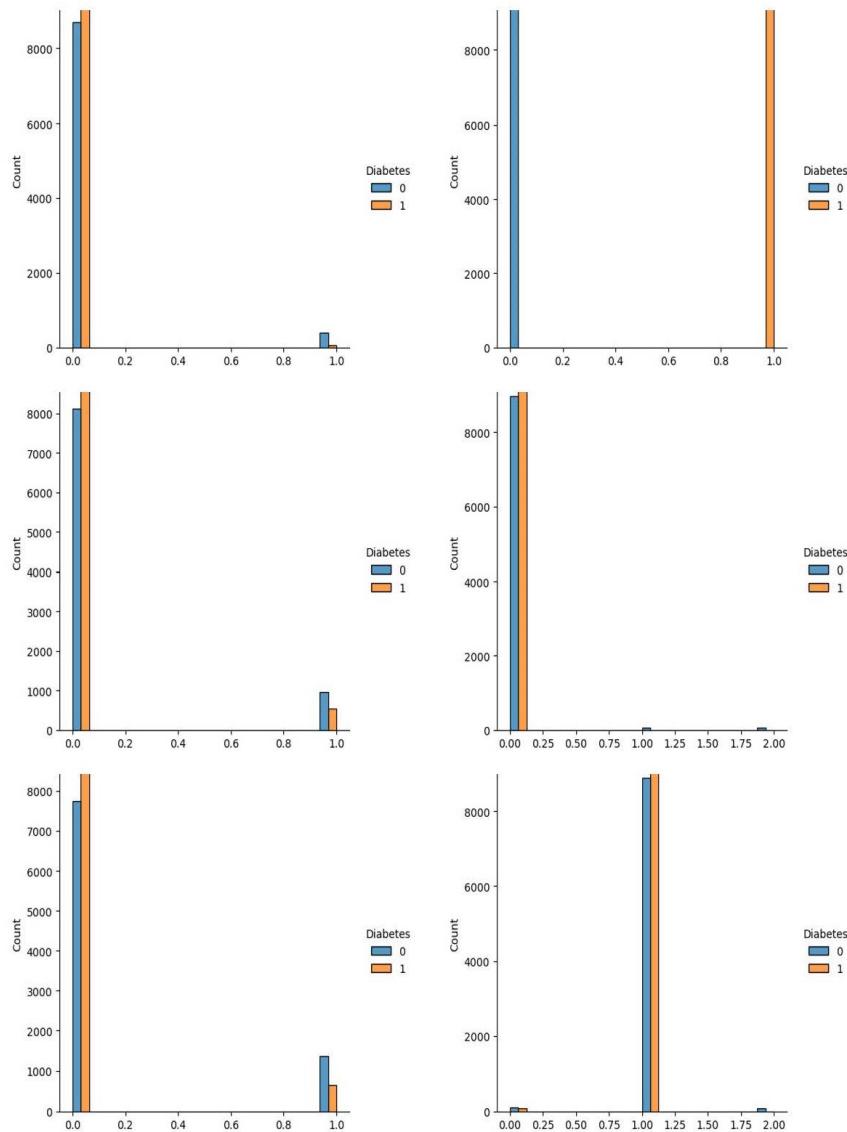
# Concatenate resampled features
#data_resampled = pd.concat([X_resampled, y_resampled], axis=1)
Dataframe9=copy.deepcopy(pd.concat([X_resampled, y_resampled], axis=1))

X_val = Dataframe9.iloc[:, :-1].values
Y1 = Dataframe9.iloc[:, -1].values
```

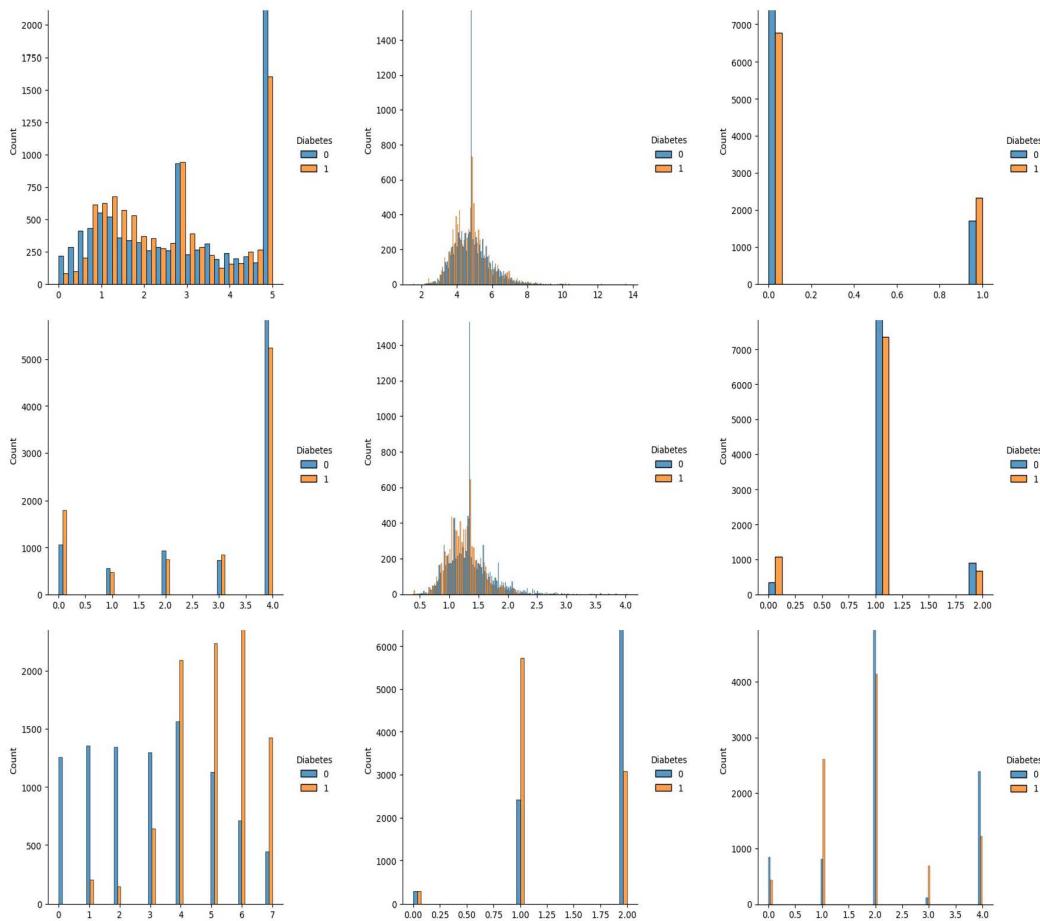
###

```
columns_to_include = ['Poverty', 'DirectChol', 'TotChol', 'Diabetes']
subset_data = Dataframe9[columns_to_include]
sns.pairplot(data = subset_data, hue = 'Diabetes')
plt.show()
```

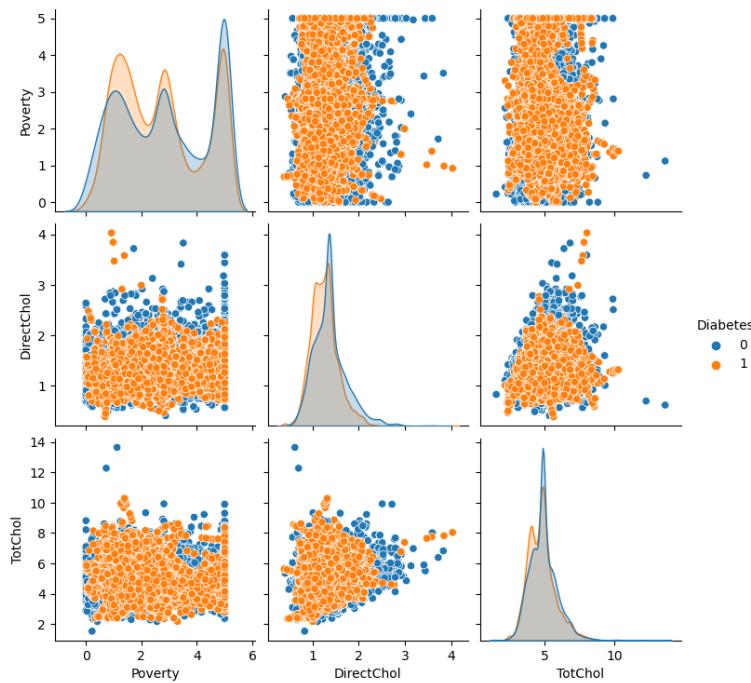
نتایجی که حاصل شد قابل قبول بود:



شکل ۴-۱۰: نمودار نمونهبرداری شده عوامل موثر در ابتلا دیابت (۱)



شکل ۱۱-۴: نمودار نمونهبرداری شده عوامل موثر در ابتلا دیابت (۲)



شکل ۱۲-۴: نمودار Pairplot بعد از نمونه‌برداری مجدد

حال پس از بخش مصوّرسازی به سراغ آماده سازی داده‌ها برای ایجاد مدل‌ها پرداخته شد. سپس ستون Y و X هایمان مشخص تا داده‌ها را به بخش‌های آموزشی و سنجش تقسیم شود. چون در ساخت مدل‌ها، باید داده‌ها را به چهار دسته‌های X آموزشی، های X سنجش، های Y آموزشی و های Y سنجش، تقسیم شوند [۱۲]. مقیاسی که برای داده‌های سنجش در نظر گرفته شده ۲۵٪ است و سپس آن‌ها را مقیاس‌بندی کرده. این بخش صرفاً برای رسم ماتریس پراکنش انجام شد.

```
from sklearn.model_selection import train_test_split
XTrain, XTest, YTrain, YTest = train_test_split(X_val, Y1, test_size = 0.25, random_state = 0)
```

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

XTrain = sc.fit_transform(X_val)
XTest = sc.transform(X_val)
```

در این بخش که مدل اصلی با توجه با روش اعتبارسنجی متقابل بحث شده انجام می‌شود، باید در هر Fold عملیات مقیاس‌بندی نیز جداگانه انجام شود. با توجه به اینکه برای هر مدل باید الگوریتم K-Fold تکرار شونده برای $K=2,5,10$ به تعداد ۲۰ مرتبه انجام شود و مقادیر ACC ، AUC و Precision و Recall به دست آیند، دوتابع ایجاد شد:

ساخت لیست مربوط به مقادیر ACC :

```
#Importing required libraries
```

```

from sklearn.model_selection import RepeatedKFold
from sklearn.metrics import accuracy_score
from numpy import mean
from numpy import std
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler

def Kfold_modulation(input_model):

    #Implementing cross validation
    k_list = [2,5,10]
    avg_acc_list = []

    for k in k_list:
        acc_scores = []
        kf = RepeatedKFold(n_splits=k, n_repeats=20, random_state=None)
        #kf = KFold(n_splits=k,shuffle=False, random_state=None)

        for train_indexes, test_indexes in kf.split(X_val):
            X_train, X_test = X_val[train_indexes], X_val[test_indexes]
            y_train, y_test = Y1[train_indexes], Y1[test_indexes]

            # Apply feature scaling
            scaler = StandardScaler()
            X_train = scaler.fit_transform(X_train)
            X_test = sc.transform(X_test)

            model = input_model
            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)

            acc_score = accuracy_score(y_test, y_pred)
            acc_scores.append(acc_score)

        avg_acc = np.mean(acc_scores)
        avg_acc_list.append(avg_acc)
    return avg_acc_list

```

ساخت لیست مربوط به مقادیر : AUC

```

def Kfold_modulation2(input_model):

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X_val) # Scaling

    k_list = [2, 5, 10]
    num_repeats = 20
    auc_list = []

    for k in k_list:
        kf = RepeatedKFold(n_splits=k, n_repeats=num_repeats)

        mean_score = cross_val_score(input_model, X_scaled, Y1, scoring="roc_auc", cv=kf).mean()

        auc_list.append(mean_score)

    return auc_list

```

ساخت لیست مربوط به مقادیر Recall :

```

from sklearn.metrics import recall_score

def Recall_f(input_model):
    k_list = [2, 5, 10]
    avg_recall_list = []

    for k in k_list:
        recall_scores = []
        kf = RepeatedKFold(n_splits=k, n_repeats=20, random_state=None)
        #kf = KFold(n_splits=k, shuffle=False, random_state=None)

        for train_indexes, test_indexes in kf.split(X_val):
            X_train, X_test = X_val[train_indexes], X_val[test_indexes]
            y_train, y_test = Y1[train_indexes], Y1[test_indexes]

            # Apply feature scaling
            scaler = StandardScaler()
            X_train = scaler.fit_transform(X_train)
            X_test = scaler.transform(X_test)

```

```

model = input_model
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

recall = recall_score(y_test, y_pred)
recall_scores.append(recall)

avg_recall = np.mean(recall_scores)
avg_recall_list.append(avg_recall)

return avg_recall_list

```

:Precision ساخت لیست مربوط به مقادیر

```

from sklearn.metrics import precision_score

def Prec_f(input_model):
    #Implementing cross validation
    k_list = [2, 5, 10]
    avg_precision_list = []

    for k in k_list:
        precision_scores = []
        kf = RepeatedKFold(n_splits=k, n_repeats=20, random_state=None)
        #kf = KFold(n_splits=k, shuffle=False, random_state=None)

        for train_indexes, test_indexes in kf.split(X_val):
            X_train, X_test = X_val[train_indexes], X_val[test_indexes]
            y_train, y_test = Y1[train_indexes], Y1[test_indexes]

            # Apply feature scaling
            scaler = StandardScaler()
            X_train = scaler.fit_transform(X_train)
            X_test = sc.transform(X_test)

            model = input_model
            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)

```

```

precision = precision_score(y_test, y_pred)
precision_scores.append(precision)

avg_precision = np.mean(precision_scores)
avg_precision_list.append(avg_precision)

return avg_precision_list

```

سپس هر مدل را ساخته و میزان ACC ، AUC ، Precision و Recall را برای هر کدام در یک لیست وارد کرده:

```

from sklearn.ensemble import RandomForestClassifier
ranfor = RandomForestClassifier()
ranfor_acc=Kfold_modulation(ranfor)
ranfor_auc=Kfold_modulation2(ranfor)
ranfor_rec=Recall_f(ranfor)
ranfor_pre=Prec_f(ranfor)

#The ranfor_pred value is later used to draw the confusion matrix.
XTrain, XTest, YTrain, YTest = train_test_split(X_val,Y1, test_size = 0.25, random_state = 0)
ranfor2 = RandomForestClassifier()
ranfor2.fit(XTrain, YTrain)
ranfor2_pred=ranfor2.predict(XTest)

```

```

# Decision Trees Algorithm
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

```

```

DecTree = DecisionTreeClassifier()
DecTree_acc=Kfold_modulation(DecTree)
DecTree_auc=Kfold_modulation2(DecTree)
Dec_pre=Prec_f(DecTree)
Dec_rec=Recall_f(DecTree)

```

```

# AdaBoost Algorithm
from sklearn.ensemble import AdaBoostClassifier
AdaBoost = AdaBoostClassifier()
AdaBoost_acc=Kfold_modulation(AdaBoost)
AdaBoost_auc=Kfold_modulation2(AdaBoost)
Ada_rec=Recall_f(AdaBoost)
Ada_pre=Prec_f(AdaBoost)

```

```
# Naive Bayes Algorithm
```

```
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB()
nb_acc=Kfold_modulation(nb)
nb_auc=Kfold_modulation2(nb)
nb_rec=Recall_f(nb)
nb_pre=Prec_f(nb)
```

سپس مقادیر Precision ، AUC ، ACC ، Recall را که در لیست‌های جداگانه قرار دارند، را باهم ترکیب و یک دیتافریم حاوی تمامی مقادیر Precision ، AUC ، ACC ، Recall را ایجاد کرده تا با استفاده از آن، نمودارهای Precision ، AUC ، ACC و ROC رسم شوند.

```
acc_list0=[ranfor_acc,DecTree_acc,AdaBoost_acc,nb_acc]
```

```
acc_list=[]
```

```
for i in acc_list0 :
```

```
my_formatted_list = [ '%.4f' % elem for elem in i ]
```

```
list=[]
```

```
#for i in my_formatted_list:
```

```
#list.append(int(float(i)*100))
```

```
acc_list.append(my_formatted_list)
```

```
auc_list0=[ranfor_auc,DecTree_auc,AdaBoost_auc,nb_auc]
```

```
auc_list=[]
```

```
for i in auc_list0 :
```

```
my_formatted_list = [ '%.4f' % elem for elem in i ]
```

```
list=[]
```

```
auc_list.append(my_formatted_list)
```

```
print(auc_list)
```

```
#recall
```

```
rec_list0=[ranfor_rec,Dec_rec ,Ada_rec,nb_rec]
```

```
rec_list=[]
```

```
for i in rec_list0 :
```

```
my_formatted_list = [ '%.4f' % elem for elem in i ]
```

```
list=[]
```

```
rec_list.append(my_formatted_list)
```

```
#Precision
```

```
pre_list0=[ranfor_pre,Dec_pre,Ada_pre,nb_pre]
```

```
pre_list=[]
```

```
for i in pre_list0 :
```

```

my_formatted_list = [ '%.4f' % elem for elem in i ]
list=[]
pre_list.append(my_formatted_list)

bar=pd.DataFrame([acc_list[0],acc_list[1],acc_list[2],acc_list[3]])
bar['algo'] = ['RF','DT','AB','NB']
#bar.insert()=['ranfor_acc','DecTree_acc','AdaBoost_acc','nb_acc']
bar.columns=['K2' , 'K5' , 'K10','Algorithm']

bar['K2']=bar['K2'].astype('float64')*100
bar['K5']=bar['K5'].astype('float64')*100
bar['K10']=bar['K10'].astype('float64')*100
print('ACC','\n',bar,'\n')

barauc=pd.DataFrame([auc_list[0],auc_list[1],auc_list[2],auc_list[3]])
barauc['algo'] = ['RF','DT','AB','NB']
#barauc.insert()=['ranfor_auc','DecTree_auc','AdaBoost_auc','nb_auc']
barauc.columns=['K2' , 'K5' , 'K10','Algorithm']
barauc['K2']=barauc['K2'].astype('float64')*100
barauc['K5']=barauc['K5'].astype('float64')*100
barauc['K10']=barauc['K10'].astype('float64')*100
print('\n','AUC','\n',barauc)

barpre=pd.DataFrame([pre_list[0],pre_list[1],pre_list[2],pre_list[3]])
barpre['algo'] = ['RF','DT','AB','NB']
barpre.columns=['K2' , 'K5' , 'K10','Algorithm']
barpre['K2']=barpre['K2'].astype('float64')*100
barpre['K5']=barpre['K5'].astype('float64')*100
barpre['K10']=barpre['K10'].astype('float64')*100
print('\n','PRE','\n',barpre)

barrec=pd.DataFrame([rec_list[0],rec_list[1],rec_list[2],rec_list[3]])
barrec['algo'] = ['RF','DT','AB','NB']
barrec.columns=['K2' , 'K5' , 'K10','Algorithm']
barrec['K2']=barrec['K2'].astype('float64')*100
barrec['K5']=barrec['K5'].astype('float64')*100
barrec['K10']=barrec['K10'].astype('float64')*100
print('\n','REC','\n',barrec)

```

سپس با کد زیر، نمودار مقایسه ACC و AUC را برای الگوریتم‌های مختلف در Fold‌های ۵، ۲ و ۱۰ رسم می‌شود:

```
# create data
```

```
# view data

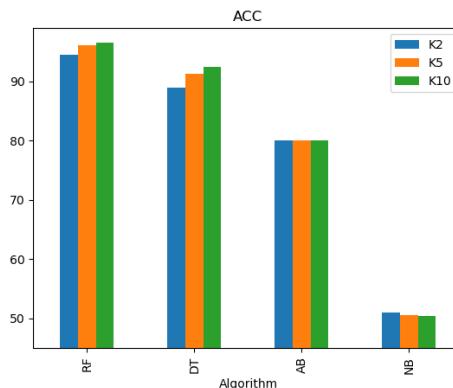
bar['K2'] = bar['K2'].astype('float64')
bar['K5'] = bar['K5'].astype('float64')
bar['K10'] = bar['K10'].astype('float64')

# plot grouped bar chart
bar.plot(x='Algorithm',
          kind='bar',
          stacked=False,
          ylim=[45, 99],
          title='ACC')
```

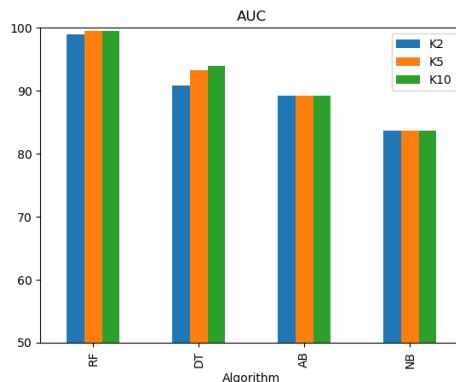
```
barauc['K2'] = barauc['K2'].astype('float64')
barauc['K5'] = barauc['K5'].astype('float64')
barauc['K10'] = barauc['K10'].astype('float64')

# plot grouped bar chart
barauc.plot(x='Algorithm',
            kind='bar',
            stacked=False,
            ylim=[50, 100],
            title='AUC')
```

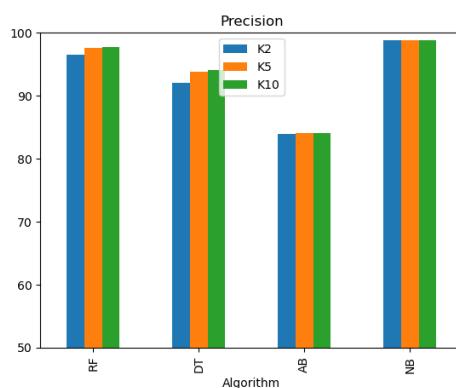
نمودار ها:



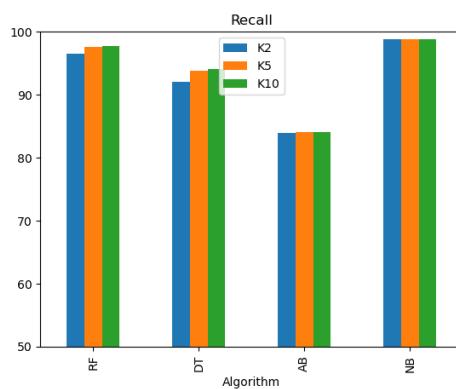
شکل ۱۳-۴: نمودار ACC



شکل ۱۴-۴: نمودار AUC

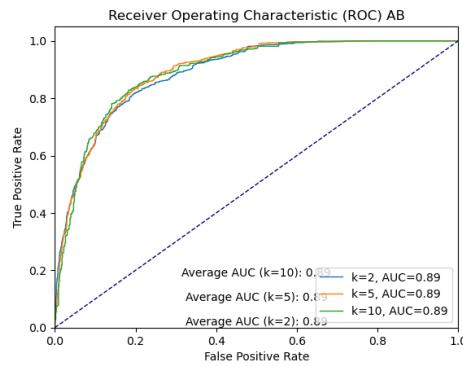


شکل ۱۵-۴: نمودار Precision

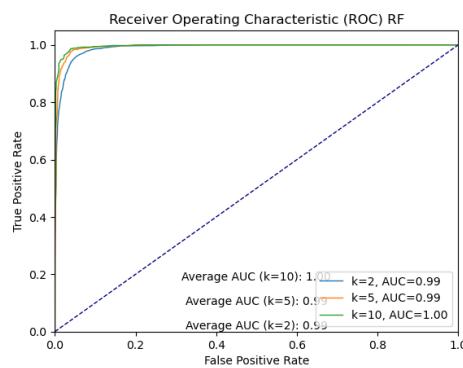


شکل ۱۶-۴: نمودار Recall

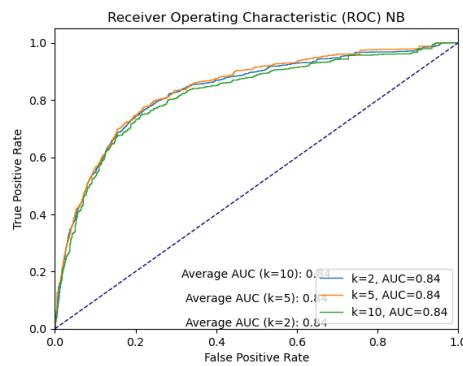
حال برای الگوریتم‌های مختلف، نمودار ROC Fold را در مختصات مختلف رسم کرده و میانگین آن نیز مشاهده می‌شود:



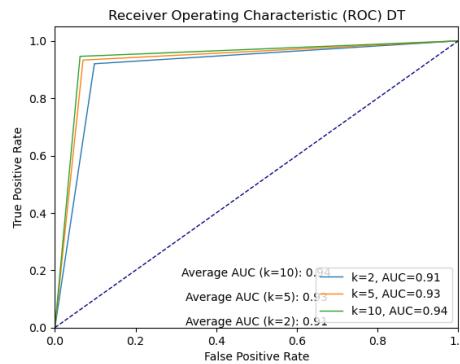
شکل ۱۷-۴: نمودار ROC برای الگوریتم AdaBoost



شکل ۱۸-۴: نمودار ROC برای الگوریتم جنگل درختان تصادفی



شکل ۱۹-۴: نمودار ROC برای الگوریتم Naïve Bayes



شکل ۴-۲۰: نمودار ROC برای الگوریتم درخت تصمیم

از کد زیر برای رسم نمودارهای فوق استفاده کردم:

```

from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import StratifiedKFold, RepeatedKFold

# Define the repeated k-fold cross-validation parameters
n_folds = [2, 5, 10]
n_repeats = 20

# Define the classifier
clf = nb
sc = StandardScaler()

# Initialize the mean AUC list for each k
mean_auc_list = []

# Perform repeated k-fold cross-validation for each k
for n_fold in n_folds:
    auc_list = []
    rskf = RepeatedKFold(n_splits=n_fold, n_repeats=n_repeats)
    #rskf = KFold(n_splits=n_fold, shuffle=False, random_state=None)

    # Perform cross-validation
    for train_index, test_index in rskf.split(X_val, Y1):
        X_train, X_test = X_val[train_index], X_val[test_index]
        Y_train, Y_test = Y1[train_index], Y1[test_index]

        X_train = sc.fit_transform(X_train)
        X_test = sc.transform(X_test)

        # Fit the classifier
        clf.fit(X_train, Y_train)

        # Predict probabilities
        probas = clf.predict_proba(X_test)[:, 1]

        # Compute ROC curve and area under the curve
        fpr, tpr, thresholds = roc_curve(Y_test, probas)
        auc_value = auc(fpr, tpr)
        auc_list.append(auc_value)

    mean_auc = np.mean(auc_list)
    mean_auc_list.append(mean_auc)

```

```

clf.fit(X_train, Y_train)
Y_prob = clf.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(Y_test, Y_prob)
auc_val = auc(fpr, tpr)
auc_list.append(auc_val)

# Calculate the mean AUC and update mean AUC list
mean_auc = np.mean(auc_list)
mean_auc_list.append(mean_auc)

# Plot ROC curve
plt.plot(fpr, tpr, lw=1, label=f'k={n_fold}, AUC={mean_auc:.2f}')

# Set plot settings
plt.plot([0, 1], [0, 1], color='navy', lw=1, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) NB')
plt.legend(loc="lower right")

# Display average AUC values
plt.text(0.5, 0.02, f'Average AUC (k=2): {mean_auc_list[0]:.2f}', transform=plt.gca().transAxes,
ha='center', va='center')
plt.text(0.5, 0.1, f'Average AUC (k=5): {mean_auc_list[1]:.2f}', transform=plt.gca().transAxes,
ha='center', va='center')
plt.text(0.5, 0.18, f'Average AUC (k=10): {mean_auc_list[2]:.2f}', transform=plt.gca().transAxes,
ha='center', va='center')
plt.show()

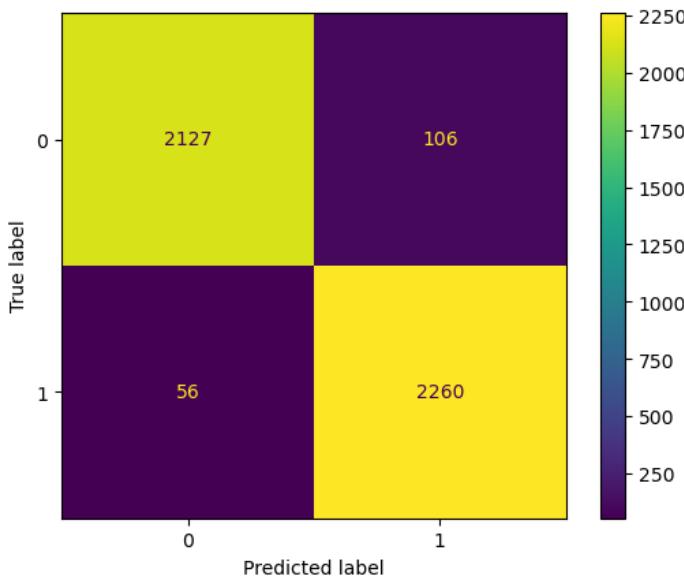
```

و در نهایت ماتریس آشتبگی را با استفاده از کد زیر برای الگوریتم جنگل تصادفی رسم شد:

```

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
YP_ranfor = ranfor2.predict(XTest)
cm = confusion_matrix(YTest, YP_ranfor, labels=None)
print(cm)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()

```



شکل ۴-۲۱: ساختار ماتریس آشفتگی برای الگوریتم جنگل درختان تصادفی

Arduino ۲-۲-۴

آردوینو یک بستر منبع باز است که برای ساخت پروژه‌های الکترونیکی استفاده می‌شود و شامل یک برد مدار فیزیکی قابل برنامه ریزی و یک نرم افزار به نام محیط توسعه یکپارچه (IDE) است که روی کامپیوتر اجرا می‌شود و برای نوشتند و آپلود کدهای کامپیوتري روی برد استفاده می‌شود. پلتفرم آردوینو به این دلیل محبوب است که برای بارگذاری کد جدید روی برد نیازی به برنامه نویس جداگانه ندارد و از نسخه ساده شده C++ برای برنامه نویسی استفاده می‌کند. این پلتفرم یک فرم فاکتور استاندارد را فراهم می‌کند که استفاده از عملکردهای میکروکنترلر را آسان‌تر می‌کند [۳۷].

بوردهای آردوینو در اتصالات برق، پین‌ها برای اتصال سیم‌ها، دکمه تنظیم مجدد، انواع نشانگرهای LED برای ارتباطات گوناگون از جمله سریال، IC اصلی و تنظیم‌کننده ولتاژ در مدل‌های مختلف تفاوت دارند [۳۷]. آردوینو می‌تواند با قطعات و سنسورهای مختلف تعامل داشته باشد و از طریق پروتکل MQTT می‌تواند با دستگاه‌های مختلف مبتنی بر اینترنت اشیاء تعامل داشته باشد [۳۸].

در مورد راه اندازی Arduino IDE دو نکته ضروری است:

۱. حتماً باید پکیج مربوط به بورد مد نظر را بروی آن نصب کنید. (برای مدلی که استفاده می‌کردم (WeMos D1 R1) آن را با لینک ^۱ در file> performances> additional board manager url وارد کردم.)

۲. برای استفاده از MQTT باید بسته esp8266client را از طریق Arduino IDE نصب کرد.

: MQTT

این پروتکل، یک پروتکل پیام رسانی است که برای ارتباط دستگاه‌ها در اینترنت اشیاء (IoT) استفاده می‌شود. سبک، کارآمد و مقیاس‌پذیر است و برای انتقال داده‌ها از طریق شبکه‌هایی با محدودیت منابع،

^۱http://arduino.esp8266.com/stable/package_esp8266com_index.json

ایده‌آل است. MQTT از پیام رسانی بین دستگاه‌ها و ابر^۲ پشتیبانی می‌کند و قابلیت اطمینان و ویژگی‌های امنیتی را ارائه می‌دهد [۳۵].

به جای ارتباط مستقیم بین کاربران و سرورها، MQTT از یک واسطه پیام برای مدیریت ارتباط بین فرستنده‌گان و دریافت کننده‌گان استفاده می‌کند. سرور، پیام‌های فرستنده‌گان را فیلتر کرده و بین گیرنده‌گان توزیع می‌کند. این کار فرستنده و گیرنده پیام‌ها را از نظر مکان و زمان جدا می‌کند [۳۵].

اجزای MQTT شامل کلاینت‌های MQTT است که دستگاه‌هایی هستند که با استفاده از MQTT ارتباط برقرار می‌کنند و یک سرور MQTT که پیام‌ها را بین کاربران روبدل می‌کند. کاربران می‌توانند پیام‌هایی را با موضوعات و داده‌ها منتشر کنند و همچنین می‌توانند برای دریافت پیام‌ها مشترک موضوعات خاصی شوند [۳۵].

همچنین (MQTT over WebSockets (WSS) یک پیاده‌سازی از MQTT است که اجازه می‌دهد داده‌ها مستقیماً در یک مرورگر وب دریافت شوند. ارتباط MQTT ایمن است و می‌توان با استفاده از پروتکل‌های SSL، گواهی‌ها و رمزهای عبور محافظت کرد [۳۵].

برای راه اندازی یک سرور MQTT به صورت محلی، می‌توان از ابزاری به نام mosquitto^۳ که در سیستم عامل‌های مختلف در دسترس است، استفاده کرد.

در مورد راه اندازی mosquitto نکته‌ای که قابل توجه است این است که بایستی پورت 1883 در تنظیمات این برنامه باز باشد و امکان اتصال به صورت ناشناس نیز ممکن باشد تا بتوان به راحتی و بدون دردرس به آن متصل شد برای این منظور مراحل زیر بایستی طی شوند.

۱. ابتدا آن را متوقف کرده:

در لینوکس: sudo systemctl stop mosquitto

در ویندوز: sc stop mosquitto

۲. سپس به فایل mosquitto.conf دسترسی پیدا کرده و آن را با یک ویرایشگر باز کرده. این فایل در سیستم عامل‌های مختلف مسیر متفاوت دارد:

در لینوکس: /etc/mosquitto

در ویندوز: C:\Program Files\mosquitto\

۳. پس از بازگشایی این فایل، دو خط زیر را به آن افزوده می‌شود:

```
allow_anonymous true
listener 1883
```

۴. دستور mosquitto.c را در خط فرمان وارد کرده.

۵. سپس برنامه را مجدد راه اندازی کرده:

در لینوکس: sudo systemctl start mosquitto

در ویندوز: sc start mosquitto

^۲Cloud

^۳mosquitto.org

در کدی که برای بورد آردوینو نوشته شد، ابتدا تنظیمات اتصال به WiFi را انجام و تنظیمات مربوط به اتصال MQTT را با وارد کردن IP سرور و تعیین نام و گذر واژه برای آن انجام داده و پس از مشخص کردن پین‌های مربوط به LED، تعیین کرده در هر ۱ ثانیه یک عدد دو رقمی تولید و از طریق پروتکل مورد بحث به سرور ارسال شود که توسط NodeRed دریافت شده و زوج و فرد بودن آن به اصطلاح مشخص می‌کند که مثلاً فرد مورد نظر دیابت دارد یا ندارد. این یک مثال است که مدل ساده‌ای را ساخته شده که با فرض اتصال ابزارهای سنجش‌گر قدمخون به صورت آنلاین که مبتنی بر اینترنت اشیاء باشند، می‌توان آخرین اطلاعات فرد بیمار را دریافت کرد.

کد بورد آردوینو:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Network credentials
const char* ssid = "WiFi6";
const char* password = "*****";
const char* mqtt_username = "mqtt";
const char* mqtt_password = "mqtt";

// MQTT broker address
const char* mqtt_server = "192.168.191.3";

// Initializing the WiFi and MQTT clients
WiFiClient DiabetesPredictor20231;
PubSubClient client(DiabetesPredictor20231);

void setup() {
    // Serial communication for debugging purposes
    pinMode(LED_BUILTIN, OUTPUT);
    digitalWrite(LED_BUILTIN, HIGH);
    Serial.begin(9600);

    // Connecting to Wi-Fi
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
}
```

```

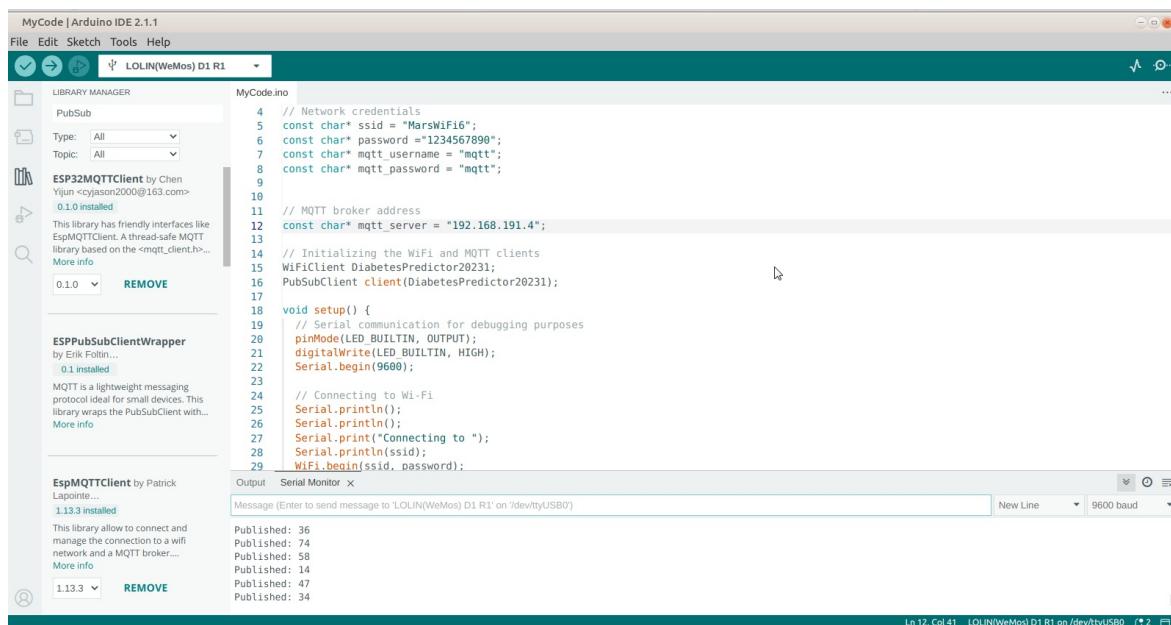
Serial.println("");
Serial.println("WiFi connected");
Serial.println(WiFi.localIP());
Serial.println('\n');

// Connecting to MQTT broker
client.setServer(mqtt_server, 1883);
while (!client.connected()) {
    Serial.print("Connecting to MQTT broker...");
    if (client.connect("DiabetesPredictor2023", mqtt_username, mqtt_password)) {
        Serial.println("connected");
        //client.subscribe("GetData2023", 2);
    } else {
        Serial.print("failed with state ");
        Serial.print(client.state());
        Serial.print("\n");
        delay(2000);
    }
}
void loop() {
    // Generating a two-digit random number and convert it to a string
    int random_num = random(10, 100);
    String payload = String(random_num);

    // Publishing the payload to the MQTT topic
    client.publish("GetData2023", payload.c_str());
    digitalWrite(LED_BUILTIN, LOW);
    Serial.print("Published: ");
    Serial.println(payload);
    delay(400);
    digitalWrite(LED_BUILTIN, HIGH);

    // Printing the payload and wait for 10 seconds before publishing again
    delay(10000);}
```

ادامه توسعه این پروژه در بخش پیانی مطرح می‌شود.



شکل ۴-۲۲: محیط توسعه Arduino در حال دریافت اعداد تولید شده توسط بورد

۳-۲-۴ ابزار Node Red

ابزار Node-RED یک چارچوب نرم افزاری مبتنی بر جریان^۴ همه کاره و کاربرپسند^۵ است که توسط IBM توسعه یافته است که توسعه اینترنت اشیاء را ساده می‌کند. این چارچوب مبتنی بر Node.js است و امکان ادغام یکپارچه با دستگاه‌های مختلف، API‌ها و انواع خدمات آنلاین را فراهم می‌کند. با رابط بصری جذاب آن، کاربران می‌توانند به سادگی با کشیدن و رها کردن، گره‌هایی را روی یک صفحه ایجاد و به هم متصل کنند تا رفتار مورد نظر برنامه خود را تعریف کنند [۳۶]. این ابزار با ادغام یکپارچه با پلتفرم‌هایی مانند Raspberry Pi، Arduino، AWS و Node.js، ضمن سادگی و تطبیق پذیری عالی، آن را به گزینه‌ای محبوب برای پروژه‌های اینترنت اشیاء و برنامه‌های اتوماسیون خانگی تبدیل کرده است [۳۶].

همچنین نصب داشبورد در این بستر، با وارد کردن دستور `npm install node-red-dashboard` در خط فرمان انجام می‌شود.

در این ابزار، یک داشبورد (شکل ۴-۲۳) طراحی شد که دو بخش دارد.

^۴Flow-Based

^۵User friendly

^۶Node

Prediction of diabetes

Prediction		Update	
AgeDecade	10-19	AgeDecade	Select option
Race	White	Race	Select option
Poverty	2	Poverty	
Work	NotWorking	Work	Select option
DirectChol	23	DirectChol	
TotChol	36	TotChol	
HealthGen	Fair	HealthGen	Select option
Depressed	Several	Depressed	Select option
SleepTrouble	No	SleepTrouble	Select option
SmokeNow	No	SmokeNow	Select option
HardDrugs	No	HardDrugs	Select option
SameSex	No	SameSex	Select option
SexOrientation	Heterosexual	SexOrientation	Select option
PregnantNow	No	PregnantNow	Select option
SEND		Diabetes Select option	
Result	You may develop diabetes!	GET THE LATEST PATIENT STATUS	
		SEND	

شکل ۴-۲۳: صفحه داشبورد پیش‌بینی کاربر و بهروزرسانی دادگان

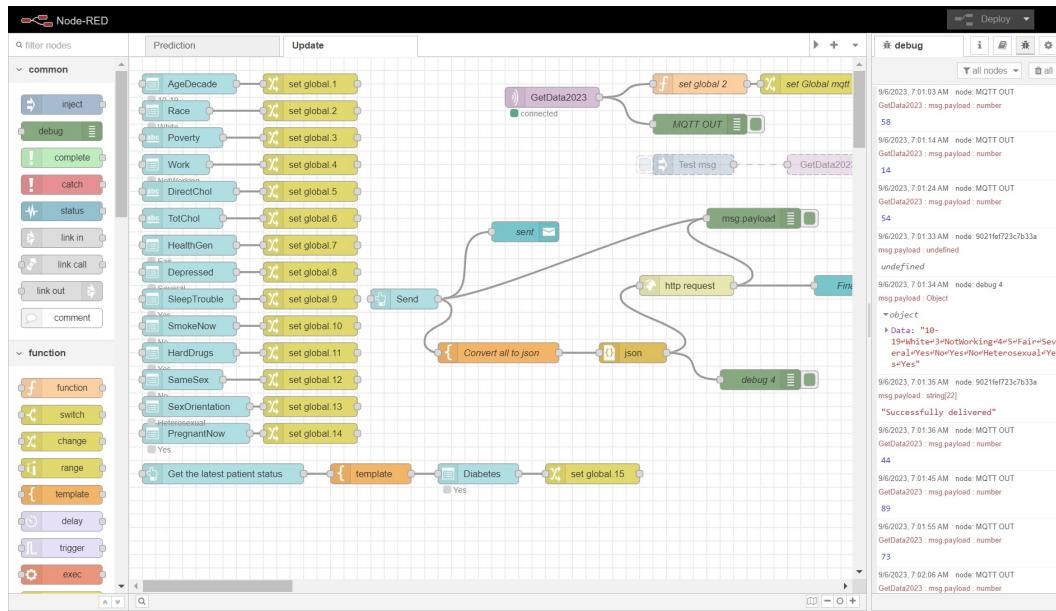
• بهروزرسانی دادگان:

در این بخش داده‌های جدید از یک سامانه دیگر دریافت شده و از طریق چارچوب Flask در فایل csv دادگان اضافه می‌شوند. بدیهیست که گزینه GET THE LAST PATIENT STATUS آخرين وضعیت بیمار را درمورد داشتن یا نداشتن دیابت را از طریق اینترنت اشیاء دریافت می‌کند. بدیهیست داشتن دادگان به روز، به صحت مدل و افزایش دقت آن، کمک شایانی می‌کند.

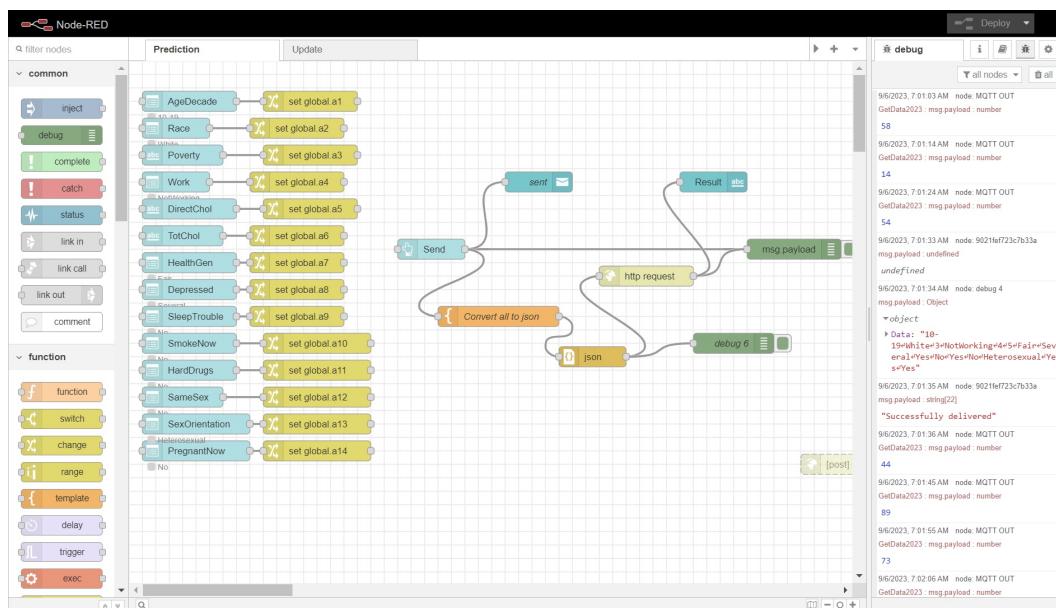
• پیش‌بینی: در این بخش، افراد با وارد کردن ویژگی‌های سلامتی خود، می‌توانند بینند که آیا احتمال ابتلا به دیابت در آن‌ها وجود دارد یا خیر.

در هر کدام از بخش‌های مطرح شده، گره‌های مربوط به دریافت مشخصات را ایجاد و مقادیر از طریق مرورگر دریافت و در آن ثبت می‌شود. سپس در ساختار داده json ثبت شده و در نهایت از طریق یک request برای http ارسال می‌شود.

بعد از دریافت صحیح اطلاعات، یک اعلان درمورد دریافت صحیح داده‌ها نیز دریافت می‌شود.



شکل ۲۴-۴: طراحی بخش بهروزرسانی در NodeRed



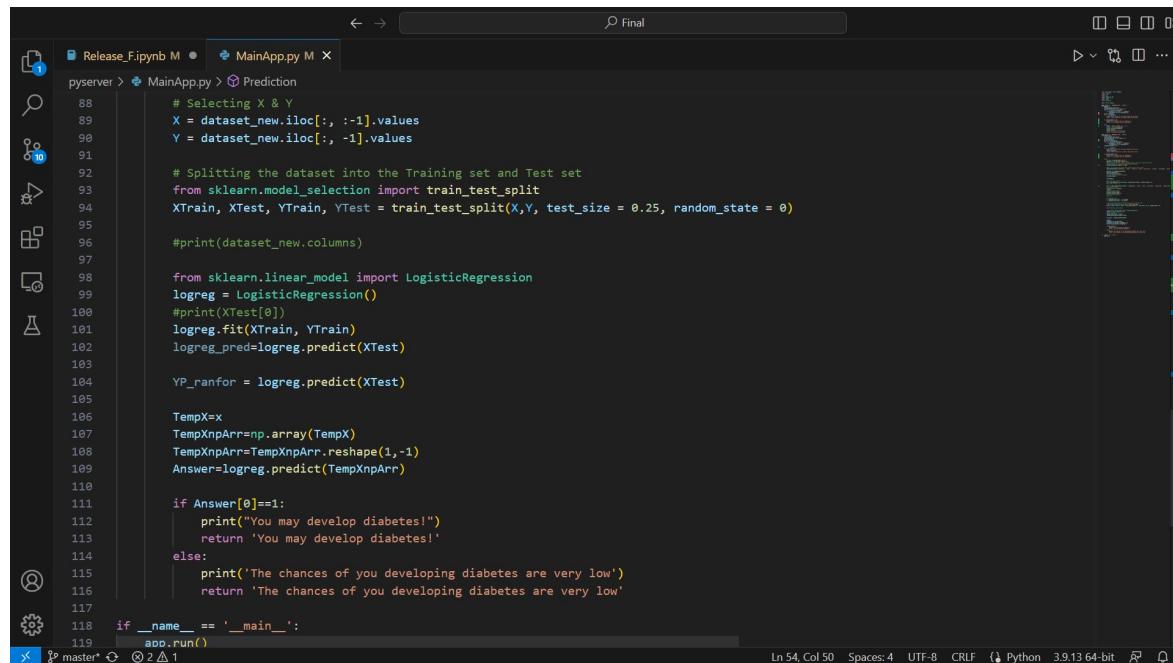
شکل ۲۵-۴: طراحی بخش پیش‌بینی در NodeRed

Flask ۴-۲-۴

می‌توان گفت Flask یک چارچوب توسعه وب سبک و کاربرپسند است که به زبان پایتون نوشته شده است. این سادگی، انعطاف‌پذیری و ساختار مدولار را برای ساخت برنامه‌های کاربردی مقیاس پذیر ارائه می‌دهد. مستندات گسترده و جامعه پر جنب و جوش این چارچوب، یادگیری و استفاده از آن را آسان می‌کند. علاوه بر این، Flask به خوبی با کتابخانه‌ها و ابزارهای دیگر ادغام می‌شود و از پایگاه داده‌های مختلف و فناوری‌های وب پشتیبانی و پردازش انواع درخواست‌ها را آسان می‌کند. این یک

انتخاب ارجح برای توسعه دهنده‌گان با تمام سطوح تجربه است و به آنها اجازه می‌دهد تا به طور کارآمد برنامه‌های کاربردی وب قوی بسازند [۳۹].

لذا در این پژوهه از این چارچوب برای استفاده از توابع و کتابخانه‌های پایتون جهت پردازش درخواست‌های کاربران استفاده کردم. وقتی از طریق داشبورد در NodeRed یک درخواست مبتنی بر به روز رسانی داده‌های دادگان یا درخواست از طرف یک فرد درمورد پیش‌بینی احتمال ابتلای آن فرد با مشخصاتش در چارچوب Flask دریافت شود، در تابع مربوطه این درخواست مورد پردازش قرار می‌گیرد و نتیجه به سادگی برای کاربر ارسال می‌شود.



```

88     # Selecting X & Y
89     X = dataset_new.iloc[:, :-1].values
90     Y = dataset_new.iloc[:, -1].values
91
92     # Splitting the dataset into the Training set and Test set
93     from sklearn.model_selection import train_test_split
94     Xtrain, XTest, Ytrain, YTest = train_test_split(X,Y, test_size = 0.25, random_state = 0)
95
96     #print(dataset_new.columns)
97
98     from sklearn.linear_model import LogisticRegression
99     logreg = LogisticRegression()
100    #print(XTest[0])
101    logreg.fit(Xtrain, Ytrain)
102    logreg_pred=logreg.predict(XTest)
103
104    YP_ranfor = logreg.predict(XTest)
105
106    TempX=x
107    TempXnpArr=np.array(TempX)
108    TempXnpArr=TempXnpArr.reshape(1,-1)
109    Answer=logreg.predict(TempXnpArr)
110
111    if Answer[0]==1:
112        print("You may develop diabetes!")
113        return 'You may develop diabetes!'
114    else:
115        print('The chances of you developing diabetes are very low')
116        return 'The chances of you developing diabetes are very low'
117
118    if __name__ == '__main__':
119        app.run()

```

شکل ۴-۲۶: بخشی از کد Flask

از این کد برای ارسال نتایج پیش‌بینی و بهره بردن از کتابخانه‌های پایتون در روند دریافت اطلاعات انجام گرفت.

۳-۴ نتیجه‌گیری

در این بخش، تلاش شد تا ضمن بررسی روش انتخاب بهترین الگوریتم پیش‌بینی دیابت با پایتون و سنجش میزان صحت و دقت روش‌ها، یک روش پیاده‌سازی مختصر نیز برای استفاده همگان از این سیستم معرفی شود. از روش SMOTE برای نمونه‌برداری و از روش RFE با مدل رگرسیون لجستیک برای انتخاب ویژگی استفاده شد.

این سنجش دقت از طریق مقایسه ROC AUC و نمودارهای آن‌ها انجام شد که در فصل قبلی نیز مقالاتی که ارائه شده بودند نیز بر این اساس مقایسه شدند که نتیجه کلی آن روش جنگل درختان تصادفی بود که به عنوان مدل برتر برگزیده شد.

نتایج Precision و Recall برای الگوریتم جنگل درختان تصادفی، گرچه اندکی کمتر از بیز تصادفی بود،

اما با توجه به عملکرد خوب این الگوریتم در کل، می‌توان نگاه ویژه‌ای به این الگوریتم برای انتخاب به عنوان مدل برتر داشت.

بدترین عملکرد مربوط به الگوریتم‌های بیز ساده و درخت تصمیم بود که مطابق نمودارهای مشاهده شده عملکرد پایین‌تری نسبت به سایرین داشتند.

و در نهایت به بررسی روش طراحی یک مدل ساده برای سامانه آنلاین پیش‌بینی دیابت با ابزارهای اینترنت اشیاء و دو چارچوب NodeRed و Flask پرداخته شد.

فصل ۵

جمع‌بندی و کارهای آتی

۱-۵ جمع‌بندی

بر هیچ کس پوشیده نیست که پیشگیری بهتر از درمان است. لذا با استفاده از راهکارهای علمی، مدلی طراحی شد تا با دقت خوبی برای افراد جامعه، بتواند پیش‌بینی کند که آیا فرد به دیابت مبتلاست یا خیر تا در صورت لروم، اقدامات پیش‌گیرانه را سریع‌تر آغاز کنند.

از روش SMOTE برای نمونه‌برداری و از روش RFE با مدل رگرسیون لجستیک برای انتخاب ویژگی استفاده شد و در نتیجه انجام فرایندهای متعدد، مطابق نمودارهای ایجاد شده در فصل قبلی، الگوریتم جنگل درختان تصادفی در اعتبارسنجی متقابل Repeated K Fold با $K=10$ برای بهترین نتیجه را با میزان حدودی $ACC=96\%$ در این پیش‌بینی به ارمغان آورد که نشان می‌دهد دفعات بیشتری که الگوریتم‌های اعتبارسنجی متقابل مدل را مورد آزمایش قرار دهن، دقت ارزیابی بالاتری به دنبال خواهد داشت. همچنین میزان Recall و Precision برای این الگوریتم که با $7\%.97$ برابر بود (که تنها حدود ۱٪ از بیز ساده کمتر بود)، نشان دهنده اطمینان خوب به صحت پیش‌بینی این مدل است.

لازم به ذکر است از طریق بررسی عملکرد با شاخصه AUC پس از الگوریتم جنگل درختان تصادفی، نمودارهای درخت تصمیم و AdaBoost به ترتیب در رتبه‌های بعدی قرار گرفتند.

البته به خوبی مشخص است که هر چه دفعات انجام الگوریتم‌های اعتبارسنجی متقابل بیشتر باشد، زمان و هزینه کار نیز بیشتر خواهد بود.

۲-۵ کارهای آتی

در مورد روش‌های توسعه این سیستم می‌توان به مواردی چون سیستم‌های یادگیری ماشین آنلاین (با الگوریتم‌های یادگیری افزایشی^۱) اشاره کرد که دادگان همواره با اطلاعات جدید به روزرسانی می‌شوند و در بازه‌های مختلف توسط این الگوریتم، بهترین نتایج ارائه می‌شود. ضمن اینکه می‌توان از اینترنت اشیاء و امکاناتی از این قبیل استفاده کرد. به تازگی گجتها و کیت‌های مخصوصی برای گوشی‌های هوشمند طراحی شده‌اند که برای سنجش پارامترهای گوناگون سلامتی مورد استفاده قرار می‌گیرند و با

^۱Incremental learning

استفاده از گسترش شبکه‌های پرسرعت اینترنت نظیر 5G به سرعت می‌توان حجم عظیمی از داده‌های جدید را دریافت کرد.

به عنوان یک نمونه ساده و سمبیلیک می‌توان از برد الکترونیکی آردوینو که قابلیت نصب گجت‌های مختلف و سنسورهای گوناگون را فراهم می‌آورد، استفاده کرد و یک سیستم یادگیری ماشین آنلاین را طراحی کرد که با سرور (مثلًا برای آردوینو NodeRed می‌باشد) تبادل دارد.

همچنین می‌توان سامانه‌های موازی با این سیستم را نیز راه اندازی کرد مثل یک سایت پیش‌بینی‌کننده احتمال ابتلا به بیماری دیابت در افراد که هر شخصی با وارد کردن پارامترهای خودش می‌تواند نسبت به وضعیت سلامتی خود برآورده تقریبی داشته باشد.

پیوست

۱- کد استانداردسازی متون فارسی آمیخته به عبارات انگلیسی

با توجه به اینکه این پایان نامه در LATEX نوشته شده شد، یکی از معضلات مربوط به تایپ فارسی و انگلیسی در این سیستم، این است که حتماً عبارات انگلیسی بایستی درون تگ lr نوشته شوند تا فونت تعیین شده بر روی آن اعمال شود. پس یک برنامه ساده به زبان پایتون نوشته شد تا عبارات انگلیسی درون متن های فارسی را در این تگ قرار دهد.

در این کد پایتونی از مبحث RegEx که در درس کامپایلر مورد بحث بود، استفاده شد. فایل ورودی متن عادی است که پس از انجام عملیات، در فایل خروجی عبارات انگلیسی درون تگ lr قرار می گیرد.

```
import re
filename='import.txt'
lines=[]
with open(filename) as file:
    lines = [str(line.rstrip()) for line in file]

def myfun(a):
    e=a.group(0)
    e=' \lr{'+e+'}'
    return e

w=[]
for i in lines:
    x =re.findall(r"[a-zA-Z0-9\s]+\b(?=\\s[^a-zA-Z0-9]*)", i)
    tempi=re.sub(r"[a-zA-Z0-9\s]+\b(?=\\s[^a-zA-Z0-9]*)", myfun, i)
    w.append(tempi)
    print(tempi)

with open(r'export_text.txt', 'w+') as fp:
    for item in w:
        fp.write("%s\n" % item)
print('Done')
```

واژه‌نامه

Missing data	داده‌های گم شده	الف
ACC(Accuracy)	دقت	اختصاصی
Sigmoid	سیگماوار	اعتبارسنجی
BMI	شاخص توده بدنی	افراز
Precision	صحت	انتخاب ویژگی
Classification	طبقه‌بندی	آشتفتگی
FrameWork	چارچوب	ب
PhysActive	فعالیت فیزیکی	بیز ساده
TotChol	کلسترول کل	Overfitting
DirectChol	کلسترول مستقیم	بیش‌برازش
Underfitting	کم‌برازش	بیماری‌های زمینه‌ای
Scatter matrix	ماتریس پراکندگی	پ
FP	مثبت کاذب	پیش‌پردازش
TP	مثبت واقعی	AdaBoost
IDE	محیط توسعه	تقویت‌کننده انطباقی
	ک	Random forest
	م	ج
	Dataset	جنگل تصادفی
	NaN	ح
	Training data	حساسیت
	Testing data	د
		دادگان
		داده نامشخص
		داده‌های آموزشی
		داده‌های سنجش

Normalization	نرمال سازی	Mode	مد (آمار)
Resampling	نمونه‌برداری مجدد	Visualization	تصویرسازی
Under sampling	نمونه‌برداری کم		منحنی مشخصه
Over sampling	نمونه‌برداری زیاد	ROC	عملکرد سیستم
Race	نژاد	FN	منفی کاذب
Heatmap	نقشه حرارتی	TN	منفی واقعی
Lineplot	نمودار میله‌ای	Mean	میانگین
			میانگین فشار خون
	و	BPDiaAve	دیاستولیک
MaritalStatus	وضعیت تأهل		میانگین فشار خون
Feature	ویژگی	BPSysAve	سیستولیک
ن			
Incremental learning	یادگیری افزایشی	AUC	ناحیه زیر منحنی
		FPR	نرخ مثبت کاذب
		TPR	نرخ مثبت واقعی

مراجع

- [۱] اسماعیلی . مهدی، مفاهیم و تکنیک‌های داده کاوی
- [۲] نعمت الهی . نادر، آمار و احتمالات مهندسی

- [3] What is diabetes?, Aoife M Egan, Sean F Dinneen
- [4] Epidemiology of diabetes,Nita Gandhi Forouh,Nicholas J Wareha
- [5] Diabetes Cookbook For Dummies, by Alan L. Rubin, MD with Cait James, MS
- [6] Classification and prediction of diabetes disease using machine learning paradigm,Md.Maniruzzaman, Md. Jahanur Rahman, Benojir Ahammed and Md. Menhazul Abedin
- [7] Prediction of Diabetes using Classification Algorithms, Deepti Sisodia ,Dilip Singh
- [8] Logistic Regression,Lynne Connelly
- [9] Interpretable Machine Learning,Christoph Molnar
- [10] An Overview of Big Data Visualization Techniques in Data Mining ,Samuel Soma Ajibade, Anthonia Adediran
- [11] Data Mining Concepts and Techniques ,Jiawei Han,Micheline Kamber,Jian Pei
- [12] Data Mining for Business Analytics - Concepts, Techniques and Applications in Python , Galit Shmueli, Peter C. Bruce, Peter Gedeck, Nitin R. Patel
- [13] Top 10 algorithms in data mining,Xindong Wu,Jiannong Cao
- [14] Machine Learning for Predictive Analysis, Amit Joshi, Mahdi Khosravy, Neeraj Gupta
- [15] Cross-validation,Daniel Berrar
- [16] Predicting diabetes mellitus using SMOTE and ensemble machine learning approach, Henry Ford Exercise Testing (FIT) project , Manal Alghamdi,Mouaz Al-Mallah,Steven Keteyian,Clinton Brawner,Jonathan Ehrman,Sherif Sakr
- [17] The Working Principle Of An Arduino, Yusuf Abdullahi Badamasi
- [18] <https://www.cdc.gov/diabetes/basics/diabetes.html>
- [19] <https://www.kaggle.com/datasets/cdc/national-health-and-nutrition-examination-survey>

- [20] <https://medium.com/@rithpansanga/logistic-regression-for-feature-selection-selecting-the-right-features-for-your-model-410ca093c5e0>
- [21] <https://www.analyticsvidhya.com/blog/2023/05/recursive-feature-elimination/>
- [22] <https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765>
- [23] <https://towardsdatascience.com/how-to-split-data-into-three-sets-train-validation-and-test-and-why-e50d22d3e54c>
- [24] <https://www.techtarget.com/searchbusinessanalytics/definition/data-visualizatio>
- [25] https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation
- [26] <https://www.blog.trainindata.com/feature-scaling-in-machine-learning/>
- [27] <https://towardsdatascience.com/laplace-smoothing-in-na%C3%AFve-bayes-algorithm-9c237a8bdece>
- [28] <https://scikit-learn.org/stable/modules/tree.html>
- [29] <https://medium.com/udacity/shannon-entropy-information-gain-and-picking-balls-from-buckets-5810d35d54b4>
- [30] <https://www.datacamp.com/tutorial/adaboost-classifier-python>
- [31] <https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/>
- [32] <https://machinelearningmastery.com/confusion-matrix-machine-learning/>
- [33] <https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/>
- [34] <https://www.investopedia.com/terms/m/mlr.asp>
- [35] <https://aws.amazon.com/what-is/mqtt/>
- [36] <https://nodered.org/docs/>
- [37] <https://learn.sparkfun.com/tutorials/what-is-an-arduino/all>
- [38] <https://docs.arduino.cc/>
- [39] <https://pythonbasics.org/what-is-Flask-python/>