دانشگاه صنعتی نوشیروانی بابل

پایان‌نامه جهت دریافت درجه کارشناسی

رشته مهندسی کامپیوتر گرایش نرم‌افزار

عنوان

تشخیص بیماری دیابت با استفاده از یادگیری ماشین

استاد راهنما

دکتر فاطمه زمانی

نگارنده

ابوالفضل حسینی فر

آذر ۱۴۰۱

قدردانی

سپاس مخصوص خداوند مهربان که به انسان توانایی و دانایی بخشید تا به بندگانش شفقت ورزد، مهربانی کند و در حل مشکلاتشان یاری شان نماید. از راحت خویش بگذرد و آسایش هم نوعان را مقدم دارد، با او معامله کند و در این خلوص انباز نگیرد و خوش باشد که پروردگار سمیع و بصیر است.

همچنین از زحمات خانم دکتر فاطمه زمانی که در تمام مراحل انجام این تحقیق از رهنمودهای ایشان بهره‌مند شده‌ام، تشکر و قدردانی می‌کنم.

فهرست مطالب

فهرست شکل‌ها

فهرست جداول

چکیده

تشخیص زود و به‌موقع بیماری دیابت، نقشی اساسی در افزایش کیفیت سلامتی دارد و می‌تواند افراد را قبل از اینکه خیلی دیر شود، از شرایط خطرناک دوری دهد. طبق گزارش سازمان بهداشت جهانی سالانه نزدیک ۴۱ میلیون نفر جان خود را به دلیل بیماری‌های غیرمسری از دست می‌دهند که دیابت و فشارخون از شایع‌ترین آن‌ها هستند.

تشخیص بیماری دیابت از طریق علائم اولیه یک چالش بزرگ در جهان امروز است. بیماری که با توجه به سبک زندگی امروزی بخشی جدانشدنی از زندگی افراد است. یک سیستم پشتیبان تصمیم‌گیری دقیق، می‌تواند به تشخیص بیماری دیابت با استفاده از علائم اولیه کمک کند. در این پایان‌نامه برای تشخیص بیماری دیابت از روی مجموعه داده[۱]، الگوریتم‌های مختلف یادگیری ماشین[۲] از قبیل رگرسیون لجستیک[۳]، شبکه عصبی[۴]، ماشین بردار پشتیبان[۵]، k نزدیک‌ترین همسایه[۶]و ... با استفاده از انتخاب ویژگی‌های[۷] متفاوت از جمله روش همبستگی[۸]، آستانه واریانس[۹]، پیشرو[۱۰]، عقبگرد[۱۱] و ... بررسی شده‌اند. همچنین در این مدل از الگوریتم جنگل ایزوله[۱۲] برای تشخیص داده‌های پرت و روش SMOTETomek[۱۳] برای بالانس کردن دیتا و در آخر برای تشخیص نهایی و دقیق‌تر از الگوریتم یادگیری گروهی[۱۴] استفاده شده است تا به بهترین نتیجه ممکن برسیم.

[1] Dataset
[2] Machine learning algorithms
[3] Logistic Regression
[4] Neural Network
[5] Support Vector Machine (SVM)
[6] K-nearest Neighbor (KNN)
[7] Feature Selection
[8] Correletion
[9] Variance Threshold
[10] Forward
[11] Backward
[12] Isolation Forest
[13] synthetic minority oversampling technique tomek link
[14] Ensemble Learning

واژگان کلیدی

مجموعه داده، رگرسیون، انتخاب ویژگی، یادگیری ماشین، همبستگی، آستانه واریانس، جنگل ایزوله، ماشین بردار پشتیبان، درخت تصمیم، شبکه تصادفی، اعتبار سنجی متقابل، ماتریس سردرگمی، یادگیری گروهی

فصل اول: کلیات پژوهش

۱-۱: مقدمه

سازمان بهداشت جهانی گزارش داد که بیماری‌های غیرواگیر سالانه باعث مرگ زودرس ۴۱ میلیون نفر می‌شود، یعنی تقریباً ۷۱ درصد از کل مرگ‌ها در سطح جهان. در صورت عدم کاهش، تعداد کل مرگ و میر ناشی از بیماری های غیرواگیر تا سال ۲۰۳۰ به ۵۲ میلیون نفر در سال خواهد رسید . شایع ترین بیماری های غیرواگیر دیابت و فشار خون هستند که به ترتیب نزدیک به ۴۶/۲ و ۴ درصد از کل مرگ و میرها را تشکیل می‌دهند.[۱]

دیابت نوع ۲ یک اختلال متابولیک مداوم است که سطح گلوکز خون را تغییر می دهد و معمولاً نتیجه ناکارآمدی بدن در استفاده از انسولین تولید شده است. افرادی که دیابت دارند احتمال بیشتری برای سکته مغزی و مرگ و میر دارند . با این حال، نظارت مداوم بر سطح گلوکز خون می تواند به طور موثر عوارض دیابت را پیشگیری کند و یا کاهش دهد. در کشورهای در حال توسعه، تخمین زده می شود که تعداد افراد مبتلا به دیابت از حدود ۸۴ به ۲۲۸ میلیون تا سال ۲۰۳۰ افزایش یابد و این امر به طور قابل توجهی بر سیستم های مراقبت‌های بهداشتی سنگینی می‌کند .[۱]

یادگیری گروهی یکی از روش‌های یادگیری ماشین است که نتایج حاصل از مدل‌های طبقه‌بندی متعدد را ترکیب می‌کند و دقت بالاتری را در مقایسه با یک روش واحد نشان داده است. چندین مطالعه قبلی با موفقیت از رویکرد یادگیری گروهی برای کمک به تصمیم گیری‌های پزشکی و پیش بینی دیابت استفاده کرده‌اند.در حوزه یادگیری ماشین، ممکن است مشکلات چالش برانگیزی مانند داده‌های پرت و مجموعه داده‌های نامتعادل ایجاد شود که باعث کاهش دقت مدل می‌شود. مطالعات قبلی نشان داده‌اند که با استفاده از روش جنگل جداسازی (iForest) برای حذف داده‌های پرت و استفاده از تکنیک اقلیت مصنوعی تامک پیوند (SMOTETomek) برای متعادل کردن داده‌های نامتعادل ، عملکرد مدل یادگیری ماشین به طور قابل توجهی بهبود یافته است.[۱]

رویکرد ما در این پایان‌نامه شامل چهار مرحله است:

۱. سیستم نظارت بر عوامل خطر رفتاری (BRFSS) یک نظرسنجی تلفنی مرتبط با سلامت است که سالانه توسط CDCجمع آوری می شود. برای این پایان‌نامه، یک csv از مجموعه داده های موجود در Kaggle برای سال ۲۰۱۵ استفاده شد[1]. این مجموعه داده اصلی شامل پاسخ‌هایی از ۴۴۱۴۵۵ نفر است و دارای ۳۳۰ ویژگی است[۲]. این ویژگی‌ها یا سؤالاتی هستند که مستقیماً از شرکت کنندگان پرسیده می شود یا متغیرهای محاسبه شده بر

[1] 253,680 survey responses from cleaned BRFSS 2015 - binary classification

اساس پاسخ‌های فردی شرکت کنندگان هستند. در این تحقیق از مجموعه داده با ۸۰۰۰ نمونه و ۲۱ ویژگی استفاده شده است.

۲. در مرحله دوم، ما الگوریتم تشخیص داده‌های پرت را پیاده‌سازی کرده و سپس با استفاده از روش SMOTETomek اقدام به متعادل کردن داده‌ها می‌کنیم و نتیجه را با حالت اول بررسی می‌کنیم.

۳. در مرحله سوم، ما الگوریتم‌های انتخاب ویژگی را پیاده‌سازی کرده و هر کدام از این الگوریتم‌ها با توجه به روشی که کار می‌کنند، یک سری از ویژگی‌های مرحله اول را حذف کرده و ویژگی‌های باقیمانده را به عنوان خروجی به ما می‌دهند.

۴. در مرحله چهارم ما خروجی هر کدام از الگوریتم‌های انتخاب ویژگی را به الگوریتم‌های یادگیری ماشین به عنوان ورودی می‌دهیم و خروجی آنها را بررسی و تحلیل و مقایسه می‌کنیم و در نهایت بهترین الگوریتم‌ها برای استفاده در روش یادگیری گروهی به‌کار می‌گیریم و در نهایت سعی می‌کنیم تا به بیشترین دقت برسیم.

۱-۲: تعریف مسئله:

دیابت و فشار خون بالا دو عامل خطر اصلی برای سکته مغزی هستند که باعث مرگ و میر می‌شوند. یک مطالعه نشان داده است که با رژیم غذایی سالم و انجام ورزش منظم روزانه می‌توان از شرایط بدتر جلوگیری کرد. بنابراین، یک مدل پیش‌بینی زودهنگام بیماری که بتواند افراد را در مورد خطر دیابت و فشار خون بالا آگاه کند، مورد نیاز است که می‌تواند به آنها فرصت انجام اقدامات پیشگیرانه را بدهد. یادگیری ماشین را می توان به عنوان مدل اولیه پیش‌بینی بیماری برای پیش‌بینی دیابت و بیماری‌های فشار خون بر اساس داده های عوامل خطر فعلی فرد مورد استفاده قرار داد.

این پایان‌نامه ، ویژگی‌های مختلف مربوط به بیماری دیابت و مدل‌ها را بر اساس الگوریتم‌های یادگیری نظارت شده[1] مانند Naïve Bayes ، درخت تصمیم[2] ، KNN و... ارائه می‌کند.

الگوریتم‌های مختلف انتخاب ویژگی روی الگوریتم‌های یادگیری ماشین بررسی می‌شوند و سعی می‌شود بهترین آن‌ها برای استفاده در یادگیری گروهی به کار گرفته شوند و به بهترین دقت ممکن برسیم.

[1] Supervised
[2] Decesion tree

۱-۳: فرضیه‌ها و حوزه تحقیق

در این پایان‌نامه از یک csv از مجموعه داده های موجود در Kaggle برای سال ۲۰۱۵ استفاده شده است. این مجموعه داده اصلی شامل پاسخ‌هایی از ۴۴۱۴۵۵ نفر است و دارای ۳۳۰ ویژگی است. در این تحقیق از مجموعه داده با ۸۰۰۰ نمونه و ۲۱ ویژگی استفاده شده است که برای اثبات عملکرد الگوریتم‌های مختلف مهم است. این پایان‌نامه با هدف پیش‌بینی احتمال ابتلا به بیماری دیابت در بیماران انجام شده است.

۱-۴: اهداف تحقیق و سوالات اصلی

هدف اصلی ما یافتن بهترین الگوریتم‌ها برای به‌کارگیری در یادگیری گروهی و همچنین یافتن بهترین الگوریتم انتخاب ویژگی است که بتوانیم به بهترین نتیجه و دقت برسیم. باید به ترکیبی از الگوریتم‌ها برسیم که در مجموع دقت بالای ۸۵ درصد را به ما ارائه دهد.

سوالاتی که مطرح خواهد شد:

- داده‌های پرت چیست؟
- مجموعه داده متعادل و نامتعادل چیست؟
- انتخاب ویژگی چیست؟
- الگوریتم‌های یادگیری چه چیزی هستند؟
- بهترین روش انتخاب ویژگی چیست؟
- یادگیری گروهی چیست؟
- بهترین ترکیب الگوریتم‌ها برای استفاده در یادگیری گروهی کدام‌ها هستند؟

تا انتهای این پایان‌نامه به همه این سوالات پاسخ داده خواهدشد.

۱-۵: روش تحقیق

روش تحقیق به این صورت است که ابتدا برای تشریح‌کردن ویژگی‌های مجموعه داده، هر کدام از ویژگی‌ها را روی نمودار بررسی می‌کنیم. سپس مجموعه داده‌ای که روی آن انتخاب ویژگی انجام نشده را به علاوه لیستی از آبجکت‌ها از الگوریتم‌های یادگیری ماشین را به تابعی می‌دهیم و این تابع پارامترهای مختلفی را برای ارزیابی به ما می‌دهد. سپس با الگوریتم جنگل ایزوله داده‌های پرت را حذف می‌کنیم و دوباره الگوریتم‌ها را با مجموعه داده جدید بررسی می‌کنیم. سپس بهترین‌ها را به عنوان الگوریتم‌های یادگیری گروهی انتخاب می‌کنیم و نتیجه را

بررسی می‌کنیم. حالا نوبت آن است تا با روش SMOTETomek مجموعه داده را متعادل کنیم. پس از انجام این کار دوباره الگوریتم‌ها را بررسی می‌کنیم. پس از این کارها نوبت استفاده از الگوریتم‌های انتخاب ویژگی است. هر کدام از این الگوریتم‌ها یک سری ویژگی را حذف می‌کنند و یک سری را نگه می‌دارند. حالا باید ببینیم که کدام از آن‌ها عملکرد بهتری دارند و در نهایت بهترین عملکرد را معرفی خواهیم‌کرد.

۱-۶: ساختار پایان‌نامه

در فصل دوم ادبیات تحقیق مورد استفاده در این پژوهش بیان می‌شود. در فصل سوم چند نمونه مطالعاتی که در این زمینه انجام شده‌اند را بررسی خواهیم‌کرد. در فصل چهارم به نحوه پیاده‌سازی پروژه و تحلیل نتایج خواهیم پرداخت و در فصل پنجم به نتیجه‌گیری و پیشنهاد پرداخته خواهد شد.

فصل دوم: مفاهیم پایه

۲-۱: مقدمه

برای پیش‌بینی بیماری دیابت از روی مجموعه داده باید با استفاده از الگوریتم‌های انتخاب ویژگی ابتدا مجموعه داده را ساده کنیم و مجموعه داده جدیدی به‌دست آوریم و سپس از آن در الگوریتم‌های یادگیری ماشین استفاده کنیم. از این رو در این بخش به بیان الگوریتم‌ها و مفاهیم پایه مرتبط با آن پرداخته می‌شود.

۲-۲: آشنایی کلی با یادگیری ماشین

۲-۲-۱: یادگیری با نظارت (Supervised learning)

یادگیری نظارت‌شده نوعی از یادگیری مربوط به یادگیری ماشین است که در آن ورودی و خروجی مشخص است و در واقع ناظر اطلاعاتی را در اختیار یادگیرنده قرار می‌دهد و به این ترتیب سیستم تابعی را از ورودی به خروجی یاد می‌گیرد و همچنین در آن از داده‌های برچسب‌گذاری شده[1] استفاده می‌شود.

برای مثال ایمیلی که به شما زده می‌شود را در نظر بگیرید . ایمیل‌ها ورودی هستند و خروجی اسپم یا غیر اسپم بودن آن‌ها است که در واقع اسپم‌ها فیلتر می‌شوند . ابتدا داده‌ها به دو صورت اسپم و غیر اسپم تقسیم می‌شوند و به ماشین آموزش داده می‌شود. از ماشین امتحان گرفته می‌شود و ایمیلی را به ماشین می‌دهیم که تشخیص می‌دهد اسپم یا غیر اسپم است. به عبارت دیگر برای ورودی ما خروجی تعریف شده‌است.

۲-۲-۲: یادگیری بدون نظارت (Unsupervised Learning)

یادگیری بدون نظارت یک روش یادگیری ماشین است که در آن کاربران نیازی به نظارت بر مدل ندارند. در عوض به مدل اجازه می‌دهد تا به تنهایی برای کشف الگوها و اطلاعاتی که قبلاً کشف نشده بودند کار کند . این کار عمدتا با داده‌های بدون برچسب سروکار دارد. در یادگیری بدون نظارت بر خلاف یادگیری نظارت‌شده، داده‌ها از قبل مشخص نشده است و هدف آن ارتباط بین ورودی و خروجی نیست و فقط دسته بندی آنها مهم است و یادگیرنده باید در داده‌ها به دنبال ساختاری خاص بگردد.

الگوریتم‌های یادگیری بدون نظارت به کاربران این امکان را می‌دهد که کارهای پردازشی پیچیده‌تری را در مقایسه با یادگیری تحت نظارت انجام دهند. اگرچه یادگیری بدون نظارت در مقایسه با سایر روش های یادگیری طبیعی، می‌تواند غیر قابل پیشبینی باشد.

[1] Tagged data

۲-۲-۳: یادگیری تقویتی (Reinforecement Learning)

با خواندن این الگوریتم، دستگاه برای تصمیم‌گیری خاص آموزش دیده است. این روش به این صورت کار می‌کند که دستگاه در معرض محیطی قرار می‌گیرد که در آن به طور مداوم با استفاده از آزمون و خطا خود را آموزش می دهد. این دستگاه از تجربیات گذشته درس می‌گیرد و سعی می‌کند بهترین دانش ممکن را برای اتخاذ تصمیمات تجاری دقیق به دست آورد. برای مثال از یادگیری تقویتی می‌توان فرآیند تصمیم‌گیری مارکوف[1] را نام برد.

۲-۲-۴: رگرسیون (Regression)

رگرسیون یعنی بازگشت، پیش‌بینی و بیان تغییرات یک متغیر بر اساس اطلاعات متغیر دیگر و به دسته‌ای از الگوریتم‌ها گفته می‌شود. رگرسیون زمانی بکار می‌رود که خروجی ما مقداری پیوسته باشد اما برای طبقه‌بندی نیز بکار می رود. مثال: رابطه بین قد و وزن انسان‌ها را در نظر بگیرید. همه می‌دانیم که این رابطه یک رابطه مستقیم ریاضی و صد درصدی نیست که لزوما هر که قد بلندتری داشته باشد وزن بیشتری دارد، اما می‌توان گفت که با احتمال قابل قبولی افراد با قد بلندتر، وزن بیشتری نیز دارند. در اینجا پیش‌بینی وزن از روی قد و بیان ارتباط بین این متغیر با روش آماری رگرسیون خطی صورت می‌پذیرد که این رابطه را به صورت کمی به ما نشان می‌دهد. در مثال فوق معادله رگرسیون خطی می‌تواند به صورت زیر باشد:

متغیر وزن = متغیر قد*a+b

ترسیم این خط پس از محاسبه ضرایب a و b ما را به خط رگرسیون می‌رساند.

۲-۲-۵: طبقه بندی (Classification)

طبقه‌بندی نظارت شده یکی از کارهایی است که اغلب توسط سیستم‌های هوشمند انجام می‌شود. بنابراین، تعداد زیادی تکنیک بر اساس هوش مصنوعی(تکنیک‌های مبتنی بر منطق[2]، تکنیک‌های مبتنی بر پرسپترون[3]) و آمار (شبکه‌های بیزی[4]، تکنیک‌های مبتنی بر نمونه[5]) توسعه یافته‌اند . طبقه‌بندی برای داده‌هایی بکار می‌رود که خروجی مورد نظر ما اعداد گسسته است مثلا در مجموعه داده‌ی این پایان نامه، خروجی ۱ یا ۰ است که به معنای وجود یا عدم وجود بیماری دیابت است.

[1] Markov Process
[2] Logic
[3] Perceptron
[4] Bayesian network
[5] Sample-based techniques

۲-۲-۶: داده‌های آموزش و آزمایش (Train and Test data)

در یادگیری ماشین، بخشی از داده‌های مجموعه داده را به عنوان داده‌های آموزش و بخشی را به عنوان داده‌های آزمایش به وسیله الگوریتم‌های مختلف جدا می‌کنیم. از قسمت داده‌های آموزش برای یادگیری و از قسمت داده های آزمایش برای ارزیابی الگوریتم و میزان دقت آن استفاده می‌کنیم. روش کلاسیک به این صورت بود که معمولا ۸۰ یا ۷۰ درصد ثابت داده را برای آموزش و ۲۰ یا ۳۰ درصد آن را برای آزمایش بکار می‌گرفتند (Train Test Split) اما در این پایان‌نامه علاوه بر آن، از روش‌های جدیدترکه k-fold و cross validation است نیز استفاده شده که چندین بار داده‌های آزمایش و آموزش را جابه‌جا می‌کند تا به بهترین دقت برسد که در ادامه این پایان نامه توضیح داده خواهد شد.

۲-۲-۷: ارزیابی مدل (Model evaluation)

دانشی که در مرحله یادگیری مدل تولید می‌شود، می‌بایست در مرحله ارزیابی مورد تحلیل قرار گیرد تا بتوان ارزش آن را تعیین نمود و در پی آن کارایی الگوریتم یادگیرنده مدل را نیز مشخص کرد. این معیارها را می‌توان هم برای مجموعه داده‌های آموزشی در مرحله یادگیری و هم برای مجموعه نمونه‌های آزمایشی در مرحله ارزیابی محاسبه نمود. همچنین لازمه موفقیت در بهره‌مندی از علم داده‌کاوی، تفسیر دانش تولید و ارزیابی‌شده است. به عبارتی ساده‌تر، ارزیابی مدل در این پایان‌نامه به معنای بررسی میزان دقت الگوریتم‌های یادگیری ماشین در پیش بینی بیماری دیابت است.

۲-۲-۸: اضافه برازش (Overfitting)

اضافه برازش به مدلی اشاره دارد که داده‌های آموزشی را خیلی خوب مدل می‌کند. تطبیق بیش از حد زمانی اتفاق می‌افتد که یک مدل جزئیات و نویز در داده‌های آموزشی را تا حدی بیاموزد که بر عملکرد مدل در داده‌های جدید تأثیر منفی بگذارد. این بدان معنی است که نویز یا نوسانات تصادفی در داده‌های آموزشی به عنوان مفاهیم توسط مدل انتخاب شده و یاد می‌گیرد. مشکل این است که این مفاهیم برای داده‌های جدید اعمال نمی‌شوند و بر توانایی تعمیم مدل تأثیر منفی می‌گذارند.

برازش بیش از حد در مدل های ناپارامتریک[1] و غیرخطی[2] که انعطاف‌پذیری بیشتری در هنگام یادگیری تابع هدف دارند، بیشتر است. به این ترتیب، بسیاری از الگوریتم‌های یادگیری ماشین ناپارامتریک نیز شامل پارامترها یا تکنیک‌هایی برای محدود کردن جزئیاتی هستند که مدل یاد می گیرد. به عنوان مثال، درختان تصمیم یک الگوریتم یادگیری ماشین ناپارامتریک هستند که بسیار منعطف هستند و در معرض داده‌های آموزشی بیش از حد

[1] Nonparametric
[2] Nonlinear

مناسب هستند. این مشکل را می‌توان با هرس‌کردن[1] یک درخت پس از یادگیری به منظور حذف بخشی از جزئیاتی که برداشت کرده است، برطرف کرد.

۲-۲-۹: کم برازش یا عدم تناسب (Underfitting)

عدم تناسب یا کم‌برازش به مدلی اطلاق می‌شود که نه می‌تواند داده‌های آموزشی را خوب مدل کند و نه می‌تواند به داده‌های جدید تعمیم دهد. یک مدل یادگیری ماشین با عدم تناسب مدل مناسبی نیست و واضح است که عملکرد ضعیفی در داده‌های آموزشی خواهد داشت.عدم تناسب اغلب مورد بحث قرار نمی‌گیرد زیرا با توجه به یک معیار عملکرد خوب، تشخیص آن آسان است. راه حل این است که الگوریتم‌های یادگیری ماشین جایگزین را امتحان کنید. با این وجود، تضاد خوبی با مشکل بیش برازش ایجاد می‌کند.

۲-۲-۱۰: یادگیری عمیق (Deep learning)

به صورت خلاصه، تعریف یادگیری عمیق را می‌توان اینگونه بیان کرد:

روش هایی از یادگیری ماشین بر پایه استفاده از شبکه های عصبی عمیق که از داده های موجود برای محاسبه رفتارها و خروجی‌های آینده استفاده می‌کند. اگر به این تعریف نگاه کنیم می‌فهمیم که در واقع یادگیری عمیق یکی از روش‌های یادگیری ماشین است. در این روش ، ماشین‌ها یاد می‌گیرند که بر اساس مدل‌هایی شبیه شبکه های عصبی مغز انسان مفاهیم سطح بالا و انتزاعی را یاد بگیرند. استفاده از یادگیری عمیق کمک می کند که ماشین‌ها بتوانند تصمیم‌هایی شبیه تصمیم های انسانی بگیرند.

در یادگیری عمیق ازچند لایه مختلف شبکه عصبی استفاده می‌شود. هرکدام از این لایه‌ها بخش‌هایی از اطلاعات ورودی را تحلیل می‌کنند. این لایه‌های چندگانه، امکان پیش‌بینی را در یادگیری عمیق افزایش می‌دهند. تعداد این لایه‌ها گاهی می تواند تا ۱۵۰ لایه برسد.

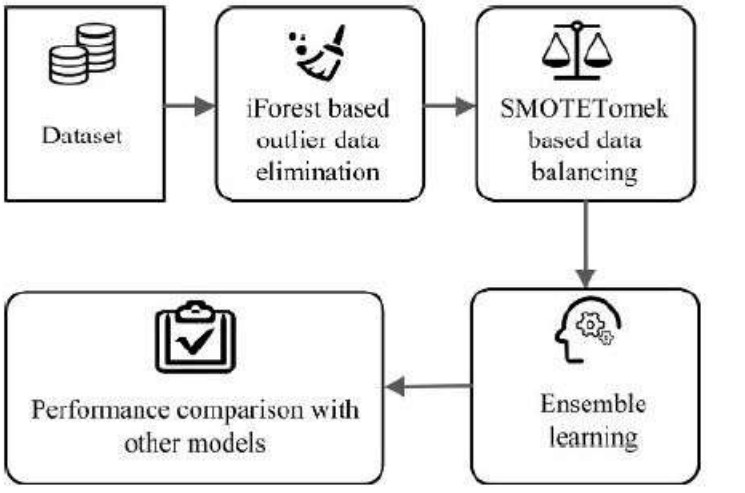
در یادگیری عمیق از روش های متفاوتی استفاده می‌شود. این روش ها بسته به کاربردهای متفاوت یادگیری عمیق و نوع داده های ورودی و خروجی موردنیاز انتخاب می شود.مانند:

- شبکه‌های عصبی کلاسیک (Classic Neural Networks)
- شبکه‌های عصبی پیچشی (Convolutional Neural Networks)
- شبکه‌های عصبی برگشتی (Recurrent Neural Networks)
- رمزگذار خودکار (Auto Encoders)

[1] Pruning

۲-۲-۱۱: یادگیری گروهی

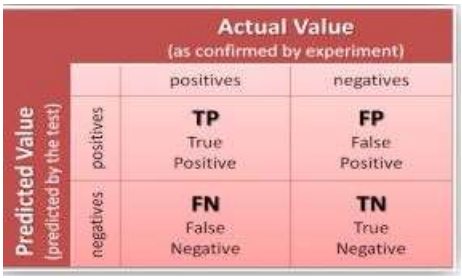
یادگیری گروهی نوعی روش طبقه‌بندی است که از طریق ترکیب الگوریتم‌های طبقه‌بندی متعدد در یک مدل واحد طراحی شده است تا بایاس و واریانس را کاهش دهد و در نتیجه دقت پیش‌بینی را بهبود ببخشد. در این پایان‌نامه، ما از یادگیری گروهی با اعتبارسنجی متقابل استفاده کردیم که دو سطح طبقه‌بندی کننده را ارائه می‌دهد: اول و دوم، که می‌تواند به جلوگیری از برازش بیش از حد کمک کند[۱]. اعتبار سنجی متقابل کمک می‌کند تا در مدل استفاده از مجموعه آموزشی مشابه برای طبقه‌بندی‌کننده‌های سطح اول و دوم جلوگیری شود. مطالعات قبلی نشان داد که پرسپترون چند لایه(MLP) ، ماشین‌های بردار پشتیبان(SVM) ، درخت تصمیم (DT)و رگرسیون لجستیک (LR) می‌توانند به عنوان مدل‌های پیش‌بینی مورد استفاده قرار گیرند و نتایج قابل توجهی در بهبود دقت طبقه‌بندی در چندین مجموعه داده سلامت نشان دادند [۱] بنابراین، ما از MLP ، SVM، و DT به عنوان طبقه‌بندی‌کننده سطح اول در بیش‌تر مواقع استفاده می‌کنیم. مطالعات قبلی نشان داده است که رگرسیون لجستیک عملکرد بسیار خوبی به عنوان طبقه‌بندی‌کننده سطح اول دارد، بنابراین در این پایان‌نامه از رگرسیون لجستیک به عنوان طبقه‌بندی‌کننده سطح اول استفاده خواهیم کرد و می‌بینیم که عملکرد بهتری نسبت به سایر الگوریتم‌ها دارد. شکل(۲-۱) ساختار مدل پیشنهادی ما را در این پایان‌نامه نشان می‌دهد:



شکل(۲-۱): مدل پیشنهادی و پیاده‌سازی شده در این مقاله

۲-۲-۱۲: ماتریس سردرگمی (Confusion matrix)

شکل (۲-۲) Confusion matrix را نشان می‌دهد:



شکل(۲-۲): **confusion matrix**

در یادگیری ماشین، ماتریسی به نام ماتریس سردرگمی که با نام ماتریس خطا نیز شناخته می‌شود وجود دارد که امکان تجسم عملکرد یک الگوریتم یادگیری را به ما می‌دهد. هر ردیف از ماتریس نمونه های یک کلاس پیش بینی شده را نشان می‌دهد در حالی که هر ستون نمونه‌های یک کلاس واقعی را نشان می‌دهد و یا برعکس.

TP (True Positive) : تعداد پیش‌بینی‌های مثبتی را نشان می‌دهد که در واقعیت نیز مثبت است.

FP (False Positive) : تعداد پیش‌بینی‌های مثبتی را نشان می‌دهد که در واقعیت منفی است.

FN (False Negative) : تعداد پیش‌بینی‌های منفی را نشان می‌دهد که در واقعیت هم منفی است.

TN (True Negative) : تعداد پیش‌بینی‌های منفی را نشان می‌دهد که در واقعیت مثبت است.

۲-۲-۱۳: دقت (Accuracy)

به ما می‌گوید که هر چند وقت یک بار می‌توانیم انتظار داشته باشیم که مدل یادگیری ماشین ما از مجموع تعداد دفعاتی که پیش‌بینی کرده است، نتیجه را به درستی پیش‌بینی کند. به عنوان مثال: فرض کنید که شما مدل یادگیری ماشین خود را با مجموعه داده‌ای متشکل از ۱۰۰ رکورد آزمایش می‌کنید و مدل یادگیری ماشین شما تمام ۹۰ مورد را به درستی پیش‌بینی می‌کند. دقت، در این مورد ۹۰٪ خواهد بود که دقت عالی است؛ اما چیزی در مورد خطاهایی که مدل های یادگیری ماشین ما روی داده‌های جدیدی که قبلاً ندیده‌ایم ایجاد می‌کنند، به ما نمی‌گوید.

۲-۲-۱۴: امتیاز یادآوری (Recall)

امتیاز یادآوری مدل نشان‌دهنده توانایی مدل در پیش‌بینی صحیح موارد مثبت از موارد مثبت واقعی است؛ این برخلاف دقتی است که تعداد پیش‌بینی‌های انجام شده توسط مدل ها را از بین همه پیش بینی‌های مثبت انجام شده در واقع مثبت می‌کند . به عنوان مثال اگر مدل یادگیری ماشین شما در تلاش است تا موارد مثبت را شناسایی کند، امتیاز یادآوری این خواهد بود که چند درصد از آن نظرات مثبت را مدل یادگیری ماشین شما به درستی به‌عنوان مثبت پیش‌بینی کرده است. به عبارت دیگر، اندازه‌گیری می‌کند که مدل یادگیری ماشین ما

چقدر در شناسایی همه موارد مثبت واقعی از بین همه موارد مثبت موجود در یک مجموعه داده خوب است. هرچه امتیاز Recall بالاتر باشد، مدل یادگیری ماشین در شناسایی نمونه‌های مثبت و منفی بهتر است. Recall به عنوان حساسیت یا نرخ مثبت واقعی نیز شناخته می‌شود. امتیاز Recall بالا نشان می‌دهد که مدل در شناسایی نمونه‌های مثبت خوب است؛ برعکس، امتیاز Recall پایین نشان می دهد که مدل در شناسایی نمونه‌های مثبت خوب نیست. Recall اغلب همراه با سایر معیارهای عملکرد مانند دقت و صحت، برای دریافت تصویر کاملی از عملکرد مدل استفاده می‌شود. از نظر ریاضی، نسبت مثبت واقعی به مجموع مثبت واقعی و منفی کاذب را نشان می‌دهد .

فرمول(۲-۱) برای محاسبه recall بکار می‌رود:

$$recall = \frac{TP}{TP + FN}$$

فرمول(۲-۱): محاسبه recall

۲-۲-۱۵: نرخ منفی واقعی (Specificity)

Specificity نسبت منفی‌های واقعی را که به درستی توسط مدل شناسایی شده‌اند اندازه‌گیری می‌کند. این بدان معناست که نسبت دیگری از منفی واقعی وجود خواهد داشت که مثبت پیش‌بینی شده و می‌تواند به عنوان مثبت کاذب نامیده شود. این نسبت را می‌توان نرخ منفی واقعی[1] نیز نامید. مجموع نرخ منفی واقعی و نرخ مثبت کاذب همیشه یک خواهد بود. Specificity بالا به این معنی است که مدل بیشتر نتایج منفی را به درستی شناسایی می‌کند. در حالیکه Specificity پایین به این معنی است که مدل بسیاری از نتایج منفی را به اشتباه به عنوان مثبت نشان می‌دهد .

ار نظر ریاضی، specificity را می‌توان با فرمول (۲-۲) محاسبه کرد:

$$specificity = \frac{TN}{TN + FP}$$

فرمول(۲-۲): محاسبه specificity

[1] TNR

۲-۲-۱۶: درستی (Precision)

در ساده ترین عبارت،Precision نسبت بین مثبت‌های واقعی و همه مثبت ها است. برای بیان مشکل ما، این معیار نسبت بیمارانی است که ما درست به‌صورت مثبت پیش‌بینی کردیم به کل بیمارانی که مثبت پیشبینی کردیم. از نظر ریاضی درستی را می توان با فرمول(۲-۳) محاسبه کرد.

$$Precision = \frac{TP}{TP + FP}$$

فرمول(۲-۳): محاسبه precision

۲-۲-۱۷: امتیاز مدل F1 (F1-Score)

امتیاز مدل F1 نشان‌دهنده امتیاز مدل به عنوان تابعی از دقت و امتیاز یادآوری است. F-score یک معیار عملکرد مدل یادگیری ماشین است که برای اندازه‌گیری عملکرد آن از نظر دقت، وزن یکسانی به Precision و Recall می‌دهد و آن را جایگزینی برای معیارهای دقت می‌کند (نیازی به دانستن تعداد کل مشاهدات نیست). اغلب به عنوان یک مقدار واحد استفاده می‌شود که اطلاعات سطح بالایی در مورد کیفیت خروجی مدل ارائه می‌دهد ؛ این یک معیار مفید برای مدل در سناریوهایی است که در آن فرد سعی می کند دقت یا امتیاز یادآوری را بهینه کند و در نتیجه عملکرد مدل آسیب می‌بیند.

F1-Scoe از فرمول(۲-۴) به‌دست می‌آید:

$$f1 - score = \frac{2 * precision * recall}{precision + recall}$$

فرمول(۲-۴): محاسبه f1-score

۲-۳: داده‌های پرت و ناهنجاری‌ها

پیدا کردن نقاط پرت(Outlier) ، کاری است که باید تقریبا در هرگونه تحلیل‌های آماری صورت گیرد. وجود نقاط نامتعارف یا ناهنجار، باعث ایجاد خطا در مدل‌های آماری شده و پیش‌بینی را با مشکل مواجه می‌کند. به همین دلیل شناسایی ناهنجاری‌ها (Anomaly Detection) در علم داده (DataScience) از اهمیت زیادی برخوردار است. در علم آمار، یک ناهنجاری یک مشاهده نامتعارف، رویداد یا مقداری است که اختلاف و انحراف قابل توجهی نسبت به سایر مقادیر داشته باشد و رفتار متفاوتی نسبت به دیگر نقاط داشته باشد.

اما چرا بایستی داده‌های پرت را مورد بررسی قرار دهیم؟

فرض کنید یک کارت بانکی دارید و به صورت معمول و عادی از این کارت استفاده‌هایی می‌کنید. مثلاً حقوق ماهیانه‌ی شما به این کارت واریز می‌شود و شما در طول ماه آرام آرام آن مبالغ را توسط دستگاه‌های POS دریافت کرده و یا به صورت اینترنتی از فروشگاه‌های مشخص خرید می‌کنید. حال کارت شما دزدیده می‌شود و این شخص سریعاً به محل دیگری رفته و با دانستن رمز کارت، سریعاً درخواست دریافت مبلغی نامتعارف را از یک دستگاه POS در یک زمان نامتعارف انجام می‌دهد. این کار یک عمل غیر طبیعی (برای کارت شما) است، و اگر یک الگوریتم تشخیص داده‌های (یا همان فرآیندهای) پرت در شبکه‌ی شتاب موجود باشد، احتمالاً می‌تواند این عملیات را شناسایی کرده و کارت بانکی را به عنوان دزدیده شده ضبط نماید (و یا درخواست رمزی مانند رمز دوم انجام شود). در حوزه‌هایی مانند ورزش فوتبال نیز می‌توان از داده‌کاوی و فرآیندهای تشخیص داده‌های پرت استفاده کرد. برای مثال از طریق سنسورهایی که به بازیکنان متصل است و با کمک تحلیل آن‌ها در شرایط مختلف، می‌توان بازیکنانی که توانایی‌های بالاتری (با توجه به شرایط) دارند را کشف کرد. برای مثال، برخی از بازیکنان در شرایط جوی بارانی، عملکرد بهتری از خود به نمایش می‌گذارند و در واقع به عنوان یک داده‌ی پرت، از سایر بازیکنان جدا شده و شناسایی می‌شوند.

۲-۳-۱: الگوریتم جنگل ایزوله برای تشخیص ناهنجاری و داده‌های پرت (Isolation Forest)

الگوریتم جنگل ایزوله (Isolation Forest) یا جنگل جداسازی، یک الگوریتم یادگیری بدون نظارت(Unsupervised Learning Algorithm) برای تشخیص ناهنجاری (Anomaly) است که برای جداسازی نقاط پرت (Outlier) به کار می‌رود .البته در اغلب روش‌های شناسایی نقاط پرت، بقیه نقاط که رفتار عادی دارند مورد ارزیابی قرار گرفته و براساس رفتار آن‌ها، نقاط پرت مشخص می‌شوند در حالی‌که در الگوریتم جنگل ایزوله از ابتدا این گونه نقاط مورد بررسی قرار می‌گیرند.

الگوریتم جنگل ایزوله (Isolation Forest) یک ویژگی (یک بعد) را به صورت تصادفی انتخاب می‌کند و سپس یک مقدار تصادفی بین کمینه (Minimum) و بیشینه‌ی (Maximum) آن انتخاب کرده و با یک خط جداساز آن بعد را جدا می‌کند. حال دوباره الگوریتم، یک ویژگی (بعد) تصادفی را انتخاب می‌کند و دوباره خطی برای جداسازی آن ویژگی به صورت تصادفی می‌کشد. در واقع الگوریتم آنقدر این تقسیم‌بندی را انجام می‌دهد تا بلاخره یک نقطه، تنهای تنها، در یکی از محوطه‌ها پیدا شود. اما می‌دانیم که در اکثر مواقع با بیشتر از دو بعد (ویژگی) سر و کار داریم، پس برای به دست آوردن نقاط تنها بایستی درخت را خیلی بیشتر از این ادامه دهیم. اینجاست که ایده اصلی جنگل ایزوله مطرح می‌شود. با روش تقسیم‌بندی درخت‌ها در جنگل ایزوله، داده‌های پرت، سریع‌تر از سایر نقاط (داده‌ها) تنها (ایزوله) می‌شوند. اگر بخواهیم از منظر درخت بگوییم، داده‌های پرت به ریشه (بالای) درخت نزدیک‌تر هستند زیرا این درخت‌ها، نقاط ایزوله را زودتر از سایرین پیدا می‌کنند و آن‌ها را ایزوله (تنها) می‌کنند. برای تولید جنگل بایستی چند درخت (مثلا ۱۰۰۰درخت) ساخته شود و هر درخت مشخص می‌کند که

کدام نقاط زودتر ایزوله شدند (داده‌ی پرت هستند)، و جنگل ایزوله هم با توجه به این درخت‌ها و تصمیماتشان، تصمیم نهایی را گرفته و به هر نقطه (به صورت میانگین) یک امتیاز بین ۰ تا ۱ می‌دهد. هر چقدر این امتیاز به ۱نزدیک‌تر باشد، این نقطه به احتمال بیشتری، داده‌ی پرت است.

۲-۴: متعادل کردن مجموعه داده

داده نامتعادل به طور ساده برمی‌گردد به مسائل طبقه‌بندی که در آنها داده‌های گروه‌ها به طور یکسان نباشد. برای مثال در یک مسئله دو کلاسه، ۱۰۰ تا نمونه داشته باشید که ۸۰ تا از این نمونه های مربوط به کلاس یک و ۲۰ تا مربوط به کلاس دو باشد. در چنین حالتی شما یک پایگاه داده نامتعادل دارید که در آن تعداد نمونه‌های کلاس یک ۴ برابر کلاس دو است! حالا فرض کنیم این داده‌ها را به یک مدل بدهیم و مدل در یک تصمیم هوشمندانه همه داده‌ها را به گروه یک دسته‌بندی کند، در این شرایط دقت خوبی هم به دست می‌آید اما این مدل در واقع توزیع داده‌ها در کلاس‌ها را نشان می‌دهد نه چیز دیگری! برای حل این مشکل یکی از راه‌ها استفاده از روش SMOTETomek است.

در دنیای واقعی، مدل‌سازی طبقه‌بندی اغلب با یک مشکل مجموعه داده نامتعادل مواجه می‌شود، که در آن تعداد کلاس اکثریت بسیار بزرگ‌تر از کلاس اقلیت است، بنابراین باعث می‌شود که مدل نتواند از کلاس اقلیت به خوبی یاد بگیرد. هنگامی که اطلاعات موجود در مجموعه داده از کلاس اقلیت مهم‌تر باشد، به عنوان مثال، مانند مجموعه‌داده‌های تشخیص بیماری و مجموعه داده های تشخیص جرم، این یک مشکل جدی محسوب می شود. یکی از روش‌های رایج برای حل این مشکل مجموعه‌داده‌های نامتعادل، نمونه‌برداری بیش از حد از کلاس اقلیت یا کم‌نمونه‌سازی از کلاس اکثریت است. این رویکردها اما ضعف‌های خاص خود را دارند برای مثال داده جدیدی به مجموعه‌داده اضافه نمی‌شود.

یکی از راه حل ها برای غلبه بر این ضعف، تولید نمونه‌های جدید است که با طبقه اقلیت موجود ترکیب شده است. این روش به عنوان روش نمونه‌برداری بیش از حد اقلیت مصنوعی یا SMOTE شناخته شده است . SMOTE انواع مختلفی دارد، اما در این پایان‌نامه، روش SMOTE-Tomek Links بررسی خواهد شد، جایی که روش oversampling از SMOTE با undersampling از Tomek links ترکیب می‌شوند.

۱-۴-۲: SMOTE

SMOTE یکی از محبوب ترین تکنیک‌های نمونه برداری بیش از حد است که توسط Chawla و همکاران توسعه داده شده است. برخلاف نمونه‌برداری بیش از حد تصادفی که فقط برخی از نمونه‌های تصادفی را از کلاس اقلیت کپی می‌کند، SMOTE بر اساس فاصله هر داده (معمولاً با استفاده از فاصله اقلیدسی) و نزدیک‌ترین همسایگان

کلاس اقلیت، نمونه‌هایی تولید می‌کند، بنابراین نمونه‌های تولید شده با کلاس اقلیت اصلی متفاوت هستند. به طور خلاصه، فرآیند تولید نمونه‌های مصنوعی به شرح زیر است.

- داده های تصادفی را از کلاس اقلیت انتخاب کنید.
- فاصله اقلیدسی بین داده های تصادفی و k نزدیکترین همسایه آن را محاسبه کنید.
- اختلاف را با یک عدد تصادفی بین ۰ و ۱ ضرب کنید، سپس نتیجه را به عنوان نمونه مصنوعی به کلاس اقلیت اضافه کنید.
- این روش را تا زمانی که نسبت مورد نظر طبقه اقلیت برآورده شود، تکرار کنید.

این روش موثر است زیرا داده‌های مصنوعی که تولید می‌شوند نسبتاً نزدیک به فضای ویژگی در کلاس اقلیت هستند، بنابراین برخلاف روش نمونه‌برداری اولیه، «اطلاعات» جدیدی روی داده‌ها اضافه می‌شود.[۳]

۲-۴-۲: Tomek Links

Tomek Links روش اصلاح شده الگوریتم نزدیک‌ترین همسایه‌های متراکم[1] است که در سال ۱۹۷۶ توسط Tomek توسعه یافته‌است. برخلاف روش نزدیک‌ترین همسایه‌های متراکم که فقط به‌طور تصادفی نمونه‌ها را با k نزدیک‌ترین همسایه‌های خود از کلاس اکثریتی که می‌خواهد حذف شود انتخاب می‌کند، روش Tomek links از قانون جفت مشاهده (مثلا a و b) استفاده می‌کند که روش کار به صورت زیر است:

- نزدیک‌ترین همسایه a، b است.
- نزدیک‌ترین همسایه b، a است.
- a و b به کلاس‌های متفاوت تعلق دارند یعنی a متعلق به کلاس اکثریت و b متعلق به کلاس اقلیت است.

از نظر ریاضی می‌توان آن را به صورت زیر بیان کرد:

فرض کنید که $d(x_i,x_j)$ فاصله اقلیدسی بین x_i و x_j باشد که x_i متعلق به کلاس اقلیت و x_j متعلق به کلاس اکثریت است. اگر نمونه‌ای مانند x_k وجود نداشته باشد که شرایط زیر را برآورده کند:

$$1)\ d(x_i,x_k) < d(x_i,x_j)\ \ or$$

$$2)\ d(x_j,x_k) < d(x_i,x_j)$$

آن‌گاه x_i,x_j یک پیوند Tomek است.

[1] Condensed Nearest Neighbors

این روش می‌تواند برای یافتن و حذف نمونه‌های مورد نظر از داده‌های کلاس اکثریت که کمترین فاصله اقلیدسی را با داده‌های کلاس اقلیت دارند (یعنی داده‌هایی از کلاس اکثریت که به داده‌های کلاس اقلیت نزدیک‌ترین هستند و تشخیص آن مشکل است و ابهام دارد) استفاده شود.[۳]

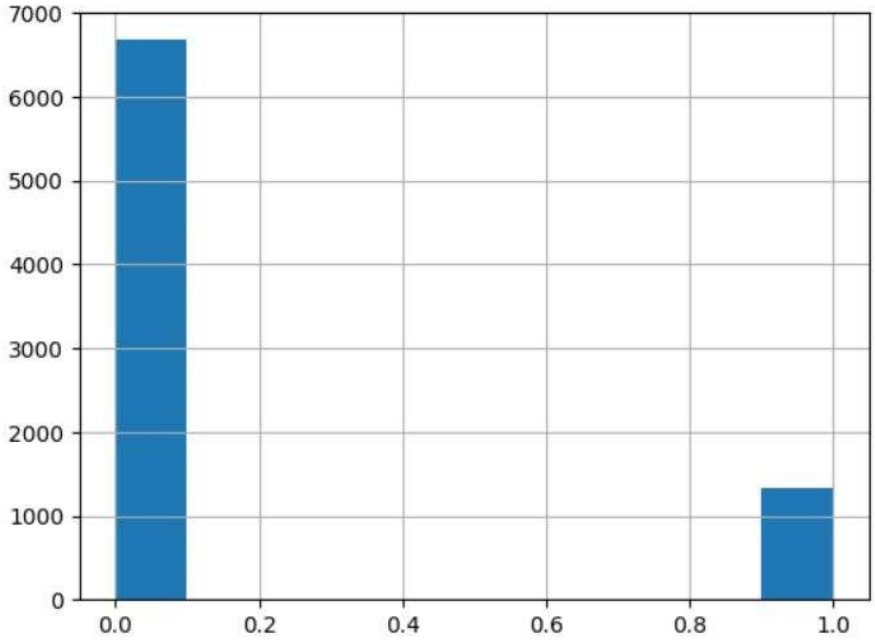
۲-۴-۳: متعادل کردن مجموعه داده با روش SMOTETomek

این روش ابتدا توسط باتیستا و همکاران در سال ۲۰۰۳ معرفی شد. این روش ترکیبی از توانایی SMOTE برای تولید داده‌های مصنوعی برای کلاس اقلیت و توانایی Tomek Links برای حذف داده‌هایی است که به عنوان پیوندهای Tomek از کلاس اکثریت شناسایی می‌شوند (یعنی نمونه‌هایی از داده‌ها از کلاس اکثریت که نزدیک‌ترین هستند به داده‌های طبقه اقلیت) فرآیند پیوندهای SMOTE-Tomek به شرح زیر است:

- (شروع SMOTE) داده‌های تصادفی را از کلاس اقلیت محاسبه کنید.
- فاصله بین داده های تصادفی و k نزدیکترین همسایه آن را محاسبه کنید.
- اختلاف را با یک عدد تصادفی بین ۰ و ۱ ضرب کنید، سپس نتیجه را به عنوان نمونه مصنوعی به کلاس اقلیت اضافه کنید.
- مرحله شماره ۲ تا ۳ را تا زمانی که نسبت مورد نظر طبقه اقلیت برآورده شود، تکرار کنید. (پایان SMOTE)
- (شروع Tomek Links) داده های تصادفی را از کلاس اکثریت انتخاب کنید.
- اگر نزدیک‌ترین همسایه داده‌های تصادفی، داده‌های کلاس اقلیت است (یعنی ایجاد پیوندTomek) ، سپس پیوند Tomek را حذف کنید.

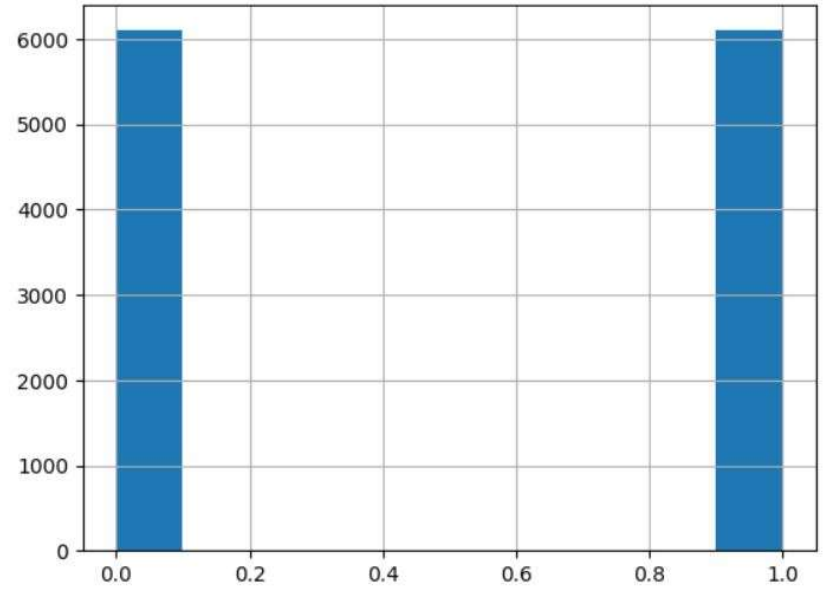
در شکل(۲-۳) مجموعه داده و به خصوص ویژگی Diabetes_binary را قبل و در شکل(۲-۴) بعد از اعمال روش SMOTETomek مشاهده می‌کنیم.

```
0.0    6675
1.0    1325
Name: Diabetes_binary, dtype: int64
```



شکل(۲-۳): ویژگی هدف (**Diabetes_binary**) در مجموعه داده قبل از اعمال روش **SMOTETomek**

```
0.0    6095
1.0    6095
Name: Diabetes_binary, dtype: int64
```



شکل(۲-۴): ویژگی هدف (**Diabetes_binary**) در مجموعه داده بعد از اعمال روش **SMOTETomek**

می‌بینیم که پس از اعمال SMOTETomek، مجموعه داده متعادل شده است. هدف طبقه بندی همیشه به حداقل رساندن خطاها در حین یادگیری است. از این رو انتظار می‌رود که دقت بهتری را می‌توان از مدلی که از مجموعه داده های متعادل استفاده می‌کند، به‌دست آورد.[۱][۳]

۲-۵: الگوریتم‌های انتخاب ویژگی چیست؟

انتخاب ویژگی ، به عنوان یک استراتژی پیش‌پردازش[1] داده، ثابت شده است که در تهیه داده‌ها (به ویژه داده‌های با ابعاد بالا) برای مشکلات مختلف داده‌کاوی و یادگیری ماشین موثر و کارآمد است. انتخاب ویژگی را می‌توان به عنوان فرآیند شناسایی ویژگی‌های مرتبط و حذف ویژگی‌های غیر مرتبط و تکراری با هدف مشاهده زیرمجموعه ای از ویژگی‌ها که مساله را به خوبی و با حداقل کاهش درجه کارایی تشریح می‌کنند، تعریف کرد. این کار مزایای گوناگونی دارد که برخی از آن‌ها در ادامه بیان شده‌اند:

- ✓ بهبود کارایی الگوریتم‌های یادگیری ماشین
- ✓ درک داده، کسب دانش درباره فرآیند و کمک به بصری سازی آن
- ✓ کاهش داده کلی، محدود کردن نیازمندی‌های ذخیره سازی و احتمالا کمک به کاهش هزینه‌ها
- ✓ کاهش مجموعه ویژگی‌ها، ذخیره‌سازی منابع در دور بعدی گردآوری داده یا در طول بهره برداری
- ✓ سادگی و قابلیت استفاده از مدل‌های ساده‌تر و افزایش سرعت

به همه دلایل گفته شده، در سناریوهای تحلیل کلان داده انتخاب ویژگی نقشی اساسی ایفا می‌کند. در ادامه الگوریتم‌های انتخاب ویژگی که در این پروژه استفاده شده‌اند را توضیح خواهیم داد.

۲-۵-۱: انتخاب ویژگی به روش همبستگی (Correleation)

در این بخش، نحوه ارزیابی خوب بودن ویژگی‌ها برای طبقه‌بندی را مورد بحث قرار می‌دهیم. به طور کلی ، یک ویژگی زمانی خوب است که با مفهوم نتیجه نهایی مرتبط باشد اما با ویژگی‌های مرتبط دیگر همبستگی نداشته باشد . اگر همبستگی بین دو متغیر را به عنوان معیار خوب بودن در نظر بگیریم، تعریف فوق به‌این معنا است که یک ویژگی خوب است اگر با نتیجه نهایی همبستگی بالایی داشته باشد اما با هیچ یک از ویژگی‌های دیگر همبستگی بالایی نداشته باشد . به عبارت دیگر، اگر همبستگی بین یک ویژگی و نتیجه نهایی به اندازه‌ای بالا باشد که آن را به نتیجه نهایی مرتبط کند و همبستگی بین آن و سایر ویژگی‌های مرتبط به سطحی نرسد که بتوان آن را پیش‌بینی کرد، هر یک از ویژگی‌های مرتبط دیگر به عنوان یک ویژگی خوب برای کار طبقه‌بندی

[1] Pre-processing

درنظر گرفته می‌شود. از این روش انتخاب ویژگی به یافتن یک معیار مناسب از همبستگی بین ویژگی‌ها و یک روش صحیح برای انتخاب ویژگی‌ها بر اساس این معیار خلاصه می‌شود.

به طور کلی دو رویکرد برای اندازه‌گیری همبستگی بین دو متغیر تصادفی وجود دارد. یکی بر اساس همبستگی خطی کلاسیک[1] و دیگری مبتنی بر نظریه اطلاعات[2] است. تحت رویکرد اول، شناخته‌شده ترین معیار، ضریب همبستگی خطی است. برای یک جفت متغیر(X,Y) ضریب همبستگی خطی r با فرمول(۲-۵) به‌دست می‌آید:

$$r = \frac{\sum_i (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_i (x_i - \overline{x_i})^2}\sqrt{\sum_i (y_i - \overline{y_i})^2}}$$

فرمول(۲-۵): محاسبه همبستگی

که در آن $\overline{x_i}$ میانگین X و $\overline{y_i}$ میانگین Y است. مقدار r شامل ۱- و ۱ است. اگر X و Y کاملاً همبسته باشند، r مقدار ۱- یا ۱ می‌گیرد. اگر X و Y کاملاً مستقل[3] باشند، r صفر است.

این یک اندازه‌گیری متقارن برای دو متغیر است. معیارهای دیگر در این دسته اساساً تغییرات فرمول فوق هستند مانند خطای رگرسیون حداقل مربعات[4] و حداکثر شاخص فشرده‌سازی اطلاعات[5].

چندین مزیت از انتخاب همبستگی خطی به‌عنوان معیار خوبی برای طبقه‌بندی وجود دارد:

- ✓ به حذف ویژگی‌هایی با همبستگی خطی نزدیک به صفر با نتیجه نهایی کمک می کند.
- ✓ به کاهش افزونگی در میان ویژگی‌های انتخاب‌شده کمک می‌کند.

مشخص است که اگر داده‌ها به صورت خطی در نمایش اصلی قابل تفکیک باشند، اگر همه به جز یکی، از گروهی از ویژگی‌های خطی وابسته حذف شوند، همچنان به صورت خطی قابل تفکیک هستند. با این حال، فرض همبستگی خطی بین ویژگی‌ها در دنیای واقعی از لحاظ از دست دادن برخی اطلاعات امن نیست. اندازه‌گیری‌های همبستگی خطی ممکن است نتوانند همبستگی‌هایی را که ماهیت خطی ندارند، ثبت کنند. محدودیت دیگر این است که محاسبه مستلزم آن است که همه ویژگی‌ها دارای مقادیر عددی باشند .

در شکل(۲-۵) کد پیاده‌سازی این الگوریتم را مشاهده می‌کنید:

[1] Classical linear correlation
[2] Information theory
[3] Independent
[4] Least square regression error
[5] Maximum information compression index

```
#Correlation with output variable
cor_target = abs(cor["Diabetes_binary"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.15]
df_select_correlation = np.array(relevant_features.index)
df_correlation = df_test.copy()
df_correlation = df_test[df_select_correlation]
df_correlation.head()
```

شکل(۲-۵): الگوریتم Correlation

با توجه به مجموعه داده‌ای که داریم، ویژگی‌های جدول(۲-۱) از الگوریتم شکل(۲-۵) استخراج می‌شود:

	Diabetes_binary	HighBP	HighChol	BMI	GenHlth	DiffWalk	Age	Income
0	0.0	1.0	1.0	40.0	5.0	1.0	9.0	3.0
3	0.0	1.0	0.0	27.0	2.0	0.0	11.0	6.0
4	0.0	1.0	1.0	24.0	2.0	0.0	11.0	4.0
5	0.0	1.0	1.0	25.0	2.0	0.0	10.0	8.0
6	0.0	1.0	0.0	30.0	3.0	0.0	9.0	7.0
7	0.0	1.0	1.0	25.0	3.0	1.0	11.0	4.0
9	0.0	0.0	0.0	24.0	2.0	0.0	8.0	3.0
10	1.0	0.0	0.0	25.0	3.0	0.0	13.0	8.0
11	0.0	1.0	1.0	34.0	3.0	1.0	10.0	1.0
12	0.0	0.0	0.0	26.0	3.0	0.0	7.0	7.0

جدول(۲-۱): خروجی انتخاب ویژگی correlation

۲-۵-۲: انتخاب ویژگی به روش Variance Threshold

در یادگیری ماشین گرفتن واریانس به ما کمک می‌کند تا تشخیص دهیم رکوردهای[1] یک ویژگی به چه میزان پخش شده‌اند یا به عبارت دیگر رکوردهای یک مجموعه داده‌ای به چه میزان از میانگین فاصله دارند. واریانس از فرمول (۲-۶) محاسبه می‌شود:

[1] records

$$\sigma^2 = \frac{\sum(X-\mu)^2}{N} \rightarrow \quad \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2}$$

فرمول(۲-۶): محاسبه واریانس و انحراف معیار

این انتخاب ویژگی را می توان به عنوان یک روش کاهش واریانس در نظر گرفت که مزایای کاهش واریانس از کاهش ابعاد را با آسیب افزایش سوگیری از حذف برخی از ویژگی‌های مربوطه مبادله می‌کند. اگر از یک روش کاهش واریانس مانند طبقه‌بندی استفاده شود، میتوان از ویژگی‌های مرتبط (ضعیف) بیشتری استفاده کرد. این روش ویژگی‌هایی از مجموعه داده را انتخاب می‌کند که کمترین واریانس را با ویژگی نتیجه نهایی داشته باشند .

در شکل(۲-۳) کد پیاده‌سازی این الگوریتم را مشاهده می‌کنید:

```
df_threshold = dff.copy()
VarThreshOld = VarianceThreshold(threshold=0.04)
VarThreshOld.fit(df_threshold)
temp=VarThreshOld.get_support()
dff_columns = dff.columns
df_select_ThreshOld=[]
for index in range(0,len(temp)):
    if (temp[index]==True ):
        print(dff_columns[index])
        df_select_ThreshOld.append(dff_columns[index])

print(df_select_ThreshOld)
df2_threshold = df_threshold[df_select_ThreshOld]
df2_threshold.head()
```

شکل(۲-۶): الگوریتم Variance threshold

با توجه به مجموعه داده‌ای که داریم، ویژگی‌های جدول(۲-۲) از الگوریتم شکل(۲-۶) استخراج می‌شود:

	Diabetes_binary	HighBP	HighChol	BMI	GenHlth	DiffWalk	Age	Income
0	0.0	1.0	1.0	40.0	5.0	1.0	9.0	3.0
3	0.0	1.0	0.0	27.0	2.0	0.0	11.0	6.0
4	0.0	1.0	1.0	24.0	2.0	0.0	11.0	4.0
5	0.0	1.0	1.0	25.0	2.0	0.0	10.0	8.0
6	0.0	1.0	0.0	30.0	3.0	0.0	9.0	7.0
7	0.0	1.0	1.0	25.0	3.0	1.0	11.0	4.0
9	0.0	0.0	0.0	24.0	2.0	0.0	8.0	3.0
10	1.0	0.0	0.0	25.0	3.0	0.0	13.0	8.0
11	0.0	1.0	1.0	34.0	3.0	1.0	10.0	1.0
12	0.0	0.0	0.0	26.0	3.0	0.0	7.0	7.0

جدول(۲-۲): خروجی انتخاب ویژگی Variance threshold

۲-۵-۳: انتخاب ویژگی به روش حذف ویژگی بازگشتی (RFE) [1]

حذف ویژگی بازگشتی اساساً یک انتخاب معکوس از پیش‌بینی‌کننده‌ها است. این تکنیک با ساختن یک مدل بر روی کل مجموعه پیش‌بینی‌کننده‌ها و محاسبه امتیاز اهمیت برای هر پیش‌بینی آغاز می‌شود. سپس کمترین پیش‌بینی‌کننده(های) مهم حذف می‌شوند، مدل دوباره ساخته می‌شود و امتیازهای اهمیت دوباره محاسبه می شوند. در عمل، تحلیلگر تعداد زیر مجموعه‌های پیش‌بینی‌کننده را برای ارزیابی و همچنین اندازه هر زیرمجموعه را مشخص می‌کند. بنابراین اندازه زیرمجموعه یک پارامتر تنظیم برای RFE است. اندازه زیرمجموعه‌ای که معیارهای عملکرد را بهینه می‌کند برای انتخاب پیش‌بینی‌کننده‌ها بر اساس رتبه‌بندی اهمیت استفاده می‌شود و سپس از زیرمجموعه بهینه برای آموزش مدل نهایی استفاده می‌شود. فرآیند نمونه‌گیری مجدد شامل روال انتخاب ویژگی است و نمونه‌های مجدد خارجی برای تخمین اندازه زیر مجموعه مناسب استفاده می‌شوند.

همه مدل‌ها را نمی‌توان با روش RFE جفت کرد و برخی از مدل‌ها بیشتر از بقیه از RFE بهره می‌برند. از آن‌جایی که RFE مستلزم آن است که مدل اولیه از مجموعه پیش‌بینی‌کننده کامل استفاده کند، پس برخی از مدل‌ها زمانی که تعداد پیش‌بینی‌کننده‌ها از تعداد نمونه‌ها بیشتر شود، قابل استفاده نیستند.

اگر مایل به استفاده از یکی از این تکنیک‌ها با RFE هستید، ابتدا باید پیش‌بینی‌کننده‌ها را شناسایی کنید . علاوه بر این، برخی از مدل‌ها از استفاده از RFE بیشتر از بقیه بهره می‌برند. جنگل تصادفی یکی از این مدل‌ها است.

در شکل(۲-۴) کد پیاده‌سازی این الگوریتم را مشاهده می‌کنید:

[1] Recursive Feature Elimination

```
lin_reg = LinearRegression()

rfe_mod = RFECV(lin_reg,cv=10)
myvalues=rfe_mod.fit(X_train_minmax,y_train_minmax)
myvalues.support_
myvalues.ranking_

print("Num Features: %s" % (myvalues.n_features_))
print("Selected Features: %s" % (myvalues.support_))
print("Feature Ranking: %s" % (myvalues.ranking_))
df_select_rfe = []
for i in range(0,len(myvalues.support_)):
    if(myvalues.support_[i]==True):
        df_select_rfe.append(df_minmax_columns[i])
df_select_rfe.append('target')
df_rfe = df_minmax.copy()
df_rfe = df_rfe[df_select_rfe]
df_rfe.head()
```

شکل(۲-۷): الگوریتم RFE

با توجه به مجموعه داده‌ای که داریم، ویژگی‌های جدول(۲-۳) از الگوریتم شکل(۲-۷) استخراج می شود:

	Diabetes_binary	HighBP	HighChol	BMI	Smoker	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
0	0.0	1.0	1.0	40.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	5.0	18.0	15.0	1.0	0.0	9.0	4.0	3.0
3	0.0	1.0	0.0	27.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	11.0	3.0	6.0
4	0.0	1.0	1.0	24.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	2.0	3.0	0.0	0.0	0.0	11.0	5.0	4.0
5	0.0	1.0	1.0	25.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	2.0	0.0	2.0	0.0	1.0	10.0	6.0	8.0
6	0.0	1.0	0.0	30.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0	14.0	0.0	0.0	9.0	6.0	7.0
7	0.0	1.0	1.0	25.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	3.0	0.0	0.0	1.0	0.0	11.0	4.0	4.0
9	0.0	0.0	0.0	24.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	2.0	0.0	0.0	0.0	1.0	8.0	4.0	3.0
10	1.0	0.0	0.0	25.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	3.0	0.0	0.0	0.0	1.0	13.0	6.0	8.0
11	0.0	1.0	1.0	34.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	3.0	0.0	30.0	1.0	0.0	10.0	5.0	1.0
12	0.0	0.0	0.0	26.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	3.0	0.0	15.0	0.0	0.0	7.0	5.0	7.0

جدول(۲-۳): خروجی انتخاب ویژگی RFE

۲-۵-۴: انتخاب ویژگی به روش روبه‌جلو یا پیشرو (Forward)

انتخاب رو به جلو یک روش کار مکرر و تکراری است که در ابتدا باید تعداد ویژگی‌هایی که می‌خواهیم در پایان اجرای الگوریتم داشته باشیم را به عنوان ورودی به الگوریتم پیشرو بدهیم. الگوریتم با نداشتن ویژگی در مدل شروع می‌کند و در هر تکرار، ویژگی‌هایی را اضافه می کند که مدل ما را به بهترین شکل بهبود می‌بخشد تا زمانی

که اضافه شدن یک ویژگی جدید عملکرد مدل را بهبود نبخشد و در نهایت ویژگی‌های انتخاب شده را به عنوان خروجی به ما می‌دهد.

در شکل(۲-۸) کد پیاده‌سازی این الگوریتم را مشاهده می‌کنید:

```
X = dfc[['HighBP', 'HighChol', 'CholCheck', 'BMI', 'Smoker', 'Stroke', 'HeartDiseaseorAttack','P
Y = dfc[['Diabetes_binary']]
X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size=0.2, random_state=4)
knn = KNeighborsClassifier(n_neighbors=5)
sfs = SequentialFeatureSelector(knn, n_features_to_select=7)
sfs.fit(np.asarray(X_train), y_train.values.ravel())
rr = sfs.get_support()
df_col = dfc.columns
df_select_forward = []
for i in range(0,len(rr)):
    if(rr[i]==True):
        df_select_forward.append(df_col[i])
df_select_forward.append('Diabetes_binary')
df_forward = dfc.copy()
df_forward = df_forward[df_select_forward]
df_forward.head(20)
```

شکل(۲-۸): کد پیاده‌سازی انتخاب ویژگی forward

با توجه به مجموعه داده‌ای که داریم، ویژگی‌های جدول(۲-۴) از الگوریتم شکل(۲-۸) استخراج می شود:

	HighBP	CholCheck	BMI	Stroke	HvyAlcoholConsump	AnyHealthcare	Education	Diabetes_binary
0	1.0	1.0	40.0	0.0	0.0	1.0	4.0	0.0
1	1.0	1.0	27.0	0.0	0.0	1.0	3.0	0.0
2	1.0	1.0	24.0	0.0	0.0	1.0	5.0	0.0
3	1.0	1.0	25.0	0.0	0.0	1.0	6.0	0.0
4	1.0	1.0	30.0	0.0	0.0	1.0	6.0	0.0
5	1.0	1.0	25.0	0.0	0.0	1.0	4.0	0.0
6	0.0	1.0	24.0	0.0	0.0	1.0	4.0	0.0
7	0.0	1.0	25.0	0.0	0.0	1.0	6.0	1.0
8	1.0	1.0	34.0	0.0	0.0	1.0	5.0	0.0
9	0.0	1.0	26.0	0.0	0.0	1.0	5.0	0.0
10	1.0	1.0	28.0	0.0	0.0	1.0	4.0	1.0

جدول(۲-۴): مجموعه داده با انتخاب ویژگی forward

۲-۵-۵: انتخاب ویژگی به روش رو به عقب یا عقبگرد (Backward)

روش عقبگرد نیز مانند روش پیشرو یک روش کار مکرر و تکراری است و برعکس روش پیشرو عمل می‌کند. در ابتدا باید تعداد ویژگی‌هایی که می‌خواهیم در پایان اجرای الگوریتم داشته باشیم را به عنوان ورودی به الگوریتم

عقبگرد بدهیم. در انتخاب ویژگی به روش عقبگرد، با شروع از مجموعه اولیه ویژگی‌ها، به طور موقت هر ویژگی را حذف می‌کنیم و ارزش معیار انتخاب را محاسبه می‌کنیم و ویژگی با بالاترین مقدار معیار انتخاب را از مجموعه حذف می‌کنیم و این حذف را آنقدر ادامه می‌دهیم تا تعداد ویژگی‌هایی موجود باشد که به عنوان ورودی داده بودیم و همچنین بهترین ویژگی‌های انتخاب شده از لحاظ عملکرد باشند.

در شکل(۲-۹) کد پیاده‌سازی این الگوریتم را مشاهده می‌کنید:

```
X = dfc[['HighBP', 'HighChol', 'CholCheck', 'BMI', 'Smoker', 'Stroke', 'HeartDisease
Y = dfc[['Diabetes_binary']]
X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size=0.2, random_sta
knn = KNeighborsClassifier(n_neighbors=5)
sfs = SequentialFeatureSelector(knn, n_features_to_select=7, direction = "backward")
sfs.fit(np.asarray(X_train), y_train.values.ravel())
rr = sfs.get_support()
df_col = dfc.columns
df_select_backward = []
for i in range(0,len(rr)):
    if(rr[i]==True):
        df_select_backward.append(df_col[i])
df_select_backward.append('Diabetes_binary')
df_backward = dfc.copy()
df_backward = df_backward[df_select_backward]
df_backward.head(20)
```

شکل(۲-۹): کد پیاده‌سازی انتخاب ویژگی backward

با توجه به مجموعه داده‌ای که داریم، ویژگی‌های جدول(۲-۵) از الگوریتم شکل(۲-۹) استخراج می‌شود:

	BMI	GenHlth	MentHlth	PhysHlth	Age	Education	Income	Diabetes_binary
0	40.0	5.0	18.0	15.0	9.0	4.0	3.0	0.0
1	27.0	2.0	0.0	0.0	11.0	3.0	6.0	0.0
2	24.0	2.0	3.0	0.0	11.0	5.0	4.0	0.0
3	25.0	2.0	0.0	2.0	10.0	6.0	8.0	0.0
4	30.0	3.0	0.0	14.0	9.0	6.0	7.0	0.0
5	25.0	3.0	0.0	0.0	11.0	4.0	4.0	0.0
6	24.0	2.0	0.0	0.0	8.0	4.0	3.0	0.0
7	25.0	3.0	0.0	0.0	13.0	6.0	8.0	1.0
8	34.0	3.0	0.0	30.0	10.0	5.0	1.0	0.0
9	26.0	3.0	0.0	15.0	7.0	5.0	7.0	0.0
10	28.0	4.0	0.0	0.0	11.0	4.0	6.0	1.0

جدول(۲-۵): مجموعه داده با انتخاب ویژگی backward

۲-۶: روش های ارزیابی

در ادامه الگوریتم‌های یادگیری ماشین که در این پایان نامه استفاده شده اند را بررسی خواهیم کرد.

۲-۶-۱: الگوریتم Naïve Bayes

در این الگوریتم از فرمول بیز برای پیش‌بینی احتمالات طبقه‌بندی استفاده می‌شود. در ریاضی بیز از فرمول(۲-۷) محاسبه می‌شود.

$$Pr[Class|Predictors] = \frac{Pr[Class] \times Pr[Predictors|Class]}{Pr[Predictors]} = \frac{Prior \times Likelihood}{Evidence}$$

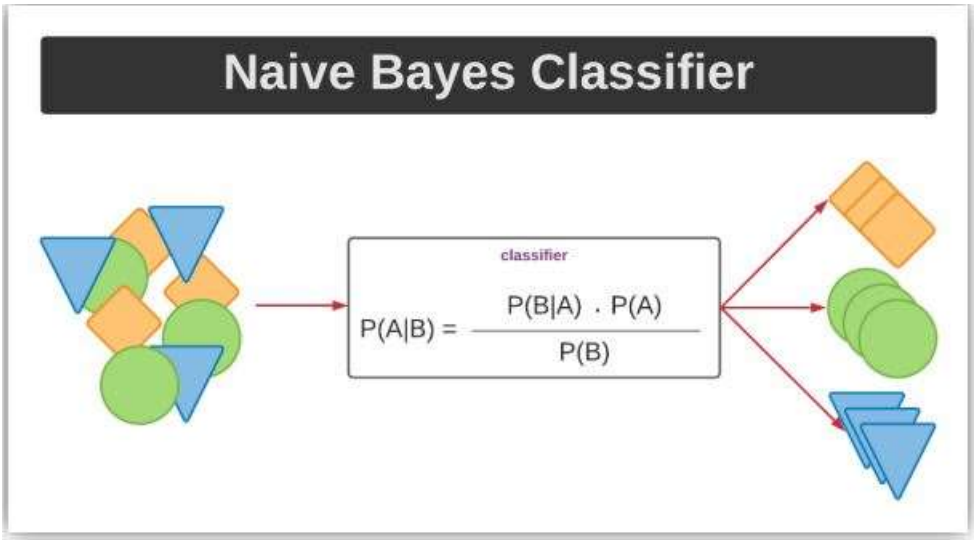
فرمول(۲-۷): محاسبه احتمال بیز

سه عنصر در این معادله وجود دارد:

۱. Pr احتمال کلی هر دسته است که می‌تواند از داده‌های آموزشی تخمین زده شود یا با استفاده از دانش متخصصین تعیین شود.
۲. Likelihood احتمال نسبی داده‌های پیش‌بینی مشاهده شده برای هر دسته را اندازه می‌گیرد.
۳. Evidence یک عامل عادی‌سازی است که می‌تواند از طریق pr و Likelihood اندازه‌گیری شود.

بیشتر محاسبات هنگام تعیین احتمال انجام می‌شود. به عنوان مثال، اگر چندین پیش‌بینی عددی وجود داشته باشد، می‌توان از توزیع احتمال چند متغیره برای محاسبات استفاده کرد. با این حال، محاسبه خارج از توزیع نرمال دو متغیره بسیار دشوار است یا ممکن است به داده‌های مجموعه آموزشی فراوانی نیاز داشته باشد. زمانی که ویژگی ها ترکیبی از مقادیر عددی و پیوسته باشند، تعیین احتمال پیچیده‌تر می‌شود. جنبه مثبت این مدل به دلیل یک فرض بسیار دقیق است که پیش‌بینی‌کننده‌ها مستقل فرض می‌شوند (اگرچه استقلال به طور کلی یک فرض ضعیف است)، این امکان را می‌دهد احتمال مشترک به عنوان محصولی از مقادیر خاص طبقه‌بندی محاسبه شود. در عمل Naïve Bayes اغلب به خوبی با طبقه‌بندی‌کننده‌های پیچیده‌تر رقابت می‌کند.

شکل(۲-۸) Naïve Bayes را نشان می دهد:



شکل(۲-۹): Naïve Bayes

۲-۶-۲: الگوریتم ماشین‌های بردار پشتیبان (SVM) [1]

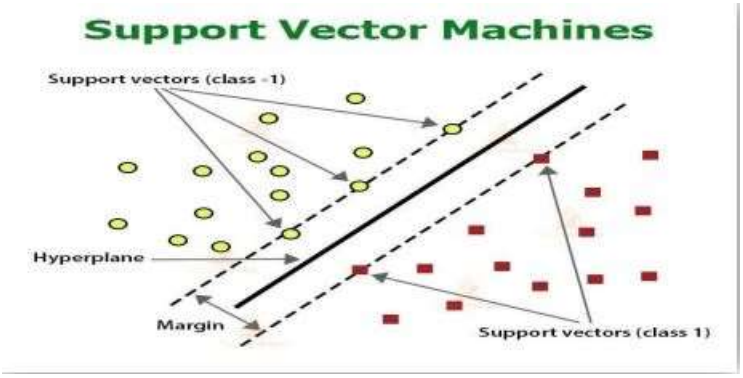
یکی از رایج‌ترین و هیجان‌انگیزترین مدل‌های یادگیری نظارت‌شده با الگوریتم‌های یادگیری مرتبط که داده‌ها را تجزیه و تحلیل می‌کند و الگوها را تشخیص می‌دهد، ماشین‌های بردار پشتیبان است. آن‌ها برای حل مسائل رگرسیون و طبقه‌بندی استفاده می‌شوند با این حال، آن‌ها بیشتر در حل مسائل طبقه‌بندی استفاده می‌شوند. ماشین‌های بردار پشتیبان اولین بار در سال ۱۹۹۲ به صورت آزمایشگاهی استفاده شدند و سپس به دلیل موفقیت در تشخیص ارقام دست نویس در سال ۱۹۹۴ محبوب شدند . الگوریتم ماشین‌های بردار پشتیبان قدرتمند است اما مفاهیم پشت آن آنقدرها پیچیده نیستند. یک الگوریتم ماشینهای بردار پشتیبان مدلی را ایجاد می‌کند که

[1] Support Vector Machines

نمونه های جدیدی را به یک دسته یا دسته دیگر اختصاص می‌دهد و آن را به یک طبقه‌بندی خطی باینری غیراحتمالی[1] تبدیل می‌کند.

هدف از به‌کارگیری ماشین های بردار پشتیبان یافتن بهترین خط در دو بعد یا بهترین ابرصفحه یا ابر جداکننده[2] در بیش از دو بعد است تا به ما کمک کند فضای خود را به کلاس‌ها تفکیک کنیم. ابر صفحه (خط) از طریق بیشترین حاشیه[3] یعنی حداکثر فاصله بین نقاط داده هر دو کلاس پیدا می‌شود. اگر نقاط داده جدیدی را اضافه کنیم، نتیجه استفاده از ابرصفحه‌های مختلف از نظر طبقه‌بندی نقطه داده جدید در کلاس مناسب، بسیار متفاوت خواهد بود.

شکل(۲-۹) ماشین‌های بردار پشتیبان را نشان می‌دهد:



شکل(۲-۱۰): ماشین‌های بردار پشتیبان

۲-۶-۳: الگوریتم Logistic Regression

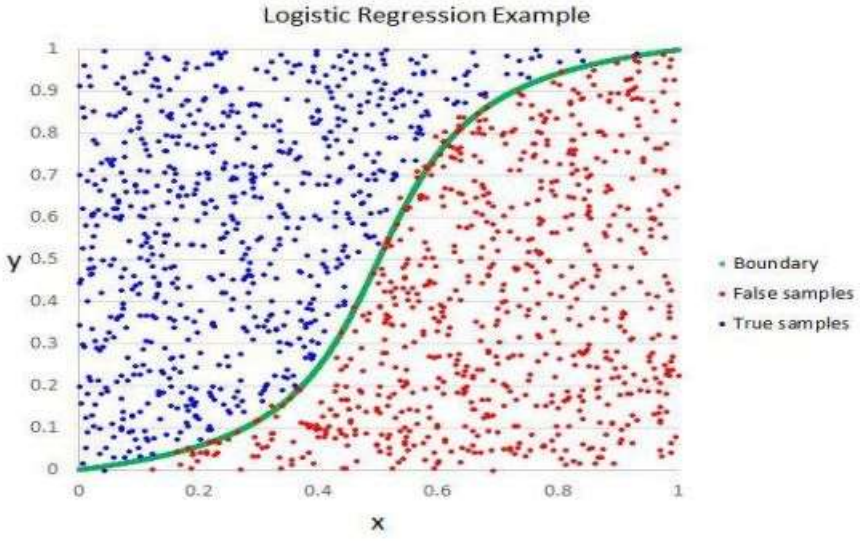
رگرسیون لجستیک تکنیک دیگری است که توسط یادگیری ماشین از حوزه آمار گرفته شده است. این روش برای مسائل طبقه‌بندی باینری[4] (مسائل با دو مقدار کلاس) است. رگرسیون لجستیک یکی از محبوب‌ترین الگوریتم های یادگیری ماشین است که تحت تکنیک یادگیری نظارتی قرار می‌گیرد. برای پیش‌بینی متغیر وابسته طبقه بندی با استفاده از مجموعه معینی از متغیرهای مستقل استفاده می شود.

تابع لجستیک، که تابع سیگموئید[5] نیز نامیده می‌شود، توسط آماردانان برای توصیف ویژگی‌های رشد جمعیت در بوم‌شناسی، افزایش سریع و به حداکثر رساندن ظرفیت تحمل محیط ایجاد شد . رگرسیون لجستیک خروجی یک متغیر وابسته قطعی را پیش‌بینی می‌کند. بنابراین نتیجه باید یک مقدار قطعی یا گسسته باشد. می‌تواند بله یا

[1] Nonprobabilistic binary linear classifier
[2] hyperplane
[3] Maximum margin
[4] binary
[5] sigmoid

خیر، ۰ یا ۱ ،درست یا نادرست و غیره باشد اما به جای دادن مقدار دقیق ۰ و ۱ ، مقادیر احتمالی را می‌دهد که بین ۰ و ۱ قرار دارند. رگرسیون لجستیک بسیار شبیه به رگرسیون خطی است و تفاوت آنها در نحوه استفاده است. از رگرسیون خطی برای حل مسائل رگرسیونی (مسائلی که خروجی آنها مقداری پیوسته است) استفاده می شود، درحالیکه از رگرسیون لجستیک برای حل مسائل طبقه‌بندی استفاده می‌شود. برای مثال منحنی تابع لجستیک احتمال مواردی مانند سرطانی بودن یا نبودن سلول‌ها، بیماری دیابت داشتن یا نداشتن، چاق بودن یا نبودن موش بر اساس وزن و غیره را نشان می‌دهد.

رگرسیون لجستیک یک الگوریتم یادگیری ماشین قابل توجه است زیرا توانایی ارائه احتمالات و طبقه‌بندی داده‌های جدید با استفاده از مجموعه داده‌های پیوسته و گسسته را دارد و همچنین می‌تواند برای طبقه‌بندی مشاهدات با استفاده از انواع مختلف داده‌ها استفاده شود و به راحتی می‌تواند مؤثرترین متغیرهای مورد استفاده برای طبقه‌بندی را تعیین کند. شکل(۲-۱۰) تابع لجستیک را نشان می‌دهد.



شکل(۲-۱۱): نحوه کار رگرسیون لجستیک

تابع سیگموئید یک تابع ریاضی است که برای ترسیم مقادیر پیش‌بینی شده به احتمالات استفاده می‌شود و هر مقدار واقعی را به مقدار دیگری در محدوده ۰ و ۱ نگاشت می‌کند. مقدار رگرسیون لجستیک باید بین ۰ و ۱ باشد و نمی‌تواند از این حد فراتر رود، بنابراین منحنی مانند فرم S را تشکیل می‌دهد. منحنی شکل S را تابع سیگموئید یا تابع لجستیک می‌نامند. در رگرسیون لجستیک، از مفهوم مقدار آستانه استفاده می‌کنیم که احتمال ۰ یا ۱ را تعریف می‌کند. مقادیر بالای مقدار آستانه به ۱ میل می‌کنند و مقادیر زیر مقدار آستانه به ۰ میل می‌کنند. معادله رگرسیون لجستیک را می‌توان از معادله رگرسیون خطی به دست آورد. مراحل ریاضی برای به‌دست‌آوردن معادلات رگرسیون لجستیک در زیر آورده شده‌است.

می‌دانیم که معادله خط مستقیم را می‌توان به‌صورت فرمول(۲-۸) نوشت:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \ldots + b_nx_n$$

فرمول(۲-۸): نرمال‌سازی معادله خط مستقیم

در رگرسیون لجستیک y فقط می‌تواند بین ۰ و ۱ باشد، بنابراین معادله فوق را بر (۱-y) تقسیم می‌کنیم مطابق با فرمول(۲-۹):

$$\frac{y}{1-y}; 0\ for\ y = 0\ ; and\ infinity\ for\ y = 1$$

فرمول(۲-۹): نرمال‌سازی معادله خط مستقیم

اما به محدوده $(-\infty , +\infty)$ نیاز داریم. پس از معادله لگاریتم می‌گیریم که فرمول (۲-۱۰) به‌دست می‌آید:

$$\log\left(\frac{y}{1-y}\right) = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \ldots + b_nx_n$$

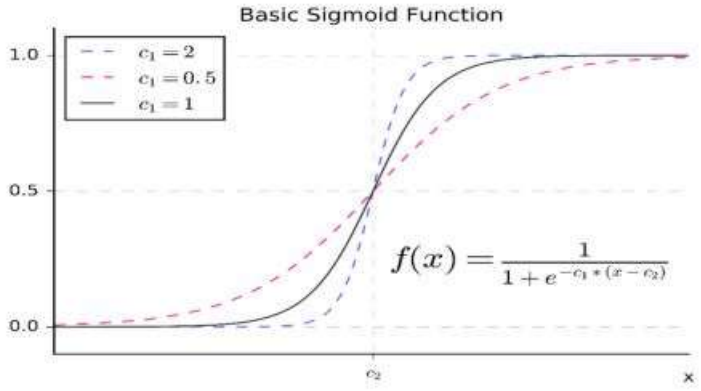
فرمول(۲-۱۰): معادله رگرسیون لجستیک

معادله فوق معادله نهایی رگرسیون لجستیک است.

بر اساس دسته‌بندی‌ها، رگرسیون لجستیک را می‌توان به سه نوع طبقه‌بندی کرد:

۱. دو جمله‌ای: در رگرسیون لجستیک دوجمله‌ای، تنها دو نوع متغیر وابسته ممکن است وجود داشته باشد مانند ۰ یا ۱ ، پیروزی یا شکست و ... .
۲. چند جمله‌ای: در رگرسیون لجستیک چند جمله‌ای ، می‌تواند ۳ یا چند نوع نامرتب متغیر وابسته وجود داشته باشد، مانند گربه، سگ یا گوسفند.
۳. ترتیبی: در رگرسیون لجستیک ترتیبی، می تواند ۳ یا چند نوع متغیر وابسته مرتب شده مانند کم، متوسط یا زیاد وجود داشته باشد.

شکل(۲-۱۲) تابع سیگوئید را نشان می‌دهد:



شکل(۲-۱۲): تابع **sigmoid**

۲-۶-۴: الگوریتم k– نزدیک‌ترین همسایه (KNN)

الگوریتم k– نزدیک‌ترین همسایه یکی از ساده‌ترین الگوریتم‌های یادگیری ماشین بر اساس تکنیک یادگیری نظارت‌شده است. الگوریتم k– نزدیک‌ترین همسایه شباهت بین داده جدید و موارد موجود اطراف را فرض می‌کند و مورد جدید را در دسته‌ای قرار می دهد که بیشترین شباهت را به دسته‌های موجود مجاور دارد.

این الگوریتم تمام داده‌های موجود را ذخیره می‌کند و یک نقطه داده جدید را بر اساس شباهت طبقه‌بندی می کند. این بدان معنی است که وقتی داده‌های جدید ظاهر می‌شوند، می‌تواند آن‌ها را به راحتی در یک دسته مناسب قرار دهد . ما باید به عنوان ورودی عدد k که تعداد نزدیک‌ترین همسایه‌هایی است که این الگوریتم برای طبقه بندی یک داده‌ی جدید بررسی می‌کند را به الگوریتم بدهیم تا اجرا شود.

این الگوریتم را می‌توان برای رگرسیون و همچنین برای طبقه‌بندی استفاده کرد اما بیشتر برای مسائل طبقه‌بندی استفاده می‌شود. الگوریتمk– نزدیک‌ترین همسایه یک الگوریتم ناپارامتریک است، به این معنی که هیچ فرضی در مورد داده‌های اساسی ایجاد نمی‌کند. به آن الگوریتم یادگیرنده تنبل[1] نیز می‌گویند زیرا بلافاصله از مجموعه آموزشی یاد نمی‌گیرد، در عوض مجموعه داده را ذخیره می‌کند و در زمان طبقه‌بندی، عملی را روی مجموعه داده انجام می‌دهد.

الگوریتم k– نزدیک‌ترین همسایه در مرحله آموزش فقط مجموعه داده را ذخیره می‌کند و هنگامی که داده‌های جدیدی دریافت می‌کند، آن داده‌ها را در دسته‌ای طبقه‌بندی می‌کند که بسیار شبیه به داده‌های جدید است.

برای مثال فرض کنید تصویری از موجودی داریم که شبیه گربه و سگ است، اما می‌خواهیم بدانیم که گربه است یا سگ. بنابراین برای این شناسایی می‌توانیم از الگوریتم-k نزدیک‌ترین همسایه استفاده کنیم، زیرا بر روی معیار

[1] Lazy learner

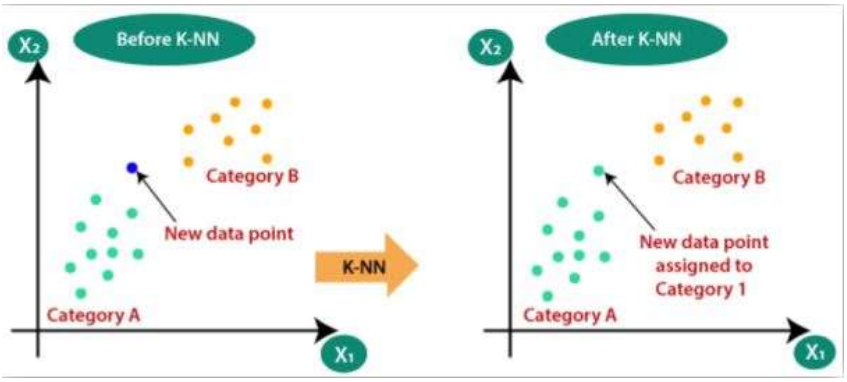
شباهت کار می‌کند . مدل *KNN* ما ویژگی‌های مشابه مجموعه داده‌های جدید را با تصاویر گربه‌ها و سگ‌ها پیدا می‌کند و بر اساس مشابه‌ترین ویژگی‌ها، آن را در دسته‌بندی گربه یا سگ قرار می‌دهد که در شکل(۲-۱۳) نشان داده شده‌است:

شکل(۲-۱۳): KNN

سوال: چرا به الگوریتم k– نزدیک‌ترین همسایه نیاز داریم؟

فرض کنید دو دسته A و B وجود دارند و ما یک نقطه داده جدید X_1 داریم، بنابراین این نقطه داده در کدام یک از این دسته‌ها قرار می‌گیرد؟

برای حل این نوع مسائل به یک الگوریتم KNN نیاز داریم. با کمک KNN، ما به راحتی می‌توانیم دسته یا کلاس یک مجموعه داده خاص را شناسایی کنیم. شکل(۲-۱۴) را در نظر بگیرید:



شکل(۲-۱۴): مثال از KNN

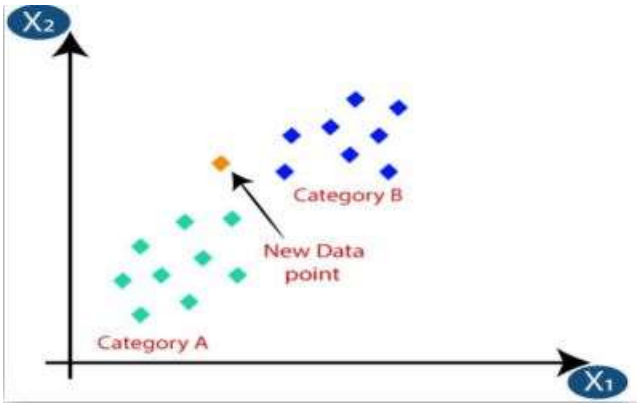
مراحل کار این الگوریتم به صورت زیر است:

۱) عدد k را انتخاب کنید.
۲) فاصله اقلیدسی این k تا همسایه را محاسبه می‌کند.
۳) K تا از نزدیک‌ترین همسایگان را بر اساس فاصله اقلیدسی محاسبه‌شده به‌ترتیب صعودی در نظر می‌گیرد.
۴) در بین این k همسایه، شمارش می‌کند که از هر دسته چندتا داریم.

۵) نقاط داده جدید را به دسته‌ای که تعداد همسایه برای آن بیشتر است، اختصاص می‌دهد.

۶) مدل ما آماده است.

فرض کنید یک نقطه داده جدید داریم و باید آن را در دسته‌بندی مورد نیاز قرار دهیم. شکل(۲-۱۵) را در نظر بگیرید:



شکل(۲-۱۵): نقطه داده‌ای جدید

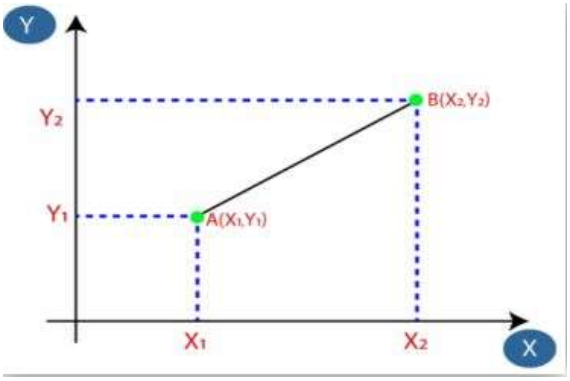
ابتدا تعداد همسایگان را انتخاب می‌کنیم، بنابراین k را مثلا برابر ۵ انتخاب می‌کنیم.

در مرحله بعد، فاصله اقلیدسی بین نقاط داده را محاسبه خواهیم کرد. می‌توان آن را با فرمول(۲-۱۱) محاسبه کرد:

$$distance\ A, B = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

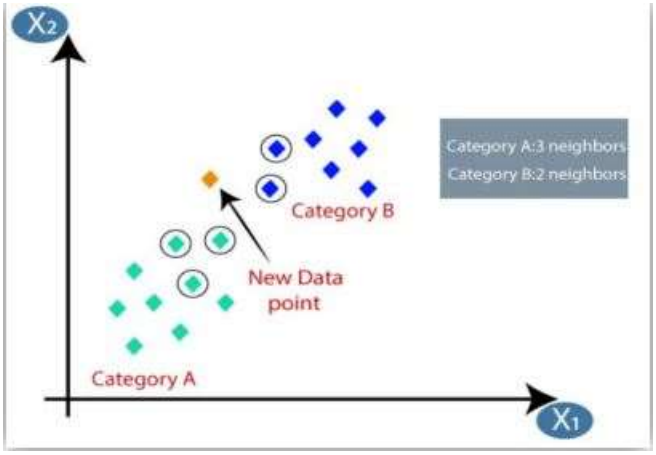
فرمول(۲-۱۱): محاسبه فاصله دو نقطه

شکل(۲-۱۶) فاصله اقلیدسی دو نقطه را نشان می‌دهد:



شکل(۲-۱۶): فاصله بین دو نقطه A و B

با محاسبه فاصله اقلیدسی نزدیک ترین همسایگان را به عنوان سه همسایه نزدیک در رده A و دو نزدیک ترین همسایه در دسته B به دست آوردیم. شکل(۲-۱۷) را در نظر بگیرید:



شکل(۲-۱۷): همسایه‌های نقطه داده‌ای جدید

همان‌طور که می‌بینیم ۳ نزدیک‌ترین همسایه از دسته A هستند، بنابراین این نقطه داده جدید باید به دسته A تعلق داشته باشد.

مزایا:

- ✓ اجرای آن ساده است.
- ✓ در برابر داده‌های آموزشی با نویز زیاد، قوی است.
- ✓ اگر داده‌های آموزشی زیاد باشد می‌تواند موثرتر باشد.

معایب:

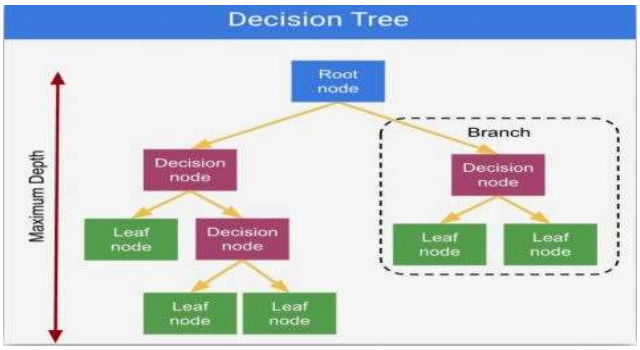
- ✓ همیشه نیاز به تعیین مقدار k دارد که ممکن است اکثر مواقع پیچیده باشد.
- ✓ هزینه محاسبات به دلیل فاصله بین نقاط داده برای همه نمونه‌های آموزشی بالا است.

۲-۶-۵: الگوریتم درخت تصمیم (Decision tree)

درخت تصمیم دقیقاً مانند یک درخت است با این تفاوت که از ریشه به سمت پایین(برگ) رشد کرده است. در الگوریتم درخت تصمیم نمونه‌ها را دسته‌بندی می‌کنیم که درواقع دسته‌ها در انتهای گره‌های برگ قرار دارد . درخت تصمیم در مسائلی کاربرد دارد که بتوان آن‌ها را به‌صورتی مطرح کرد که پاسخ واحدی به‌صورت نام یک دسته یا کلاس ارائه دهند . روش‌های ساخت درخت تصمیم معمولا به صورت بالا به پایین عمل می‌کنند به این

معنی که ابتدا فضای ورودی به فضاهای کوچک‌تر تقسیم می‌شود، سپس فرآیند تقسیم‌بندی برای هر یک از این قسمت‌ها تکرار می‌شود؛ ابتدا ریشه ساخته می‌شود، سپس هر یک از زیرشاخه‌ها به شاخه‌های دیگری تقسیم می شود و این فرآیند تکرار می‌شود. یک گره ریشه در بالای آن قرار دارد و برگ‌های آن در پایین هستند . یک رکورد در گره ریشه وارد می‌شود و در این گره یک آزمایش صورت می‌گیرد تا مشخص شود که این رکورد به کدام یک از گره‌های فرزند خواهد رفت. درخت تصمیم از تعدادی گره و شاخه تشکیل شده‌است که در آن نمونه‌ها را به نحوی دسته‌بندی می‌کند که از ریشه به سمت پایین رشد می‌کند و درنهایت به گره‌های برگ می‌رسد. هر گره داخلی یا غیر برگ با یک ویژگی مشخص می‌شود. این ویژگی سؤالی را در رابطه با مثال ورودی مطرح می‌کند. در هر گره داخلی به تعداد جواب‌های ممکن با این سؤال شاخه وجود دارد که هریک با مقدار آن جواب مشخص می شوند. برگ‌های این درخت با یک کلاس و یا یک طبقه از جواب‌ها مشخص میشوند.

شکل(۲-۱۷) درخت تصمیم را نشان می‌دهد:

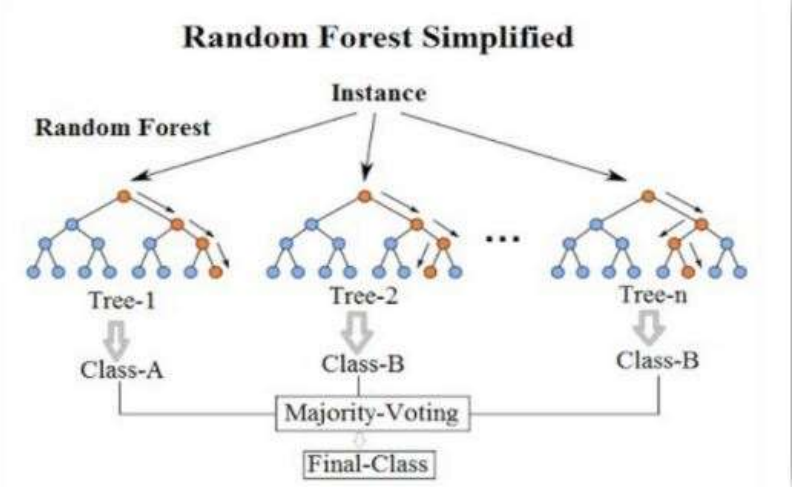


شکل(۲-۱۷): درخت تصمیم

۲-۶-۶: الگوریتم جنگل تصادفی (Random forest)

جنگل تصادفی همانطور که از نامش پیداست، از تعداد زیادی درخت تصمیم‌گیری فردی تشکیل شده‌است که به عنوان یک مجموعه عمل می‌کنند. هر درخت منفرد در جنگل تصادفی یک پیش‌بینی کلاس را منتشر می‌کند و کلاسی که بیشترین رأی را دارد، پیش‌بینی مدل ما می‌شود. شکل زیر ساختار کلی از جنگل تصادفی را نشان می دهد که درنهایت بین پیش‌بینی‌های انجام شده، رأی‌گیری انجام می‌دهد .

شکل (۲-۱۹) درخت تصمیم را نشان می‌دهد:



شکل(۲-۱۹): جنگل تصادفی

۲-۶-۷: الگوریتم شبکه‌های عصبی پرسپترون چند لایه (MLP) [1]

پرسپترون یک الگوریتم یادگیری ماشین است که در دسته الگوریتم‌های یادگیری با نظارت قرار می‌گیرد. الگوریتم پرسپترون یکی از الگوریتم‌های دسته‌بندی باینری (دودویی) محسوب می‌شود و این به معنای این است که الگوریتم پرسپترون امکان این را دارد که تعدادی عضو را دسته‌بندی کند و مشخص کند یک عضو متعلق به یک گروه است یا خیر. الگوریتم پرسپترون را به دلیل این که عملیات شناسایی را به صورت ترتیبی و یک به یک انجام می‌دهد، الگوریتم خطی می‌دانند. شبکه عصبی پرسپترون از جمله ساده‌ترین معماری‌های شبکه عصبی مصنوعی است.

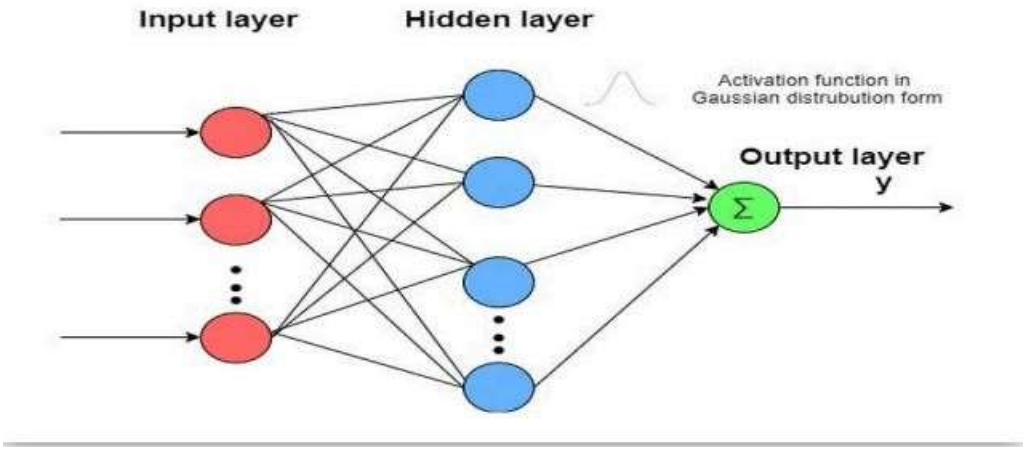
پرسپترون چند لایه(MLP) یک شبکه عصبی مصنوعی عمیق است که از بیش از یک پرسپترون تشکیل شده است .آن‌ها از یک لایه ورودی برای دریافت سیگنال تشکیل شده‌اند، یک لایه خروجی که در مورد ورودی تصمیم می‌گیرد یا پیش‌بینی می‌کند، و در بین این دو، تعدادی دلخواه از لایه‌های پنهان که موتور محاسباتی واقعی MLP هستند قرار دارد. MLP ها با یک لایه مخفی قادر به تقریب هر عملکرد پیوسته‌ای هستند.

شبکه عصبی پرسپترون چند لایه اغلب برای مشکلات یادگیری تحت نظارت به کار گرفته می‌شوند. آن‌ها بر روی مجموعه‌ای از جفت‌های ورودی و خروجی آموزش می‌بینند و یاد می‌گیرند که همبستگی (یا وابستگی‌ها) بین ورودی‌ها و خروجی‌ها را مدل کنند ؛ مرحله آموزش شامل تنظیم پارامترها، یا وزن‌دهی و تنظیم بایاس‌های مدل به منظور به حداقل رساندن خطا است.

[1] Multilayer Preceptron neural networks

از الگوریتم Backpropagation برای تعدیل وزن و میزان بایاس نسبت به خطا استفاده می‌شود و خود خطا را می‌توان به‌روش‌های مختلفی از جمله خطای مربع میانگین ریشه (RMSE)[1] اندازه‌گیری کرد. شبکه‌های پیشرو مانند MLP همانند تنیس هستند. آنها عمدتاً در دو حرکت دخیل هستند، یک حرکت رفت و یک حرکت برگشت. شما می‌توانید در این تنیس حدس‌ها و پاسخ‌ها را نوعی علم در نظر بگیرید؛ زیرا هر حدس، آزمایشی است از آنچه ما فکر می‌کنیم و می‌دانیم، و هر پاسخ بازخوردی است که به ما می‌گوید چقدر اشتباه می‌کنیم. در پاس رو به جلو، جریان سیگنال از لایه ورودی از طریق لایه‌های پنهان به لایه خروجی حرکت می‌کند و مقدار لایه خروجی با برچسب‌های درست اندازه‌گیری می‌شود. در برگشت، خطا و بایاس محاسبه می‌شوند؛ این عمل، به ما یک چشم انداز خطا می‌دهد که در طول آن پارامترها ممکن است تنظیم شوند؛ زیرا MLP را یک قدم به حداقل خطا نزدیک می‌کنند. این شبکه بازی تنیس را آنقدر ادامه می‌دهد تا زمانی که خطا کمینه شود. این حالت به همگرایی معروف است.

شکل(۲-۲۰) لایه‌های شبکه عصبی را نشان می‌دهد:



شکل(۲-۲۰): لایه‌های شبکه عصبی

شبکه‌های عصبی در یادگیری ماشین از مدل‌های ریاضی یا محاسباتی برای پردازش اطلاعات استفاده می‌کنند. این شبکه‌های عصبی معمولاً غیرخطی هستند که به آن ها اجازه می‌دهد تا روابط پیچیده بین ورودی و خروجی داده را مدل‌سازی کنند و الگوهایی را در یک مجموعه داده پیدا کنند.

کاربرد شبکه‌های عصبی در یادگیری ماشین یکی از این سه دسته کلی را شامل می شود:

۱. طبقه‌بندی که به موجب آن یک شبکه عصبی می‌تواند الگوها و توالی‌ها را تشخیص دهد.

[1] Root Mean Square Error

۲. تقریب تابعی و تحلیل رگرسیون
۳. پردازش داده‌ها شامل خوشه‌بندی[1] و فیلترکردن داده‌ها

استفاده ار شبکه‌های عصبی برای یادگیری ماشین دارای مزایایی است از جمله:

- ✓ آن‌ها اطلاعات را در کل شبکه ذخیره می‌کنند ؛ به این معنی که شبکه عصبی می‌تواند به کار خود ادامه دهد حتی اگر برخی از اطلاعات از یک قسمت از شبکه عصبی از بین برود.
- ✓ هنگامی که شبکه‌های عصبی با مجموعه داده‌های با کیفیت آموزش داده می‌شوند، در هزینه‌ها و زمان صرفه‌جویی می‌کنند؛ زیرا زمان کوتاه‌تری برای تجزیه و تحلیل داده‌ها و ارائه نتایج می‌گیرند. آن‌ها همچنین کمتر مستعد خطا هستند؛ به خصوص اگر با داده‌های با کیفیت بالا آموزش داده شوند.
- ✓ شبکه‌های عصبی کیفیت و دقت نتایج را ارائه می‌دهند.
- ✓ قابلیت یادگیری مدل‌های غیر خطی

معایب شبکه‌های عصبی چندلایه عبارتند از:

- MLP نیاز به تنظیم تعدادی از پارامترهای فوق العاده مانند تعداد سلول‌های عصبی پنهان، لایه‌ها و تکرارها دارد.
- MLP به مقیاس‌بندی ویژگی‌ها حساس است.

۲-۶-۸: اعتبارسنجی متقابل ((Cross validation(CV)

اعتبارسنجی متقابل تکنیکی برای ارزیابی یک مدل یادگیری ماشین و آزمایش عملکرد آن است که به مقایسه و انتخاب یک مدل مناسب برای مسئله مدل‌سازی پیش‌بینی‌کننده خاص کمک می‌کند. درک اعتبارسنجی متقابل آسان است، پیاده‌سازی آن آسان است و تمایل به بایاس کمتری نسبت به سایر روش‌های مورد استفاده برای شمارش امتیازهای کارایی مدل دارد. همه‌ی این‌ها اعتبارسنجی متقابل را به ابزاری قدرتمند برای انتخاب بهترین مدل برای کار تبدیل می‌کند. تکنیک‌های مختلفی وجود دارد که ممکن است برای اعتبارسنجی متقابل یک مدل استفاده شود. با این حال، همه آنها یک الگوریتم مشابه دارند:

۱. مجموعه داده را به دو بخش تقسیم کنید؛ یکی برای آموزش، دیگری برای آزمایش
۲. مدل را روی مجموعه آموزشی آموزش دهید
۳. اعتبار مدل را در مجموعه آزمایشی تأیید کنید

[1] Clustering

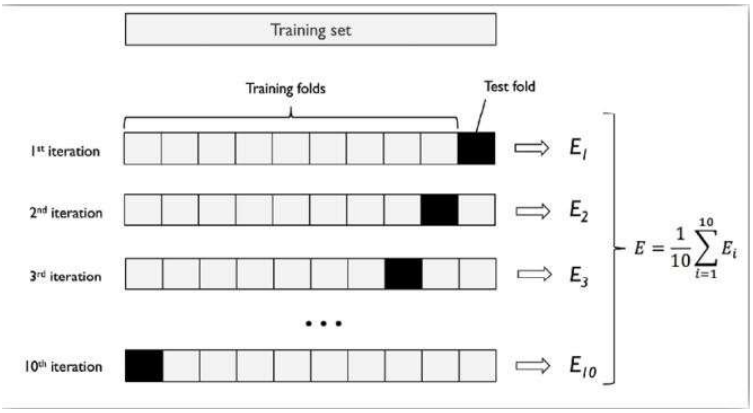
مراحل ۱ تا ۳ را چند بار تکرار کنید. تعداد تکرار به روش CV که استفاده می‌کنید بستگی دارد. تکنیک‌های CV زیادی وجود دارد. برخی از آنها معمولا استفاده می‌شوند، برخی دیگر فقط در تئوری کار می‌کنند. در ادامه روش اعتبارسنجی متقابلی را که در این پایان نامه پوشش داده خواهد شد، بررسی می‌کنیم.

۲-۶-۹: روش k-fold

k-fold CV روش جدیدی را برای تقسیم مجموعه داده معرفی می‌کند که بر غلبه بر روش سنتی که میزان داده های ثابتی را برای آموزش و آزمایش جدا می‌کردیم (روش ۸۰-۲۰ یا ۷۰-۳۰) غلبه می‌کند. الگوریتم روش k-fold که در شکل(۲-۲۱) نیز نمایش داده شده‌است، عبارت است از:

۱) تعداد k دسته را انتخاب کنید. معمولاً k مساوی ۵ یا ۱۰ است، اما می‌توانید هر عددی را انتخاب کنید که کمتر از طول مجموعه داده باشد.
۲) مجموعه داده را (در صورت امکان) به k قسمت مساوی تقسیم کنید که به آنها folds گفته می‌شود.
۳) k-1 Folds را انتخاب کنید که مجموعه آموزشی خواهد بود. باقی‌مانده مجموعه آزمایشی خواهند بود.
۴) مدل را روی مجموعه آموزشی آموزش دهید. در هر تکرار از اعتبارسنجی متقابل، باید یک مدل جدید مستقل از مدل آموزش داده شده در تکرار قبلی آموزش دهید.
۵) در مجموعه آزمایشی اعتبارسنجی کنید.
۶) نتیجه اعتبارسنجی را ذخیره کنید.
۷) مراحل ۳ تا ۶ را k بار تکرار کنید. هر بار از fold باقی‌مانده به عنوان مجموعه آزمون استفاده کنید. در پایان، شما باید مدل را روی هر fold که دارید اعتبارسنجی کرده باشید.
۸) برای به دست آوردن امتیاز نهایی، از نتایجی که در مرحله ۶ به دست آوردید میانگین بگیرید.

نحوه اجرای الگوریتم k-fold در شکل(۲-۲۱) نشان داده شده است:



شکل(۲-۲۱): الگوریتم k-fold

فصل سوم: مروری بر مطالعات انجام شده

۳-۱: مقدمه

در این بخش پژوهش‌های گفته شده از دیدگاه‌های مختلف اعم از روش استخراج ویژگی، روش طبقه‌بندی و ارزیابی کارآمدی الگوریتم‌های به کار رفته شده مورد بررسی قرار می‌گیرند.

۳-۲: مرور کارهای پیشین

در [۴] به بررسی میزان دقت الگوریتم‌های مختلف با چند تکنیک یادگیری ماشین پرداخته است و همچنین مدلی برای یادگیری گروهی ارائه می‌کند. این مقاله از الگوریتم‌های Naïve bayes، رگرسیون لجستیک، یادگیرنده مبتنی بر نمونه و SVM به عنوان طبقه‌بندی‌کننده‌های سطح یک و از جنگل تصادفی به عنوان طبقه‌بندی‌کننده سطح دو استفاده کرده است. نتایج را در جدول(۳-۱) مشاهده می‌کنید.

پارامترها	مقادیر
accuracy	**0/862**
specificity	**0/763**
F1-score	**0/842**

جدول(۳-۱): نتایج مقاله شماره ۱

در [۵] به تفاوت عملکرد طبقه‌بندی‌کننده‌های بر پایه جنگل تصادفی و غیر جنگل تصادفی پرداخته است که نتیجه accuracy را در جدول(۳-۲) مشاهده می‌کنید.

پارامترها	Base classifiers	RF classifiers
accuracy	0/721	**0/744**
heart	0/775	**0/804**
parkinson	0/844	**0/871**

جدول(۳-۲): نتایج مقاله شماره ۲

فصل چهارم: پیاده‌سازی و تجزیه و تحلیل داده‌ها

۴-۱: مجموعه داده

مجموعه داده استفاده شده در این پروژه شامل سن، جنسیت، BMI و ... که در [۲] قابل مشاهده است. این مجموعه داده شامل ۲۱ ویژگی و ۱۵۰۰۰ نمونه است. قبل از شروع کار باید کتابخانه‌های مورد نیاز را به پروژه خود اضافه کنیم که در شکل(۴-۱) و شکل (۴-۲) مشاهده می‌کنید.

```
from numpy import hstack
from numpy import array
from sklearn.feature_selection import SequentialFeatureSelector
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import KFold
from KFOLD import k_fold_results
from without_kfold import without_k_fold_results
from sklearn.model_selection import train_test_split, StratifiedKFold
from sklearn.metrics import precision_recall_fscore_support
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import RFECV
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
```

شکل(۴-۱): library

```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import IsolationForest
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import StackingClassifier
from sklearn.metrics import (confusion_matrix, ConfusionMatrixDisplay,
                             precision_score, recall_score, f1_score,
                             plot_roc_curve, plot_precision_recall_curve,
                             accuracy_score)
df = pd.read_csv("diabetes_binary_health_indicators_BRFSS2015.csv" , nrows=15000)
len(df)
df.head(100)
```

شکل(۴-۲): library

حال که کتابخانه‌های مدنظر را اضافه کردیم باید نگاهی اجمالی به مجموعه داده داشته باشیم.

بخشی از مجموعه داده در جدول (۴-۱) نشان داده شده است:

	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCost	GenHlth
0	0.0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	0.0	0.0	...	1.0	0.0	5.0
1	0.0	0.0	0.0	0.0	25.0	1.0	0.0	0.0	1.0	0.0	...	0.0	1.0	3.0
2	0.0	1.0	1.0	1.0	28.0	0.0	0.0	0.0	0.0	1.0	...	1.0	1.0	5.0
3	0.0	1.0	0.0	1.0	27.0	0.0	0.0	0.0	1.0	1.0	...	1.0	0.0	2.0
4	0.0	1.0	1.0	1.0	24.0	0.0	0.0	0.0	1.0	1.0	...	1.0	0.0	2.0
5	0.0	1.0	1.0	1.0	25.0	1.0	0.0	0.0	1.0	1.0	...	1.0	0.0	2.0
6	0.0	1.0	0.0	1.0	30.0	1.0	0.0	0.0	0.0	0.0	...	1.0	0.0	3.0
7	0.0	1.0	1.0	1.0	25.0	1.0	0.0	0.0	1.0	0.0	...	1.0	0.0	3.0
8	1.0	1.0	1.0	1.0	30.0	1.0	0.0	1.0	0.0	1.0	...	1.0	0.0	5.0
9	0.0	0.0	0.0	1.0	24.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	2.0
10	1.0	0.0	0.0	1.0	25.0	1.0	0.0	0.0	1.0	1.0	...	1.0	0.0	3.0
11	0.0	1.0	1.0	1.0	34.0	1.0	0.0	0.0	0.0	1.0	...	1.0	0.0	3.0
12	0.0	0.0	0.0	1.0	26.0	1.0	0.0	0.0	0.0	0.0	...	1.0	0.0	3.0
13	1.0	1.0	1.0	1.0	28.0	0.0	0.0	0.0	0.0	0.0	...	1.0	0.0	4.0
14	0.0	0.0	1.0	1.0	33.0	1.0	1.0	0.0	1.0	0.0	...	1.0	1.0	4.0
15	0.0	1.0	0.0	1.0	33.0	0.0	0.0	0.0	1.0	0.0	...	1.0	0.0	2.0

جدول(۴-۱): مجموعه داده

توضیحات هر یک از ویژگی‌های مجموعه داده در شکل(۴-۳) آمده است.

```
Diabetes_binary:               target variable Diabetes_binary has 2 classes. 0 is for no diabetes, and 1 is for diabetes
HighBP:         1: high blood pressure, 0: no high blood pressure
HighChol:               1: high cholesterol, 0: no high cholesterol
CholCheck:              1: cholesterol checked within past five years , 0: not checked within 5 years
BMI:            claculated BMI
Smoker:         1: smoker , 0: not smoker
Stroke:         1: had stroke , 0: didn't have stroke
HeartDiseaseorAttack:           1: had heart disease or heart attack , 0: didn't have heart disease or heart attack
PhysActivity:           1: Meet Aerobic Recommendations ( physical activity more than 150 minutes per week ,yes) , 0: no or low
physical activity
Fruits:         1: more than 1 fruit per day , 0: no fruits per day
Veggies:                1: Consume Vegetables 1 or more times per day , not consumed vegetables
HvyAlcoholConsump:              1: having more than 14 drinks per week for men or 7 for women , 0 : not drinking too much ( low
er than 14 per week for men or 7 for women)
AnyHealthcare:          About how long has it been since you last visited a doctor for a routine checkup? 1: within past year ,
0: more than 1 year
NoDocbcCost:            1: yes 0: no ( Was there a time in the past 12 months when you needed to see a doctor but could not bec
ause of cost?)
GenHlth:                Would you say that in general your health is? 1: yes, 2: Very good, 3: Good, 4: fair, 5:poor
MentHlth:               Now thinking about your mental health, which includes stress, depression, and problems with emotions, f
or how many days during the past 30 days was your mental health not good?
PhysHlth:               During the past 30 days, for about how many days did poor physical or mental health keep you from doing
your usual activities, such as self-care, work, or recreation?
DiffWalk:               Do you have serious difficulty walking or climbing stairs? 1: yes, 0: no
Sex:            1: male , 0: female
Age:            Fourteen-level age category , 1: Age 18 to 24 , 2: Age 25 to 29 , 3: Age 30 to 34 , ... , 13: Age 80 or older
Education:              1: Never attended school or only kindergarte , 2: Grades 1 through 8 (Elementary), 3: Grades 9 through
11 (Some high school), 4:Grade 12 or GED (High school graduate), 5:College 1 year to 3 years (Some college or technical schoo
l), 6:College 4 years or more (College graduate)
Income:         what Is your annual household income from all sources: 1:1 Less than $10,000 , 2: $10,000 to less than $15,000,
3: $15,000 to less than $20,000, 4: $20,000 to less than $25,000, 5:$25,000 to less than $35,000, ... , 8: $75,000 or more
```

شکل(۴-۳): توضیحات ویژگی‌های مجموعه داده

اطلاعات بیش‌تری از مجموعه داده را در جدول(۴-۲) مشاهده می‌کنید:

	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...
count	8000.000000	8000.000000	8000.000000	8000.000000	8000.000000	8000.000000	8000.000000	8000.000000	8000.000000	8000.000000	...
mean	0.165625	0.474875	0.448625	0.959375	28.795125	0.449500	0.048250	0.100250	0.736375	0.590250	...
std	0.371767	0.499400	0.497385	0.197432	6.382338	0.497474	0.214307	0.300352	0.440626	0.491818	...
min	0.000000	0.000000	0.000000	0.000000	14.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
25%	0.000000	0.000000	0.000000	1.000000	24.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
50%	0.000000	0.000000	0.000000	1.000000	28.000000	0.000000	0.000000	0.000000	1.000000	1.000000	...
75%	0.000000	1.000000	1.000000	1.000000	32.000000	1.000000	0.000000	0.000000	1.000000	1.000000	...
max	1.000000	1.000000	1.000000	1.000000	74.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...

جدول(۴-۲): خروجی تابع describe برای مجموعه داده (برخی از ویژگی‌ها)

برای کار با مجموعه داده، باید عناصر آن از یک نوع باشند. در این مجموعه داده همه داده ها از نوع float64 هستند که در شکل (۴-۴) قابل مشاهده است.

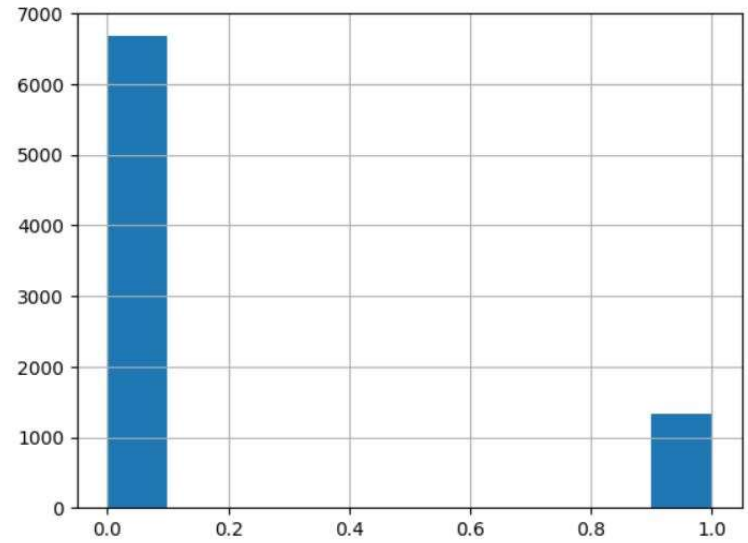
```
RangeIndex: 8000 entries, 0 to 7999
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Diabetes_binary       8000 non-null   float64
 1   HighBP                8000 non-null   float64
 2   HighChol              8000 non-null   float64
 3   CholCheck             8000 non-null   float64
 4   BMI                   8000 non-null   float64
 5   Smoker                8000 non-null   float64
 6   Stroke                8000 non-null   float64
 7   HeartDiseaseorAttack  8000 non-null   float64
 8   PhysActivity          8000 non-null   float64
 9   Fruits                8000 non-null   float64
 10  Veggies               8000 non-null   float64
 11  HvyAlcoholConsump     8000 non-null   float64
 12  AnyHealthcare         8000 non-null   float64
 13  NoDocbcCost           8000 non-null   float64
 14  GenHlth               8000 non-null   float64
 15  MentHlth              8000 non-null   float64
 16  PhysHlth              8000 non-null   float64
 17  DiffWalk              8000 non-null   float64
 18  Sex                   8000 non-null   float64
 19  Age                   8000 non-null   float64
 20  Education             8000 non-null   float64
 21  Income                8000 non-null   float64
dtypes: float64(22)
```

شکل(۴-۴) : نوع داده‌ای ویژگی‌های مجموعه داده

در مرحله بعد برای آن که دید نسبتاً خوبی به مجموعه داده پیدا کنیم بهتر است هر یک از ویژگی ها را روی نمودار بررسی کنیم.

در شکل(۴-۵) ویژگی Diabetes_binary که ویژگی هدف ما است را مشاهده می‌کنید.

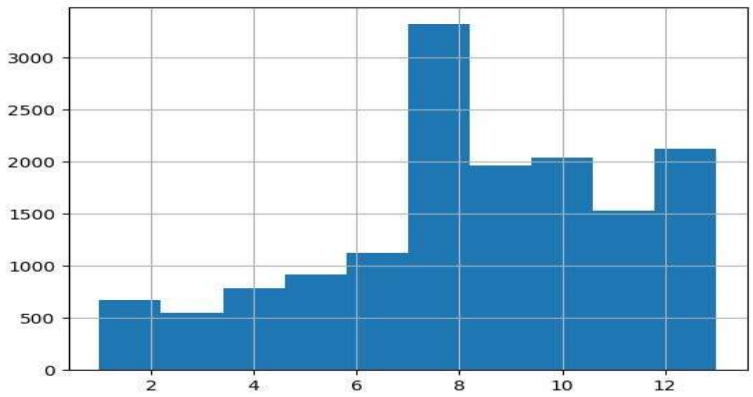
```
0.0    6675
1.0    1325
Name: Diabetes_binary, dtype: int64
```



شکل(۴-۵): ویژگی Diabetes_binary

در شکل(۴-۶) ویژگی age را مشاهده می‌کنید.

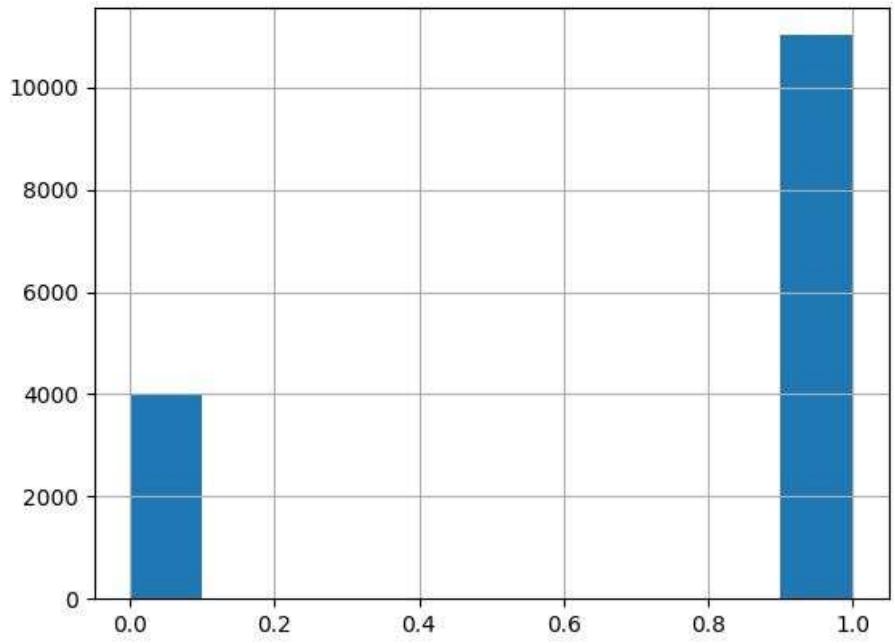
```
10.0    2034
9.0     1961
8.0     1788
11.0    1531
7.0     1528
6.0     1119
13.0    1063
12.0    1063
5.0      912
4.0      784
3.0      548
2.0      389
1.0      280
Name: Age, dtype: int64
```



شکل(۴-۶): ویژگی age

در شکل(۴-۷) ویژگی PhysActivity را مشاهده می‌کنید:

```
1.0    11013
0.0     3987
Name: PhysActivity, dtype: int64
```



شکل(۴-۷): ویژگی physActivity

روش کار در این پروژه به این صورت است که ابتدا مجموعه داده بدون انتخاب ویژگی را به مدل‌ها می‌دهیم و نتیجه را بررسی می‌کنیم. سپس با الگوریتم iforest سعی می‌کنیم تا outlier data را حذف کنیم. سپس نتیجه الگوریتم‌ها را این بار با مجموعه داده جدید بررسی می‌کنیم. حالا سعی می‌کنیم تا با تکنیک SMOTETomek مجموعه داده را متعادل کنیم. اگر دقت کنیم، می‌بینیم که زیر ۲۰ درصد افراد در این مجموعه داده دیابت مثبت دارند. همین کم بودن شدید مثبت‌ها نسبت به منفی‌ها می‌تواند باعث کاهش امتیاز recall شود. پس از این کار دوباره نتیجه الگوریتم‌ها را بررسی می‌کنیم.

ابتدا به روش train_test_split به میزان ۸۰-۲۰ داده‌های آزمایش و آموزش را جدا کرده و از ۲۰ درصد داده آزمایش برای تست نهایی الگوریتم یادگیری گروهی استفاده خواهد شد. برای این کار از تابع k_fold استفاده می‌کنیم که حجم زیادی از کدهای پیاده‌سازی شده کاهش یافت. فقط کافی است تا لیستی از آبجکت‌های مدل الگوریتم‌های یادگیری ماشین را با ورودی‌های خاص هر آبجکت به این تابع بدهیم و مقادیر پارامترهای accuracy، recall، specificity، precision و f1-score را در یک جدول به عنوان خروجی ببینیم.

```
algo={
    'NB':MultinomialNB(alpha=1.0),
    'Logistic Regression':LogisticRegression(C=0.01, solver='liblinear',max_iter=1000),
    'SVM':SVC(kernel='rbf', gamma=0.1 ),
    'KNN':KNeighborsClassifier(n_neighbors=5),
    'Dtree':DecisionTreeClassifier(),
    'Multi-layer-Perceptron':MLPClassifier(random_state=3, max_iter=500),
}
```

شکل(۴-۸): لیست ورودی تابع k-fold

۴-۲: یادگیری با روش‌های مختلف انتخاب ویژگی

۴-۲-۱: یادگیری بدون استفاده از انتخاب ویژگی

در ابتدا مجموعه داده را بدون استفاده از الگوریتم‌های انتخاب ویژگی به مدل‌ها می‌دهیم و نتایج را بررسی می‌کنیم. در جدول(۴-۳) مقادیر accuracy، recall، specificity، precision و f1-score را برای الگوریتم‌های یادگیری ماشین مختلف مشاهده می‌کنید.

	accuracy	recall	specificity	precision	F1
SVM	0.832969	0.025604	0.992694	0.456144	0.048212
Multi-layer-Perceptron	0.832500	0.159153	0.965919	0.479427	0.233668
Logistic Regression	0.832031	0.078787	0.980897	0.442670	0.133283
KNN	0.815000	0.175221	0.941374	0.370474	0.237805
Dtree	0.753594	0.325455	0.838415	0.284107	0.303066
NB	0.741875	0.366315	0.816133	0.282624	0.318851

جدول(۴-۳): یادگیری بدون انتخاب ویژگی

می‌بینیم که بهترین مقدار accuracy به ترتیب برای الگوریتم ماشین بردار پشتیبان با مقدار ۰/۸۳۲۹ و mlp و رگرسیون لجستیک و KNN با مقادیر ۰/۸۳۲۵ ، ۰/۸۳۲۰ و ۰/۸۱۵ است. نکته قابل توجه مقدار recall پایین و برعکس آن مقدار specificity بالا برای تمامی الگوریتم‌هاست و دلیل آن متعادل و بالانس نبودن متغیر هدف یعنی Diabetes_binary است که قبلا گفته شد زیر ۲۰ درصد آن مقدار یک دارد و بقیه ۰ است. همین نشان می‌دهد که الگوریتم‌ها در خروجی خود مقدار ۰ بیشتر دارند، در نتیجه با توجه به فرمول recall و specificity به طور خودکار مقدار آن‌ها به همین صورت خواهد بود.

حالا باید از تعدادی از این الگوریتم‌ها را به عنوان تخمین‌گرهای مرتبه صفر و یکی از آن‌ها را به عنوان تخمین‌گر نهایی و مرتبه یک انتخاب کنیم. ۴ الگوریتم اول را که بهترین accuracy را دارند به عنوان الگوریتم‌های مرتبه صفر انتخاب می‌کنیم و از آن‌جایی که رگرسیون لجستیک معمولاً به عنوان تخمین‌گر نهایی بهتر عمل می‌کند و

accuracy آن با بهترین الگوریتم یعنی SVM اختلاف ناچیزی دارد، آن را به عنوان تخمین‌گر نهایی انتخاب می‌کنیم. توجه کنید که در این لحظه ما هنوز داده‌های پرت را حذف نکرده‌ایم. نتیجه را در شکل(۴-۴) مشاهده می‌کنید.

	accuracy	recall	specificity	precision	F1
Ensemble	0.846250	0.175373	0.981231	0.652778	0.276471
SVM	0.832969	0.025604	0.992694	0.456144	0.048212
Multi-layer-Perceptron	0.832500	0.159153	0.965919	0.479427	0.233668
Logistic Regression	0.832031	0.078787	0.980897	0.442670	0.133283
KNN	0.815000	0.175221	0.941374	0.370474	0.237805
Dtree	0.753594	0.325455	0.838415	0.284107	0.303066
NB	0.741875	0.366315	0.816133	0.282624	0.318851

جدول(۴-۴): یادگیری گروهی بدون انتخاب ویژگی

می‌بینیم که با استفاده از یادگیری گروهی، مقدار accuracy به مقدار ۰/۰۱۴ بهبود پیدا کرده است.

حالا وقت آن است که داده‌های پرت را حذف کنیم و نتیجه را با مقادیر جدول(۴-۴) مقایسه کنیم.

۴-۲-۲: حذف داده‌های پرت با Iforest

در مجموعه‌های داده معمولاً مقادیری وجود دارد که درست و واقعی نیست. برای مثال ممکن است در یک مجموعه داده مقدار سن یک نمونه صفر باشد! و اگر تعداد این نمونه‌ها زیاد باشد روی عملکرد الگوریتم‌های یادگیری ماشین تاثیرگذار خواهد بود. در ابتدا مجموعه داده شامل ۸۰۰۰ نمونه بود که پس از استفاده از الگوریتم iforest مقدار نمونه‌ها به تعداد ۷۲۰۰ کاهش یافت.

در شکل(۴-۵) نتیجه الگوریتم‌های مختلف را برای مجموعه داده جدید مشاهده می‌کنید.

	accuracy	recall	specificity	precision	F1
SVM	0.846007	0.023619	0.994898	0.416667	0.044039
Logistic Regression	0.845139	0.072015	0.985018	0.487363	0.124107
Multi-layer-Perceptron	0.841493	0.140263	0.969237	0.505711	0.199977
KNN	0.830556	0.160557	0.951662	0.376512	0.224385
Dtree	0.780035	0.337039	0.860222	0.303290	0.318599
NB	0.774132	0.301643	0.859602	0.281265	0.290001

جدول(۴-۵): نتیجه یادگیری با حذف داده‌های پرت و بدون انتخاب ویژگی

به وضوح میزان بهبود accuracy در الگوریتم‌ها مشخص است. همچنان ماشین بردار پشتیبان بهترین عملکرد را دارد. به ترتیب بهترین عملکرد را الگوریتم‌های SVM، رگرسیون لجستیک، mlp و KNN با مقادیر ۰/۸۴۶، ۰/۸۴۵، ۰/۸۴۱ و ۰/۸۳۰ دارند که همه‌ی آن‌ها نسبت به قبل بهبود یافته‌اند. حالا نوبت آن است تا با روش SMOTETomek مجموعه داده را متعادل کنیم.

۴-۲-۳: متعادل‌کردن مجموعه داده با SMOTETomek

علاوه بر این، مجموعه داده نامتعادل در جایی اتفاق می‌افتد که تعداد نمونه‌های کلاس اکثریت تا حد زیادی از نمونه‌های کلاس اقلیت فراتر می‌رود و می‌تواند به طور قابل‌توجهی بر عملکرد مدل یادگیری ماشین تأثیر بگذارد (یعنی نتایج مغرضانه و غیرقابل اعتماد). مطالعات گذشته نشان داده است که مدل های طبقه‌بندی پس از حل مشکل داده های نامتعادل بهبود یافته است. با ذکر این نکته که این روش روی مجموعه داده‌ای که داده‌های پرت آن حذف شده‌است، اجرا شده است نتایج الگوریتم‌های مختلف را در جدول(۴-۶) مشاهده می‌کنید.

	accuracy	recall	specificity	precision	F1
SVM	0.863000	0.913333	0.812873	0.829701	0.869434
Dtree	0.857566	0.867267	0.847914	0.850917	0.858841
KNN	0.801168	0.970001	0.632623	0.725060	0.829771
Multi-layer-Perceptron	0.791123	0.856766	0.727324	0.764322	0.804012
Logistic Regression	0.726414	0.761423	0.691429	0.711276	0.735391
NB	0.583266	0.360640	0.805815	0.649848	0.463748

جدول(۴-۶): نتیجه یادگیری با استفاده از SMOTETomek و *IForest*

با توجه به مقادیر جدول(۴-۶) می‌بینیم که نتایج جالبی به‌دست آمده است. برای مثال عملکرد الگوریتم رگرسیون لجستیک بسیار بد بود و دقت آن به زیر ۷۵ درصد یعنی ۰/۷۲۶ رسیده است! و باز هم عملکرد عالی الگوریتم SVM. الگوریتم‌هایی که بهترین accuracy را داشتند به ترتیب SVM، درخت تصمیم، KNN و MLP با مقادیر

۰/۸۶۳، ۰/۸۵۷، ۰/۸۰۱ و ۰/۷۹۱ داشتند. نکته مهمی که باید به آن توجه کنیم افزایش مقادیر recall در تمامی الگوریتم‌هاست. با روش SMOTETomek ما نسبت مقادیر مثبت و منفی متغیر هدف یعنی Diabetes_binary که ۸۰ به ۲۰ به سود منفی بود را به ۵۰ – ۵۰ رساندیم و با این کار مجموعه داده متعادل می‌شود. همین امر باعث می شود تا مقادیر واقعی recall را ببینیم. همچنین مقادیر specificity هم نسبتاً خوب است. حتی بدون استفاده از الگوریتم‌های انتخاب ویژگی هم دقت ما بسیار بالا و خوب است. حالا این ۴ الگوریتم اول را به عنوان تخمین‌گرهای مرتبه صفر و مثل همیشه رگرسیون لجستیک را به عنوان تخمین‌گر مرتبه یک به الگوریتم یادگیری گروهی می‌دهیم و نتایج را در جدول(۴-۷) مشاهده می‌کنید.

	accuracy	recall	specificity	precision	F1
Ensemble	0.922888	0.925471	0.920296	0.920945	0.923203
SVM	0.863000	0.913333	0.812873	0.829701	0.869434
Dtree	0.857566	0.867267	0.847914	0.850917	0.858841
KNN	0.801168	0.970001	0.632623	0.725060	0.829771
Multi-layer-Perceptron	0.791123	0.856766	0.727324	0.764322	0.804012
Logistic Regression	0.726414	0.761423	0.691429	0.711276	0.735391
NB	0.583266	0.360640	0.805815	0.649848	0.463748

جدول(۴-۷): نتیجه یادگیری گروهی با استفاده از *SMOTETomek* و *IForest*

با توجه به مقادیر جدول (۴-۷) می‌بینیم که ensemble learner به مقدار accuracy ۰/۹۲۲ رسیده است! و نکته قابل توجه امتیاز بقیه معیارها است که همه بالای ۹۲ درصد است. البته همیشه وقتی نتایج اینقدر خوب است و مخصوصاً در مقادیر بالاتر باید نگران overfitting باشیم.

حالا نوبت آن است تا الگوریتم‌های انتخاب ویژگی را روی مجموعه داده آزمایش کنیم و ببینیم که آیا اصلا به مقادیری بهتر از ۰/۹۲۲ برای accuracy می‌رسیم یا خیر. به این نکته هم توجه کنیم که انتخاب ویژگی روی مجموعه داده کنونی اعمال خواهد شد.

۴-۲-۴: ارزیابی یادگیری با انتخاب ویژگی correlation

از مجموعه داده‌ای که با استفاده از الگوریتم انتخاب ویژگی correlation به‌دست آمده به‌عنوان ورودی استفاده می‌کنیم. نتایج در جدول(۴-۸) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
Dtree	0.864949	0.824433	0.905357	0.896771	0.859008
Multi-layer-Perceptron	0.802090	0.818933	0.785183	0.793014	0.805288
KNN	0.765174	0.830304	0.700433	0.734777	0.779383
SVM	0.761074	0.833214	0.689219	0.728168	0.777027
Logistic Regression	0.727950	0.800949	0.655239	0.699058	0.746333
NB	0.662733	0.701434	0.624444	0.651451	0.675244

جدول(۴-۸): نتیجه یادگیری با روش انتخاب ویژگی *correlation*

بهترین مقدار accuracy برای الگوریتم درخت تصمیم و سپس MLP به ترتیب با مقادیر۰/۸۶۴۹ و ۰/۸۰۲ است که مشاهده می‌کنید برای الگوریتم درخت تصمیم این مقدار افزایش یافته و برای الگوریتمMLP هم این مقدار افزایش یافته است. برای برخی الگوریتم‌ها هم حتی کاهش یافته‌است مانند KNN و علت این کاهش‌ها از دست رفتن اطلاعات برخی از ویژگی‌هایی است که حذف شده‌اند .بالاترین recall مربوط به الگوریتم درخت تصمیم و سپس MLP به ترتیب با مقادیر ۰/۸۲۴ و ۰/۸۱۸ است که نسبت به حالت بدون انتخاب ویژگی کاهش یافته‌است و این بدان معناست که توانایی این الگوریتم‌ها در این حالت برای شناسایی موارد مثبت کاهش یافته است . بالاترین specificity مربوط به الگوریتم درخت تصمیم با مقدار۰/۹۰۵ است که از حالت بدون انتخاب ویژگی بیشتر است و یعنی این در این حالت نتایج منفی درست‌تر نشان داده می‌شوند. بالاترین precision مربوط به الگوریتم درخت تصمیم و سپس MLP به ترتیب با مقادیر ۰/۸۹۶ و ۰/۷۹۳ است. یعنی در این دو الگوریتم موارد مثبتی که پیش‌بینی شد و واقعا هم مثبت بودند، نسبت به حالت قبل افزایش یافته‌است. بیشترین مقدار-f1 score هم در بین این الگوریتم‌ها مربوط به درخت تصمیم با مقدار ۰/۸۵۹ است که به میزان کمی در حد اعشار نسبت به حالت بدون انتخاب ویژگی افزایش داشته‌است. با توجه به نتایج بهترین الگوریتم‌های برای یادگیری گروهی، درخت تصمیم، MLP، KNN و SVM خواهندبود.

نتایج در جدول(۴-۹) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
Ensemble	0.878589	0.872236	0.884963	0.883817	0.877988
Dtree	0.864949	0.824433	0.905357	0.896771	0.859008
Multi-layer-Perceptron	0.802090	0.818933	0.785183	0.793014	0.805288
KNN	0.765174	0.830304	0.700433	0.734777	0.779383
SVM	0.761074	0.833214	0.689219	0.728168	0.777027
Logistic Regression	0.727950	0.800949	0.655239	0.699058	0.746333
NB	0.662733	0.701434	0.624444	0.651451	0.675244

جدول(۴-۹): نتیجه یادگیری گروهی با انتخاب ویژگی *correlation*

مقدار accuracy برای یادگیری گروهی ۰/۸۷۸ است که نسبت به حالت بدون انتخاب ویژگی کاهش یافته‌است. همچنین بقیه معیارها نیز نسبت به حالت بدون انتخاب ویژگی کاهش داشته‌اند. یک نکته هم این است که ما برای یادگیری گروهی از ۴ الگوریتم استفاده کردیم که دو تا از آن‌ها یعنی KNN و SVM مقدار accuracy زیر ۰/۸۰ داشتند. برای یادگیری گروهی تنها از دو الگوریتم درخت تصمیم و MLP نیز استفاده کردیم که نتایج را در جدول (۴-۱۰) مشاهده می‌کنید.

	accuracy	recall	specificity	precision	F1
Ensemble	0.878999	0.881245	0.876746	0.877651	0.879444
Dtree	0.864949	0.824433	0.905357	0.896771	0.859008
Multi-layer-Perceptron	0.802090	0.818933	0.785183	0.793014	0.805288
KNN	0.765174	0.830304	0.700433	0.734777	0.779383
SVM	0.761074	0.833214	0.689219	0.728168	0.777027
Logistic Regression	0.727950	0.800949	0.655239	0.699058	0.746333
NB	0.662733	0.701434	0.624444	0.651451	0.675244

جدول(۴-۱۰): نتیجه یادگیری گروهی تنها با دو الگوریتم درخت تصمیم و MLP

مقدار accuracy در این حالت نسبت به حالت قبل در حد ۰/۰۰۱۴افزایش یافته‌است. همچنین امتیاز recall و f1-score نسبت به حالت قبل به میزان اندکی افزایش داشته‌اند و شاهد کاهش امتیازهای specificity و precision نسبت به قبل هستیم.

۴-۲-۵: ارزیابی یادگیری با انتخاب ویژگی Variance Threshold

از مجموعه داده‌ای که با استفاده از الگوریتم انتخاب ویژگی variance threshold به‌دست آمده به عنوان ورودی استفاده می‌کنیم. نتایج در جدول(۴-۱۱) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
Dtree	0.856234	0.870970	0.841578	0.846061	0.858214
SVM	0.851106	0.903480	0.798909	0.817737	0.858390
KNN	0.808244	0.965327	0.651561	0.734709	0.834220
Multi-layer-Perceptron	0.767947	0.827826	0.708314	0.740256	0.780981
Logistic Regression	0.724671	0.761087	0.688325	0.709158	0.734141
NB	0.578549	0.351228	0.805812	0.643840	0.454408

جدول(۴-۱۱): نتیجه یادگیری با انتخاب ویژگی variance threshold

بهترین مقدار accuracy برای الگوریتم درخت تصمیم و سپس SVM به ترتیب با مقادیر۰/۸۵۶ و ۰/۸۵۱ است که مشاهده می‌کنید برای هر دو الگوریتم این مقدار کاهش یافته‌است. تنها برای الگوریتمKNN این مقدار افزایش اندکی داشته‌است. بالاترین recall مربوط به الگوریتم KNN و سپسSVM به ترتیب با مقادیر ۰/۹۶۵ و ۰/۹۰۳ است که نسبت به حالت بدون انتخاب ویژگی کاهش یافته‌است و این بدان معناست که توانایی این الگوریتم‌ها در این حالت برای شناسایی موارد مثبت کاهش یافته است .بالاترین specificity مربوط به الگوریتم درخت تصمیم با مقدار۰/۸۴۱ است که از حالت بدون انتخاب ویژگی کمتر است. بالاترین precision مربوط به الگوریتم درخت تصمیم و سپس SVM به ترتیب با مقادیر ۰/۸۴۶ و ۰/۸۱۷ است. یعنی در این دو الگوریتم موارد مثبتی که پیش بینی شد و واقعا هم مثبت بودند، نسبت به حالت قبل کاهش یافته‌اند. بیشترین مقدارf1-score هم در بین این الگوریتم‌ها مربوط به SVM با مقدار ۰/۸۵۸۳ است که به میزان کمی نسبت به حالت بدون انتخاب ویژگی کاهش داشته‌است. با توجه به نتایج بهترین الگوریتم‌های برای یادگیری گروهی، درخت تصمیم، SVM، KNN و MLP خواهندبود. نتایج در جدول(۴-۱۲) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
Ensemble	0.914274	0.921376	0.907149	0.908724	0.915006
Dtree	0.856234	0.870970	0.841578	0.846061	0.858214
SVM	0.851106	0.903480	0.798909	0.817737	0.858390
KNN	0.808244	0.965327	0.651561	0.734709	0.834220
Multi-layer-Perceptron	0.767947	0.827826	0.708314	0.740256	0.780981
Logistic Regression	0.724671	0.761087	0.688325	0.709158	0.734141
NB	0.578549	0.351228	0.805812	0.643840	0.454408

جدول(۴-۱۲): نتیجه یادگیری گروهی با انتخاب ویژگی variance threshold

مقدار accuracy برای یادگیری گروهی ۰/۹۱۴ است که نسبت به حالت بدون انتخاب ویژگی کاهش یافته‌است. همچنین بقیه معیارها نیز نسبت به حالت بدون انتخاب ویژگی کاهش داشته‌اند البته عملکرد انتخاب ویژگی variacne threshold از correlation بهتر است و تنها به میزان کمی با حالت بدون انتخاب ویژگی اختلاف دارد.

۴-۲-۶: ارزیابی یادگیری با انتخاب ویژگی RFECV

از مجموعه داده‌ای که با استفاده از الگوریتم انتخاب ویژگی RFECV به‌دست آمده به عنوان ورودی استفاده می کنیم. نتایج در جدول(۴-۱۳) قابل مشاهده است.

	accuracy	recall	specificity	precision	F1
Dtree	0.856335	0.868650	0.844088	0.847738	0.857969
SVM	0.846901	0.908256	0.785746	0.808853	0.855615
Multi-layer-Perceptron	0.808555	0.849742	0.767018	0.786248	0.816249
KNN	0.797580	0.970608	0.624737	0.720993	0.827329
Logistic Regression	0.726824	0.761875	0.691813	0.711686	0.735824
NB	0.590034	0.359853	0.820121	0.666360	0.467233

جدول(۴-۱۳): نتیجه یادگیری با انتخاب ویژگی RFECV

بهترین مقدار accuracy برای الگوریتم درخت تصمیم و سپس SVM به ترتیب با مقادیر۰/۸۵۶ و ۰/۸۴۶ است که مشاهده می‌کنید برای هر دو الگوریتم این مقدار کاهش یافته‌است. تنها برای الگوریتم KNN این مقدار افزایش اندکی داشته‌است. بالاترین recall مربوط به الگوریتم KNN و سپس SVM به ترتیب با مقادیر ۰/۹۷۰ و ۰/۹۰۸ است که برای knn نسبت به حالت بدون انتخاب ویژگی افزایش و برای SVM کاهش داشته‌است و این بدان معناست که توانایی این الگوریتم در این حالت برای شناسایی موارد مثبت کاهش یافته است. بالاترین

specificityمربوط به الگوریتم درخت تصمیم با مقدار۰/۸۴۴ است که باز هم از حالت بدون انتخاب ویژگی کمتر است. بالاترینprecision مربوط به الگوریتم درخت تصمیم و سپس SVM به ترتیب با مقادیر ۰/۸۴۷ و ۰/۸۰۸ است. یعنی در این دو الگوریتم موارد مثبتی که پیش‌بینی شد و واقعا هم مثبت بودند، نسبت به حالت قبل کاهش یافته‌اند. بیشترین مقدارf1-score هم در بین این الگوریتم‌ها مربوط به درخت تصمیم با مقدار ۰/۸۵۷۹ است که به میزان کمی نسبت به حالت بدون انتخاب ویژگی کاهش داشته‌است. با توجه به نتایج بهترین الگوریتم‌های برای یادگیری گروهی، درخت تصمیم، SVM، KNN و MLP خواهندبود. نتایج در جدول(۴-۱۴) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
Ensemble	0.905660	0.914005	0.897288	0.899275	0.906580
Dtree	0.856335	0.868650	0.844088	0.847738	0.857969
SVM	0.846901	0.908256	0.785746	0.808853	0.855615
Multi-layer-Perceptron	0.808555	0.849742	0.767018	0.786248	0.816249
KNN	0.797580	0.970608	0.624737	0.720993	0.827329
Logistic Regression	0.726824	0.761875	0.691813	0.711686	0.735824
NB	0.590034	0.359853	0.820121	0.666360	0.467233

جدول(۴-۱۴): نتیجه یادگیری گروهی با انتخاب ویژگی RFECV

مقدار accuracy برای یادگیری گروهی ۰/۹۰۵ است که نسبت به حالت بدون انتخاب ویژگی کاهش یافته‌است. همچنین بقیه معیارها نیز نسبت به حالت بدون انتخاب ویژگی کاهش داشته‌اند.

۴-۲-۷: ارزیابی یادگیری با انتخاب ویژگی Forward

از مجموعه داده‌ای که با استفاده از الگوریتم انتخاب ویژگیForward به‌دست آمده به عنوان ورودی استفاده می کنیم. نتایج در جدول(۴-۱۵) قابل مشاهده است.

	accuracy	recall	specificity	precision	F1
Dtree	0.855925	0.764487	0.947326	0.935555	0.841201
KNN	0.795527	0.728864	0.861965	0.840906	0.780776
Multi-layer-Perceptron	0.758202	0.831240	0.685130	0.725421	0.774590
SVM	0.711239	0.763132	0.659361	0.691152	0.725287
NB	0.694523	0.767145	0.622159	0.669929	0.715093
Logistic Regression	0.691140	0.736529	0.645849	0.675145	0.704415

جدول(۴-۱۵): نتیجه یادگیری با انتخاب ویژگی forward

بهترین مقدار accuracy برای الگوریتم درخت تصمیم و سپس KNN به ترتیب با مقادیر۰/۸۵۵۹ و ۰/۷۹۵ است که مشاهده می‌کنید برای هر دو الگوریتم این مقدار کاهش یافته‌است. بالاترین recall مربوط به الگوریتم MLP و سپس NB به ترتیب با مقادیر ۰/۸۳۱ و ۰/۷۶۷ است که نسبت به حالت بدون انتخاب ویژگی کاهش یافته‌است و این بدان معناست که توانایی این الگوریتم‌ها در این حالت برای شناسایی موارد مثبت کاهش یافته است . بالاترین specificity مربوط به الگوریتم درخت تصمیم با مقدار۰/۹۴۷ است که از حالت بدون انتخاب ویژگی بیشتر است. بالاترین precision مربوط به الگوریتم درخت تصمیم و سپس KNN به ترتیب با مقادیر ۰/۹۳۵ و ۰/۸۴۰ است. یعنی در این دو الگوریتم موارد مثبتی که پیش‌بینی شد و واقعا هم مثبت بودند، نسبت به حالت قبل افزایش یافته‌اند. بیشترین مقدارf1-score هم در بین این الگوریتم‌ها مربوط به درخت تصمیم با مقدار ۰/۸۴۱ است که به میزان کمی نسبت به حالت بدون انتخاب ویژگی کاهش داشته‌است. با توجه به نتایج بهترین الگوریتم‌های برای یادگیری گروهی، درخت تصمیم، SVM، KNN و MLP خواهندبود. نتایج در جدول(۴-۱۶) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
Dtree	0.855925	0.764487	0.947326	0.935555	0.841201
Ensemble	0.853158	0.810811	0.895645	0.886303	0.846878
KNN	0.795527	0.728864	0.861965	0.840906	0.780776
Multi-layer-Perceptron	0.758202	0.831240	0.685130	0.725421	0.774590
SVM	0.711239	0.763132	0.659361	0.691152	0.725287
NB	0.694523	0.767145	0.622159	0.669929	0.715093
Logistic Regression	0.691140	0.736529	0.645849	0.675145	0.704415

جدول(۴-۱۶): نتیجه یادگیری گروهی با انتخاب ویژگی forward

مقدار accuracy برای یادگیری گروهی ۰/۸۵۳ است که نسبت به حالت بدون انتخاب ویژگی کاهش یافته‌است این مقدار حتی از الگوریتم درخت تصمیم هم کمتر است و نشان‌دهنده این است که یادگیری گروهی عملکرد

ضعیف‌تری نسبت به درخت تصمیم به تنهایی داشته‌است. البته مقدار recall و f1-score در حالت یادگیری گروهی بیشتر از مقادیر درخت تصمیم است. در کل همه مقادیر نسبت به حالت بدون انتخاب ویژگی کاهش داشته‌اند.

۴-۲-۸: ارزیابی یادگیری با انتخاب ویژگی Backward

از مجموعه داده‌ای که با استفاده از الگوریتم انتخاب ویژگیBackward به‌دست آمده به عنوان ورودی استفاده می کنیم. نتایج در جدول(۴-۱۷) قابل مشاهده است.

	accuracy	recall	specificity	precision	F1
Dtree	0.854389	0.848865	0.859959	0.858419	0.853532
SVM	0.817266	0.874502	0.760110	0.784412	0.826979
KNN	0.811423	0.948031	0.675020	0.744537	0.833991
Multi-layer-Perceptron	0.733901	0.800679	0.666877	0.706048	0.750133
Logistic Regression	0.696778	0.716247	0.677207	0.689046	0.702334
NB	0.566757	0.329873	0.803516	0.626718	0.432148

جدول(۴-۱۷): نتیجه یادگیری با انتخاب ویژگی backward

بهترین مقدار accuracy برای الگوریتم درخت تصمیم و سپس SVM به ترتیب با مقادیر۰/۸۵۴۳ و ۰/۸۱۷ است که مشاهده می‌کنید برای هر دو الگوریتم این مقدار کاهش یافته‌است. بالاترین recall مربوط به الگوریتم KNN و سپس SVM به ترتیب با مقادیر ۰/۹۴۸ و ۰/۸۷۴ است که نسبت به حالت بدون انتخاب ویژگی کاهش یافته است و این بدان معناست که توانایی این الگوریتم‌ها در این حالت برای شناسایی موارد مثبت کاهش یافته است . بالاترین specificity مربوط به الگوریتم درخت تصمیم با مقدار۰/۸۵۹ است که از حالت بدون انتخاب ویژگی بیشتر است. بالاترینprecision مربوط به الگوریتم درخت تصمیم و سپس SVM به ترتیب با مقادیر ۰/۸۵۸ و ۰/۷۸۴ است. یعنی در این دو الگوریتم موارد مثبتی که پیش‌بینی شد و واقعا هم مثبت بودند، نسبت به حالت قبل افزایش یافته‌اند. بیشترین مقدارf1-score هم در بین این الگوریتم‌ها مربوط به درخت تصمیم با مقدار ۰/۸۵۳ است که به میزان کمی نسبت به حالت بدون انتخاب ویژگی کاهش داشته‌است. با توجه به نتایج بهترین الگوریتم‌های برای یادگیری گروهی، درخت تصمیم، SVM، KNN و MLP خواهندبود. نتایج در جدول(۴-۱۸) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
Ensemble	0.897457	0.904177	0.890715	0.892482	0.898291
Dtree	0.854389	0.848865	0.859959	0.858419	0.853532
SVM	0.817266	0.874502	0.760110	0.784412	0.826979
KNN	0.811423	0.948031	0.675020	0.744537	0.833991
Multi-layer-Perceptron	0.733901	0.800679	0.666877	0.706048	0.750133
Logistic Regression	0.696778	0.716247	0.677207	0.689046	0.702334
NB	0.566757	0.329873	0.803516	0.626718	0.432148

جدول(۴-۱۸): نتیجه یادگیری گروهی با انتخاب ویژگی backward

مقدار accuracy برای یادگیری گروهی ۰/۸۹۷ است که نسبت به حالت بدون انتخاب ویژگی کاهش یافته‌است. همچنین در تمامی معیارها، امتیاز یادگیری گروهی با انتخاب ویژگی backward از حالت بدون انتخاب ویژگی کمتر است.

فصل پنجم: نتیجه‌گیری و پیشنهاد

با توجه به جداولی که در فصل چهارم بررسی کردیم، باید تصمیم بگیریم از کدام الگوریتم انتخاب ویژگی و کدام تکنیک یادگیری ماشین برای پیش‌بینی استفاده کنیم.

ابتدا باید مشخص کنیم به دنبال افزایش کدام یک از پارامترها هستیم.

اگر به دنبال افزایش accuracy باشیم می‌توانیم از مجموعه داده که داده‌های پرت آن حذف شده است و با تکنیک SMOTETomek متعادل شده است بدون انتخاب ویژگی استفاده کنیم که در این حالت مقدار accuracy برای یادگیری بالای ۰/۹۲ خواهد بود که مقداری عالی است.

اگر به دنبال افزایش recall باشیم می‌توانیم از انتخاب ویژگی RFECV و الگوریتم KNN به تنهایی استفاده کنیم که در این حالت امتیاز recall به بالای ۹۷ درصد می‌رسد. البته این عدد نشان دهنده این است که به احتمال قوی overfitting اتفاق افتاده است که می‌توانیم از همان حالت اول استفاده کنیم که به ما مقدار بالای ۰/۹۲ می‌دهد.

اگر به دنبال افزایش specificity باشیم می‌توانیم از انتخاب ویژگی Forward و الگوریتم درخت تصمیم به تنهایی استفاده کنیم که به مقدار بالای ۰/۹۴ می‌رسیم.

اگر به دنبال افزایش f1-score باشیم می‌توانیم از همان حالت اول یعنی مجموعه داده بدون انتخاب ویژگی استفاده کنیم که به مقدار بالای ۰/۹۲ در یادگیری گروهی می‌رسیم.

اما اگر هدف ما این باشد که همه پارامترها را بهبود ببخشیم و همه پارامترها به بهترین مقدار نزدیک باشند:

- بدون انتخاب ویژگی می‌توانیم در یادگیری گروهی می‌توانیم از رگرسیون لجستیک به عنوان تخمین‌گر نهایی و الگوریتم‌های درخت تصمیم، SVM، MLP و KNN به عنوان تخمین‌گر های اولیه استفاده کنیم که به مقادیر جدول(۵-۱) می‌رسیم.

پارامترها	مقدار
Accuracy	**0/9228**
Recall	**0/9254**
Specificity	**0/9202**
Precision	**0/9209**
F1-score	**0/9232**

جدول(۵-۱): پارامترهای یادگیری گروهی

- با استفاده از انتخاب ویژگی variance threshold و همان ترکیب الگوریتم‌های مورد اول، به مقادیر جدول(۵-۲) می‌رسیم.

پارامترها	مقدار
Accuracy	**0/9142**
Recall	**0/9213**
Specificity	**0/9071**
Precision	**0/9087**
F1-score	**0/9150**

جدول(۵-۲): پارامترهای یادگیری گروهی

کد پیاده‌سازی این پایان‌نامه را می‌توانید از https://github.com/abolfazl-hosseinifar/diabetes_prediction دریافت کنید.

مراجع

[١] N. L. Fitriyani, M. Syafrudin, G. Alfian and J. Rhee, "Development of Disease Prediction Model Based on Ensemble Learning Approach for Diabetes and Hypertension," in *IEEE Access*, vol. 7, pp. 144777-144789, 2019, doi: 10.1109/ACCESS.2019.2945129.

[٢] https://www.kaggle.com/code/alexteboul/diabetes-health-indicators-dataset-notebook

[٣] https://towardsdatascience.com/imbalanced-classification-in-python-smote-tomek-links-method-6e48dfe69bbc

[٤] S. Bashir, U. Qamar and F. H. Khan, "IntelliHealth: A medical decision support application using a novel weighted multi-layer classifier ensemble framework", *J. Biomed. Inform.*, vol. 59, pp. 185-200, Feb. 2016.

[٥] A. Ozcift and A. Gulten, "Classifier ensemble construction with rotation forest to improve medical diagnosis performance of machine learning algorithms", *Comput. Methods Programs Biomed.*, vol. 104, no. 3, pp. 443-451, Dec. 2011.

Abstract

Early diseases prediction plays an important role for improving healthcare quality and can help individuals avoid dangerous health situations before it is too late. This paper proposes a disease prediction model (DPM) to provide an early prediction for type 2 diabetes and hypertension based on individual's risk factors data. The proposed DPM consists of isolation forest (iForest) based outlier detection method to remove outlier data, synthetic minority oversampling technique tomek link (SMOTETomek) to balance data distribution, and ensemble approach to predict the diseases. Four datasets were utilized to build the model and extract the most significant risks factors. The results showed that the proposed DPM achieved highest accuracy when compared to other models and previous studies. We also developed a mobile application to provide the practical application of the proposed DPM. The developed mobile application gathers risk factor data and send it to a remote server, so that an individual's current condition can be diagnosed with the proposed DPM. The prediction result is then sent back to the mobile application; thus, immediate and appropriate action can be taken to reduce and prevent individual's risks once unexpected health situations occur (i.e., type 2 diabetes and/or hypertension) at early stages.

Babol Noshirvani
University of Technology

Babol Noshirvani University of Technology

Department of Electrical and Computer Engineering

Subject

Development of Diabetes Disease Prediction Model Based on Ensemble Learning Approach

Supervisor

Dr. Fateme Zamani

By

Abolfazl Hosseinifar

November 2022