پایان نامه جهت دریافت درجه کارشناسی

رشته مهندسی کامپیوتر گرایش نرم افزار

عنوان

بررسی تاثیر انواع الگوریتم‌های انتخاب ویژگی در تکنیک‌های یادگیری ماشین به منظور افزایش دقت در پیش بینی و تشخیص بیماری‌های قلبی

استاد راهنما

دکتر فاطمه زمانی

نگارنده

امیررضا زارع

مرداد ۱۴۰۱


بسم الله الرحمن الرحيم

قدردانی

سپاس بی‌نهایت خدای را که دریای بی منت بخشش است و بال فضل بر کائنات گسترده و با منت خود مرا به زینت ایمان آراسته و در خیمه لطف منزل داده است. چگونه شکر او را گویم که منت بر من تمام کرده و از سر رحمت خود، خلعت تحصیل بر من پوشانیده. پروردگارا مرا یاری کن تا دانش اندکم گامی باشد برای تجلیل از تو و تعالی ساختن زندگی خود و دیگران و افتخاری باشد برای کشورم.

از زحمات استاد راهنمای محترم، خانم دکتر فاطمه زمانی که نه تنها به عنوان استاد بلکه همچون همکاری در تمام مراحل انجام این تحقیق از رهنمودها و کمک‌های بیدریغ ایشان بهره‌مند شده‌ام تشکر و قدردانی می‌کنم.

فهرست مطالب

فهرست شکل‌ها

فهرست جداول

چکیده

قلب مهم‌ترین قسمت بدن انسان است که وظیفه پمپاژ خون غنی از اکسیژن را از طریق شبکه‌ای از شریان‌ها و وریدها به سایر قسمت‌های بدن بر عهده دارد. هر نوع اختلالی که بر قلب ما تأثیر بگذارد یک بیماری قلبی است. بر اساس گزارش سازمان بهداشت جهانی منتشر شده در سال ۲۰۱۹، سالانه حدود ۱۷ میلیون نفر در سراسر جهان بر اثر بیماری قلبی جان خود را از دست می دهند.

تشخیص بیماری قلبی از طریق علائم اولیه یک چالش بزرگ در سناریوی کنونی جهان است و اگر به موقع تشخیص داده نشود ممکن است علت مرگ باشد. در کشورهای در حال توسعه که پزشکان متخصص قلب در مناطق دورافتاده، نیمه شهری و روستایی در دسترس نیستند، یک سیستم پشتیبانی تصمیم‌گیری دقیق می‌تواند نقشی حیاتی در تشخیص بیماری قلبی در مراحل اولیه داشته باشد. در این پایان نامه برای تشخیص بیماری قلبی 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 غیره با استفاده از انتخاب ویژگی‌های[۷] متفاوت از قبیل روش همبستگی[۸]، آستانه واریانس[۹]، پیشرو[۱۰]، عقبگرد[۱۱] و غیره بررسی شده‌اند و نتایج مختلفی بدست آمده که در اکثر موارد الگوریتم رگرسیون لجستیک عملکرد خوبی داشته است.

با استفاده از انتخاب ویژگی همبستگی و پیشرو با تکنیک رگرسیون لجستیک و همچنین با روش انتخاب ویژگی آستانه واریانس با تکنیک LightGBM[۱۲] روی این مجموعه داده به دقت بالای ۰.۸۴ می‌رسیم.

در این پایان نامه الگوریتم‌های یادگیری ماشین را با هرکدام از الگوریتم‌های انتخاب ویژگی بررسی و تحلیل خواهیم‌کرد و در پایان بهترین روش‌های پیش‌بینی بیماری قلبی را معرفی خواهیم کرد.

[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] Correlation
[9] Variance threshold
[10] Forward
[11] Backward
[12] Light Gradient Boosting Machine

واژگان کلیدی

مجموعه داده، رگرسیون، انتخاب ویژگی، یادگیری ماشین، همبستگی، آستانه واریانس، پیشرو، عقبگرد، مرزبندی، ماشین بردار پشتیبان، درخت تصمیم، جنگل تصادفی، شبکه عصبی، اعتبار سنجی متقابل، دقت، ماتریس سردرگمی

فصل اول: کلیات پژوهش

۱–۱: مقدمه

بیماری قلبی یکی از مهم‌ترین علل مرگ و میر در جهان امروز است. پیش‌بینی بیماری‌های قلبی عروقی یک چالش حیاتی در حوزه تحلیل داده‌های بالینی است.

امروزه کاملا واضح است که یادگیری ماشین در کمک به تصمیم‌گیری و پیش‌بینی از مقدار زیادی داده تولید شده توسط صنعت مراقبت‌های بهداشتی[۱۳]، موثر است. مطالعات مختلف تنها نگاهی اجمالی به پیش‌بینی بیماری قلبی با تکنیک‌های یادگیری ماشین دارند.

در این پایان نامه، ما روش‌های مختلف را بررسی کرده و در نهایت بهترین روش‌ها را بسته به هدف پیشنهاد می‌کنیم و آن هم یافتن ویژگی‌های مهم با استفاده از تکنیک‌های یادگیری ماشین است که منجر به بهبود دقت در پیش‌بینی بیماری‌های قلبی می‌شود و مدل‌های پیش‌بینی با ترکیب‌های مختلف ویژگی‌ها و چندین تکنیک طبقه‌بندی[۱۴] شناخته شده معرفی خواهندشد[۱][۲].

رویکرد ما در این پایان نامه شامل سه مرحله است:

۱. در مرحله اول ما یک مجموعه داده از مردم کلیولند[۱۵] را انتخاب می‌کنیم که شامل ۱۴ ویژگی بالینی مهم است به عنوان مثال سن، جنسیت، نوع درد قفسه سینه، ضربان بر ثانیه، کلسترول، قند خون ناشتا، حداکثر ضربان قلب، آنژین ناشی از ورزش و غیره.
۲. در مرحله دوم، 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.
۳. در مرحله سوم 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 الگوریتم

[13] Healthcare industry
[14] Classification
[15] Cleveland

انتخاب ویژگی و یادگیری ماشین را که بیشترین دقت در تشخیص بیماری قلبی را در این پایان نامه داشتند، معرفی می‌کنیم.

۱–۲: تعریف مسئله

بیماری قلبی که به عنوان بیماری قلبی عروقی شناخته می‌شود، شرایط مختلفی را در بر می‌گیرد که بر قلب تأثیر می‌گذارد و پایه اصلی مرگ و میر در سراسر جهان در طول چند دهه گذشته است. بسیاری از عوامل خطر در بیماری قلبی و نیاز به زمان برای دستیابی به رویکردهای دقیق، قابل اعتماد و معقول برای تشخیص زودهنگام برای دستیابی به مدیریت سریع بیماری، علم داده‌کاوی[۱۶] را به میان می‌کشد. داده کاوی یک تکنیک رایج برای پردازش داده‌های عظیم در حوزه‌های مختلف است. محققان چندین تکنیک داده کاوی و یادگیری ماشین را برای تجزیه و تحلیل داده‌های پیچیده پزشکی استفاده می‌کنند و به متخصصان مراقبت‌های بهداشتی کمک می‌کنند تا بیماری قلبی را پیش‌بینی کنند. این پایان نامه ، ویژگی‌های مختلف مربوط به بیماری قلبی و مدل‌ها را بر اساس الگوریتم‌های یادگیری نظارت‌شده[۱۷] مانند Naïve Bayes ، درخت تصمیم[۱۸]، KNN و الگوریتم جنگل تصادفی[۱۹] و غیره ارائه می‌کند.

الگوریتم‌های مختلف انتخاب ویژگی روی الگوریتم‌های یادگیری ماشین بررسی می‌شوند و بهترین الگوریتم انتخاب ویژگی و اینکه روی کدام الگوریتم یادگیری ماشین بیشترین دقت را به عنوان خروجی می‌دهد بررسی خواهدشد.

۱–۳: فرضیه‌ها و حوزه تحقیق

در این پایان نامه از مجموعه داده موجود از پایگاه داده کلیولند از پایگاه UCI بیماران قلبی استفاده می‌کند. مجموعه داده شامل ۳۰۲ نمونه و ۷۶ ویژگی است. از این ۷۶ ویژگی، ۱۴ ویژگی برای آزمایش در این پایان نامه درنظر گرفته شده‌است که برای اثبات عملکرد الگوریتم‌های مختلف مهم است. این پایان نامه با هدف پیش‌بینی احتمال ابتلا به بیماری قلبی در بیماران انجام شده است[۳].

[16] Data mining
[17] Supervised
[18] Decision tree
[19] Random forest

۱–۴: اهداف تحقیق و سوالات اصلی

با این فرض که از مجموعه داده مردم کلیولند به روش یادگیری ماشین می‌خواهیم احتمال وجود بیماری قلبی را در فردی تشخیص دهیم، هدف اصلی ما یافتن بهترین الگوریتم انتخاب ویژگی است که روی الگوریتم یادگیری ماشین مشخصی بهترین نتیجه و دقت را به عنوان خروجی تولید کرده‌است. باید یک روش انتخاب ویژگی کارآمد انتخاب کنیم که دقت آن روی الگوریتم مشخصی از یادگیری ماشین بالای ۸۰ درصد باشد.

سوالاتی که مطرح خواهد شد:

- ? انتخاب ویژگی چیست
- ? الگوریتم‌های یادگیری چه چیزهایی هستند
- ? بهترین روش انتخاب ویژگی چیست
- ? بهترین الگوریتم یادگیری ماشین در این پایان نامه کدام است

تا انتهای این پایان نامه به همه این سوالات پاسخ داده خواهدشد.

۱–۵: روش تحقیق

روش تحقیق به این صورت است که ابتدا برای تشریح کردن 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 آبجکت ها از الگوریتم‌های یادگیری ماشین به یک کتابخانه که برای اعتبار سنجی متقابل[۲۰] نوشته شده‌است می‌دهیم و این کتابخانه پارامترهای مختلفی برای ارزیابی به ما می‌دهد. سپس الگوریتم‌های انتخاب ویژگی را روی مجموعه داده اصلی اجرا می‌کنیم و هرکدام یک مجموعه داده جدید به ما می‌دهند که این مجموعه داده‌ها را به همراه لیستی از آبجکت‌ها به تابع k_fold_results() از کتابخانه cross-validation می‌دهیم و نتایج را با هم مقایسه خواهیم‌کرد و بهترین عملکرد را معرفی می‌کنیم.

[20] Cross-validation

۱–۶: ساختار پایان نامه

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

فصل دوم: مفاهیم پایه

۲–۱: مقدمه

برای پیش‌بینی بیماری قلبی از روی مجموعه داده باید با استفاده از الگوریتم‌های انتخاب ویژگی ابتدا مجموعه داده را ساده کنیم و مجموعه داده جدیدی بدست 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[۲۱] 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[۴].

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 دسته‌بندی آن‌ها مهم است و یادگیرنده باید در داده‌ها به دنبال ساختاری خاص بگردد.

[21] Tagged data

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 روش‌های یادگیری طبیعی، می‌تواند غیر قابل پیش‌بینی باشد[۵].

۲-۲-۳: یادگیری تقویتی (Reinforcement 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 می‌دهد. 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 تصمیم گیری مارکوف[۲۲]را نام برد[۶][۷].

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 می‌رود.

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[۸].

[22] Markov Process

۲-۲-۵: طبقه‌بندی (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[۲۳]، تکنیک‌های مبتنی بر پرسپترون[۲۴]) و آمار (شبکه‌های بیزی[۲۵]، تکنیک‌های مبتنی بر نمونه[۲۶]) 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 قلبی است[۹].

۲-۲-۶: داده های 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 داده‌های 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 پایان نامه علاوه بر آن، از روش‌های جدیدتر که k-fold و cross validation است نیز استفاده شده که چندین بار داده‌های آزمایش و آموزش را جابجا می‌کند تا به بهترین دقت برسد که در ادامه این پایان نامه توضیح داده خواهدشد[۱۰].

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 پایان نامه به معنای بررسی میزان دقت الگوریتم‌های یادگیری ماشین در پیش‌بینی بیماری قلبی است[۱۱].

---

[23] logic
[24] Perceptron
[25] Bayesian network
[26] Sample-based techniques

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 مدل‌های ناپارامتریک[۲۷] و غیرخطی[۲۸] 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 محدودکردن جزئیاتی هستند که مدل یاد می‌گیرد. به عنوان مثال، درختان تصمیم یک الگوریتم یادگیری ماشین ناپارامتریک هستند که بسیار منعطف هستند و در معرض داده‌های آموزشی بیش از حد مناسب هستند. این مشکل را می‌توان با هرس‌کردن[۲۹] یک درخت پس از یادگیری به منظور حذف بخشی از جزئیاتی که برداشت کرده است، برطرف کرد[۱۲][۱۳].

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

عدم تناسب یا کم برازش 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[۱۴].

[27] Nonparametric
[28] Nonlinear
[29] pruning

۲-۲-۱۰: یادگیری عمیق (Deep learning)

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

روش‌هایی از یادگیری ماشین بر پایه استفاده از شبکه‌های عصبی عمیق که از داده‌های 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 شبکه‌های عصبی مغز انسان مفاهیم سطح بالا و انتزاعی را یاد بگیرند. استفاده از یادگیری عمیق کمک می‌کند که ماشین‌ها بتوانند تصمیم‌هایی شبیه تصمیم‌های انسانی بگیرند.

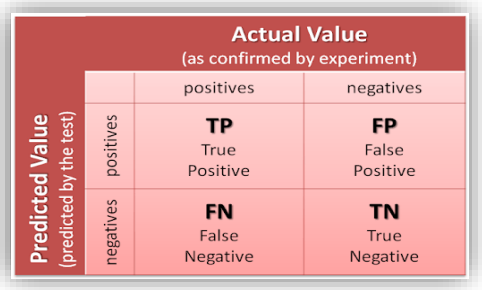
در یادگیری عمیق از چند لایه مختلف شبکه عصبی استفاده می‌شود. هرکدام از این لایه‌ها بخش‌هایی از اطلاعات ورودی را تحلیل می‌کنند. این لایه‌های چندگانه، امکان پیش‌بینی را در یادگیری عمیق افزایش می‌دهند. تعداد این لایه‌ها گاهی می‌تواند تا ۱۵۰ لایه برسد[۱۵].

در یادگیری عمیق از روش‌های متفاوتی استفاده می‌شود. این روش‌ها بسته به کاربردهای متفاوت یادگیری عمیق و نوع داده‌های ورودی و خروجی موردنیاز انتخاب می‌شود. مانند:

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

۲-۲-۱۱: ماتریس سردرگمی (confusion matrix)

شکل(۲-۱) Confusion matrix را نشان می‌دهد:



شکل(۲-۱): 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 نمونه‌های یک کلاس پیش‌بینی 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 پیش بینی‌های 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 پیش بینی‌های منفی را نشان می‌دهد که در واقعیت مثبت است.

۲-۲-۱۲: دقت (Accuracy)

به ما می‌گوید که هر چند وقت یک‌بار 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 پیش بینی می‌کند. دقت، در این مورد ۹۰% خواهد بود که دقت عالی است؛ اما چیزی در مورد خطاهایی که مدل‌های یادگیری ماشین ما روی داده‌های جدیدی که قبلاً ندیده‌ایم ایجاد می‌کنند، به ما نمی‌گوید[۴۹].

۲-۲-۱۳: امتیاز یادآوری (Recall)

امتیاز یادآوری مدل نشان دهنده توانایی مدل در پیش‌بینی صحیح موارد مثبت از موارد مثبت واقعی است؛ این برخلاف دقتی است که تعداد پیش‌بینی‌های انجام‌شده توسط مدل‌ها را از بین همه پیش‌بینی‌های مثبت انجام‌شده 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 مدل یادگیری ماشین شما را به درستی به عنوان مثبت پیش‌بینی کرده‌است. 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 ماشینی 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 می‌دهد 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[۵۰].

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

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

فرمول(۲-۱): محاسبه recall

۲-۲-۱۴: 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 خواهدداشت که مثبت پیش‌بینی شده و می‌تواند به عنوان مثبت کاذب نامیده شود. این نسبت را می‌توان نرخ منفی واقعی[۳۰] نیز نامید. مجموع نرخ منفی واقعی و نرخ مثبت کاذب همیشه یک خواهدبود. 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[۵۱].

از نظر ریاضی، Specificity را می‌توان با فرمول(۲-۲) محاسبه کرد:

[30] TNR

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

فرمول(۲-۲): محاسبه specificity

۲-۲-۱۵: درستی (Precision)

در ساده‌ترین عبارت، Precision نسبت بین مثبت‌های واقعی و همه مثبت‌ها است. برای بیان مشکل ما، این معیار نسبت بیمارانی است که ما درست بصورت مثبت پیش‌بینی کردیم به کل بیمارانی که مثبت پیش‌بینی کردیم[۵۲]. از نظر ریاضی درستی را می‌توان با فرمول(۲-۳) محاسبه کرد.

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

فرمول(۲-۳): محاسبه precision

۲-۲-۱۶: 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 ماشینی 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 می‌کند دقت یا امتیاز یادآوری را بهینه کند و در نتیجه عملکرد مدل آسیب می‌بیند[۵۳].

F1-Score از فرمول(۲-۴) بدست می‌آید:

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

فرمول(۲-۴): محاسبه f1-Score

۲-۳: الگوریتم های انتخاب ویژگی چیست؟

انتخاب ویژگی، به عنوان یک استراتژی پیش‌پردازش[۳۱] 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 زیرمجموعه‌ای 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 بصری‌سازی آن
- ☑ کاهش داده کلی، محدود کردن نیازمندی‌های ذخیره‌سازی و احتمالا کمک به کاهش هزینه‌ها
- ☑ کاهش مجموعه ویژگی‌ها، ذخیره‌سازی منابع در دور بعدی گردآوری داده یا در طول بهره‌برداری
- ☑ 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[۱۶].

در ادامه الگوریتم های انتخاب ویژگی که در این پروژه استفاده شده‌اند را توضیح خواهیم داد.

۲-۳-۱: انتخاب ویژگی به روش همبستگی (Correlation)

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 بدین 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 درنظر گرفته

[31] pre-processing

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[۱۷][۱۸].

به طور کلی دو رویکرد برای اندازه‌گیری همبستگی بین دو متغیر تصادفی وجود دارد. یکی بر اساس همبستگی خطی کلاسیک[۳۲] و دیگری مبتنی بر نظریه اطلاعات[۳۳] 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) بدست 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 ضریب 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[۳۴] 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[۳۵] و حداکثر شاخص فشرده‌سازی اطلاعات[۳۶].

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

- ☑ به حذف ویژگی‌هایی با همبستگی خطی نزدیک به صفر با نتیجه نهایی کمک می‌کند.
- ☑ 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[۱۹].

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

---

[32] Classical linear correlation
[33] information theory
[34] Independent
[35] Least square regression error
[36] Maximum information compression index

```
#Correlation with output variable
cor_target = abs(cor["target"])
#Selecting highly correlated features
relevant_features = cor_target[cor_target>0.4]
df_select_correlation = np.array(relevant_features.index)
df_correlation = df_minmax.copy()
df_correlation = df_minmax[df_select_correlation]
df_correlation.head()
```

شکل(۲-۲): الگوریتم Correlation

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

	cp	thalach	exang	oldpeak	ca	thal	target
0	1.000000	0.282443	1	0.241935	1.000000	0.0	1
1	1.000000	0.442748	1	0.419355	0.666667	1.0	1
2	0.666667	0.885496	0	0.564516	0.000000	0.0	0
3	0.333333	0.770992	0	0.225806	0.000000	0.0	0
4	0.333333	0.816794	0	0.129032	0.000000	0.0	0

جدول(۲-۱): خروجی انتخاب ویژگی correlation

۲-۳-۲: انتخاب ویژگی به روش Variance Threshold

در یادگیری ماشین گرفتن واریانس به ما کمک می‌کند تا تشخیص دهیم رکوردهای[37] 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:

[37] Records

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

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

این انتخاب ویژگی را می‌توان 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[۲۰].

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

```
VarThreshOld = VarianceThreshold(threshold=0.04)
VarThreshOld.fit(df_minmax)
temp=VarThreshOld.get_support()
df_minmax_columns = df_minmax.columns
df_select_ThreshOld=[]
for index in range(0,len(temp)):
    if (temp[index]==True):
        df_select_ThreshOld.append(df_minmax_columns[index])

df_ThreshOld = df_minmax.copy()
df_ThreshOld = df_ThreshOld[df_select_ThreshOld]
df_ThreshOld.head()
```

شکل(۲-۳): الگوریتم Variance threshold

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

	gender	cp	fbs	restecg	exang	thal	target	target
0	1	1.000000	0	1.0	1	0.0	1	1
1	1	1.000000	0	1.0	1	1.0	1	1
2	1	0.666667	0	0.0	0	0.0	0	0
3	0	0.333333	0	1.0	0	0.0	0	0
4	1	0.333333	0	0.0	0	0.0	0	0

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

۲-۳-۳: انتخاب ویژگی به روش حذف ویژگی بازگشتی (RFE)[۳۸]

حذف ویژگی بازگشتی اساساً یک انتخاب معکوس از پیش‌بینی کننده‌ها است. این تکنیک با ساختن یک مدل بر روی کل مجموعه پیش‌بینی کننده‌ها و محاسبه امتیاز اهمیت برای هر پیش‌بینی آغاز می‌شود. سپس کمترین پیش‌بینی کننده (های) مهم حذف می‌شوند، مدل دوباره ساخته می‌شود و امتیازهای اهمیت دوباره محاسبه می‌شوند. در عمل، تحلیلگر تعداد زیر مجموعه‌های پیش‌بینی کننده را برای ارزیابی و همچنین اندازه هر زیر مجموعه را مشخص می‌کند. بنابراین اندازه زیر مجموعه یک پارامتر تنظیم برای RFE است. اندازه زیر مجموعه‌ای که معیارهای عملکرد را بهینه می‌کند برای انتخاب پیش‌بینی کننده‌ها بر اساس رتبه‌بندی اهمیت استفاده می‌شود و سپس از زیر مجموعه 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 نمی توان 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 پیش‌بینی کننده‌ها 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[۲۱][۲۲].

[38] Recursive Feature Elimination

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

```
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(۲-۴): الگوریتم RFE

با توجه به مجموعه داده‌ای که داریم، ویژگی‌های جدول(۲-۳) از الگوریتم شکل(۲-۴) استخراج می‌شود:

	gender	cp	trestbps	thalach	oldpeak	ca	thal	target
0	1	1.000000	0.622642	0.282443	0.241935	1.000000	0.0	1
1	1	1.000000	0.245283	0.442748	0.419355	0.666667	1.0	1
2	1	0.666667	0.339623	0.885496	0.564516	0.000000	0.0	0
3	0	0.333333	0.339623	0.770992	0.225806	0.000000	0.0	0
4	1	0.333333	0.245283	0.816794	0.129032	0.000000	0.0	0

جدول(۲-۳): خروجی انتخاب ویژگی RFE

۲-۳-۴: انتخاب ویژگی به روش رو به جلو 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 می‌کند 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 می دهد[۲۳].

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

```
knn = KNeighborsClassifier(n_neighbors=5)
X = df[['age','gender','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpeak','slope','ca','thal']]
Y = df[['target']]

sfs_forward = SFS(knn,
          k_features=7,
          forward=True,
          floating=True,
          scoring='accuracy',
          cv=5,
          n_jobs=-1)
sfs_forward = sfs_forward.fit(X, Y,custom_feature_names=df_minmax_col)

pd.DataFrame.from_dict(sfs_forward.get_metric_dict()).T
```

	feature_idx	cv_scores	avg_score	feature_names	ci_bound	std_dev	std_err
1	(12,)	[0.8032786885245902, 0.8524590163934426, 0.716...	0.767814	(thal,)	0.085846	0.066791	0.033396
2	(11, 12)	[0.819672131147541, 0.8852459016393442, 0.7333...	0.784317	(ca, thal)	0.089933	0.069971	0.034985
3	(2, 11, 12)	[0.8688524590163934, 0.9180327868852459, 0.816...	0.84071	(cp, ca, thal)	0.059356	0.046181	0.023091
4	(2, 8, 11, 12)	[0.8360655737704918, 0.9016393442622951, 0.816...	0.840874	(cp, exang, ca, thal)	0.040418	0.031446	0.015723
5	(2, 5, 8, 11, 12)	[0.8360655737704918, 0.9180327868852459, 0.816...	0.84082	(cp, fbs, exang, ca, thal)	0.050552	0.039331	0.019665
6	(1, 2, 5, 8, 11, 12)	[0.7868852459016393, 0.9016393442622951, 0.816...	0.831038	(gender, cp, fbs, exang, ca, thal)	0.049287	0.038347	0.019173
7	(1, 2, 5, 8, 10, 11, 12)	[0.8360655737704918, 0.9508196721311475, 0.8, ...	0.84071	(gender, cp, fbs, exang, slope, ca, thal)	0.080002	0.062244	0.031122

```
df_select_forward = ['gender', 'cp', 'fbs', 'exang', 'slope', 'ca', 'thal']
df_select_forward.append('target')
df_forward = df_minmax.copy()
df_forward = df_forward[df_select_forward]
df_forward.head()
```

شکل(۲-۵): الگوریتم forward

با توجه به مجموعه داده‌ای که داریم، ویژگی‌های جدول(۲-۴) از الگوریتم شکل(۲-۵) استخراج می‌شود:

	gender	cp	fbs	exang	slope	ca	thal	target
0	1	1.000000	0	1	0.5	1.000000	0.0	1
1	1	1.000000	0	1	0.5	0.666667	1.0	1
2	1	0.666667	0	0	1.0	0.000000	0.0	0
3	0	0.333333	0	0	0.0	0.000000	0.0	0
4	1	0.333333	0	0	0.0	0.000000	0.0	0

جدول(۲-۴): خروجی انتخاب ویژگی forward

۲-۳-۵: 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[۲۴].

در شکل(۲-۶) کد پیاده‌سازی این الگوریتم را مشاهده می‌کنید:

```
sfs_Backward = SFS(knn,
          k_features=7,
          forward=False,
          floating=True,
          scoring='accuracy',
          cv=3,
          n_jobs=-1)
sfs_Backward = sfs_Backward.fit(X, Y,custom_feature_names=df_minmax_col)
import pandas as pd
pd.DataFrame.from_dict(sfs_Backward.get_metric_dict()).T
```

	feature_idx	cv_scores	avg_score	feature_names	ci_bound	std_dev	std_err
13	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)	[0.6237623762376238, 0.6138613861386139, 0.68]	0.639208	(age, gender, cp, trestbps, chol, fbs, restecg...	0.065544	0.029126	0.020595
12	(0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12)	[0.6237623762376238, 0.6138613861386139, 0.7]	0.645875	(age, gender, cp, trestbps, chol, fbs, thalach...	0.086605	0.038485	0.027213
11	(0, 1, 2, 3, 4, 5, 7, 8, 10, 11, 12)	[0.6237623762376238, 0.6138613861386139, 0.7]	0.645875	(age, gender, cp, trestbps, chol, fbs, thalach...	0.086605	0.038485	0.027213
10	(0, 1, 2, 3, 4, 5, 7, 8, 11, 12)	[0.6237623762376238, 0.6138613861386139, 0.7]	0.645875	(age, gender, cp, trestbps, chol, fbs, thalach...	0.086605	0.038485	0.027213
9	(0, 1, 2, 3, 4, 5, 7, 11, 12)	[0.6237623762376238, 0.6138613861386139, 0.7]	0.645875	(age, gender, cp, trestbps, chol, fbs, thalach...	0.086605	0.038485	0.027213
8	(0, 1, 2, 3, 4, 7, 11, 12)	[0.6237623762376238, 0.6138613861386139, 0.7]	0.645875	(age, gender, cp, trestbps, chol, thalach, ca,...	0.086605	0.038485	0.027213
7	(0, 2, 3, 4, 7, 11, 12)	[0.6237623762376238, 0.6138613861386139, 0.7]	0.645875	(age, cp, trestbps, chol, thalach, ca, thal)	0.086605	0.038485	0.027213

```
df_select_backward = ['age', 'cp', 'trestbps', 'chol', 'thalach', 'ca', 'thal']
df_select_backward.append('target')
df_backward = df_minmax.copy()
df_backward = df_backward[df_select_backward]
df_backward.head()
```

شکل(۲-۶): الگوریتم Backward

با توجه به مجموعه داده‌ای که داریم، ویژگی‌های جدول(۲-۵) از الگوریتم شکل(۲-۶) استخراج می‌شود:

	age	cp	trestbps	chol	thalach	ca	thal	target
0	0.791667	1.000000	0.622642	0.365297	0.282443	1.000000	0.0	1
1	0.791667	1.000000	0.245283	0.235160	0.442748	0.666667	1.0	1
2	0.166667	0.666667	0.339623	0.283105	0.885496	0.000000	0.0	0
3	0.250000	0.333333	0.339623	0.178082	0.770992	0.000000	0.0	0
4	0.562500	0.333333	0.245283	0.251142	0.816794	0.000000	0.0	0

جدول(۲-۵): خروجی انتخاب ویژگی Backward

۲-۳-۶: انتخاب ویژگی به روش Ridge

یکی از مهم‌ترین موارد در مورد رگرسیون **Ridge** این است که بدون هدر دادن اطلاعات در مورد پیش‌بینی‌ها، سعی می‌کند متغیرهایی را تعیین کند که اثرات دقیقاً صفر دارند. رگرسیون **Ridge** محبوب است زیرا از منظم-سازی برای پیش‌بینی استفاده می‌کند و منظم‌سازی برای حل مشکل بیش‌برازش درنظر گرفته شده‌است.

ما عمدتاً متوجه می‌شویم که بیش‌برازش جایی است که اندازه داده‌ها بسیار بزرگ است و رگرسیون خطی با جریمه کردن ضریب ویژگی‌ها کار می‌کند و همچنین خطاهای پیش‌بینی را به حداقل می‌رساند.

از رگرسیون **Ridge** برای خلاص شدن از شر برازش استفاده می‌شود که با تطبیق مدل تنها با ویژگی‌های مهم می‌توان آن را کاهش داد. **Ridge** همچنین می‌تواند در انتخاب ویژگی به ما کمک کند تا ویژگی‌های مهم مورد نیاز برای اهداف مدل‌سازی را دریابیم. می‌توان رگرسیون **Ridge** را به عنوان روشی برای تخمین ضریب مدل‌های رگرسیون چندگانه در نظر گرفت. ما عمدتاً نیاز رگرسیون **Ridge** را در جایی پیدا می‌کنیم که متغیرها در داده‌ها همبستگی بالایی دارند. همچنین می‌توانیم رگرسیون **Ridge** را به عنوان راه‌حلی برای عدم دقت تخمین‌گر حداقل مربعات[۳۹] در نظر بگیریم که در آن متغیرهای مستقل هر مدل رگرسیون خطی همبستگی بالایی دارند.

با استفاده از **Ridge** می‌توانیم تخمین دقیق‌تری به دست آوریم زیرا واریانس کوچک‌تر و برآوردگر میانگین مربع را ارائه می‌کند[۲۵].

در شکل(۲-۷) کد پیاده‌سازی این الگوریتم را مشاهده می‌کنید:

[39] Least Squares Estimator (LSE)

```python
ridgeRegressor = Ridge(alpha = 0.5)
ridgeRegressor.fit(X_train_minmax, y_train_minmax)
y_predicted_ridge = ridgeRegressor.predict(X_test_minmax)

R_squared = r2_score(y_predicted_ridge,y_test_minmax)
print("R squared Error on test set : ", R_squared)
feature_names = df.columns

coefficient_df = pd.DataFrame()
coefficient_df["Column_Name"] = feature_names
coefficient_df['Coefficient_Value'] = pd.Series(ridgeRegressor.coef_)
print(coefficient_df.head(15))
plt.rcParams["figure.figsize"] = (15,6)
plt.bar(coefficient_df["Column_Name"],coefficient_df["Coefficient_Value"])
ridgeRegressor.coef_ = abs(ridgeRegressor.coef_)
df_select_ridge= []
for i in range (0,len(ridgeRegressor.coef_)):
    if (ridgeRegressor.coef_[i] > 0.1):
        df_select_ridge.append(df_minmax_columns[i])
df_select_ridge.append('target')
df_ridge = df_minmax.copy()
df_ridge = df_ridge[df_select_ridge]
df_ridge.head()
```

شکل(۲-۷): الگوریتم Ridge

با توجه به مجموعه داده‌ای که داریم، ویژگی‌های جدول(۲-۶) از الگوریتم شکل(۲-۷) استخراج می‌شود:

	gender	cp	trestbps	thalach	oldpeak	ca	thal	target
0	1	1.000000	0.622642	0.282443	0.241935	1.000000	0.0	1
1	1	1.000000	0.245283	0.442748	0.419355	0.666667	1.0	1
2	1	0.666667	0.339623	0.885496	0.564516	0.000000	0.0	0
3	0	0.333333	0.339623	0.770992	0.225806	0.000000	0.0	0
4	1	0.333333	0.245283	0.816794	0.129032	0.000000	0.0	0

جدول(۲-۶): خروجی انتخاب ویژگی Ridge

۲-۴: روش های ارزیابی

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

۲-۴-۱: الگوریتم Naïve Bayes

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

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

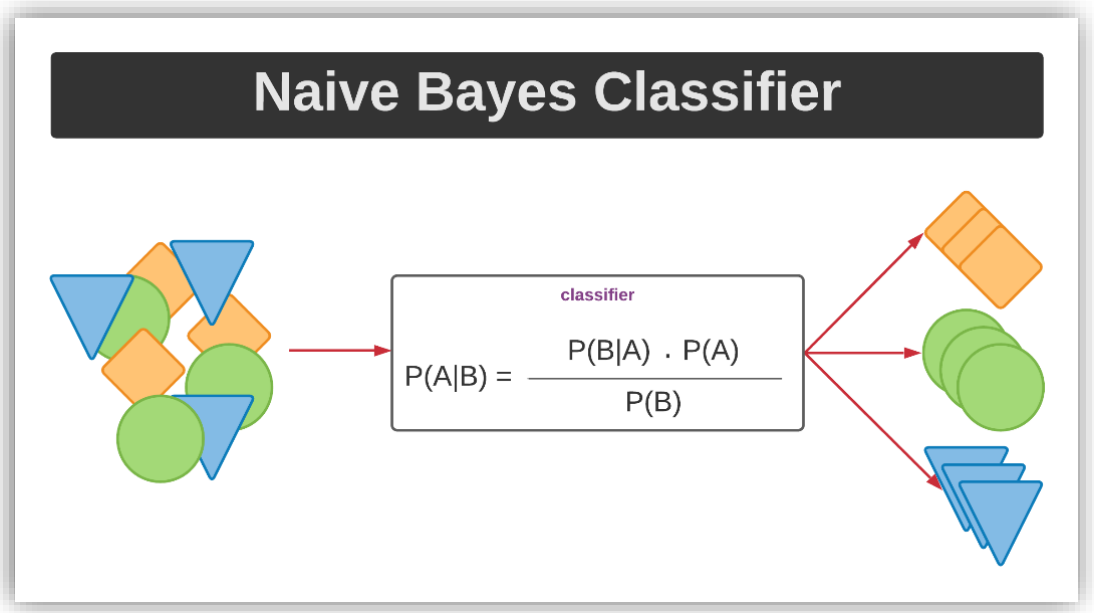
فرمول(۲-۷): محاسبه احتمال بیز

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 متخصص 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 ویژگی‌ها ترکیبی از مقادیر عددی و پیوسته باشند، تعیین احتمال پیچیده‌تر می‌شود. جنبه مثبت این مدل به دلیل یک فرض بسیار دقیق است که پیش‌بینی کننده‌ها 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 طبقه‌بندی کننده‌های پیچیده‌تر رقابت می‌کند[۲۶].

شکل(۲-۸) Naïve Bayes را نشان می‌دهد:



شکل(۲-۸): **Naïve Bayes**

۲-۴-۲: الگوریتم ماشین‌های بردار پشتیبان (SVM)[40]

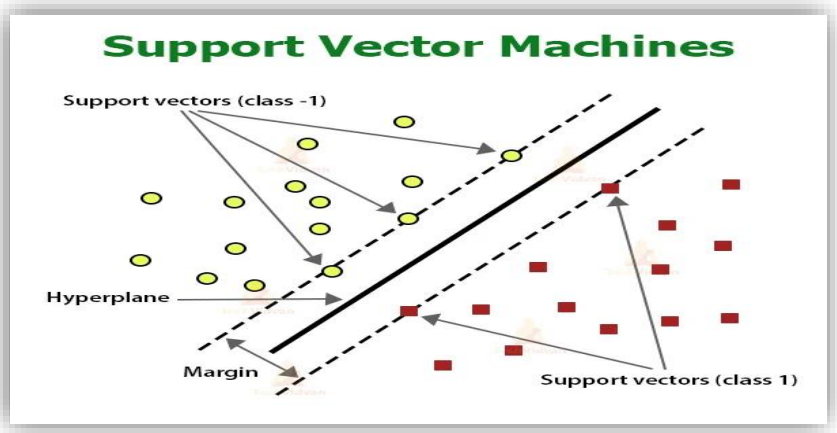
یکی از رایج‌ترین و هیجان‌انگیزترین مدل‌های یادگیری نظارت‌شده با الگوریتم‌های یادگیری مرتبط که داده‌ها را تجزیه و تحلیل می‌کند و الگوها را تشخیص می‌دهد، ماشین‌های بردار پشتیبان است. آن‌ها برای حل مسائل رگرسیون و طبقه‌بندی استفاده می‌شوند با این حال، آن‌ها بیشتر در حل مسائل طبقه‌بندی استفاده می‌شوند. ماشین‌های 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 نمونه‌های جدیدی را به یک دسته یا دسته دیگر اختصاص می‌دهد و آن را به یک طبقه‌بندی خطی باینری غیراحتمالی[41] تبدیل می‌کند.

هدف از بکارگیری ماشین های بردار پشتیبان یافتن بهترین خط در دو بعد یا بهترین ابرصفحه یا ابر جداکننده[42] در بیش از دو بعد است تا به ما کمک کند فضای خود را به کلاس‌ها تفکیک کنیم. ابر صفحه (خط) از طریق

[40] Support vector machines
[41] nonprobabilistic binary linear classifier
[42] hyperplane

بیشترین حاشیه[۴۳] یعنی حداکثر فاصله بین نقاط داده هر دو کلاس پیدا می شود. 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[۲۷].

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



شکل(۲-۹): ماشین‌های بردار پشتیبان

۲-۴-۳: الگوریتم Logistic Regression

رگرسیون لجستیک تکنیک دیگری است که توسط یادگیری ماشین از حوزه آمار گرفته شده است. این روش برای مسائل طبقه‌بندی باینری[۴۴] (مسائل با دو مقدار کلاس) است.

رگرسیون لجستیک یکی از محبوب‌ترین الگوریتم‌های یادگیری ماشینی است که تحت تکنیک یادگیری نظارتی قرار می‌گیرد. برای پیش‌بینی متغیر وابسته طبقه‌بندی با استفاده از مجموعه معینی از متغیرهای مستقل استفاده می‌شود[۲۸].

تابع لجستیک، که تابع سیگموئید[۴۵] 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

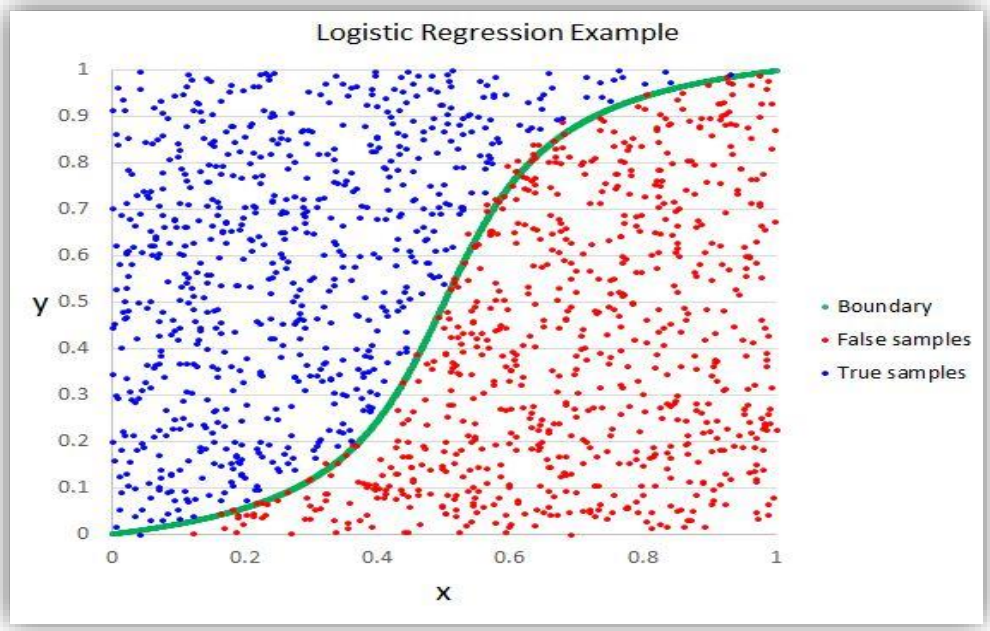
[43] maximum margin
[44] binary
[45] sigmoid

یا 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 می‌شود، درحالیکه از رگرسیون لجستیک برای حل مسائل طبقه‌بندی استفاده می‌شود برای مثال منحنی تابع لجستیک احتمال مواردی مانند سرطانی بودن یا نبودن سلول‌ها، بیماری قلبی 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 داده-های 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 طبقه-بندی را تعیین کند[۲۹]. شکل(۲-۱۰) تابع لجستیک را نشان می‌دهد.



شکل(۲-۱۰): 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 بدست آوردن معادلات رگرسیون لجستیک در زیر آورده شده‌است.

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

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

فرمول(۲-۸): معادله خط مستقیم

در رگرسیون لجستیک y فقط می‌تواند بین ۰ و ۱ باشد، بنابراین معادله فوق را بر $(1-y)$ تقسیم می‌کنیم مطابق با فرمول(۲-۹):

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

فرمول(۲-۹): نرمال سازی معادله خط مستقیم

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

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

فرمول(۲-۱۰): معادله رگرسیون لجستیک

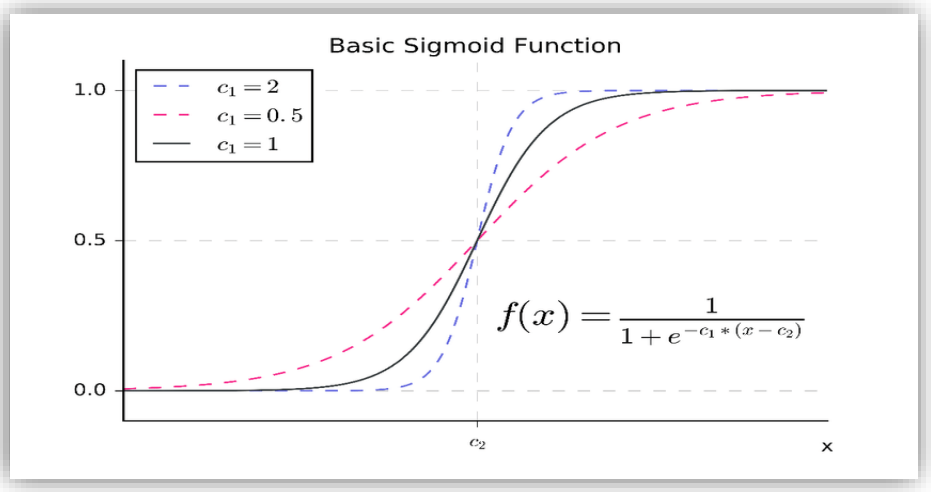
معادله فوق معادله نهایی رگرسیون لجستیک است[۳۰].

بر اساس دسته بندی‌ها، 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 غیره.
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، می‌تواند ۳ یا چند نوع متغیر وابسته مرتب شده مانند کم، متوسط یا زیاد وجود داشته باشد.

شکل(۲-۱۱) تابع سیگوئید را نشان می‌دهد:



شکل(۲-۱۱): تابع sigmoid

۲-۴-۴: الگوریتم k-نزدیک ترین 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 می‌دهد 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 می‌کند. 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 طبقه‌بندی 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. الگوریتم k-نزدیک‌ترین همسایه یک الگوریتم ناپارامتریک است، به این معنی که هیچ فرضی در مورد داده‌های اساسی ایجاد نمی‌کند. به آن الگوریتم یادگیرنده تنبل[۴۶] نیز می‌گویند زیرا بلافاصله از مجموعه

[46] lazy learner

آموزشی یاد نمی‌گیرد، در عوض مجموعه داده را ذخیره می‌کند و در زمان طبقه‌بندی، عملی را روی مجموعه داده انجام می‌دهد[۳۱].

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.

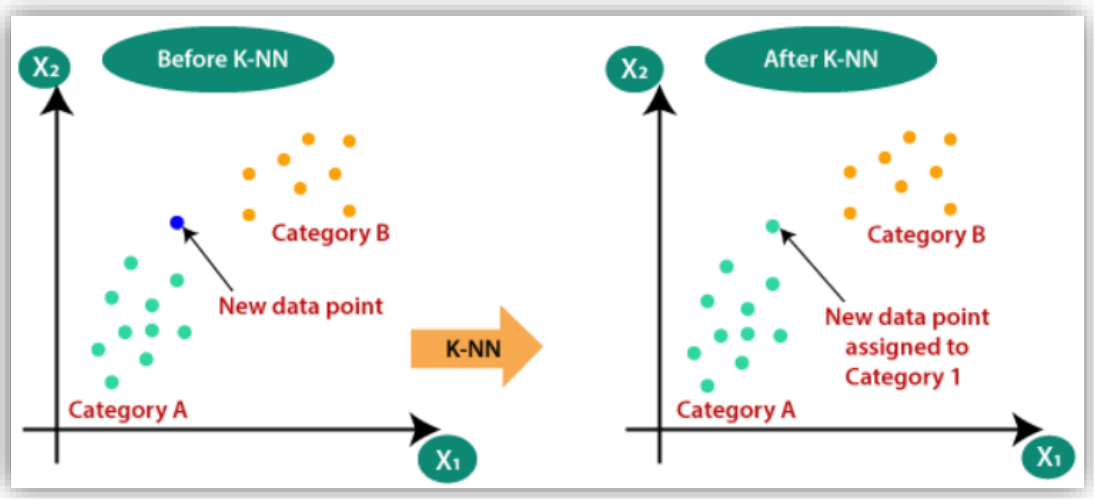
برای مثال فرض کنید تصویری از موجودی داریم که شبیه گربه و سگ است، اما می‌خواهیم بدانیم که گربه است یا سگ. بنابراین برای این شناسایی می‌توانیم از الگوریتم k-نزدیک‌ترین همسایه استفاده کنیم، زیرا بر روی معیار شباهت کار می‌کند. مدل KNN ما ویژگی‌های مشابه مجموعه داده‌های جدید را با تصاویر گربه‌ها و سگ‌ها پیدا می‌کند و بر اساس مشابه‌ترین ویژگی‌ها، آن را در دسته‌بندی گربه یا سگ قرار می‌دهد که در شکل(۲-۱۲) نشان داده شده‌است:

شکل(۲-۱۲): KNN

سوال: چرا به الگوریتم k-نزدیک ترین همسایه نیاز داریم؟

فرض کنید دو دسته A و B وجود دارند و ما یک نقطه داده جدید X1 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(۲-۱۳) را در نظر بگیرید:



شکل(۲-۱۳): 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 به ترتیب صعودی در نظر می‌گیرد.

۴) در بین این k همسایه‌ها، 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(۲-۱۴) را در نظر بگیرید:

شکل(۲-۱۴): نقطه داده‌ای جدید

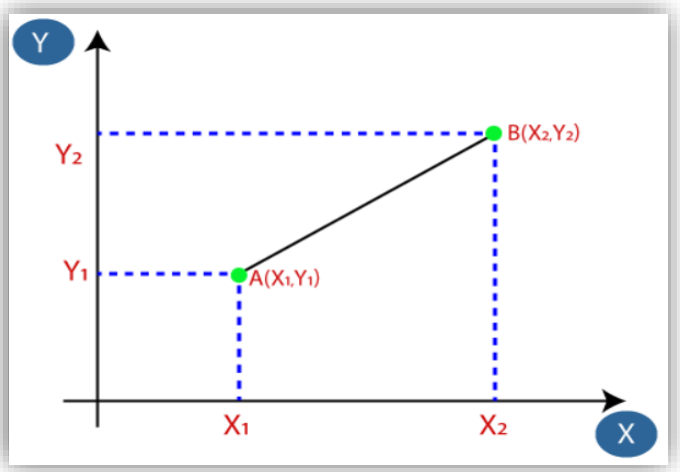
ابتدا تعداد همسایگان را انتخاب می‌کنیم، بنابراین k را مثلا برابر ۵ انتخاب می‌کنیم.

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

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

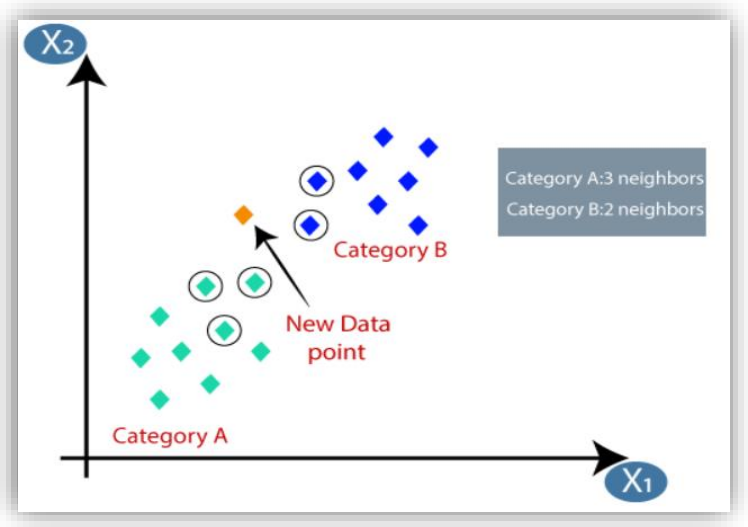
فرمول(۲-۱۱): محاسبه فاصله دو نقطه

شکل(۲-۱۵) فاصله اقلیدسی دو نقطه را نشان می‌دهد:



شکل(۲-۱۵): فاصله بین دو نقطه A و B

با محاسبه فاصله اقلیدسی نزدیک‌ترین همسایگان را به عنوان سه همسایه نزدیک در رده A و دو نزدیک‌ترین همسایه در دسته B به دست آوردیم[۳۲]. شکل(۲-۱۶) را در نظر بگیرید:



شکل(۲-۱۶): همسایه‌های نقطه داده ای 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.

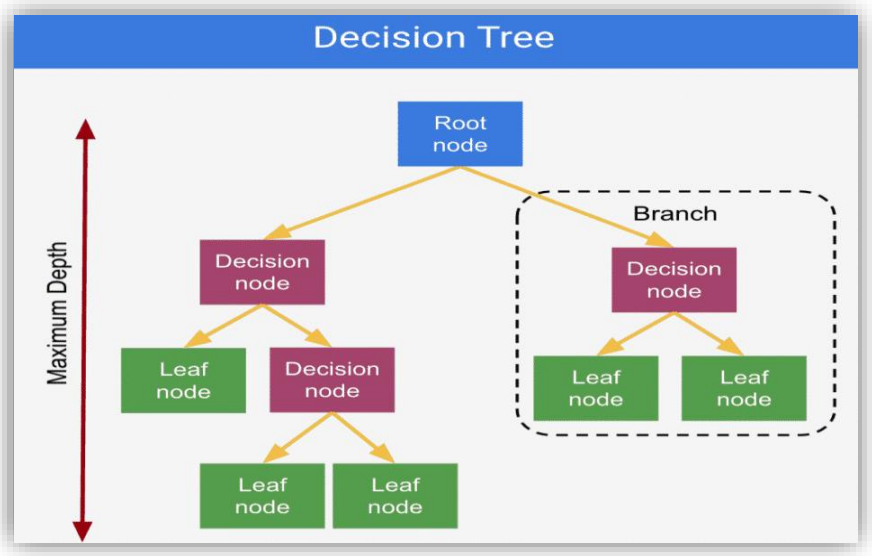
معایب:

- ☒ همیشه نیاز به تعیین مقدار K دارد که ممکن است اکثر مواقع پیچیده باشد.
- ☒ هزینه محاسبات به دلیل محاسبه فاصله بین نقاط داده برای همه نمونه‌های آموزشی بالا است[۳۳].

۲-۴-۵: الگوریتم درخت تصمیم (Decision tree)

درخت تصمیم دقیقاً مانند یک درخت است با این تفاوت که از ریشه به سمت پایین (برگ)[۴۷] 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 می‌شود 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 خواهدرفت. 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 می‌شوند. برگ‌های این درخت با یک کلاس و یا یک طبقه از جواب‌ها مشخص می‌شوند[۳۴].

شکل(۲-۱۷) درخت تصمیم را نشان می‌دهد:



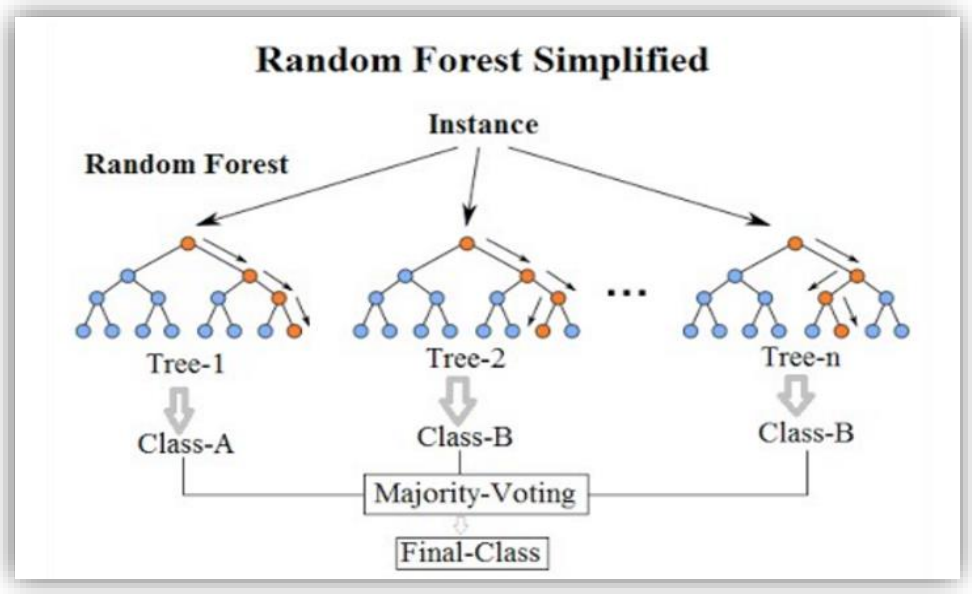
شکل(۲-۱۷): درخت تصمیم

[47] leaf

۲-۴-۶: 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 می‌دهد که درنهایت بین پیش‌بینی‌های انجام‌شده، رأی‌گیری انجام می‌دهد[۳۵].

شکل(۲-۱۸) درخت تصمیم را نشان می‌دهد:



شکل(۲-۱۸): جنگل تصادفی

۲-۴-۷: الگوریتم شبکه های عصبی پرسپترون چند لایه (MLP)[۴۸]

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

[48] Multilayer Preceptron neural networks

می‌دهد، یک الگوریتم خطی می‌دانند. شبکه عصبی پرسپترون از جمله ساده‌ترین معماری‌های شبکه عصبی مصنوعی است.

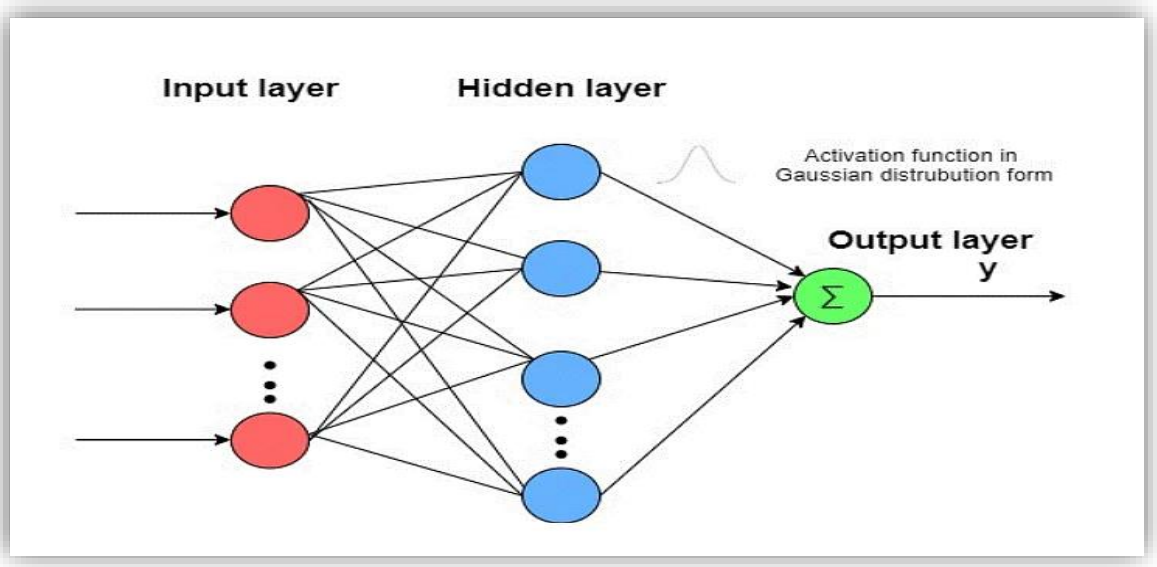
پرسپترون چند لایه (MLP) یک شبکه عصبی مصنوعی عمیق است که از بیش از یک پرسپترون تشکیل شده‌است. آن‌ها از یک لایه ورودی برای دریافت سیگنال تشکیل شده‌اند، یک لایه خروجی که در مورد ورودی تصمیم می‌گیرد یا پیش‌بینی می‌کند، و در بین این دو، تعدادی دلخواه از لایه‌های پنهان که موتور محاسباتی واقعی MLP هستند قرار دارد. MPLها 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.

از الگوریتم Backpropagation برای تعدیل وزن و میزان بایاس نسبت به خطا استفاده می‌شود[۳۶] و خود خطا را می‌توان به روش‌های مختلفی از جمله خطای مربع میانگین ریشه (RMSE)[۴۹] اندازه‌گیری کرد. شبکه های 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 چشم‌انداز 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.

[49] Root Mean Square Error

شکل(۲-۱۹) لایه‌های شبکه عصبی را نشان می‌دهد:



شکل(۲-۱۹): 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 آن‌ها 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.
۲. تقریب تابعی و تحلیل رگرسیون
۳. پردازش داده‌ها شامل خوشه‌بندی[۵۰] و فیلترکردن داده‌ها

استفاده از 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.

[50] Clustering

- ☑ 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 مدل های 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[۳۷].

۲-۴-۸: الگوریتم XGBoost[51]

Boosting یک مدل‌سازی گروهی[52] است؛ به عبارتی تکنیکی که تلاش می‌کند یک طبقه‌بندی قوی از تعداد طبقه‌بندی کننده‌های ضعیف بسازد که با ساخت مدل با استفاده از مدل‌های ضعیف به صورت سری انجام می‌شود. ابتدا یک مدل از داده‌های آموزشی ساخته می‌شود و سپس مدل دوم ساخته می‌شود که سعی می‌کند خطاهای موجود در مدل اول را اصلاح کند. این روش ادامه می‌یابد و مدل‌ها اضافه می‌شوند تا زمانی که مجموعه داده‌های آموزشی کامل به درستی پیش‌بینی شود یا حداکثر تعداد مدل‌ها اضافه شود.

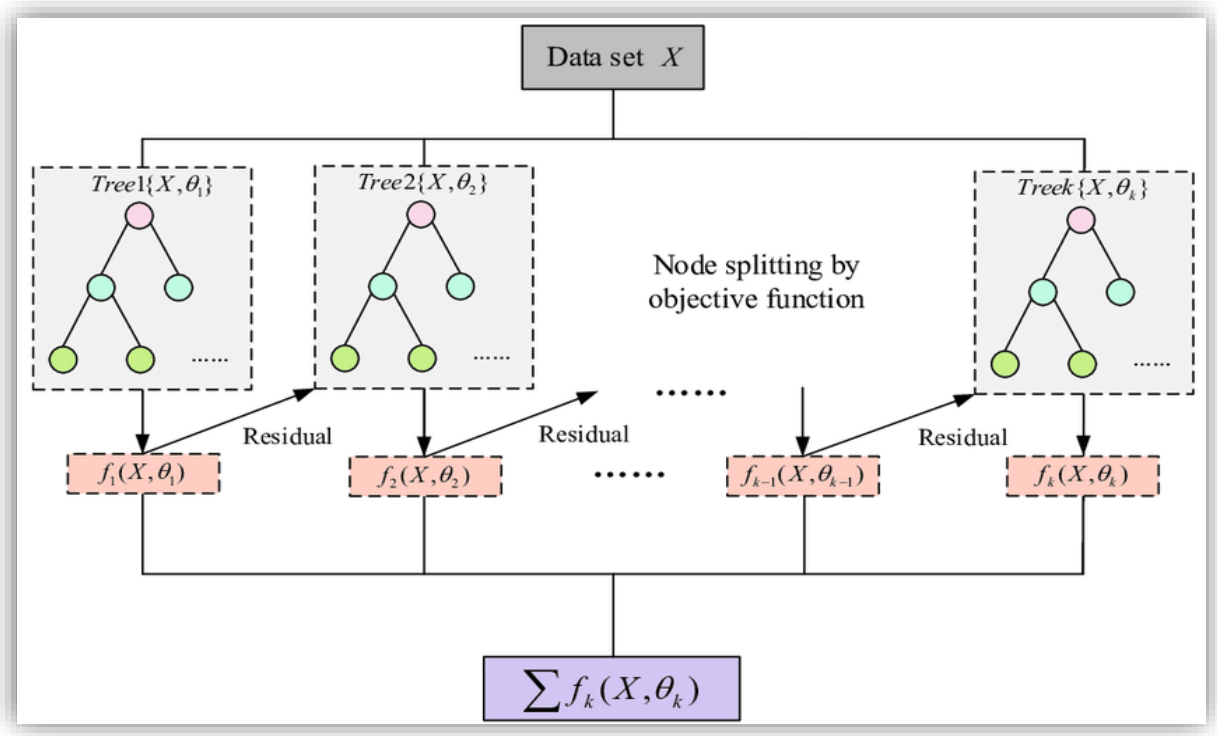
الگوریتم XGBoost به عنوان یک پروژه تحقیقاتی در دانشگاه واشنگتن توسعه داده شده است که در مسابقات یادگیری ماشین کاربردی و مسابقات Kaggle برای داده‌های ساختاریافته یا جدولی بخاطر سرعتش معروف شده است. XGBoost پیاده‌سازی درخت تصمیم تقویت‌شده با گرادیان است که برای سرعت و عملکرد طراحی شده است.

در این الگوریتم درخت‌های تصمیم به صورت متوالی ایجاد می‌شوند و وزن‌ها نقش مهمی در XGBoost دارند. وزن‌ها به همه متغیرهای مستقل اختصاص داده می‌شوند و سپس به درخت تصمیم که نتایج را پیش‌بینی می‌کند

[51] eXtreme Gradient Boosting (XGBoost)
[52] Ensemble

وارد می‌شوند. وزن متغیرهای پیش‌بینی شده‌ی اشتباه توسط درخت افزایش می‌یابد و این متغیرها سپس به درخت تصمیم دوم تغذیه می‌شوند. سپس این طبقه‌بندی‌کننده‌ها یا پیش‌بینی کننده‌های تنها، برای ارائه یک مدل قوی و دقیق‌تر جمع می‌شوند. این الگوریتم می‌تواند روی رگرسیون، طبقه‌بندی، رتبه‌بندی و مسائل پیش‌بینی تعریف شده توسط کاربر کار کند[۳۸].

شکل(۲-۲۰) این الگوریتم را نشان می‌دهد:



شکل(۲-۲۰): الگوریتم XGBoost

۲-۴-۹: الگوریتم کاهش گرادیان تصادفی (SGD)[۵۳]

کاهش گرادیان تصادفی یک الگوریتم بهینه‌سازی عمومی است که قادر به یافتن راه‌حل‌های بهینه برای طیف وسیعی از مسائل است؛ اگرچه الگوریتم کاهش گرادیان تصادفی برای مدت طولانی در جامعه یادگیری ماشین وجود داشته است، اخیراً در زمینه یادگیری در مقیاس بزرگ توجه زیادی به آن شده‌است.

[53] Stochastic Gradient Descent

کاهش گرادیان تصادفی با موفقیت برای مسائل یادگیری ماشین در مقیاس بزرگ و پراکنده که اغلب در طبقه‌بندی متن و پردازش زبان طبیعی با آن مواجه می‌شوند، استفاده شده‌است و صرفاً با خانواده خاصی از مدل‌های یادگیری ماشین مطابقت ندارد.

ایده کلی این است که پارامترها را به طور مکرر به منظور به حداقل رساندن تابع هزینه تغییر دهد.

یک پارامتر مهم Gradient Descent (GD) اندازه مراحل است که توسط فراپارامترهای نرخ یادگیری تعیین می‌شود. اگر نرخ یادگیری خیلی کم باشد، الگوریتم باید چندین بار تکرار شود تا همگرا شود که زمان زیادی طول می‌کشد و اگر خیلی زیاد باشد، ممکن است از مقدار بهینه پرش کنیم.

کلمه تصادفی به معنای سیستم یا فرآیندی است که با یک احتمال تصادفی مرتبط است. از این رو، در کاهش گرادیان تصادفی چند نمونه بجای کل مجموعه داده‌ها برای هر تکرار، به‌طور تصادفی انتخاب می‌شوند. در کاهش گرادیان تصادفی، اصطلاحی به نام batch وجود دارد که به تعداد کل نمونه‌های یک مجموعه داده که برای محاسبه گرادیان برای هر تکرار استفاده می‌شود، اشاره می‌کند. در بهینه‌سازی Gradient Descent معمولی، مانند Batch Gradient Descent، دسته به عنوان کل مجموعه داده در نظر گرفته می‌شود. اگرچه استفاده از کل مجموعه داده واقعاً برای رسیدن به حداقل‌ها به شیوه‌ای کم نویز و تصادفی کمتر مفید است، اما این مشکل زمانی به وجود می‌آید که مجموعه داده ما بزرگ شود[۳۹].

فرض کنید شما یک میلیون نمونه در مجموعه داده خود دارید بنابراین اگر از یک تکنیک معمولی بهینه‌سازی کاهش گرادیان استفاده می‌کنید، باید از تمام یک میلیون نمونه برای تکمیل یک تکرار در حین انجام کاهش گرادیان استفاده کنید و این کار باید برای هر تکرار تا رسیدن به حداقل انجام شود؛ از این رو انجام آن از نظر محاسباتی بسیار گران می‌شود.

این مشکل توسط کاهش گرادیان تصادفی حل می شود. در SGD ، برای انجام هر تکرار فقط از یک نمونه استفاده می‌کند به عنوان مثال، اندازه یک دسته نمونه به طور تصادفی برای انجام تکرار انتخاب می‌شود.

بنابراین، در SGD به جای مجموع گرادیان تابع هزینه تمام مثال‌ها، گرادیان تابع هزینه یک مثال را در هر تکرار می‌یابیم. فرمول(۲-۱۲) برای محاسبه SGD بکار می‌رود:

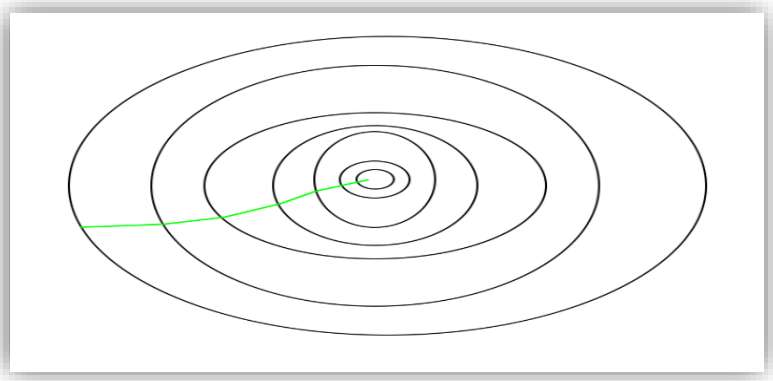
$$\text{for i in range(m):}\ \ \theta_j = \theta_j - \alpha(\hat{y}^i - y^i)x^i{}_j$$

فرمول(۲-۱۲): محاسبه SGD

در SGD از آنجایی که تنها یک نمونه از مجموعه داده به صورت تصادفی برای هر تکرار انتخاب می‌شود، مسیری که الگوریتم برای رسیدن به حداقل‌ها طی می‌کند معمولاً از الگوریتم Gradient Descent معمولی دارای نویز

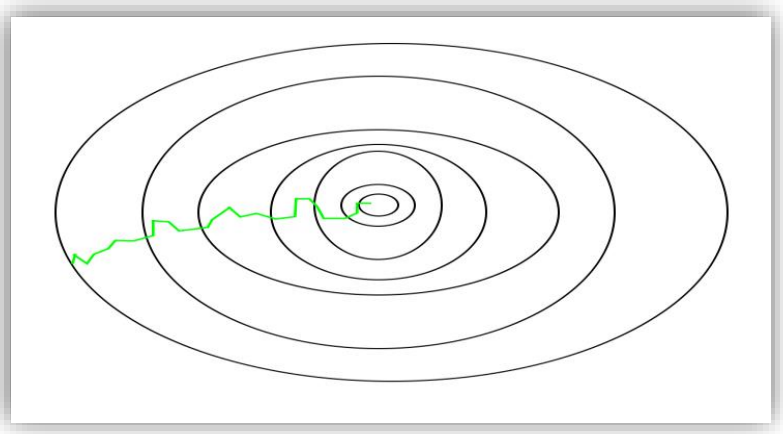
بیشتری است. اما این خیلی مهم نیست، زیرا مسیر طی شده توسط الگوریتم مهم نیست؛ تا زمانی که به حداقل‌ها و با زمان آموزش بسیار کوتاه‌تر برسیم[۴۰].

در شکل(۲-۲۱) مسیر طی شده توسط الگوریتم کاهش گرادیان ساده را مشاهده می‌کنید:



شکل(۲-۲۱): کاهش گرادیان ساده

و شکل(۲-۲۲) مسیری که الگوریتم کاهش گرادیان تصادفی طی می‌کند را نشان می‌دهد:



شکل(۲-۲۲): کاهش گرادیان تصادفی

مشاهده می‌کنید که کاهش گرادیان تصادفی دارای نویز بیشتری است.

نکته‌ای که باید به آن توجه داشت این است که، از آنجایی که SGD به طور کلی نویزی‌تر از Gradient Descent معمولی است؛ به دلیل تصادفی بودن آن در کاهش، معمولاً تعداد تکرارهای بیشتری برای رسیدن به حداقل نیاز است. حتی با وجود اینکه برای رسیدن به مینیمم نسبت به Gradient Descent معمولی به تعداد تکرارهای بیشتری نیاز دارد، هنوز از نظر محاسباتی بسیار کمتر از Gradient Descent معمولی است. از این رو، در اکثر سناریوها، SGD برای بهینه‌سازی یک الگوریتم یادگیری نسبت به Batch Gradient Descent ترجیح داده می‌شود[۴۱].

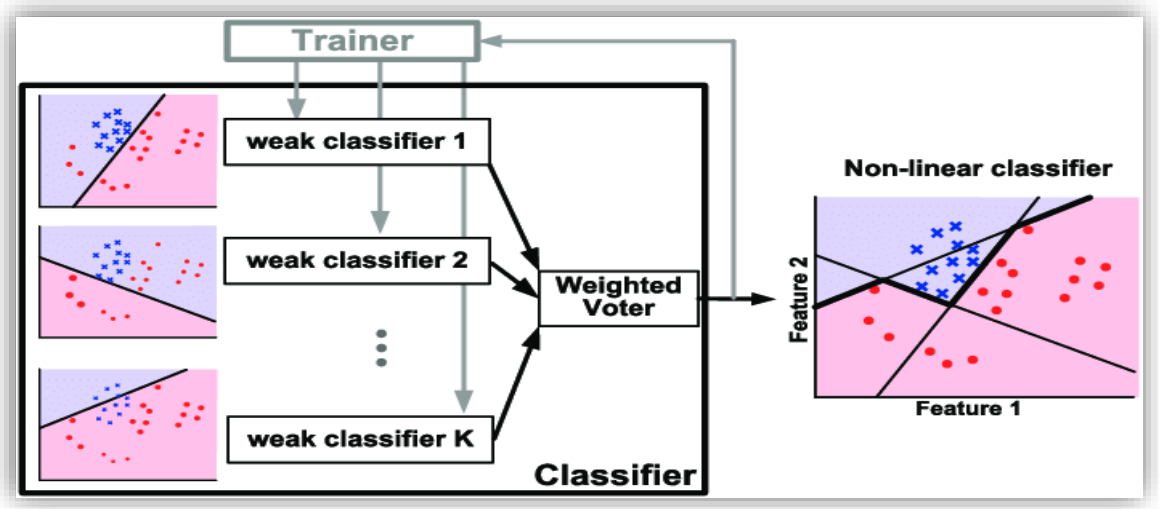
۲-۴-۱۰: الگوریتم AdaBoost[۵۴]

الگوریتم AdaBoost اولین الگوریتم تقویتی واقعا موفقی بود که با هدف طبقه‌بندی باینری توسعه یافت. AdaBoost مخفف Adaptive Boosting است و یک تکنیک تقویتی بسیار محبوب است؛ که چندین طبقه‌بندی ضعیف را در یک طبقه‌بندی قوی ترکیب می‌کند[۴۲]. مراحل اجرای این الگوریتم به صورت زیر است:

۱) مجموعه داده را مقداردهی اولیه می‌کند و به هر یک از نقاط داده وزن مساوی اختصاص می‌دهد.
۲) این را به عنوان ورودی مدل ارائه می‌دهد و نقاط داده‌ای که به اشتباه طبقه‌بندی شده‌اند را شناسایی می‌کند.
۳) وزن نقاط داده به اشتباه طبقه‌بندی شده را افزایش می‌دهد.
۴) اگر نتایج لازم بدست آمد به مرحله **۵** می‌رود در غیر این صورت به مرحله **۲** می‌رود.
۵) پایان اجرای الگوریتم

شکل(۲-۲۳) الگوریتم AdaBoost را نشان می‌دهد:

[54] Adaptive Boosting



شکل(۲-۲۳): الگوریتم **AdaBoost**

۲-۴-۱۱: الگوریتم LightGBM[۵۵]

الگوریتم LightGBM یک چارچوب تقویت‌کننده گرادیان است که بر اساس درخت تصمیم برای افزایش کارایی مدل و کاهش استفاده از حافظه است. از دو تکنیک جدید استفاده می‌کند:

۱) نمونه‌برداری یک طرفه مبتنی بر گرادیان[۵۶]

۲) دسته‌بندی ویژگی انحصاری[۵۷] که محدودیت‌های الگوریتم مبتنی بر هیستوگرام[۵۸] را برآورده می‌کند که عمدتاً در همه چارچوب‌های درخت تصمیم تقویت گرادیان[۵۹] استفاده می‌شود.

LightGBM درخت را از نظر برگ تقسیم می‌کند برخلاف سایر الگوریتم‌های تقویت‌کننده که در سطح درخت رشد می‌کنند.

رشد درخت به صورت برگ ممکن است پیچیدگی مدل را افزایش دهد و ممکن است به بیش‌برازش در مجموعه داده‌های کوچک منجر شود[۴۳].

---

[55] Light Gradient Boosting Machine
[56] One-way gradient-based sampling
[57] Exclusive feature category
[58] Histogram
[59] Gradient boosting decision tree

برای توزیع و کارآمدی با مزایای زیر طراحی شده است:

- ☑ سرعت تمرین بیشتر و راندمان بالاتر
- ☑ مصرف حافظه کمتر
- ☑ دقت بهتر
- ☑ توانایی مدیریت داده‌های در مقیاس بزرگ

۲-۴-۱۲: الگوریتم catBoost[۶۰]

الگوریتم catBoost یکی از جدیدترین الگوریتم‌های تقویت‌کننده منتشرشده در سال ۲۰۱۷ است. یک کتابخانه تقویت‌کننده منبع باز از دسته الگوریتم‌های یادگیری ماشین با نظارت برای تقویت گرادیان در درختان تصمیم است که توسط محققان و مهندسان Yandex توسعه یافته‌است و در بسیاری از شرکت‌های دیگر از جمله CERN ، Cloudflare و Careem taxi نیز استفاده می‌شود. علاوه بر رگرسیون و طبقه‌بندی، الگوریتم catBoost را می‌توان در رتبه‌بندی، سیستم‌های توصیه، پیش‌بینی، دستیارهای شخصی، اتومبیل‌های خودران، پیش‌بینی آب و هوا و بسیاری از وظایف استفاده کرد.

الگوریتم catBoost مشابه الگوریتم‌های SGD و XGboost کار می‌کند؛ اما دارای برخی ویژگی‌های پیشرفته است که آن را قابل اعتماد، سریع‌تر و دقیق‌تر می‌کند[۴۴].

مزایای این الگوریتم عبارتند از:

- ☑ درخت‌های تصمیم فراموش کار[۶۱] را پیاده‌سازی می‌کند (درخت دودویی که در آن از ویژگی‌های یکسان برای تقسیم چپ و راست برای هر سطح درخت استفاده می‌شود) در نتیجه، تقسیم ویژگی‌ها در هر سطح را محدود می‌کند که به کاهش زمان پیش‌بینی کمک می‌کند.
- ☑ ویژگی‌های دسته بندی را به طور موثر کنترل می‌کند.
- ☑ استفاده از آن در زبان‌های R و Python آسان است.
- ☑ با پارامترهای پیش فرض استفاده موثری دارد و در نتیجه زمان لازم برای تنظیم پارامتر را کاهش می‌دهد.

[60] Categorical Boosting
[61] oblivious decision trees

۲-۴-۱۳: اعتبار سنجی 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 پیش‌بینی کننده 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

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[۴۵].

در ادامه روش اعتبارسنجی متقابلی را که در این پایان نامه پوشش داده خواهدشد، بررسی می‌کنیم.

۲-۴-۱۴: روش k-fold

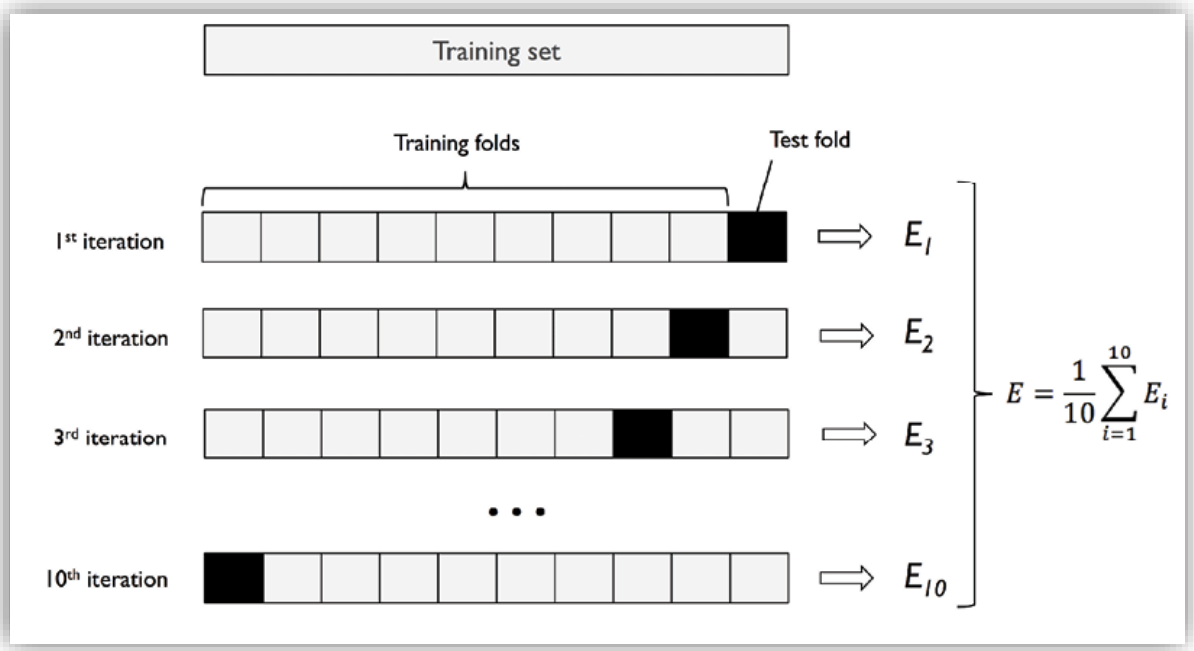
k-fold CV روش جدیدی را برای تقسیم مجموعه داده معرفی می‌کند که بر غلبه بر روش سنتی که میزان داده‌های ثابتی را برای آموزش و آزمایش جدا می‌کردیم (روش ۸۰-۲۰ یا ۷۰-۳۰) غلبه می‌کند. الگوریتم روش k-fold که در شکل۲-۱۷ نیز نمایش داده شده‌است، عبارت است از:

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 خواهدبود. باقی‌مانده مجموعه آزمایشی خواهندبود.
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 اعتبار سنجی 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(۲-۲۴) نشان داده شده‌است:



شکل(۲-۲۴): الگوریتم k-fold

فصل سوم: مروری بر مطالعات انجام شده

۳–۱: مقدمه

در این بخش توضیحی مختصر در مورد [۴۶] و [۱] و [۴۷] پرداخته شده‌است. 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 [۴۶] به بررسی میزان دقت الگوریتم‌های مختلف با چند تکنیک یادگیری ماشین پرداخته است. برای انتخاب ویژگی از روش RFE استفاده کرده‌است.

این مقاله برای پیش‌بینی از الگوریتم‌های naïve bayes ، SVM ، KNN ، Decision tree و Random forest استفاده کرده است. میزان دقت بدست آمده از هر کدام از این تکنیک‌ها در جدول(۳-۱) آمده است:

model	accuracy
naïve bayes	83.49 %
SVM	84.81 %
KNN	83.16 %
Decision tree	78.46 %
Random forest	86 %

جدول(۳-۱): نتایج مقاله شماره ۴۰

در [۱] برای انتخاب ویژگی از روش correlation استفاده کرده است و برای مدل کردن هم از الگوریتم‌های یادگیری ماشین Logistic regression ، Naïve bayes ، Random forest و Decision tree استفاده کرده است.

میزان دقت بدست آمده از هر کدام از این تکنیک‌ها در جدول(۳-۲) آمده است:

model	accuracy
naïve bayes	84.25 %
Random forest	83.26 %
Decision tree	81.97 %
Logistic regression	84.25 %

جدول(۳-۲): نتایج مقاله شماره ۱

در مقاله [۴۷] برای انتخاب ویژگی از الگوریتم FCBF[۶۲] استفاده کرده است.

برای پیش‌بینی هم از الگوریتم‌های یادگیری ماشین KNN ، SVM ، Random forest ، Naïve bayes و Neural network استفاده کرده است و دقت پیش‌بینی را قبل و بعد از استفاده از الگوریتم انتخاب ویژگی بدست آورده و مقایسه کرده است.

دقت مدل‌ها قبل از استفاده از الگوریتم انتخاب ویژگی در جدول(۳-۳) آمده است:

model	accuracy
KNN	75.3 %
SVM	86.7 %
Random forest	84.7 %
Naïve bayes	86%
Neural network	83.3 %

جدول(۳-۳): نتایج مقاله شماره ۴۱ بدون انتخاب ویژگی

دقت مدل‌ها بعد از استفاده از الگوریتم انتخاب ویژگی FCBF در جدول(۳-۴) آمده است:

model	accuracy
KNN	83.3 %
SVM	86 %
Random forest	84.7 %
Naïve bayes	80%
Neural network	78%

جدول(۳-۴): نتایج مقاله شماره ۴۱ با انتخاب ویژگی

[62] Fast Correlation Based Filter

فصل چهارم: پیاده‌سازی و تجزیه و تحلیل داده‌ها

۴-۱: مجموعه داده

مجموعه داده استفاده شده در این پروژه از اطلاعات بالینی مردم کلیولند است؛ مانند سن، جنسیت، میزان فشار خون، کلسترول و غیره که در [۴۸] قابل مشاهده است.

این مجموعه داده شامل ۱۴ ویژگی و ۳۰۲ 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:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pylab as pl
import scipy.optimize as opt
from sklearn import preprocessing
import os
import sys
module_path = os.path.abspath(os.path.join('..'))
if module_path not in sys.path:
    sys.path.append(module_path+"\\SoftwareProject")
from cross_valdiation_score import k_fold_results
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
import xgboost as xgb
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from sklearn import svm
from keras.models import Sequential
from sklearn.feature_selection import RFE
from sklearn.feature_selection import RFECV
from keras.layers import Dense
from sklearn.ensemble import RandomForestClassifier
```

شکل(۴-۱): **library**

```
from sklearn.tree import DecisionTreeClassifier
import warnings
from sklearn.feature_selection import VarianceThreshold
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.kernel_approximation import RBFSampler
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
from sklearn.ensemble import GradientBoostingClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
from pyspark.ml.classification import MultilayerPerceptronClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.naive_bayes import MultinomialNB
warnings.filterwarnings('ignore')
%matplotlib inline
```

شکل(۴-۲): library

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

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

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	67	1	4	160	286	0	2	108	1	1.5	2	3	3	1
1	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
2	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
3	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
4	56	1	2	120	236	0	0	178	0	0.8	1	0	3	0

جدول(۴-۱): مجموعه داده

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

```
age:            age
gender:         1: male, 0: female
cp:             chest pain type, 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic
trestbps:               resting blood pressure
chol:            serum cholestoral in mg/dl
fbs:            fasting blood sugar > 120 mg/dl
restecg:                resting electrocardiographic results (values 0,1,2)
thalach:                 maximum heart rate achieved
exang:          exercise induced angina
oldpeak:                oldpeak = ST depression induced by exercise relative to rest
slope:          the slope of the peak exercise ST segment
ca:             number of major vessels (0-3) colored by flourosopy
thal:           thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
target:         final result
```

شکل(۴-۳): توضیحات ویژگی‌های مجموعه داده

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

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000
mean	54.307947	0.682119	3.149007	131.728477	246.390728	0.142384	0.993377	149.609272	0.324503	1.033113	1.589404	0.668874	4.705298	0.453642
std	9.053984	0.466426	0.958083	17.609245	51.628879	0.350024	0.994982	22.881167	0.468966	1.162236	0.612945	0.934548	1.938225	0.498673
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000	3.000000	0.000000
25%	47.000000	0.000000	3.000000	120.000000	211.250000	0.000000	0.000000	133.250000	0.000000	0.000000	1.000000	0.000000	3.000000	0.000000
50%	55.000000	1.000000	3.000000	130.000000	240.500000	0.000000	1.000000	153.000000	0.000000	0.750000	2.000000	0.000000	3.000000	0.000000
75%	61.000000	1.000000	4.000000	140.000000	274.000000	0.000000	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000	7.000000	1.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000	7.000000	1.000000

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

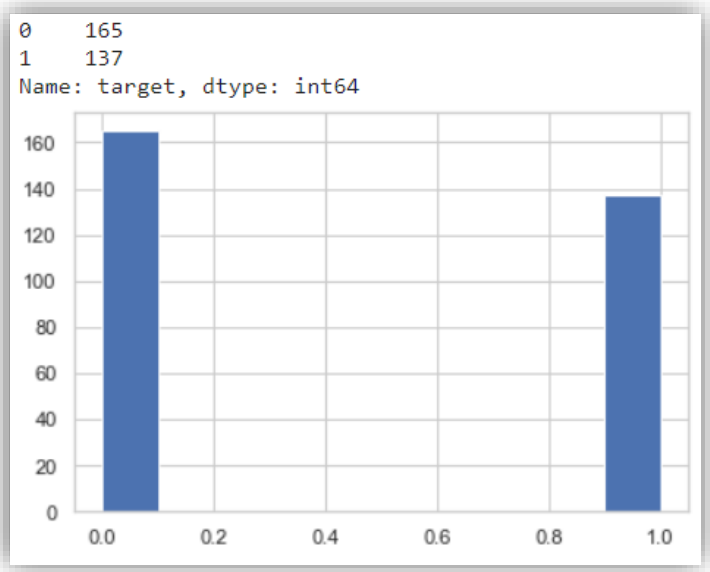
برای کار با مجموعه داده، باید عناصر آن از یک نوع باشند. پس طی فرآیندی همه عناصر مجموعه داده را به عدد (int) تبدیل می‌کنیم که نتیجه آن در شکل۴-۴ قابل مشاهده است:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 302 entries, 0 to 301
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       302 non-null    int64
 1   gender    302 non-null    int64
 2   cp        302 non-null    int64
 3   trestbps  302 non-null    int64
 4   chol      302 non-null    int64
 5   fbs       302 non-null    int64
 6   restecg   302 non-null    int64
 7   thalach   302 non-null    int64
 8   exang     302 non-null    int64
 9   oldpeak   302 non-null    float64
 10  slope     302 non-null    int64
 11  ca        302 non-null    int64
 12  thal      302 non-null    int64
 13  target    302 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.2 KB
```

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

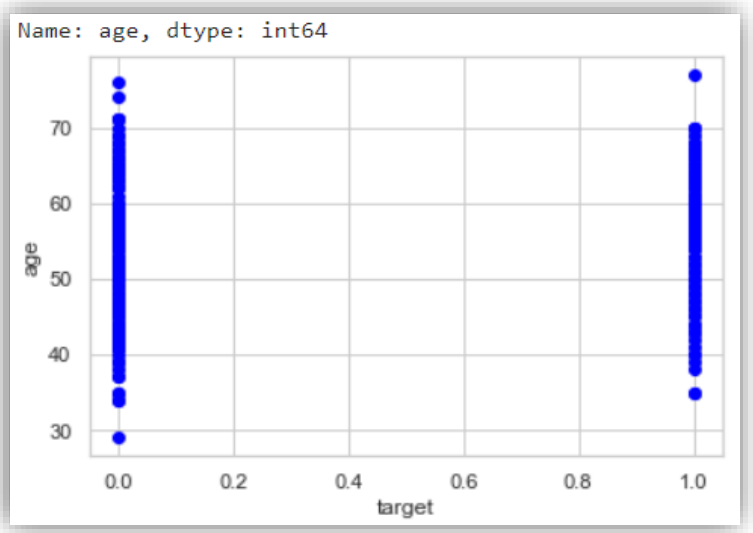
در مرحله بعد برای آن که دید نسبتاً خوبی به مجموعه داده پیدا کنیم بهتر است هر یک از ویژگی‌ها را روی نمودار بررسی کنیم.

در شکل(۴-۵) ویژگی target که نتیجه نهایی است را مشاهده می‌کنید.



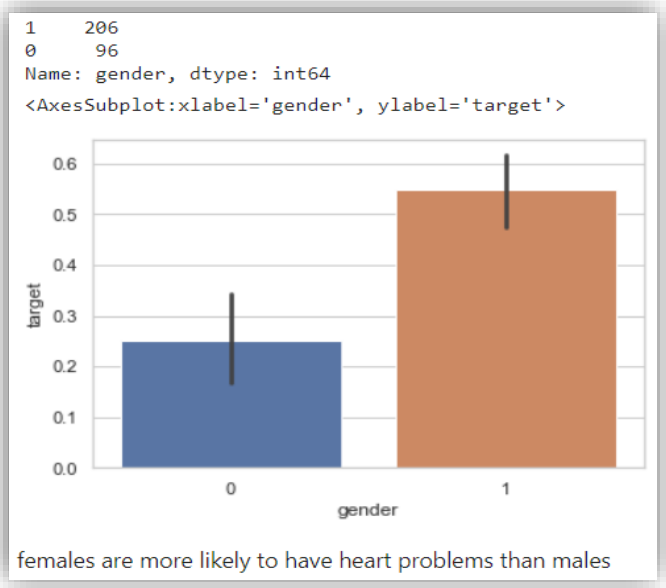
شکل(۴-۵): ویژگی target

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



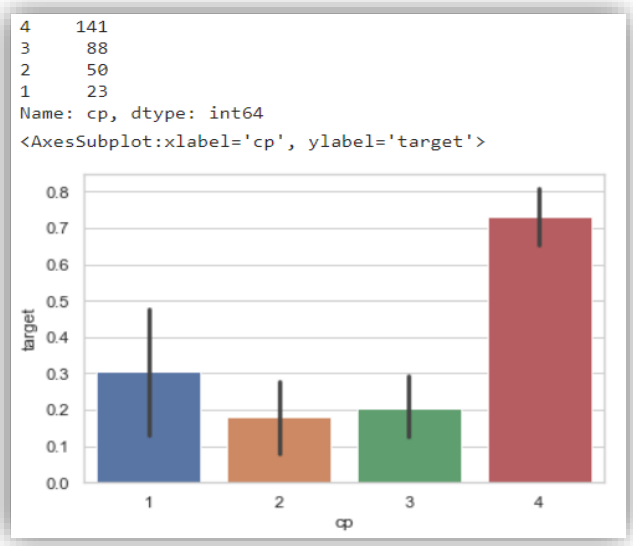
شکل(۴-۶): ویژگی age

در شکل(۴-۷) ویژگی gender را مشاهده می‌کنید.



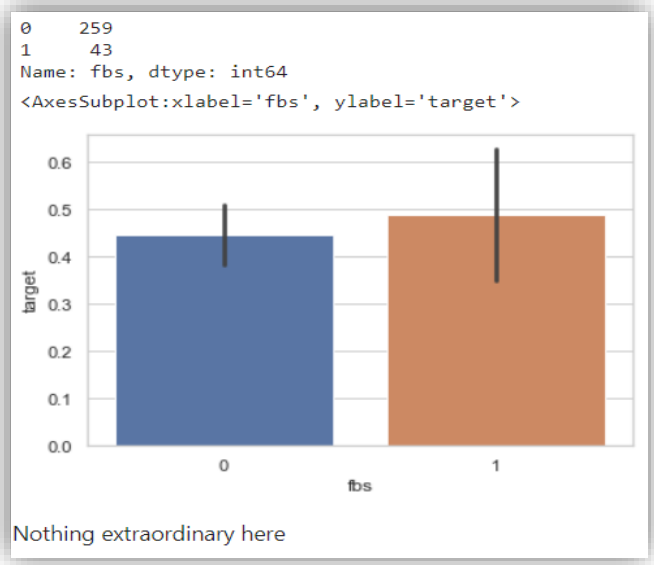
شکل(۴-۷): ویژگی gender

در شکل(۴-۸) ویژگی cp را مشاهده می‌کنید.



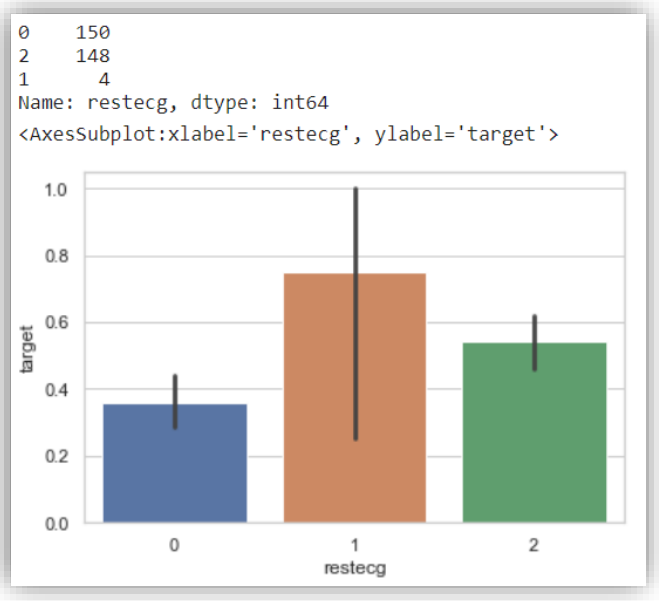
شکل(۴-۸): ویژگی cp

در شکل(۴-۹) ویژگی fbs را مشاهده می‌کنید.



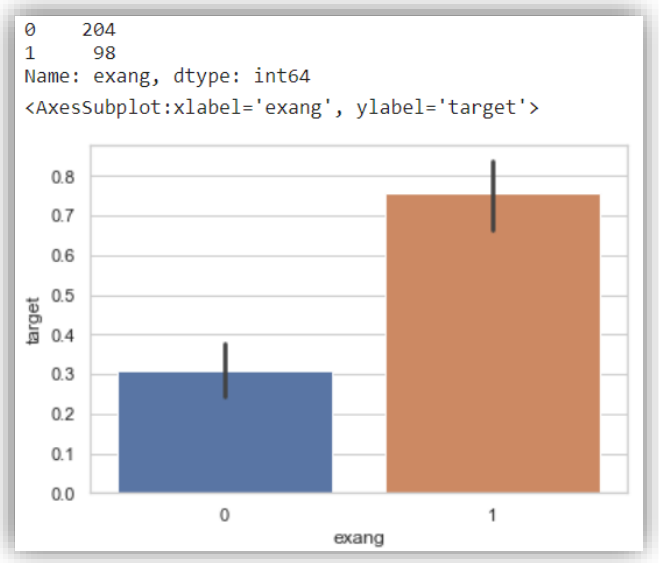
شکل(۴-۹): ویژگی fbs

در شکل(۴-۱۰) ویژگی restecg را مشاهده می‌کنید.



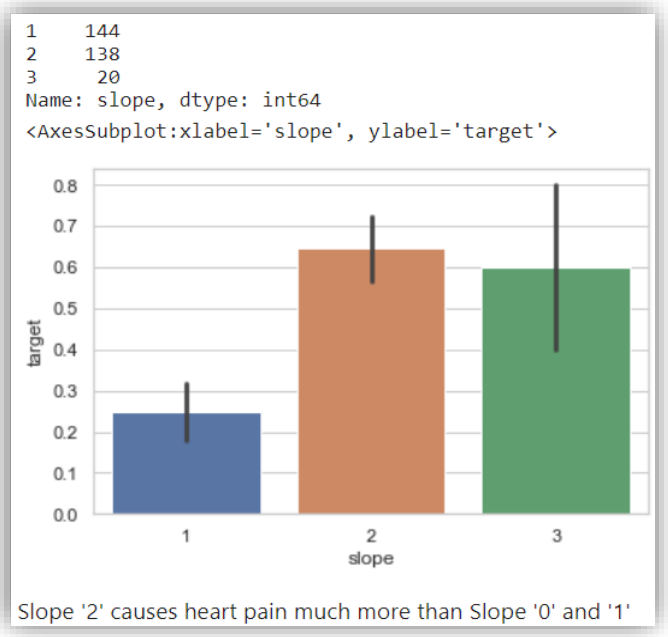
شکل(۴-۱۰): ویژگی restecg

در شکل(۴-۱۱) ویژگی exange را مشاهده می‌کنید.



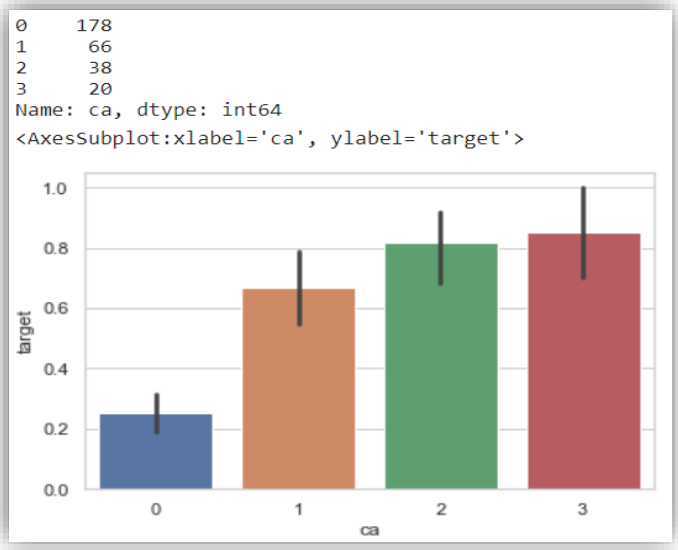
شکل(۴-۱۱): ویژگی exang

در شکل(۴-۱۲) ویژگی slope را مشاهده می‌کنید.



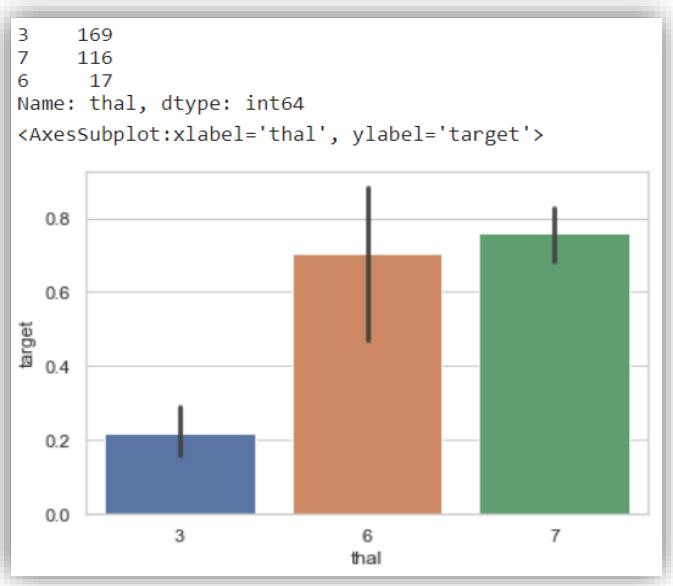
شکل(۴-۱۲): ویژگی slope

در شکل(۴-۱۳) ویژگی ca را مشاهده می‌کنید.



شکل(۴-۱۳): ویژگی ca

در شکل(۴-۱۴) ویژگی thal را مشاهده می‌کنید.



شکل(۴-۱۴): ویژگی thal

چون اعداد در این مجموعه داده در بازه‌های متفاوتی هستند، باید توسط الگوریتم minmax آن‌ها را تبدیل به اعداد در بازه‌ی ۰ تا ۱ کنیم تا بتوانیم این مجموعه داده را به عنوان ورودی به الگوریتم‌های انتخاب ویژگی بدهیم. به این کار نرمال‌سازی[۶۳] مجموعه داده گفته می‌شود.

بخشی از مجموعه داده‌ی نرمال‌شده را در جدول(۴-۳) مشاهده می‌کنید:

[63] Normalization

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	0.791667	1	1.000000	0.622642	0.365297	0	1.0	0.282443	1	0.241935	0.5	1.000000	0.0	1
1	0.791667	1	1.000000	0.245283	0.235160	0	1.0	0.442748	1	0.419355	0.5	0.666667	1.0	1
2	0.166667	1	0.666667	0.339623	0.283105	0	0.0	0.885496	0	0.564516	1.0	0.000000	0.0	0
3	0.250000	0	0.333333	0.339623	0.178082	0	1.0	0.770992	0	0.225806	0.0	0.000000	0.0	0
4	0.562500	1	0.333333	0.245283	0.251142	0	0.0	0.816794	0	0.129032	0.0	0.000000	0.0	0

جدول(۴-۳): مجموعه داده ویرایش شده

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 و سپس الگوریتم‌های انتخاب ویژگی را روی مجموعه داده اعمال کرده و هر کدام از مجموعه داده‌های جدید را به مدل‌ها می‌دهیم و نتایج را بررسی و با هم مقایسه می‌کنیم.

در این پروژه برای تعیین داده‌های آموزش و آزمایش از تابع k_fold_resault() از کتابخانه‌ی cross_validation_score که برای این پایان نامه پیاده‌سازی شده و از قبل آماده نبوده استفاده شده است ابتدا به روش train_test_split به میزان ۷۰-۳۰ داده‌های آزمایش و آموزش را جدا کرده و سپس این کتابخانه بهترین مجموعه‌های آموزش و آزمایش را با روش k-fold CV و طی چند مرحله آزمایش بدست می‌آورد.

به وسیله این تابع حجم زیادی از کدهای پیاده‌سازی کاهش یافت؛ زیرا بدون این تابع باید تمام الگوریتم‌های یادگیری ماشین بکار رفته در پروژه پیاده‌سازی میشد و به ازای هر کدام از الگوریتم‌های انتخاب ویژگی، مجموعه داده آن به هر یک از الگوریتم‌های یادگیری ماشین داده میشد. با وجود این تابع، ما یک لیستی از آبجکت‌های مدل الگوریتم‌های یادگیری ماشین با ورودی‌های خاص هر آبجکت که مورد نیاز هر الگوریتم است را ایجاد می‌کنیم که در شکل(۴-۱۵) مشاهده می‌کنید. برای هر الگوریتم انتخاب ویژگی فقط کافیست نام این لیست و مجموعه داده‌ی مشخص را به عنوان ورودی به این تابع بدهیم و سپس مقادیر پارامترهای accuracy ، recall ، specificity ، precision و f1-score را در یک جدول به عنوان خروجی برمی‌گرداند.

```
algo={
    'NB':MultinomialNB(alpha=1.0),
    'Logistic Regression':LogisticRegression(),
    'SVM':svm.SVC(kernel='poly',random_state=1),
    'KNN':KNeighborsClassifier(n_neighbors=11),
    'Dtree':DecisionTreeClassifier(),
    'Random Forest':RandomForestClassifier(random_state=1),
    'Multi-layer-Perceptron':MLPClassifier(solver='lbfgs', alpha=1e-5,random_state=3),
    'XGBoost':xgb.XGBClassifier(objective="binary:logistic", random_state=3),
    'Stochastic-Gradient-Descent':SGDClassifier(loss="hinge", penalty="l2", max_iter=8),
    'AdaBoost':AdaBoostClassifier(n_estimators=100, random_state=3),
    'LightGBM ':LGBMClassifier(),
    'catBoost':CatBoostClassifier(verbose=0, n_estimators=100),
}
```

شکل(۴-۱۵): لیست وردی تابع k_fold_results

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(۴-۴) مقادیر accuracy ، recall ، specificity ، precision و f1-Score را برای الگوریتم‌های یادگیری ماشین مختلف مشاهده می‌کنید.

	accuracy	recall	specificity	precision	F1
SVM	0.665591	0.521020	0.793024	0.684098	0.576713
Logistic Regression	0.837634	0.790223	0.882577	0.855234	0.814679
NB	0.774839	0.740221	0.809954	0.774513	0.748464
KNN	0.655806	0.594574	0.722234	0.639783	0.603076
Dtree	0.744516	0.745637	0.748481	0.713548	0.723367
Random Forest	0.820753	0.773439	0.865846	0.833590	0.796767
Multi-layer-Perceptron	0.824301	0.778293	0.871987	0.841103	0.799116
XGBoost	0.797742	0.771132	0.825124	0.786666	0.773846
Stochastic-Gradient-Descent	0.592688	0.506009	0.705384	0.614566	0.447018
AdaBoost	0.797527	0.751125	0.841825	0.804699	0.771656
LightGBM	0.804301	0.760601	0.848492	0.802548	0.775169
catBoost	0.823978	0.786127	0.861240	0.824994	0.799307

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

بهترین مقدار accuracy به ترتیب برای الگوریتم رگرسیون لجستیک با مقدار ۰.۸۳ و جنگل تصادفی، MLP و catBoost به صورت برابر با مقدار ۰.۸۲ است.

بالاترین recall مربوط به الگوریتم رگرسیون لجستیک و سپس catBoost به ترتیب با مقادیر ۰.۷۹ و ۰.۷۸ است یعنی این دو الگوریتم در شناسایی نمونه‌های مثبت نسبت به بقیه بهتر عمل می‌کنند.

بالاترین specificity مربوط به الگوریتم رگرسیون لجستیک و سپس MLP به ترتیب با مقادیر ۰.۸۸ و ۰.۸۷ است. یعنی این دو الگوریتم نتایج منفی را نسبت به بقیه درست‌تر نشان می‌دهند.

بالاترین precision مربوط یه الگوریتم رگرسیون لجستیک و سپس MLP به ترتیب با مقادیر ۰.۸۵ و ۰.۸۴ است یعنی در این دو الگوریتم موارد مثبتی که پیش‌بینی شد و واقعا هم مثبت بودند، نسبت به بقیه بیشتر است. بیشترین مقدار f1-Score هم در بین این الگوریتم‌ها مربوط به رگرسیون لجستیک با مقدار ۰.۸۱ است.

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

از مجموعه داده‌ای که با استفاده از الگوریتم انتخاب ویژگی correlation بدست آمده به عنوان ورودی استفاده می‌کنیم. نتایج در جدول(۴-۵) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
SVM	0.811505	0.740189	0.880227	0.852992	0.780716
Logistic Regression	0.840968	0.789960	0.887221	0.872508	0.818907
NB	0.817957	0.763939	0.871562	0.850665	0.793602
KNN	0.817742	0.767156	0.863526	0.835006	0.793261
Dtree	0.754624	0.726125	0.782931	0.736752	0.726571
Random Forest	0.801290	0.752535	0.844374	0.808901	0.771694
Multi-layer-Perceptron	0.804624	0.773030	0.839082	0.810716	0.782352
XGBoost	0.774946	0.744695	0.807346	0.774233	0.750688
Stochastic-Gradient-Descent	0.771720	0.745304	0.818677	0.783859	0.740342
AdaBoost	0.787957	0.750477	0.818000	0.791566	0.757645
LightGBM	0.774731	0.746887	0.804924	0.767215	0.749682
catBoost	0.821075	0.761246	0.874998	0.850841	0.794105

جدول(۴-۵): نتیجه یادگیری با انتخاب ویژگی Correlation

بهترین مقدار accuracy برای الگوریتم رگرسیون لجستیک و سپس catBoost به ترتیب با مقادیر ۰.۸۴ و ۰.۸۲ است که مشاهده می‌کنید برای الگوریتم رگرسیون لجستیک این مقدار افزایش یافته و برای الگوریتم catBoost ثابت مانده است. برای برخی الگوریتم‌ها هم حتی کاهش یافته‌است مانند AdaBoost و علت این کاهش‌ها از دست رفتن اطلاعات برخی از ویژگی‌هایی است که حذف شده‌اند.

بالاترین recall مربوط به الگوریتم رگرسیون لجستیک و سپس MLP به ترتیب با مقادیر ۰.۷۸ و ۰.۷۷ 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 رگرسیون لجستیک و SVM با مقدار ۰.۸۸ است که کمی در حد اعشار از حالت بدون انتخاب ویژگی بیشتر است و یعنی این در این حالت نتایج منفی درست‌تر نشان داده می‌شوند.

بالاترین precision مربوط یه الگوریتم رگرسیون لجستیک و سپس SVM به ترتیب با مقادیر ۰.۸۷ و ۰.۸۵ است. یعنی در این دو الگوریتم موارد مثبتی که پیش‌بینی شد و واقعا هم مثبت بودند، نسبت به حالت قبل افزایش یافتند.

بیشترین مقدار f1-Score هم در بین این الگوریتم‌ها مربوط به رگرسیون لجستیک با مقدار ۰.۸۱ است که به میزان کمی در حد اعشار نسبت به حالت بدون انتخاب ویژگی افزایش داشته‌است.

۴-۲-۳: ارزیابی یادگیری با انتخاب ویژگی Variance Threshold

از مجموعه داده‌ای که با استفاده از الگوریتم انتخاب ویژگی Variance Threshold بدست آمده به عنوان ورودی استفاده می‌کنیم. نتایج در جدول(۴-۶) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
SVM	0.807634	0.772157	0.843326	0.797227	0.777614
Logistic Regression	0.837634	0.789318	0.881338	0.857073	0.811398
NB	0.745806	0.678768	0.812965	0.750569	0.700652
KNN	0.797957	0.772920	0.821498	0.793587	0.774897
Dtree	0.790968	0.751395	0.818268	0.789705	0.765792
Random Forest	0.800968	0.789202	0.817970	0.787960	0.784101
Multi-layer-Perceptron	0.754839	0.725599	0.791557	0.745398	0.726441
XGBoost	0.791183	0.744960	0.835256	0.785311	0.758831
Stochastic-Gradient-Descent	0.774409	0.775108	0.776854	0.789801	0.729475
AdaBoost	0.820860	0.789723	0.844666	0.817582	0.791514
LightGBM	0.847204	0.796012	0.889735	0.855285	0.814798
catBoost	0.830968	0.795223	0.865256	0.830362	0.804304

جدول(۴-۶): نتیجه یادگیری با Variance Threshold

بهترین مقدار accuracy برای الگوریتم LightGBM و سپس رگرسیون لجستیک به ترتیب با مقادیر ۰.۸۴ و ۰.۸۳ است که مشاهده می‌کنید برای الگوریتم LightGBM این مقدار از ۰.۸۰ به ۰.۸۴ افزایش یافته و برای الگوریتم رگرسیون لجستیک تقریبا ثابت مانده است. برای برخی الگوریتم‌ها هم حتی کاهش یافته‌است مانند Random Forest که به علت از دست رفتن اطلاعات برخی ویژگی‌هایی که حذف شده‌اند، از ۰.۸۲ به ۰.۸۰ کاهش یافته‌است.

بالاترین recall مربوط به الگوریتم LightGBM و catBoost با مقدار ۰.۷۹ و سپس AdaBoost با مقدار ۰.۷۸ است که نسبت به حالت بدون انتخاب ویژگی، افزایش یافته‌است و این بدان معناست که توانایی الگوریتم‌ها برای شناسایی موارد مثبت افزایش یافته‌است.

بالاترین specificity مربوط به الگوریتم LightGBM با مقدار ۰.۸۸ است که کمی در حد اعشار از حالت بدون انتخاب ویژگی که مربوط به الگوریتم رگرسیون لجستیک بود بیشتر است و یعنی این در این حالت، نتایج منفی درست‌تر نشان داده می‌شوند.

بالاترین precision مربوط یه الگوریتم رگرسیون لجستیک و سپس LightGBM هردو با مقدار ۰.۸۵ است.

یعنی در این دو الگوریتم موارد مثبتی که پیش‌بینی شد و واقعا هم مثبت بودند، نسبت به حالت بدون انتخاب ویژگی، کمی افزایش یافتند.

بیشترین مقدار f1-Score هم در بین این الگوریتم‌ها مربوط به رگرسیون لجستیک و LightGBM با مقدار ۰.۸۱ است که نسبت به حالت بدون انتخاب ویژگی، افزایش داشته‌است.

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

از مجموعه داده‌ای که با استفاده از الگوریتم انتخاب ویژگی RFE بدست آمده به عنوان ورودی استفاده می‌کنیم.

نتایج در جدول(۴-۷) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
SVM	0.798172	0.732881	0.861665	0.823119	0.766745
Logistic Regression	0.824516	0.794433	0.856109	0.833953	0.806125
NB	0.798172	0.768144	0.836694	0.806967	0.777256
KNN	0.801398	0.752124	0.851304	0.819540	0.773113
Dtree	0.735054	0.705367	0.765602	0.718741	0.704606
Random Forest	0.774946	0.728291	0.821625	0.778149	0.744310
Multi-layer-Perceptron	0.728280	0.716746	0.750937	0.705763	0.701711
XGBoost	0.751720	0.735599	0.769598	0.741110	0.730778
Stochastic-Gradient-Descent	0.785269	0.692953	0.866524	0.848980	0.741384
AdaBoost	0.774839	0.735592	0.813686	0.767947	0.744748
LightGBM	0.781720	0.730072	0.831333	0.793580	0.752999
catBoost	0.824624	0.754053	0.893103	0.869971	0.797868

جدول(۴-۷): نتیجه یادگیری با RFE

بهترین مقدار accuracy برای الگوریتم catBoost و سپس رگرسیون لجستیک هر دو با مقدار ۰.۸۲ است که نسبت به حالت بدون انتخاب ویژگی، کاهش یافته‌است. در این حالت خیلی از الگوریتم‌ها مقدارشان کاهش یافته اما برای بعضی‌ها هم خیلی افزایش یافته است برای مثال الگوریتم KNN از ۰.۶۵ به ۰.۸۰ افزایش یافته‌است.

بالاترین recall مربوط به الگوریتم رگرسیون لجستیک با مقدار ۰.۷۹ که نسبت به حالت بدون انتخاب ویژگی، در در حد اعشار افزایش یافته‌است و این بدان معناست که توانایی این الگوریتم برای شناسایی موارد مثبت افزایش یافته‌است اما به طور کلی در این حالت مقدار recall برای بسیاری از الگوریتم‌ها نسبت به حالت بدون انتخاب ویژگی کاهش یافته‌است.

بالاترین specificity مربوط به الگوریتم catBoost با مقدار ۰.۸۹ است و از حالت بدون انتخاب ویژگی که الگوریتم رگرسیون لجستیک با مقدار ۰.۸۸ بود، بیشتر است و یعنی در این حالت نتایج منفی درست‌تر نشان داده می‌شوند.

بالاترین precision مربوط به الگوریتم catBoost با مقدار ۰.۸۶ است که نسبت به حالت بدون انتخاب ویژگی که مربوط به الگوریتم رگرسیون لجستیک با مقدار ۰.۸۵ بود، افزایش یافته‌است و یعنی در الگوریتم موارد مثبتی که پیش‌بینی شد و واقعا هم مثبت بودند، نسبت به حالت بدون انتخاب ویژگی افزایش یافت.

بیشترین مقدار f1-Score هم در بین این الگوریتم‌ها، مربوط به رگرسیون لجستیک با مقدار ۰.۸۰ است که نسبت به حالت بدون انتخاب ویژگی، کاهش یافته‌است.

۴-۲-۵: ارزیابی یادگیری با انتخاب ویژگی Forward

از مجموعه داده‌ای که با استفاده از الگوریتم انتخاب ویژگی Forward بدست آمده به عنوان ورودی استفاده می‌کنیم.

نتایج در جدول(۴-۸) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
SVM	0.807742	0.753176	0.859082	0.826484	0.778684
Logistic Regression	0.840860	0.796248	0.882907	0.863868	0.817339
NB	0.739032	0.688122	0.799334	0.739650	0.696447
KNN	0.811290	0.770605	0.851825	0.830769	0.788474
Dtree	0.771075	0.688822	0.847351	0.804648	0.730575
Random Forest	0.814194	0.791395	0.835714	0.805129	0.793310
Multi-layer-Perceptron	0.764409	0.758062	0.773978	0.740943	0.739934
XGBoost	0.817527	0.776775	0.854145	0.821400	0.789055
Stochastic-Gradient-Descent	0.737849	0.804845	0.699223	0.731258	0.739676
AdaBoost	0.820753	0.769723	0.866041	0.835844	0.789282
LightGBM	0.837204	0.793588	0.872415	0.839197	0.808379
catBoost	0.810860	0.760864	0.859866	0.824260	0.781273

جدول(۴-۸): نتیجه یادگیری با Forward

بهترین مقدار accuracy برای الگوریتم رگرسیون لجستیک با مقدار ۰.۸۴ است که نسبت به حالت بدون انتخاب ویژگی، افزایش یافته است و برای بعضی الگوریتم‌ها هم کاهش داشتیم مانند MLP که از ۰.۸۲ به ۰.۷۶ کاهش پیدا کرده‌است.

بالاترین recall مربوط به الگوریتم SGD با مقدار ۰.۸۰ است که نسبت به حالت بدون انتخاب ویژگی که مربوط به الگوریتم رگرسیون لجستیک با مقدار ۰.۷۹ بود، افزایش یافته‌است و این بدان معناست که توانایی این الگوریتم برای شناسایی موارد مثبت افزایش یافته‌است.

بالاترین specificity مربوط به الگوریتم رگرسیون لجستیک با مقدار ۰.۸۸ است که با حالت بدون انتخاب ویژگی، برابر است.

بالاترین precision مربوط به الگوریتم رگرسیون لجستیک با مقدار ۰.۸۶ است که نسبت به حالت بدون انتخاب ویژگی که مربوط به الگوریتم همین الگوریتم با مقدار ۰.۸۵ بود، افزایش یافته‌است یعنی در الگوریتم موارد مثبتی که پیش‌بینی شد و واقعا هم مثبت بودند، نسبت به حالت بدون انتخاب ویژگی افزایش یافت.

بیشترین مقدار f1-Score هم در بین این الگوریتم‌ها، مربوط به رگرسیون لجستیک با مقدار ۰.۸۱ است که با حالت بدون انتخاب ویژگی برابر است.

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

از مجموعه داده‌ای که با استفاده از الگوریتم انتخاب ویژگی Backward بدست آمده به عنوان ورودی استفاده می‌کنیم. نتایج در جدول(۴-۹) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
SVM	0.817634	0.757908	0.873756	0.846843	0.791675
Logistic Regression	0.817849	0.787767	0.851665	0.825434	0.797664
NB	0.784839	0.771624	0.802152	0.783677	0.769242
KNN	0.814409	0.794696	0.840485	0.816642	0.795098
Dtree	0.741505	0.696393	0.780442	0.738183	0.708198
Random Forest	0.807742	0.753549	0.855617	0.820214	0.780905
Multi-layer-Perceptron	0.751613	0.736150	0.756620	0.723547	0.724559
XGBoost	0.741505	0.683543	0.791558	0.740411	0.702082
Stochastic-Gradient-Descent	0.794731	0.846154	0.715384	0.781912	0.793007
AdaBoost	0.781613	0.746361	0.816035	0.767512	0.748187
LightGBM	0.764516	0.727524	0.798682	0.745885	0.729785
catBoost	0.830860	0.789838	0.867840	0.845540	0.808391

جدول(۴-۹): نتیجه یادگیری با Backward

بهترین مقدار accuracy برای الگوریتم catBoost با مقدار ۰.۸۳ است که نسبت به حالت بدون انتخاب ویژگی که همین مقدار و مربوط به الگوریتم رگرسیون لجستیک بود در حد چند هزارم اعشار کاهش یافت.

بالاترین recall مربوط به الگوریتم KNN با مقدار ۰.۷۹ است که با حالت بدون انتخاب ویژگی برابر است.

بالاترین specificity مربوط به الگوریتم SVM با مقدار ۰.۸۷ است که نسبت به حالت بدون انتخاب ویژگی، کمی کاهش داشته‌است.

بالاترین precision مربوط به الگوریتم SVM و catBooost با مقدار ۰.۸۴ است که نسبت به حالت بدون انتخاب ویژگی که مربوط به الگوریتم رگرسیون لجستیک با مقدار ۰.۸۵ بود، کمی کاهش یافته‌است و این یعنی در الگوریتم موارد مثبتی که پیش‌بینی شد و واقعا هم مثبت بودند، نسبت به حالت بدون انتخاب ویژگی کاهش یافت. بیشترین مقدار f1-Score هم در بین این الگوریتم‌ها مربوط به catBoost با مقدار ۰.۸۰ است که نسبت به حالت بدون انتخاب ویژگی که مربوط به الگوریتم رگرسیون لجستیک با مقدار ۰.۸۱ بود، کمی کاهش یافته‌است.

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

از مجموعه داده‌ای که با استفاده از الگوریتم انتخاب ویژگی Ridge بدست آمده به عنوان ورودی استفاده می‌کنیم.

نتایج در جدول(۴-۱۰) قابل مشاهده است:

	accuracy	recall	specificity	precision	F1
SVM	0.794731	0.739170	0.851596	0.816324	0.762731
Logistic Regression	0.827849	0.786741	0.868332	0.847696	0.806728
NB	0.794946	0.762881	0.836694	0.805452	0.772740
KNN	0.824624	0.780342	0.870846	0.843725	0.802856
Dtree	0.728280	0.719610	0.741254	0.708939	0.704377
Random Forest	0.794516	0.763817	0.829959	0.791736	0.770572
Multi-layer-Perceptron	0.741398	0.744080	0.749202	0.730293	0.721004
XGBoost	0.778065	0.755983	0.802638	0.773417	0.753723
Stochastic-Gradient-Descent	0.775269	0.636334	0.908693	0.897990	0.694709
AdaBoost	0.794731	0.755214	0.833784	0.795001	0.767486
LightGBM	0.784731	0.743028	0.830544	0.792013	0.755498
catBoost	0.817634	0.768439	0.864764	0.834998	0.791059

جدول(۴-۱۰): نتیجه یادگیری Ridge

بهترین مقدار accuracy برای الگوریتم رگرسیون لجستیک با مقدار ۰.۸۲ است که نسبت به حالت بدون انتخاب ویژگی که مربوط به همین الگوریتم با مقدار ۰.۸۳ بود کمی کاهش‌یافت.

بالاترین recall مربوط به الگوریتم رگرسیون لجستیک با مقدار ۰.۷۸ است که با حالت بدون انتخاب ویژگی برابر است.

بالاترین specificity مربوط به الگوریتم SGD با مقدار ۰.۹۰ است که نسبت به حالت بدون انتخاب ویژگی که مربوط به الگوریتم رگرسیون لجستیک با مقدار ۰.۸۸ بود، افزایش داشته‌است.

بالاترین precision مربوط به الگوریتم SGD با مقدار ۰.۸۹ است که نسبت به حالت بدون انتخاب ویژگی که مربوط به الگوریتم رگرسیون لجستیک با مقدار ۰.۸۵ بود، افزایش داشته‌است و یعنی موارد پیش‌بینی مثبتی که واقعا مثبت بودند افزایش یافته‌است.

بیشترین مقدار f1-Score هم در بین این الگوریتم‌ها مربوط به KNN و رگرسیون لجستیک با مقدار ۰.۸۰ است که نسبت به حالت بدون انتخاب ویژگی که مربوط به الگوریتم رگرسیون لجستیک با مقدار ۰.۸۱ بود کاهش یافته‌است.

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 انتخاب ویژگی correlation با تکنیک رگرسیون لجستیک، انتخاب ویژگی Variance Threshold با تکنیک LightGBM و یا انتخاب ویژگی Forward با تکنیک رگرسیون لجستیک استفاده کنیم که در همه موارد به accuracy با مقدار بالای ۰.۸۴ می‌رسیم.

اگر به دنبال افزایش recall باشیم می‌توانیم از انتخاب ویژگی Backward با تکنیک SGD استفاده کنیم و که به recall با مقدار بالای ۰.۸۴ می‌رسیم.

اگر به دنبال افزایش specificity باشیم می‌توانیم از انتخاب ویژگی Ridge با تکنیم SGD استفاده کنیم و به Specificity با مقدار ۰.۹۰ می‌رسیم.

اگر به دنبال افزایش F1-Score باشیم می‌توانیم از انتخاب ویژگی Correlation با تکنیک رگرسیون لجستیک، انتخاب ویژگی Variance Threshold با تکنیک رگرسیون لجستیک، انتخاب ویژگی Variance Threshold با تکنیک LightGBM و یا انتخاب ویژگی Forward با تکنیک رگرسیون لجستیک استفاده کنیم که در همه موارد به F1-Score با مقدار بالای ۰.۸۱ می‌رسیم.

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

- بدون استفاده از انتخاب ویژگی می‌توانیم از الگوریتم رگرسیون لجستیک استفاده کنیم که به مقادیر جدول(۵-۱) می‌رسیم.

پارامترها	مقدار
Accuracy	0.837634
Recall	0.790223
Specificity	0.882577
Precision	0.855234
F1-Score	0.814679

جدول(۵-۱): پارامترهای یادگیری با رگرسیون لجستیک

- با استفاده از انتخاب ویژگی correlation می‌توانیم از تکنیک رگرسیون لجستیک استفاده کنیم که به مقادیر جدول(۵-۲) می‌رسیم:

پارامترها	مقدار
Accuracy	0.840968
Recall	0.789960
Specificity	0.887221
Precision	0.872508
F1-Score	0.818907

جدول(۵-۲): پارامترهای یادگیری با Correlation

- با استفاده از انتخاب ویژگی Variance Threshold می‌توانیم از تکنیک رگرسیون لجستیک استفاده کنیم که به مقادیر جدول(۵-۳) می‌رسیم:

پارامترها	مقدار
Accuracy	0.837634
Recall	0.789318
Specificity	0.881338
Precision	0.857073
F1-Score	0.811398

جدول(۵-۳): پارامترهای یادگیری با Variance threshold

- با استفاده از انتخاب ویژگی RFE می‌توانیم از تکنیک رگرسیون لجستیک و یا catBoost استفاده کنیم که به مقادیر جدول(۵-۴) می‌رسیم:

تکنیک یادگیری	catBoost	رگرسیون لجستیک
Accuracy	0.824624	0.824516
Recall	0.754053	0.794433
Specificity	0.893103	0.856109
Precision	0.869971	0.833953
F1-Score	0.797868	0.806125

جدول(۵-۴): پارامترهای یادگیری با RFE

- با استفاده از انتخاب ویژگی Forward می‌توانیم از تکنیک رگرسیون لجستیک استفاده کنیم که به مقادیر جدول(۵-۵) می‌رسیم:

پارامترها	مقدار
Accuracy	0.840860
Recall	0.796248
Specificity	0.882907
Precision	0.863868
F1-Score	0.817339

جدول(۵-۵): پارامترهای یادگیری با Forward

- با استفاده از انتخاب ویژگی Backward می‌توانیم از تکنیک catBoost استفاده کنیم که به مقادیر جدول(۵-۶) می‌رسیم:

پارامترها	مقدار
Accuracy	0.830860
Recall	0.789838
Specificity	0.867840
Precision	0.845540
F1-Score	0.808391

جدول(۵-۶): پارامترهای یادگیری با Backward

- با استفاده از انتخاب ویژگی Ridge میتوانیم از تکنیک KNN استفاده کنیم که به مقادیر جدول(۵-۷) می‌رسیم:

پارامترها	مقدار
Accuracy	0.824624
Recall	0.780342
Specificity	0.870846
Precision	0.843725
F1-Score	0.802856

جدول(۵-۷): پارامترهای یادگیری با Ridge

کد پیاده‌سازی این پایان نامه را می‌توانید از https://github.com/amirrezazare1379/Heart-disease-prediction.git **دریافت کنید.**

مراجع

[1] C. Boukhatem, H. Y. Youssef, and A. B. Nassif, "Heart Disease Prediction Using Machine Learning," *2022 Adv. Sci. Eng. Technol. Int. Conf. ASET 2022*, vol. 9, no. 04, pp. 659–662, 2022, doi: 10.1109/ASET53988.2022.9734880.

[2] S. Mohan, C. Thirumalai, and G. Srivastava, "Effective heart disease prediction using hybrid machine learning techniques," *IEEE Access*, vol. 7, pp. 81542–81554, 2019, doi: 10.1109/ACCESS.2019.2923707.

[3] S. Guruprasad, V. L. Mathias, and W. Dcunha, "Heart Disease Prediction Using Machine Learning Techniques," *2021 5th Int. Conf. Electr. Electron. Commun. Comput. Technol. Optim. Tech. ICEECCOT 2021 - Proc.*, vol. 1, no. 6, pp. 762–766, 2021, doi: 10.1109/ICEECCOT52851.2021.9707966.

[4] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," in *Springer Tracts in Advanced Robotics*, vol. 61, Springer, 2010, pp. 7–13.

[5] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Trans. Neural Networks*, vol. 20, no. 3, p. 542, 2009.

[6] C. M. Bishop, "Model-based machine learning," *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, vol. 371, no. 1984, p. 20120222, 2013.

[7] V. Gullapalli, "A stochastic reinforcement learning algorithm for learning real-valued functions," *Neural Networks*, vol. 3, no. 6, pp. 671–692, 1990, doi: 10.1016/0893-6080(90)90056-Q.

[8] S. Humpage, "An introduction to regression analysis," *Sensors (Peterborough, NH)*, vol. 17, no. 9, pp. 68–74, 2000, doi: 10.1002/9781118267912.ch6.

[9] A. D. Gordon, *Classification*. CRC Press, 1999.

[10] L. Phipps, "We need to talk about consumption," in *GreenBiz Group*, 2021, pp. 4485–4494.

[11] P. Rani, R. Kumar, and A. Jain, "Multistage model for accurate prediction of missing values using imputation methods in heart disease dataset," in *Lecture Notes on Data Engineering and Communications Technologies*, vol. 59, Springer, 2021, pp. 637–653.

[12] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proceedings - IEEE Computer Security Foundations Symposium*, 2018, vol. 2018-July, pp. 268–282, doi: 10.1109/CSF.2018.00027.

[13] X. Ying, "An Overview of Overfitting and its Solutions," in *Journal of Physics: Conference Series*, 2019, vol. 1168, no. 2, p. 22022, doi: 10.1088/1742-6596/1168/2/022022.

[14] H. K. Jabbar and R. Z. Khan, "Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study)," *Comput. Sci. Commun. Instrum. Devices*, vol. 70, pp. 163–172, 2015, doi: 10.3850/978-981-09-5247-1_017.

[15] I. G. and Y. B. and A. Courville, "Deep learning 简介 ー、什么是 Deep Learning ？," *Nature*, vol. 29, no. 7553, pp. 1–73, 2016, [Online]. Available: http://deeplearning.net/.

[16] J. Li *et al.*, "Feature selection: A data perspective," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 1–45, 2017, doi: 10.1145/3136625.

[17] L. Yu and H. Liu, “Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution,” in *Proceedings, Twentieth International Conference on Machine Learning*, 2003, vol. 2, pp. 856–863.

[18] M. A. Hall, “Correlation-based Feature Selection for Machine Learning,” no. April, 1999.

[19] S. K. das Subrata, “Feature Selection with a Linear Dependence Measure,” *IEEE Trans. Comput.*, vol. C–20, no. 9, pp. 1106–1109, 1971, doi: 10.1109/T-C.1971.223412.

[20] I. Iguyon and A. Elisseeff, “An introduction to variable and feature selection,” *J. Mach. Learn. Res.*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[21] B. Butcher and B. J. Smith, “Feature Engineering and Selection: A Practical Approach for Predictive Models,” *The American Statistician*, vol. 74, no. 3. Taylor & Francis, pp. 308–309, 2020, doi: 10.1080/00031305.2020.1790217.

[22] M. L. Samb, F. Camara, S. Ndiaye, and Y. Slimani, “A Novel RFE-SVM-based Feature Selection Approach for Classification,” *Int. J. Adv. Sci. Technol.*, vol. 43, no. 1, pp. 27–36, 2012.

[23] D. Ververidis and C. Kotropoulos, “Sequential forward feature selection with low computational cost,” in *13th European Signal Processing Conference, EUSIPCO 2005*, 2005, pp. 1063–1066.

[24] B. Remeseiro and V. Bolon-Canedo, “A review of feature selection methods in medical applications,” in *Computers in Biology and Medicine*, 2019, vol. 112, pp. 1200–1205, doi: 10.1016/j.compbiomed.2019.103375.

[25] G. C. McDonald, “Ridge regression,” *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 1, no. 1, pp. 93–100, 2009, doi: 10.1002/wics.14.

[26] I. Rish, “IBM Research Report An empirical study of the naive Bayes classifier,” in *Science*, 2001, vol. 22230, no. 22, pp. 41–46.

[27] G. Turbine, E. Using, S. Vector, and A. N. Network, “Support Vector Machine과 인공신경망을 이용한 가스터빈 엔진의 복합 결함 진단에 관한 연구 인하대학교 대학원 항공공학과 박 준철 Support Vector Machine과 인공신경망을 이용한 가스터빈 엔진의 복합 결함 진단에 관한 연구 인하대학교 대학원 항공공학과 박준철,” in **情報処理学会研究報告. 自然言語処理研究会報告**, vol. 2001, no. 112, Springer, 2001, pp. 33–38.

[28] R. E. Wright, “Logistic regression.,” 1995.

[29] H. Chen *et al.*, “Logistic regression over encrypted data from fully homomorphic encryption,” *BMC Med. Genomics*, vol. 11, no. 4, pp. 3–12, 2018, doi: 10.1186/s12920-018-0397-z.

[30] S. Menard, *Applied Logistic Regression Analysis*, no. 106. Sage, 2011.

[31] J. M. Keller and M. R. Gray, “A Fuzzy K-Nearest Neighbor Algorithm,” *IEEE Trans. Syst. Man Cybern.*, vol. SMC-15, no. 4, pp. 580–585, 1985, doi: 10.1109/TSMC.1985.6313426.

[32] P. E. Danielsson, “Euclidean distance mapping,” *Comput. Graph. Image Process.*, vol. 14, no. 3, pp. 227–248, 1980, doi: 10.1016/0146-664X(80)90054-4.

[33] M. E. Yahia and B. A. Ibrahim, “K-nearest neighbor and C4.5 algorithms as data mining methods: advantages and difficulties,” *Comput. Syst. Appl.*, vol. 103, p. 103, 2004, doi: 10.1109/aiccsa.2003.1227535.

[34] B. R. Patel and K. K. Rana, "A Survey on Decision Tree Algorithm For Classification," *Ijedr*, vol. 2, no. 1, pp. 1–5, 2014.

[35] M. Pal, "Random forest classifier for remote sensing classification," *Int. J. Remote Sens.*, vol. 26, no. 1, pp. 217–222, 2005, doi: 10.1080/01431160412331269698.

[36] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*, Elsevier, 1989, pp. 593–605.

[37] G. Asadollahfardi, "Artificial Neural Network," in *Interdisciplinary computing in java programming*, Springer, 2015, pp. 77–91.

[38] K. Budholiya, S. K. Shrivastava, and V. Sharma, "An optimized XGBoost based diagnostic system for effective prediction of heart disease," *J. King Saud Univ. - Comput. Inf. Sci.*, 2020, doi: 10.1016/j.jksuci.2020.10.013.

[39] T. Chen and T. He, "xgboost: Extreme Gradient Boosting," *R Lect.*, vol. 1, no. 2016, pp. 1–84, 2014.

[40] L. Bottou, "Stochastic gradient descent tricks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7700 LECTU, Springer, 2012, pp. 421–436.

[41] S. Song, K. Chaudhuri, and A. D. Sarwate, "Stochastic gradient descent with differentially private updates," in *2013 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013 - Proceedings*, 2013, pp. 245–248, doi: 10.1109/GlobalSIP.2013.6736861.

[42] X. Li, L. Wang, and E. Sung, "AdaBoost with SVM-based component classifiers," *Eng. Appl. Artif. Intell.*, vol. 21, no. 5, pp. 785–795, 2008, doi: 10.1016/j.engappai.2007.07.001.

[43] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, pp. 3147–3155, 2017.

[44] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," *Adv. Neural Inf. Process. Syst.*, vol. 2018-Decem, pp. 6638–6648, 2018.

[45] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation.," *Encycl. database Syst.*, vol. 5, pp. 532–538, 2009.

[46] R. Williams, T. Shongwe, A. N. Hasan, and V. Rameshar, "Heart Disease Prediction using Machine Learning Techniques," *2021 Int. Conf. Data Anal. Bus. Ind. ICDABI 2021*, vol. 7, no. 2.8, pp. 118–123, 2021, doi: 10.1109/ICDABI53623.2021.9655783.

[47] Y. Khourdifi and M. Bahaj, "K-Nearest Neighbour Model Optimized by Particle Swarm Optimization and Ant Colony Optimization for Heart Disease Classification," *Stud. Big Data*, vol. 53, no. 1, pp. 215–224, 2019, doi: 10.1007/978-3-030-12048-1_23.

[48] R. Detrano, "Long Beach and Cleveland Clinic Foundation," *VA Med. Cent. http//archive. ics. uci. edu/ml/datasets/Heart+ Dis.*, 1989.

[49] M. Yin, J. W. Vaughan, and H. Wallach, "Understanding the effect of accuracy on trust in machine learning models," in *Conference on Human Factors in Computing Systems - Proceedings*, 2019, pp. 1–12, doi: 10.1145/3290605.3300509.

[50] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *ACM International Conference Proceeding Series*, 2006, vol. 148, pp. 233–240, doi:

10.1145/1143844.1143874.

[51] Y. Xiong, Y. Qiao, D. Kihara, H.-Y. Zhang, X. Zhu, and D.-Q. Wei, "Survey of Machine Learning Techniques for Prediction of the Isoform Specificity of Cytochrome P450 Substrates," *Curr. Drug Metab.*, vol. 20, no. 3, pp. 229–235, 2018, doi: 10.2174/1389200219666181019094526.

[52] D. Bzdok and A. Meyer-Lindenberg, "Machine Learning for Precision Psychiatry: Opportunities and Challenges," *Biol. Psychiatry Cogn. Neurosci. Neuroimaging*, vol. 3, no. 3, pp. 223–230, 2018, doi: 10.1016/j.bpsc.2017.11.007.

[53] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 1–13, 2020, doi: 10.1186/s12864-019-6413-7.

[54] K. de Jong, "Learning with Genetic Algorithms: An Overview," *Mach. Learn.*, vol. 3, no. 2, pp. 121–138, 1988, doi: 10.1023/A:1022606120092.

[55] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011, doi: 10.1016/j.swevo.2011.02.002.

[56] H. M. Gomes, J. P. Barddal, A. F. Enembreck, and A. Bifet, "A survey on ensemble learning for data stream classification," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 1–36, 2017, doi: 10.1145/3054925.

[57] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 8, no. 4, p. e1249, 2018.

Abstract

The heart is the most important part of the human body, which is responsible for pumping oxygen-rich blood to other parts of the body through a network of arteries and veins. Any type of disorder that affects our heart is a heart disease. According to the World Health Organization report published in 2019, about 17 million people worldwide die from heart disease every year.

Diagnosing heart disease through early symptoms is a big challenge in the current world scenario. If not diagnosed in time, it may be the cause of death. In developing countries where cardiologists are not available in remote, semi-urban and rural areas, an accurate decision support system can play a vital role in diagnosing heart disease in the early stages. In this thesis, in order to diagnose heart disease from the data set, different machine learning algorithms such as logistic regression, neural network, support vector machine, k nearest neighbor, etc., using the selection of different features such as correlation method, variance threshold , forward, backward, etc. have been investigated and various results have been obtained, which in most cases, the logistic regression algorithm has performed well.

Using correlation and leading feature selection with logistic regression technique as well as variance threshold feature selection method with LightGBM technique, we reach a high accuracy of 0.84%.

In this thesis, we will review and analyze machine learning algorithms with each of the feature selection algorithms, and finally, we will introduce the best methods for predicting heart disease.

Babol Noshirvani University of Technology

Department of Electrical and Computer Engineering

Subject

Check the effect of feature selections in machine learning techniques to increase the accuracy of heart disease prediction

Supervisor

Dr. Fateme Zamani

By

Amirreza Zare

July 2022