



دانشگاه صنعتی امیر کبیر
(پلی تکنیک تهران)



پژوهشگاه دانش‌های پیوادی

گزارش کارآموزی

دانشکده مهندسی کامپیوتر

بررسی الگوریتم های یادگیری ماشین در صنعت

نگارش

مریم کرمانشاهانی

استاد راهنما

دکتر سید احمد جوادی

شهریور ۱۴۰۰

پاسکزاری

بدین وسیله مراتب قدردانی و امتنان خود را خدمت،

جناب آقای امیرحسین هادیان و جناب آقای محمد اخوان بابت آموزش ها، کمک بی دریغ و راهنمایی های همیشگی شان،

جناب آقای دکتر دارا رحمتی مدیر محترم مرکز پردازش سریع پژوهشگاه دانش های بنیادی

جناب آقای دکتر دارا رحمتی، مدیر مرکز پردازش سریع پژوهشگاه دانش های بنیادین

جناب آقای دکتر سید احمد جوادی، استاد گرانقدر کارآموزی

سرکار خانم دکتر سعیده ممتازی، معاون محترم پژوهشی و عضو هیئت علمی دانشکده مهندسی کامپیوتر دانشگاه صنعتی
امیرکبیر

و سرکار خانم سلیمانی مسئول دفتر دانشکده و امور پژوهشی

ابراز و از تمامی زحمات آنان تشکر می نمایم.

مریم کراثانی

شهریور ۱۴۰۱

چکیده

ما احتمالا در تعیین کننده‌ترین دوره تاریخ بشر زندگی می‌کنیم. آنچه که این دوره را تعیین کننده می‌کند مسیری است که هوش مصنوعی و زیرشاخه‌های آن مقابله می‌دهد که یکی از این زیر شاخه‌ها، یادگیری ماشین می‌باشد. یادگیری ماشین یک زیرشاخه معروف در هوش مصنوعی است که به ماشین‌ها یا رایانه‌ها کمک می‌کند که بتوانند بدون برنامه‌ریزی قبلی و با الگو گرفتن از عملکرد خودشان، تصمیم بگیرند و عمل کنند. این الگوریتم‌ها به چهار دسته کلی الگوریتم‌های یادگیری نظارت شده، نظارت نشده، نیمه نظارتی و یادگیری تقویتی تقسیم می‌شوند. تمرکز این دوره کارآموزی روی دو دسته الگوریتم اول می‌باشد.

الگوریتم‌های یادگیری نظارت شده و نظارت نشده، کاربردهای بسیاری در صنعت دارند. از جمله این کاربردها می‌توان به کاربرد در تشخیص بیماری‌ها، سیستم‌های پیشنهاد دهنده، تشخیص چهره، تشخیص دست خط، تشخیص ارقام، ساخت ربات‌ها و غیره دارند. با این حال هنوز چالش‌های زیادی در زمینه علوم داده و یادگیری ماشین وجوددارد که در انتظار حل شدن اند. این گزارش نیز، مروری جامع بر الگوریتم‌های یادگیری ماشین در دو دسته ذکر شده می‌کند.

واژه‌های کلیدی:

یادگیری ماشین، الگوریتم‌های طبقه‌بندی، یادگیری عمیق، یادگیری ماشین نظارت شده، یادگیری ماشین نظارت نشده

فهرست مطالب

عنوان	صفحه
مقدمه	۱.....
آشنایی با پژوهشگاه دانش‌های بنیادین	۴.....
۲-۱ آشنایی با پژوهشگاه	۵.....
۱-۱-۲ وظایف و فعالیت‌ها	۵
۲-۱-۲ ساختار پژوهشی	۶.....
۱-۲-۳ طرح‌های ملی	۶.....
۱-۲-۴ شبکه ارتباطات الکترونیکی و دامنه اینترنتی	۷
۱-۲-۵ محل استقرار پژوهشگاه دانش‌های بنیادی	۷
جمع بندی	۸.....
آشنایی اولیه با یادگیری ماشین	۹.....
۱-۳ تعریف اولیه	۱۰
۲-۳ انواع یادگیری ماشین	۱۰
۳-۳ کاربردهای یادگیری ماشین	۱۱.....
۴-۳ فرآیندهای یادگیری ماشین	۱۳.....
جمع بندی	۱۴.....
یادگیری ماشین ناظارت شده	۱۵.....
۴-۱ تعاریف اولیه	۱۶
۴-۲ انواع مسائل یادگیری ماشین ناظارت شده	۱۶
۱-۲-۴ رگرسیون	۱۷.....
۱-۱-۲-۴ رگرسیون خطی ساده و چندگانه	۱۷.....

۸۴.....	۵-۳-۴ طبقه‌بندی مجموعه داده تخمین قیمت خانه‌های سنگاپور
۸۴.....	۱-۴-۳-۴ فرآیند مدلسازی سایکیتلرن روی مجموعه داده خانه‌های سنگاپور
۱۰۰	یادگیری ماشین ناظارت‌نشده
۱۰۱	یادگیری ناظارت‌نشده
۱۰۱	۱-۵ تعاریف اولیه
۱۰۱	۲-۵ انواع مسائل یادگیری ماشین بدون ناظر
۱۰۱.....	۱-۲-۵ خوشبندی
۱۰۲.....	۱-۱-۲-۵ الگوریتم خوشبندی میانگین کا
۱۰۴.....	۲-۱-۲-۵ الگوریتم خوشبندی انتقال میانگین
۱۰۶.....	۳-۱-۲-۵ الگوریتم خوشبندی فضایی مبتنی بر چگالی کاربردهای دارای نویز (دی‌بی‌اسکن)
۱۱۰	۲-۲-۵ الگوریتم‌های کاهش ابعاد
۱۱۰	۱-۲-۲-۵ الگوریتم تحلیل مولفه اساسی
۱۱۱	۵-۳ پیاده‌سازی
۱۱۱.....	۱-۳-۵ خوشبندی مجموعه داده خرده فروشی آنلайн
۱۱۲.....	۱-۳-۱-۵ فرآیند مدلسازی سایکیتلرن روی مجموعه داده خرده فروشی آنلайн
۱۲۳	نتیجه گیری و پیشنهادها
۱۲۴	نتیجه گیری
۱۲۷	پیشنهادها
۱۳۸	منابع و مراجع

فهرست اشکال

..... ۳	شکل ۱ - نمودار مراحل بیان مطالب در این گزارش
..... ۱۷	شکل ۲ - یادگیری نظارت شده
..... ۲۴	شکل ۳ - پیاده‌سازی الگوریتم کا همسایه نزدیک با زبان پایتون با یک مثال ساده
..... ۲۵	شکل ۴ - شکل خط رگرسیون بردار پشتیبان
..... ۲۶	شکل ۵ - منطق شهودی رگرسیون بردار پشتیبان در فضای دو بعدی به همراه داده‌های دورافتاده
..... ۲۶	شکل ۶ - تعمیم رگرسیون بردار پشتیبان
..... ۲۷	شکل ۷ - پیاده‌سازی رگرسیون بردار پشتیبان با زبان پایتون با یک مثال ساده
..... ۲۸	شکل ۸ - نمودار درخت تصمیم یک مثال
..... ۲۹	شکل ۹ - طبقه‌بندی چند کلاسه در فضای دو بعدی
..... ۳۰	شکل ۱۰ - تفاوت طبقه‌نده چند کلاسه و چند برچسبی
..... ۳۰	شکل ۱۱ - طبقه‌بندی نامتوازن در فضای دو بعدی
..... ۳۱	شکل ۱۲ - مثال شهودی از ورودی خروجی رگرسیون لجیستیک
..... ۳۲	شکل ۱۳ - تفاوت رگرسیون خطی و لجیستیک
..... ۳۲	شکل ۱۴ - نمودارتابع سیگموید
..... ۳۴	شکل ۱۵ - توزیع داده‌ها روی نمودار
..... ۳۴	شکل ۱۶ - تعیین کلاس نمونه جدید
..... ۳۵	شکل ۱۷ - آموزش مدل کا - همسایه نزدیک
..... ۳۵	شکل ۱۸ - سنجیدن مدل کا - همسایه نزدیک
..... ۳۵	شکل ۱۹ - خروجی سنجش مدل کا - همسایه نزدیک
..... ۳۷	شکل ۲۰ - آموزش مدل بیز ساده
..... ۳۷	شکل ۲۱ - سنجش مدل بیز ساده
..... ۳۸	شکل ۲۲ - خروجی سنجش مدل بیز ساده
..... ۴۰	شکل ۲۳ - درخت تصمیم مجموعه داده تایتانیک
..... ۴۰	شکل ۲۴ - آموزش مدل درخت تصمیم
..... ۴۰	شکل ۲۵ - سنجیدن مدل درخت تصمیم
..... ۴۱	شکل ۲۶ - خروجی سنجیدن مدل درخت تصمیم
..... ۴۱	شکل ۲۷ - آموزش مدل جنگل تصادفی
..... ۴۲	شکل ۲۸ - سنجش مدل جنگل تصادفی

شکل ۲۹- خروجی سنجش مدل جنگل تصادفی.....	۴۲
شکل ۳۰- شکل مرزبندی بین دو کلاس خرگوش و بیر.....	۴۳
شکل ۳۱- شکل معرفی مفاهیم مهم ماشین بردار پشتیبان.....	۴۳
شکل ۳۲- شکل داده جدید نمودار.....	۴۳
شکل ۳۳- شکل مرزبندی نهایی داده جدید	۴۴
شکل ۳۴- شکل داده های غیرخطی.....	۴۶
شکل ۳۵- شکل داده ها بعد از انتقال به فضای سه بعدی	۴۶
شکل ۳۶- تفکیک داده توسط ماشین بردار پشتیبان در فضای جدید.....	۴۷
شکل ۳۷- تفکیک داده توسط ماشین بردار پشتیبان در فضای قلبی.....	۴۷
شکل ۳۸- پیاده سازی ماشین بردار پشتیبان با زبان پایتون با یک مثال ساده	۴۸
شکل ۳۹- مسیر کلی فرآیند مدل سازی در سایکیت لرن	۴۸

فصل اول

مقدمه

مقدمه

در این دوره کارآموزی، در ابتدا مروی بر جبر خطی و ریاضیات پایه یادگیری ماشین انجام شد. سپس با پکیج‌های مختلف مورد استفاده در یادگیری ماشین، شامل کتابخانه‌های نامپای، پانداز، مت‌پلات‌لیب، سیبورن و سایکیت‌لرن آشنا شدیم و نحوه استفاده از آن‌ها را یادگرفتیم. پس از آن وارد مبحث یادگیری ماشین شدیم. آموختیم که الگوریتم‌های یادگیری ماشین به صورت کلی به چهار دسته الگوریتم‌های یادگیری ماشین نظارت شده، نظارت نشده، نیمه نظارتی و یادگیری تقویتی تقسیم می‌شوند که در این دوره کارآموزی تمرکز بر روی دو دسته اول بود. در ادامه الگوریتم‌های یادگیری ماشین نظارت شده و یادگیری ماشین نظارت نشده و نحوه پیاده‌سازی آن‌ها با کتابخانه سایکیت‌لرن را مطالعه کردیم. در نهایت هم تعدادی مجموعه داده، به ما سپرده شد تا الگوریتم‌های مطالعه شده را روی آن‌ها پیاده کنیم.

یادگیری ماشین نظارت شده، براساس مشاهدات انجام شده کار می‌کند؛ این بدان معناست که، کاربر ماشین‌ها را بر روی مجموعه داده‌هایی که دارای "برچسب" هستند، آموزش می‌دهد و بر اساس آموزش صورت گرفته، ماشین خروجی یا رویداد آینده را پیش‌بینی می‌کند. کلمه "برچسب" یا لیبل مشخص کننده این است که مجموعه‌ای از داده‌ها و سوابق وجود دارد (ورویدی‌ها) که قبلاً یک تصمیم درست (خرجی صحیح) برای آن‌ها در دیتاست ثبت شده است که بسیار با ارزش هستند. حال می‌توان ماشین را با ورودی‌ها و خروجی‌های موجود در دیتاست آموزش دهیم و سپس از ماشین بخواهیم با توجه به آموزشی که دیده است، خروجی صحیح را برای ورودی‌های جدید پیش‌بینی کند. این دسته از الگوریتم‌ها کاربردهای بسیار زیادی مانند تشخیص چهره، تشخیص دست‌خط، بیماری و غیره دارند که به برخی از این کاربردها در این دوره کارآموزی پرداخته شد.

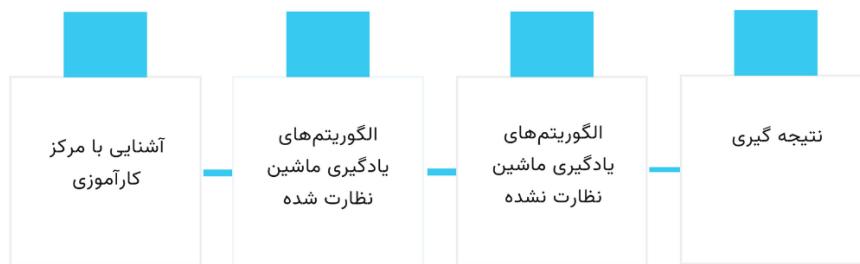
الگوریتم‌های یادگیری ماشین نظارت شده به دو دسته الگوریتم‌های طبقه‌بندی برای داده‌های گستته و الگوریتم‌های رگرسیون تقسیم می‌شوند. الگوریتم‌های کی‌همسایه نزدیک، رگرسیون لاجیستیک، درخت تصمیم، جنگل تصادفی، بیز ساده و ماشین بردار پشتیبان از جمله الگوریتم‌های طبقه‌بندی‌اند. الگوریتم‌های رگرسیون نیز به دو دسته کلی الگوریتم‌های رگرسیون خطی و غیر خطی تقسیم می‌شوند. الگوریتم‌های رگرسیون خطی شامل الگوریتم رگرسیون خطی ساده، رگرسیون چندجمله‌ای، لاسو و ستیغی شده و الگوریتم‌های رگرسیون خطی شامل الگوریتم‌های کی-همسایه نزدیک، ماشین بردار پشتیبان، درخت تصمیم، جنگل تصادفی‌اند.

الگوریتم‌های یادگیری ماشین نظارت نشده نیز به دو دسته خوش‌بندی و کاهش بعد تقسیم می‌شوند. الگوریتم‌های مطالعه شده در دسته خوش‌بندی شامل کی-میانگین، شیفت میانگین و خوش‌بندی فضایی می‌باشند که بر چگالی برنامه‌های کاربردی با نویز (دی‌بی‌اسکن) بود و در دسته الگوریتم‌های کاهش بعد هم الگوریتم تحلیل مولفه اصلی مطالعه شد. از جمله کاربردهای این الگوریتم‌ها نیز می‌توان به تشخیص الگو، بررسی شخصیت مشتری، سیستم‌های پیشنهاد دهنده اشاره کرد.

در این دوره کارآموزی این الگوریتم‌ها روی چندین مجموعه داده پیاده سازی شدند. از جمله مجموعه داده‌های مهمی که در دوره کارآموزی با در مجموعه داده مربوط به شاخص سلامت قلبی افراد، جدول داده‌ای شامل یک سری ویژگی‌ها مانند سن،

جنسیت، تحصیلات، حقوق و غیره در کنار بیماری قلبی داشتن یا نداشتن این افراد جمع آوری شده است و ما سعی کردیم با الگوریتم‌های دسته‌بندی^۱ مدلی ایجاد کنیم که بتواند، براساس این ویژگی‌ها و با کمترین خطأ، پیش‌بیماری داشتن یا نداشتن را انجام دهد. به صورت کلی، مجموعه داده‌هایی که به اینجانب انتساب شد، شامل مجموعه داده‌های شاخص سلامت قلبی افراد، کلیک روی تبلیغات، کمپین‌های بازاریابی مستقیم یک موسسه بانکی، مسافران کشتی تایتانیک، پیش‌بینی قیمت خانه‌های سنگاپور و مجموعه داده مربوط به خرده فروشی آنلاین بود.

انگیزه‌ی این گزارش، مروری بر هدف کلی این دوره کارآموزی یعنی بررسی الگوریتم‌های ماشین نظارت شده و نظارت نشده و پیاده سازی آن‌ها روی شش مجموعه داده بود. در ابتدای این گزارش، به معرفی محل کارآموزی می‌پردازیم. در فصل سوم به بررسی الگوریتم‌های یادگیری نظارت شده و پیاده‌سازی آن روی مجموعه داده‌های شاخص سلامت قلبی افراد، کلیک روی تبلیغات، کمپین‌های بازاریابی مستقیم یک موسسه بانکی، مسافران کشتی تایتانیک می‌پردازیم. در فصل چهارم نیز به بررسی الگوریتم‌های یادگیری نظارت نشده و پیاده‌سازی آن‌ها بر روی مجموعه داده‌های مربوطه می‌پردازیم. در فصل آخر نیز به نتیجه گیری، نگاهی اجمالی به موارد ذکر شده خواهیم داشت.



شکل ۱- نمودار مراحل بیان مطلب در این گزارش

¹ Classification

فصل دوم

آشنایی با پژوهشگاه دانش‌های بنیادین

آشنایی با پژوهشگاه دانش‌های بنیادین

در این فصل به معرفی پژوهشگاه دانش‌های بنیادین، مرکز پردازش سریع در این پژوهشگاه و حوزه فعالیت این مرکز پرداخته می‌شود.

۱-۲ آشنایی با پژوهشگاه

پژوهشگاه دانش‌های بنیادی مؤسسه‌ای وابسته به وزارت علوم، تحقیقات، و فناوری است که در سال ۱۳۶۸ با نام «مرکز تحقیقات فیزیک نظری و ریاضیات» تأسیس شد و هدف اولیه آن پیشبرد پژوهش و نوآوری در این دو رشته، و ضمناً فراهم آوردن الگویی بود که به ترویج و اعتلای فرهنگ پژوهش در سطح کشور کمک کند. مرکز فعالیت خود را با ۳ هسته تحقیقاتی در فیزیک نظری و ۳ هسته تحقیقاتی در ریاضیات و با امکاناتی اندک آغاز کرد ولی به تدریج با توسعه امکانات و جذب دانشورانی از رشته‌های دیگر، فعالیت آن به حیطه‌های دیگری گسترش یافت و در سال ۱۳۷۶ نام آن به «پژوهشگاه دانش‌های بنیادی» تغییر کرد. این پژوهشگاه در حال حاضر با ۹ پژوهشکده در زمینه‌های گوناگون علوم بنیادی و برخورداری از زیرساخت‌ها و امکانات لازم (شبکه الکترونیکی، کامپیوتر، آزمایشگاه‌ها، و کتابخانه مجهز و روزآمد) که دائماً هم رو به توسعه است، حضور فعالی در جریان پژوهشی کشور در این دانش‌ها دارد. ابتکار پژوهشگاه در ایجاد شبکه ارتباطی الکترونیکی (به نام شبکه علمی-تحقیقاتی ایران یا ایرانت) در سال ۱۳۷۱، که به تدریج و برای اولین بار مراکز علمی-تحقیقی و دانشگاه‌های ایران را به یکدیگر و به جهان علم در خارج مربوط ساخت، و اهتمام پژوهشگاه به اجرای طرح‌های ملی، مانند رصدخانه ملی ایران و شتابگر ملی، از جلوه‌ها و ثمرات این نگرش ملی است.

۱-۱-۲ وظایف و فعالیت‌ها

- انجام تحقیقات در زمینه‌های مرتبط با موضوع تأسیس پژوهشگاه به طور مستقل و یا با همکاری مراکز علمی و پژوهشی داخل و خارج کشور
- ایجاد ارتباط فعال و سازنده با سایر مؤسسات و جوامع علمی و پژوهشی در داخل و خارج از کشور از طریق برگزاری انواع همایش‌ها، مبادله محقق و اجرای طرح‌های مشترک
- همکاری با دانشگاه‌ها و مراکز آموزش عالی و مؤسسات پژوهشی کشور و سایر نهادها در راستای پیشبرد اهداف موضوع تأسیس پژوهشگاه از طریق ارایه تسهیلات مختلف، پذیرش طرح‌های تحقیقاتی، ایجاد امکان گذراندن فرصت‌های مطالعاتی در پژوهشگاه
- ایجاد زمینه‌های مناسب برای جذب دانشمندان و پژوهشگران ایرانی
- کمک به پرورش محقق در زمینه‌های موضوع تأسیس از طریق دایر کردن دوره‌های تحصیلات تكمیلی و اعطای کمک هزینه تحصیل

- نشر و ترویج یافته‌های علمی در زمینه‌های فعالیت پژوهشگاه از طریق انتشار کتب و نشریات و تشکیل تجمعات پژوهشی و آموزشی
- ارایه خدمات علمی و فنی در چارچوب فعالیت‌های پژوهشگاه
- بررسی و شناسایی نیازهای پژوهشی در زمینه دانش‌های بنیادی
- تأسیس مرکز خدمات شبکه‌ای با هدف برقراری ارتباطات شبکه‌ای جهت ارایه خدمات به پژوهشگاه و سایر مراکز علمی و پژوهشی و متقارضیان دیگر و همچنین تلاش برای توسعه فنون مربوط به شبکه در کشور

۲-۱-۲ ساختار پژوهشی

بخش‌های تحقیقاتی پژوهشگاه متشکل از پژوهشکده‌ها و مراکز تحقیقاتی وابسته است. پژوهشکده‌ها در حال حاضر، پژوهشگاه شامل نه پژوهشکده است:

- پژوهشکده ذرات و شتابگرهای
- پژوهشکده ریاضیات
- پژوهشکده علوم زیستی
- پژوهشکده علوم شناختی
- پژوهشکده علوم کامپیوتر
- پژوهشکده علوم نانو
- پژوهشکده فلسفه تحلیلی
- پژوهشکده فیزیک
- پژوهشکده نجوم

این پژوهشکده‌ها طبق اساسنامه پژوهشگاه از استقلال داخلی برخوردارند.

۳-۱-۲ طرح‌های ملی

این طرح‌ها پروژه‌های تحقیقاتی در مقیاس ملی هستند که اجرای آنها به پژوهشگاه دانش‌های بنیادی واگذار شده و از حمایت مالی دولت برخوردارند. این طرح‌ها در حال حاضر عبارت‌اند از:

- رصد خانه ملی ایران

- همکاری با سرن^۱
- گرید برای محاسبات گسترده علمی
- شتابگر خطی Mev10
- چشم نور ایران
- شبکه علمی دانشگاه‌ها و دامنه کشوری

۴-۱-۲ شبکه ارتباطات الکترونیکی و دامنه اینترنتی

از جمله هدف‌هایی که پژوهشگاه از بدو تأسیس خود پیگیرانه در تحقق آن می‌کوشیده است، ایجاد و توسعه شبکه ارتباطی بین دانشگاه‌ها و مؤسسات علمی و تحقیقاتی داخل و خارج کشور بوده است. به این منظور، شبکه علمی-تحقیقاتی ایران (ایرانت) در سال ۱۳۷۱ زیرنظر مرکز تحقیقات تأسیس شد که از آن زمان تاکنون دائماً در حال گسترش امکانات و خدمات خود به جامعه علمی ایران بوده است. همچنین مجوز ثبت دامنه ir. (دامنه اینترنتی کشوری ایران) از فروردین ۱۳۷۳ به این پژوهشگاه واگذار شده و از مهرماه همان سال، مرکز ثبت دامنه ir. به عنوان بخشی در داخل پژوهشگاه به فعالیت پرداخته است.

۴-۱-۳ محل استقرار پژوهشگاه دانش‌های بنیادی

پژوهشگاه دانش‌های بنیادی اکنون در چهار ساختمان در شمال شرق تهران مستقر است.

- ساختمان نیاورن (ساختمان مرکزی):
 - دفتر ریاست پژوهشگاه؛ معاونت علمی؛ پژوهشکده ریاضیات؛ پژوهشکده علوم‌شناسخی؛ پژوهشکده فلسفه تحلیلی؛ دبیرخانه؛ بخش اداری؛ مرکز اطلاع‌رسانی.
- ساختمان فرمانیه:
 - پژوهشکده فیزیک؛ پژوهشکده علوم نانو؛ پژوهشکده علوم زیستی؛ پژوهشکده علوم کامپیوتر؛ مهمانسری.
- باغ لارک:
 - پژوهشکده نجوم؛ طرح رصدخانه ملی؛ پژوهشکده ذرات و شتابگر؛ طرح چشم نور ایران؛ طرح تورین ملی؛ دفتر محاسبات پژوهشکده علوم نانو؛ کتابخانه فیزیک.
- ساختمان اختیاریه:
 - مرکز شبکه؛ کتابخانه.

¹ European Council for Nuclear Research (CERN)

جمع بندی

در این بخش ابتدا با پژوهشگاه دانش‌های بنیادی که شامل آشنایی کلی، وظایف و فعالیت‌ها، ساختار پژوهشی، طرح‌های ملی، و محل استقرار آن بود آشنا شدیم. در ادامه با الگوریتم‌های ماشین لرینینگ نظارت شده و نظارت نشده و پیاده‌سازی آن‌ها، آشنا خواهیم شد.

فصل سوم

آشنایی اولیه با یادگیری ماشین

فصل سوم

آشنایی اولیه با یادگیری ماشین

۱-۳ تعریف اولیه

یادگیری ماشین، یکی از زیر مجموعه‌های هوش مصنوعی است که به سیستم‌ها امکان یادگیری خودکار را می‌دهد. در علم یادگیری ماشین، به موضوع طراحی ماشین‌هایی پرداخته می‌شود که با استفاده از مثال‌های داده شده به آن‌ها و تجربیات خودشان، بیاموزند. در واقع، در این علم تلاش می‌شود تا با بهره‌گیری از الگوریتم‌ها، یک ماشین به شکلی طراحی شود که بدون آنکه صراحتا برنامه‌ریزی و تک تک اقدامات به آن دیکته شود بتواند بیاموزد و عمل کند. در یادگیری ماشین، به جای برنامه‌نویسی همه چیز، داده‌ها به یک الگوریتم عمومی داده می‌شوند و این الگوریتم است که براساس داده‌هایی که به آن داده شده منطق خود را می‌سازد. یادگیری ماشین روش‌های گوناگونی دارد که از آن جمله می‌توان به یادگیری نظارت شده، نظارت نشده، نیمه نظارتی و یادگیری تقویتی اشاره کرد. الگوریتم‌های مورد استفاده در یادگیری ماشین جزو این چهار دسته هستند. در ادامه به تعریف این چهار دسته خواهیم پرداخت.

۲-۳ انواع یادگیری ماشین

همانطور که بالاتر ذکر کردیم، با توجه به الگوریتم‌هایی که استفاده می‌کنیم و داده‌هایی که به سیستم ارائه می‌دهیم، یادگیری ماشین به چهار دسته تقسیم می‌شود که در ادامه به توضیح مختصراً از هر کدام خواهیم پرداخت.

(۱) یادگیری نظارت شده

در این نوع یادگیری متخصصان داده به عنوان یک ناظر، داده‌هایی را در اختیار ماشین می‌گذارند و انواع داده‌ها را با برچسب‌هایی نام‌گذاری می‌کنند. مثل یک فروشنده که نام هر دسته از محصولات خود را روی برچسبی می‌نویسد و در طبقه‌ای که محصولات چیده شده‌اند می‌چسباند. در این نوع یادگیری ورودی و خروجی مشخص شده است و ماشین تلاش می‌کند تا الگویی از رساندن ورودی به خروجی مورد انتظار را یاد بگیرد. یکی از مثال‌های مرسوم در این نوع یادگیری، تشخیص و فیلتر کردن پیام‌های اسپم از بین دیگر پیام‌های است. متخصصان ابتدا تعداد زیادی پیام را به دو دسته پیام‌های اسپم و پیام‌های واقعی تقسیم می‌کنند و آنها را به ماشین نشان می‌دهند. سپس ماشین با استفاده از ویژگی‌های مشترکی که در این دو دسته از پیام‌ها پیدا می‌کند به تدریج متوجه تفاوت پیام‌های اسپم و واقعی می‌شود و می‌تواند آنها را از هم جدا کند. در پایان با دادن پیام‌های جدید از ماشین امتحان می‌گیرند تا ببینند به درستی تشخیص میدهد یا خیر.

(۲) یادگیری نظارت نشده

در یادگیری نظارت نشده، الگوریتم باید خود به تنها یی به دنبال ساختارهای جالب موجود در داده‌ها باشد. به بیان ریاضی، یادگیری نظارت نشده مربوط به زمانی است که در مجموعه داده فقط متغیرهای ورودی وجود داشته باشند و هیچ متغیر داده خروجی موجود نباشد. به این نوع یادگیری، نظارت نشده گفته می‌شود زیرا برخلاف یادگیری نظارت شده، هیچ پاسخ صحیح داده شده‌ای وجود ندارد و ماشین خود باید به دنبال پاسخ باشد. به بیان دیگر، هنگامی که الگوریتم برای کار کردن از مجموعه داده‌ای بهره گیرد که فاقد داده‌های برچسب‌دار (متغیرهای خروجی) است، از مکانیزم دیگری برای یادگیری و تصمیم‌گیری استفاده می‌کند. به چنین نوع یادگیری، نظارت نشده گفته می‌شود. یادگیری نظارت نشده قابل تقسیم به مسائل خوشه‌بندی و انجمانی است.

(۳) یادگیری نیمه نظارتی

در این روش که ترکیبی از دو روش قبل است، بخش اندکی از داده آموزشی دسته‌بندی شده و مابقی دسته‌بندی نشده‌اند. در این حالت، ربات مانند کودکی است که نقشه راه به او داده شده است و خودش باید مسیر را با خواندن نقشه پیدا کند. جالب است که چنین روشی دقیق یادگیری را به خوبی افزایش می‌دهد.

(۴) یادگیری تقویتی

یک برنامه رایانه‌ای که با محیط پویا در تعامل است باید به هدف خاصی دست‌یابد (مانند بازی کردن با یک رقیب یا راندن خودرو). این برنامه بازخوردهایی را با عنوان پاداش‌ها و تنبیه‌ها فراهم و فضای مساله خود را بر همین اساس هدایت می‌کند. با استفاده از یادگیری تقویتی، ماشین می‌آموزد که تصمیمات مشخصی را در محیطی که دائم در معرض آزمون و خطأ است اتخاذ کند.

۳-۳ کاربردهای یادگیری ماشین

برنامه‌های یادگیری ماشین به صورت نامحدود وجود دارند و تعداد زیادی از الگوریتم‌ها نیز برای یادگیری ماشین در دسترس هستند. آن‌ها از انواع ساده گرفته تا بسیار پیچیده در دسترس هستند. در ادامه چند کاربرد یادگیری ماشین را ذکر می‌کنیم تا با اهمیت این علم بیشتر آشنا شوید.

- (۱) تشخیص تصویر: تشخیص تصویر یکی از رایج‌ترین کاربردهای یادگیری ماشین است. همچنین می‌توان از آن به عنوان یک تصویر دیجیتالی نام برد. برای این تصاویر، اندازه‌گیری، خروجی هر پیکسل در یک تصویر را توصیف می‌کند. تشخیص چهره نیز یکی از ویژگی‌های عالی است که فقط توسط یادگیری ماشین ایجاد شده است. این تکنولوژی کمک می‌کند تا چهره‌ها شناسایی شوند و اعلان‌های مربوط به آن برای افراد ارسال شوند.
- (۲) تشخیص صدا: یادگیری ماشین همچنین به توسعه برنامه برای تشخیص صدا نیز کمک می‌کند. از یادگیری ماشین به عنوان دستیار شخصی مجازی نیز یاد می‌شود. این به شما کمک می‌کند تا در صورت درخواست صوتی، اطلاعات مورد نظر را پیدا کنید. پس از پرسیدن سوال‌تان، آن دستیار به دنبال داده‌ها یا اطلاعاتی می‌گردد که شما از آن پرسیده‌اید و اطلاعات موردنیاز را جمع آوری می‌کند تا بهترین پاسخ را به شما ارائه دهد.

- (۳) پیش‌بینی‌ها (برای مثال در سرویس‌های مسیریابی): این کاربرد ماشین لرنینگ در ساخت برنامه‌هایی که می‌توانند مواردی مانند قیمت تاکسی یا مسافرت را برای مدت زمان خاص و ازدحام ترافیکی را پیش‌بینی کنند، کمک می‌کند. هنگام سفارش تاکسی برنامه می‌تواند تخمین بزند که قیمت تقریبی سفر چقدر است. این کاربرد فقط توسط ماشین لرنینگ ارائه می‌شود. یا هر زمانی که از سرویس مکان‌یابی، برای بررسی مسیر از مبدأ به مقصد استفاده می‌کنیم، برنامه مسیرهای مختلف را به ما نشان می‌دهد و میزان تراکم ترافیک در آن لحظه را بسته به میزان تعداد وسایل نقلیه نشان می‌دهد و مسیری را که در آن وسایل نقلیه کمتری قرار دارند انتخاب می‌کند. همه این کارها با استفاده از برنامه‌ی یادگیری ماشین انجام یا بازیابی می‌شوند.
- (۴) پلتفرم رسانه‌های اجتماعی: به دلیل علاقه کاربران، از رسانه‌های اجتماعی برای تهییه بهتر اخبار و تبلیغات استفاده می‌شود و این کارها عمدتاً فقط با استفاده از یادگیری ماشین قابل انجام هستند. در این مورد مثال‌های زیادی مانند وجود گزینه‌های پیشنهادات دوستی، پیشنهاد صفحه برای فیس بوک، پیشنهاد آهنگ‌ها و فیلم‌ها در یوتیوب وجود دارند. این کار عمدتاً بر اساس تجربه‌های کاربر در رسانه‌ها، مانند ارتباط برقرار کرن شان و پروفایل‌ها یا وب سایت‌هایی که اغلب از آن‌ها بازدید می‌کنند، صورت می‌گیرد و متناسب با آن به کاربر پیشنهاداتی ارائه می‌شوند. همچنین یک روش برای استخراج اطلاعات مفید از تصاویر و فیلم‌ها نیز فراهم می‌کند.
- (۵) هرزنامه یا اسپم و بدافزار: کاربران ایمیل از تعدادی فیلتر اسپم استفاده می‌کنند، که این فیلترها به طور مداوم در حال به روزرسانی هستند و این کار عمدتاً با استفاده از یادگیری ماشین انجام می‌شود. به همین ترتیب، شناسایی تعدادی بدافزار عمدتاً توسط برنامه‌های امنیتی سیستم شناسایی می‌شوند، که این امر هم بیشتر توسط ماشین لرنینگ انجام می‌گردد.
- (۶) موتور جستجو: موتورهای جستجو در هنگام جستجوی کاربر در دسترس هستند تا بهترین نتیجه را به کاربران ارائه دهند. الگوریتم‌های یادگیری ماشین زیادی برای جستجوی درخواست خاص یک کاربر، مانند گوگل ایجاد شده‌اند. هر صفحه‌ای که توسط کاربران برای یک موضوع خاص به طور مکرر باز می‌شود، برای مدت طولانی در بالای صفحه باقی خواهد ماند.
- (۷) تشخیص بیماری: از یادگیری ماشین می‌توان در تکنیک‌ها و ابزارهایی که برای تشخیص بیماری‌ها کاربرد دارند استفاده کرد. از این تکنولوژی می‌توان برای تجزیه و تحلیل پارامترهای بالینی و ترکیب آن‌ها برای پیش‌بینی آگاهی از پیشرفت بیماری، استخراج اطلاعات پزشکی، تحقیقات برای رسیدن به نتیجه، برنامه‌ریزی درمانی و نظارت بر بیمار استفاده کرد. این موارد از کاربردهای موفق استفاده از متدهای یادگیری ماشین می‌باشد. استفاده از الگوریتم‌های یادگیری ماشین همچنین می‌تواند به ادغام سیستمهای کامپیوتری و بخش‌های مراقبت بهداشتی نیز کمک کند.
- (۸) دسته‌بندی: منظور از دسته‌بندی قراردادن هر فرد، شیء و ... در دسته‌های مختلف تحت مطالعه می‌باشد. دسته‌بندی به تجزیه و تحلیل اندازه‌گیری‌های یک شیء کمک می‌کند تا دسته‌های را که به آن تعلق دارد را شناسایی کنیم. برای ایجاد یک رابطه کارآمد، تحلیلگران از داده‌ها استفاده می‌کنند. به عنوان مثال، قبل از اینکه یک بانک تصمیم به توزیع وام بگیرد، مستریان دارای توانایی بازپرداخت وام را ارزیابی می‌کند. بانک در حقیق با در نظر گرفتن عواملی

مانند درآمد مشتری، پس انداز و سابقه مالی و ... این کار را انجام می‌دهد. این اطلاعات از تجزیه و تحلیل و دسته‌بندی داده‌های گذشته در مورد وام بدست می‌آید.

- (۹) امور مالی
- (۱۰) پردازش زبان طبیعی
- (۱۱) زیست شناسی محاسباتی

۴-۳ فرآیندهای یادگیری ماشین

جمع آوری داده

به فرآیند استخراج داده خام برای وظایف یادگیری ماشین، جمع آوری داده می‌گویند. این داده از روش‌های مختلفی مانند منابع آنلاین منبع باز یا منابع غیر رایگان، به دست می‌آید. شاید این مرحله را بتوان مهمترین مرحله یادگیری ماشین نامید. اگر داده جمع آوری شده کیفیت پایینی داشته یا غیر مرتبط باشد، مدل آموزش دیده نیز کیفیت پایینی دارد.

پیش‌پردازش داده

بعد از گردآوری داده‌های مرتبط، نیاز به پیش‌پردازش آنها داریم تا اطمینان حاصل شود داده در فرمت قابل استفاده برای آموزش مدل‌های یادگیری ماشین است. این مرحله شامل مدیریت داده‌های گمشده یا داده‌های پرت می‌باشد.

مهندسی ویژگی

زمانی که دیتاست جمع آوری و پیش‌پردازش شد، ممکن است نیاز به تبدیل یا حذف بعضی از ویژگی‌های دیتاست باشد تا مدل آموزشی بهینه تری به دست آید.

انتخاب مدل

براساس دیتاست، یک مدل یادگیری ماشین انتخاب می‌کنیم. این کار یکی از وظایف مهم مهندسان صنعت است. به جای به کارگیری مدل‌های کاملاً جدید، بیشتر وظایف یادگیری ماشین با روش‌های موجود یا ترکیب روش‌های کنونی، قابل انجام است.

آموزش مدل و پایپ‌لاین داده

بعد از انتخاب مدل، یک پایپ‌لاین (خط لوله) داده برای آموزش مدل ایجاد می‌شود. یعنی جریان پیوسته ایی از داده‌های دسته‌ای ایجاد می‌شود تا مدل را به صورت مناسبی آموزش بینند. از آنجا که آموزش می‌تواند طولانی شود، بهتر است پایپ‌لاین داده تا حد ممکن کارآمد باشد.

اعتبارسنجی مدل

بعد از آموزش، اعتبارسنجی کارایی مدل بر روی بخشی از دیتاست انجام می شود. این داده ها باید توزیع اصولی مشابه مجموعه داده آموزشی داشته باشند، اما باید داده های متفاوتی باشند که مدل قبل آنها را ندیده است.

پایداری مدل

در نهایت، پس از آموزش و اعتبارسنجی عملکرد مدل، باید وزن های مدل به درستی ذخیره شوند و مدل به سمت تولید سوق داده شود. این بدان معنی است که فرایندی تنظیم شوند که کاربران جدید بتوانند به راحتی از مدل از پیش آموزش دیده، برای پیش بینی استفاده کنند و نیاز به آموزش مجدد مدل نباشد.

جمع‌بندی

در این فصل ما ابتدا مفهوم کلی یادگیری ماشین آشنا شدیم. سپس توضیح مختصری انواع یادگیری ماشین ارائه دادیم. انواع یادگیری ماشین شامل یادگیری ناظارت شده، یادگیری ناظارت نشده، یادگیری نیمه ناظارتی و یادگیری تقویتی هستند. پس از آن نیز نمونه کاربردهایی از یادگیری ماشین ارائه کردیم تا مسوجه اهمیت این علم در صنعت امروز بشویم. در بخش های بعدی وارد جزئیات بیشتر شده و آنچه که انجام شده را بیشتر توضیح خواهیم داد.

فصل چهارم

یادگیری ماشین نظارت شده

یادگیری ماشین ناظارت شده

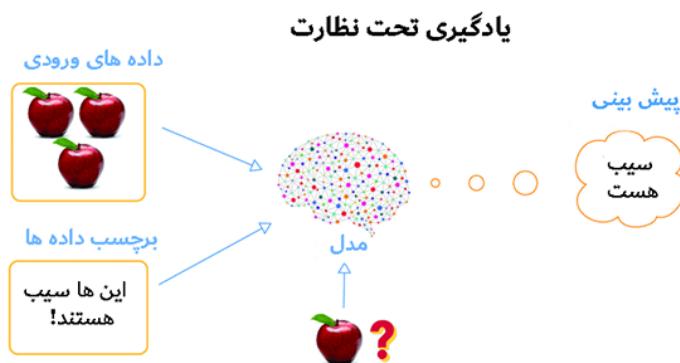
یادگیری ناظارت شده، زیرمجموعه از یادگیری ماشین است. همانطور که در فصل قبل توضیح داده شد، این فرآیند با یادگیری از روی داده‌های برچسب دار به ما در تشخیص خروجی داده‌هایی کمک می‌کند که قبلاً دیده نشده اند. در ادامه به توضیح بیشتر خواهیم پرداخت.

۱-۴ تعاریف اولیه

در یادگیری ناظارت شده کار با اضافه کردن مجموعه داده‌های شامل ویژگی‌های آموزشی و ویژگی‌های هدف آغاز می‌شود. الگوریتم‌های یادگیری ناظارت شده رابطه بین مثال‌های آموزش و متغیرهای هدف مختص آنها را به دست می‌آورد و آن رابطه یاد گرفته شده را برای دسته‌بندی ورودی‌های کاملاً جدید مورد استفاده قرار می‌دهد. برای نمایش اینکه یادگیری ناظارت شده چگونه کار می‌کند، یک مثال از پیش‌بینی نمرات دانش‌آموزان برپایه ساعت‌های مطالعه آنها ارائه می‌شود. از منظر ریاضی:

$$Y = F(x) + C$$

که در آن $F(x)$ رابطه بین نمرات و تعداد ساعاتی است که دانش‌آموزان به منظور آماده شدن برای امتحانات به مطالعه می‌پردازن. x ورودی (تعداد ساعاتی که دانش‌آموز خود را آماده می‌کند)، Y خروجی (نمراتی که دانش‌آموزان در آزمون کسب کرده‌اند) و C یک خطای تصادفی است. هدف نهایی یادگیری ناظارت شده پیش‌بینی Y با حداقل دقت برای ورودی جدید داده شده x است. شکل زیر تعریف فوق را بهتر مطرح می‌کند.



شکل ۲- یادگیری ناظارت شده

۲-۴ انواع مسائل یادگیری ماشین ناظارت شده

چندین راه برای پیاده‌سازی یادگیری ناظارت شده وجود دارد. برخی از متدالوں ترین رویکردها در ادامه مورد بررسی قرار می‌گیرند. برپایه مجموعه داده موجود، مساله یادگیری ماشین در دو نوع "دسته‌بندی" و "رگرسیون" قرار می‌گیرد. زمانی که متغیر خروجی یک مقدار پیوسته باشد مانند حقوق، وزن یا قیمت فروش مسئله در دسته رگرسیون و زمانی که متغیر خروجی

در یک دسته^۱ باشد مسئله جزو گروه مسائل دسته‌بندی (طبقه‌بندی) قرار می‌گیرد. در ادامه به توضیح بیشتر الگوریتم‌های موجود در هر کدام خواهیم پرداخت.

۱-۲-۴ رگرسیون

رگرسیون از ازارهای آماری و یادگیری ماشین مهم و پرکاربرد است. هدف اصلی وظایف مبتنی بر رگرسیون پیش‌بینی برچسب‌های خروجی یا پاسخ‌هایی است که مقادیر عددی پیوسته برای داده‌های ورودی داده شده است. خروجی بر اساس آنچه مدل در مرحله آموزش آموخته است، خواهد بود. الگوریتم‌های رگرسیون به دو دسته کلی خطی و غیر خطی تقسیم می‌شوند که هر دو در ادامه بیشتر توضیح داده شده اند.

۱-۲-۴ رگرسیون خطی ساده و چندگانه

رگرسیون خطی یکی از روش‌های تحلیل رگرسیون است. رگرسیون خطی نوعیتابع پیش‌بینی کننده خطی است که در آن متغیر وابسته، متغیری که قرار است پیش‌بینی شود به صورت ترکیبی خطی از متغیرهای مستقل پیش‌بینی می‌شود، بدین معنی که هر کدام از متغیرهای مستقل در ضریبی که در فرایند تخمین برای آن متغیر به دست آمده ضرب می‌شود؛ جواب نهایی مجموع حاصل ضرب‌ها به علاوه یک مقدار ثابت خواهد بود که آن هم در فرایند تخمین به دست آمده است. فرق رگرسیون خطی با سایر مدل‌های رگرسیون در این است که در این مدل رابطه بین متغیرهای مستقل و متغیر وابسته یک رابطه خطی فرض می‌شود. در رگرسیون خطی ساده که تنها یک متغیر مستقل وجود دارد، پیش‌بینی متغیر وابسته شکل یک خط مستقیم به خود می‌گیرد؛ در رگرسیون خطی با دو متغیر شکل پیش‌بینی یک صفحه خواهد بود، و در رگرسیون خطی با بیش از دو متغیر مستقل پیش‌بینی متغیر وابسته به صورت یک آبرصفحه خواهد بود. رگرسیون‌های خطی به دسته زیر تقسیم می‌شوند

(۱) رگرسیون خطی ساده: زمانی به یک الگوریتم رگرسیون خطی ساده گفته می‌شود که برای پیش‌بینی مقدار متغیر عددی وابسته از یک متغیر مستقل واحد استفاده شود.

(۲) رگرسیون خطی چندگانه: اگر برای پیش‌بینی مقدار متغیر وابسته عددی بیش از یک متغیر مستقل استفاده شود، به این مدل، رگرسیون خطی چندگانه گفته می‌شود.

معادله کلی رگرسیون به صورت زیر است:

$$y = b_0 + b_1x + \epsilon$$

¹category
² Hyperplane

در معادله بالا این رابطه، معادله یک خط است که جمله خطای همان اپسیلون به آن اضافه شده است. پارامترهای این مدل خطی عرض از مبدا و شیب خط اند. شیب خط در حالت رگرسیون خطی ساده، نشان می‌دهد که میزان حساسیت متغیر وابسته به متغیر مستقل چقدر است. به این معنی که با افزایش یک واحد به مقدار متغیر مستقل چه میزان متغیر وابسته تغییر خواهد کرد. عرض از مبدا نیز بیانگیر مقداری از متغیر وابسته است که به ازای مقدار متغیر مستقل برابر با صفر محاسبه می‌شود. به شکل دیگر می‌توان مقدار ثابت یا عرض از مبدا را مقدار توسط متغیر وابسته به ازای حذف متغیر مستقل در نظر گرفت.

معادله رگرسیون خطی چندگانه نیز به صورت زیر است:

$$y = b_0 + b_1 x_1 + \dots + b_n x_n + \epsilon$$

در رگرسیون خطی سعی می‌شود به کمک معادله خطی که توسط روش رگرسیون معرفی می‌شود، برآورد مقدار متغیر وابسته به ازای مقدارهای مختلف متغیر مستقل توسط خط رگرسیون بدست آید. به منظور برآورد پارامترهای مناسب برای مدل، کوشش می‌شود براساس داده‌های موجود، مدلی انتخاب شود که کمترین خطای ساده را داشته باشد. روش‌های مختلفی برای تعریف خطای ساده را در مدل رگرسیون خطی معرفی کردند. میانگین مربع خطای ساده به کار می‌رود، کمینه کردن مربع خطای ساده باشد.

به منظور بدست آوردن بهترین خط میان نقاط موجود در داده‌ها، می‌توانیم از معیاری به نام مجموع خطای مربع استفاده کنیم تا با کاهش خطای ساده توانیم این را بیابیم. منظور از خطای تفاوت میان مقدار واقعی و مقدار پیش‌بینی شده است. تابع هزینه‌های مختلفی برای محاسبه خطای مربع دارد که ما مجموع خطای مربع را به صورت زیر تعریف می‌کنیم. این تابع به ما کمک می‌کند تا بهترین مقادیر ممکن برای ضرایب را بدست آوریم.

$$SSE = \sum_{i=1}^n (y_i - \bar{y})^2$$

الگوریتم گرادیان نزولی

مفهوم مهم بعدی موردنیاز برای درک رگرسیون خطی گرادیان نزولی است. گرادیان نزولی یک روش به روزرسانی پارامترها برای کاهش تابع زیان است. همان‌طور که قبلاً هم اشاره کردیم، ما با مقادیر رندوم شروع می‌کنیم و سپس این مقادیر را به طور مکرر تغییر می‌دهیم تا خطای کاهش یابد. گرادیان نزولی به ما در نحوه تغییر مقدار این پارامترها کمک می‌کند. برای درک بهتر، بیایید گودالی را به شکل یو انگلیسی تصور کنیم که در بالاترین نقطه‌ی گودال ایستاده‌ایم و هدف ما رسیدن به پایین گودال است، اما یک مشکل وجود دارد، آن‌هم این است که فقط می‌توانید تعداد مشخصی گام جداگانه برای رسیدن به پایین گودال برداریم. اگر تصمیم بگیریم گام به گام جلو برویم، درنهایت به انتهای گودال می‌رسیم، اما مدت زمان بیشتری طول می‌کشد.

¹ Sum of squared error

اگر هر بار قدم‌های بلندتری برداریم، زودتر به آن می‌رسیم، اما این احتمال وجود دارد که از انتهای گودال فراتر برویم و از آن رد شویم. در الگوریتم گرادیان نزولی، تعداد گام‌هایی که بر می‌داریم برای رسیدن به خطای حداقل، نرخ یادگیری نامیده می‌شود. این مقدار تعیین می‌کند که الگوریتم با چه سرعتی به حداقل خطأ برسد.

طرز کار با گرادیان نزولی برای به روزرسانی پارامترهای معادله

شاید برای تان این سوال پیش آمده باشد که چگونه می‌توان از گرادیان نزولی برای به روزرسانی پارامترهای معادله استفاده کرد. برای به روزرسانی پارامترها، گرادیان تابع زیان را محاسبه می‌کنیم. برای یافتن این گرادیان مشتقات جزئی را نسبت به دو پارامتر شیب و عرض از مبدا محاسبه می‌کنیم. معادله خط ساده را در نظر بگیرید.

حال باید در نظر بگیریم که تابع زیان ما خطای میانگین مربع^۱ است. این تابع یکی دیگر از معروف‌ترین توابع زیانی است که برای رگرسیون خطی استفاده می‌شود. در این تابع ما مربع اختلاف خطرا برای تمامی نقاط داده محاسبه می‌کنیم؛ سپس این مقادیر را با هم جمع و آن مقدار را بر تعداد کل نقاط داده تقسیم می‌کنیم. فرمول این تابع به این شکل است:

$$J = \frac{1}{N} \sum_{i=1}^N (y_i - (b_0 + b_1 X_i))^2 = \frac{1}{N} \sum_{i=1}^N (y_i - pred_i)^2$$

حال باید مشتق جزئی این تابع را در نسبت با پارامترهای معادله خطی به دست آوریم. طبق فرمول تابع زیان، مشتق جزئی این تابع در نسبت با دو پارامتر b_0 و b_1 به این شکل محاسبه می‌شود:

$$\begin{aligned} \frac{\partial J}{\partial \beta_0} &= \frac{2}{N} \sum_{i=1}^N (y_i - (\beta_0 + \beta_1 X_i)) \Rightarrow \frac{\partial J}{\partial \beta_0} = \frac{2}{N} \sum_{i=1}^N (y_i - pred_i) \\ \frac{\partial J}{\partial \beta_1} &= \frac{2}{N} \sum_{i=1}^N (y_i - (\beta_0 + \beta_1 X_i)) \cdot X_i \Rightarrow \frac{\partial J}{\partial \beta_1} = \frac{2}{N} \sum_{i=1}^N (y_i - pred_i) \cdot X_i \end{aligned}$$

رگرسیون لاسو: بیش‌برازش و کم‌برازش، از مسائلی است که ممکن است که در رگرسیون چندگانه رخ دهد. یکی از راه‌های جلوگیری از این مشکلات، قاعده‌سازی است. به این معنی که مدل رگرسیونی را با توجه به تعداد پارامترهای آن جریمه کرد تا تعداد آن‌ها به یک مقدار بهینه برسد. به این ترتیب پیچیدگی مدل کاهش یافته، بدون آنکه از کارایی آن کاسته شود.

قاعده‌سازی بخصوص در موارد زیر کارساز است:

- تعداد زیاد متغیرهای توصیفی
- زیاد بودن تعداد متغیرها نسبت به تعداد مشاهدات

^۱ Mean squared error

• همخطی یا همخطی چندگانه در بین متغیرهای توصیفی

اصصلاح لاسو مخفف عبارت عملگر گزینش و انقباض کمترین قدر مطلق^۱ است. در این مدل، نحوه قاعده‌سازی براساس تابع زیان قدر مطلق انجام می‌شود. تابع هدف در رگرسیون لاسو به صورت زیر نوشته می‌شود:

$$\frac{1}{N} \sum_{i=1}^N (y_i - \sum_{j=1}^p X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^p |\beta_j|$$

در این رابطه لامبدا، پارامتر تنظیم کننده است، به این معنی که اگر مقدارش برابر با صفر باشد، مدل به رگرسیون عادی تبدیل شده و همه متغیرها در آن حضور خواهند داشت و اگر مقدار آن افزایش یابد تعداد متغیرهای مستقل در مدل کاهش خواهد یافت. بنابراین با انتخاب بی نهایت برای لامبدا عملکرد متغیری در مدل وجود ندارد. تعیین‌مقدار برای این پارامتر معمولاً توسط روش اعتبارسنج متقابل انجام می‌شود.

ممکن است به جای استفاده از پارامتر λ ، شرطی که باعث کنترل تعداد متغیرهای مستقل می‌شود را به صورت زیر نوشت که در آن t پارامتر تنظیم کننده مدل است. ولی به هر حال باید توجه داشت که تعداد پارامترها براساس مجموع قدر مطلق ضرایب کنترل می‌شوند.

$$\sum_{j=1}^p |\beta_j| < t$$

پیچیدگی مدل‌های پارامتری با تعداد پارامترهای مدل و مقادیر آن‌ها سنجیده می‌شود. هرچه این پیچیدگی بیشتر باشد خطر بیش‌برازش^۲ برای مدل بیشتر است. پدیده بیش‌برازش زمانی رخ می‌دهد که مدل به جای یادگیری الگوهای موجود در داده، خود داده را به خاطر می‌سپارد. در این حالت، مدل برای آن مجموعه داده به خصوص خوب عمل می‌کند اما برای داده‌های مشابه دیگر عملکرد خوبی ندارد، که یعنی عمل یادگیری به خوبی انجام نشده است. برای جلوگیری از بیش‌برازش در مدل‌های خطی مانند رگرسیون خطی، یک «جریمه» (به تابع هزینه اضافه می‌شود تا از افزایش زیاد پارامترها جلوگیری شود. به این کار تنظیم مدل گفته می‌شود. در روش لاسو ضریبی از نُرم $L1$ به تابع هزینه اضافه می‌شود.

$$S(\beta) = \frac{1}{N} \sum_{i=1}^N |y_i - \sum_{j=1}^p X_{ij}\beta_j|^2 = \frac{1}{N} \| y - X\beta \|_2^2$$

رگرسیون ستیغی:

در مباحث مربوط به رگرسیون چندگانه، تعیین تعداد متغیرهای مستقلی که باید در مدل به کار گرفته شوند، یک مشکل محسوب می‌شود. با افزایش تعداد متغیرها مشکل هم خطی^۳ و بیش‌برازش ظاهر شده و با کاهش آن‌ها، واریانس مدل افزایش خواهد یافت. یکی دیگر از روش‌های غلبه بر این مسائل در رگرسیون چندگانه، استفاده از مدل «رگرسیون ستیغی» است. از آنجایی که در زمانی که متغیرهای مدل، زیاد و یا همخطی چندگانه وجود داشته باشد، واریانس برآوردگرها به شکل قله ستیغ (در می‌آید، از همین روی، این روش رگرسیونی که بر این مشکل غلبه می‌کند، رگرسیون ستیغی نام‌گذاری شده است).

¹ Least Absolute Shrinkage and Selection Operator

وجود اریبی^۳ یا واریانس^۴ زیاد برای برآوردهای مدل رگرسیونی از معایب آن مدل محسوب می‌شود که هر یک ممکن است به بیش‌برازش یا کم‌برازش منجر شوند.

برای برآورد کردن پارامترهای رگرسیونی در روش سنتیگی، قیدی روی پارامترها گذاشته شده است. این قید به صورت زیر نوشته می‌شود.

$$\beta_0^2 + \beta_1^2 + \dots + \beta_p^2 \leq C^2$$

این محدودیت، مشخص می‌کند که باید مجموع مربعات پارامترها از مقدار ثابت یا آستانه‌ای کمتر باشند. این شرط را به صورت ماتریسی به صورت زیر نشان می‌دهیم.

$$\|\beta\|_2^2 \leq C^2$$

حال این قید را نیز به MSE اضافه کرده و تابع هدف حاصل را ملاک برآورد پارامترهای مدل رگرسیون خطی قرار خواهیم داد. در نتیجه شیوه برآورد پارامترها به صورت زیر در خواهد آمد.(در روش سنتیگی ضریبی از نرم L2² به تابع هزینه اضافه می‌شود).

$$\operatorname{argmin} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2$$

پارامتر λ در اینجا میزان جریمه نامیده می‌شود زیرا با توجه به افزایش تعداد پارامترها، مقدار MSE را افزایش می‌دهد. با تغییر مقدار λ و برآورد پارامترهای مدل، مدلی که کمترین مقدار مربع خطای داشته باشد، به عنوان مدل مناسب در نظر گرفته می‌شود.

با افزایش میزان جریمه(λ)، مقدار پارامترها به صفر نزدیک می‌شوند ولی هرگز به صفر نمی‌رسند. به این ترتیب در روش سنتیگی برای متغیرها کم اهمیت ضریب β کوچک شده ولی هیچگاه صفر نمی‌شوند. به این شکل می‌توان متغیرهای پراهمیت را در مدل حفظ و بقیه را نادیده گرفت.

رگرسیون شبکه الاستیک!⁵ رگرسیون شبکه الاستیک، با ترکیب رگرسیون لاسو و رگرسیون سنتیگی، بر معایب آن‌ها غلبه کرده و جایگزین مطمئن آن‌ها است. به این ترتیب اگر با مدلی مواجه هستید که متغیرهای توصیفی آن با یکدیگر همبستگی دارند، بهتراست از رگرسیون شبکه الاستیک استفاده کنید. تابع هدف در رگرسیون شبکه الاستیک با در نظر گرفتن مدل رگرسیون خطی چندگانه، به صورت زیر نوشته می‌شود:

$$\operatorname{argmin} \|y - X\beta\|_2^2 + \lambda_1 \|\beta\|_2^2 + \lambda_2 \|\beta\|_1$$

۲-۱-۲-۴ رگرسیون غیرخطی

اگر رابطه بین متغیرهای مستقل و وابسته به شکل یک تابع غیرخطی نسبت به پارامترها باشد، می‌توان برآورد پارامترهای مدل را به کمک رگرسیون غیرخطی بدست آورد.

¹ Elastic net

۱-۲-۱-۴ رگرسیون کا-همسایه نزدیک^۱

الگوریتم کا-همسایه نزدیک یکی از ساده‌ترین الگوریتم‌های یادگیری ماشین با ناظر است که برای حل مسائل طبقه‌بندی و رگرسیون استفاده می‌شود. این الگوریتم همچنین به عنوان یک مدل مبتنی بر نمونه یا یک یادگیرنده تبل شناخته می‌شود؛ زیرا یک مدل داخلی ایجاد نمی‌کند و از داده‌های آموزش عملکرد متمایز را یاد نمی‌گیرد؛ فقط نمونه‌های آموزشی را حفظ می‌کند که به عنوان دانش برای مرحله پیش‌بینی استفاده می‌شود. برای مسائل رگرسیون کا-نزدیک ترین همسایه را پیدا و با محاسبه میانگین مقدار نزدیک‌ترین همسایه‌ها، مقدار مدنظر را پیش‌بینی می‌کند.

مراحل الگوریتم رگرسیون کا-نزدیک ترین همسایه:

- (۱) داده‌ها را بارگذاری می‌کنیم.
- (۲) مقدار کا را تعیین می‌کنیم که همان تعداد نزدیک‌ترین همسایه‌ها هستند.
- (۳) برای هر نمونه داده: فاصله‌ی میان نمونه داده‌ی جدید را با نمونه داده‌های موجود محاسبه می‌کنیم و فاصله و شاخص هر نمونه را به یک فهرست وارد می‌کنیم.
- (۴) کل لیست را بر اساس فاصله‌ی نمونه داده‌ها، از کمترین به بیشترین فاصله، مرتب می‌کنیم.
- (۵) «کا» تا اولین نمونه‌های فهرست مرتب شده را به عنوان کا-نزدیک ترین همسایه انتخاب می‌کنیم.
- (۶) برچسب این «کا» نمونه را بررسی می‌کنیم.
- (۷) میانگین برچسب‌های این «کا» نمونه داده، برچسب نمونه داده جدیدمان خواهد بود.

تکنیک‌های مختلفی برای نحوه محاسبه فاصله میان نمونه‌ها وجود دارد:

- فاصله‌ی اقلیدسی
- فاصله‌ی منهتن
- فاصله‌ی مینکوفسکی

یکی از تکنیک‌هایی که بسیار استفاده می‌شود فاصله‌ی اقلیدسی است.

فاصله‌ی اقلیدسی برای محاسبه فاصله‌ی میان دو نقطه در یک صفحه یا یک فضای سه‌بعدی استفاده می‌شود. فاصله‌ی اقلیدسی براساس قضیه‌ی فیثاغورس است.

انتخاب مقدار مناسب برای کا

برای انتخاب مقدار مناسب برای این پارامتر برای داده‌های خود، الگوریتم کا-همسایه نزدیک را چندین بار با مقادیر مختلف کا اجرا می‌کنیم و مقداری را انتخاب می‌کنیم که مقدار خطأ را کاهش می‌دهد و در عین حال توانایی الگوریتم را برای انجام دقیق پیش‌بینی‌ها در هنگام مواجهه با داده‌های جدید افزایش می‌دهد.

¹ K nearest neighbors

هرچه این مقدار را کاهش می‌دهیم و به یک نزدیک می‌شود، پیش‌بینی‌های ما از پایداری کمتری برخوردار می‌شود. بر عکس، با افزایش این مقدار پیش‌بینی‌های ما به دلیل رای اکثریت/میانگین‌گیری، پایدارتر می‌شوند. بنابراین، به احتمال زیاد پیش‌بینی‌های دقیق‌تری انجام می‌دهیم. البته تا یک مقدار خاصی و بعد از آن مقدار ما شاهد افزایش فزاینده خطاهای هستیم. در این مرحله است که می‌دانیم مقدار را خیلی دور کرده‌ایم.

پیاده‌سازی

```
from sklearn.neighbors import KNeighborsRegressor
X = [[0], [1], [2], [3]]
y = [0, 0, 1, 1]

knn = KNeighborsRegressor(n_neighbors=2)
knn.fit(X, y)

print("Test Prediction: ", knn.predict([[1.5]]))
```

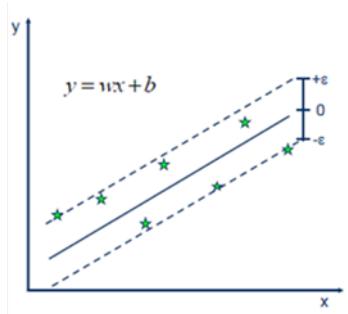
Test Prediction: [0.5]

شکل ۳- پیاده‌سازی الگوریتم کا همسایه نزدیک با زبان پایتون با یک مثال ساده

۱-۲-۲-۴ رگرسیون بردار پشتیبان^۱

در مسائل رگرسیون ما به دنبال رابطه بین ورودی و خروجی هستیم که این رابطه میتواند خطی یا غیر خطی باشد. فرض کنیم که رابطه بین ورودی و خروجی یک رابطه خطی هست و میخواهیم بهترین رابطه خطی بین ورودی و خروجی را بدست بیاوریم. در چنین حالتی رگرسیون بردار ماشین دنبال رابطه خطی هست که از وسط خروجی‌ها عبور کند، و مارجین داشته باشد که در آن خروجی همه نمونه‌ها در داخل مارجین قرار بگیرند. با چنین رویکردی میتوان بهترین رابطه خطی ممکن بین ورودی و خروجی بدست آورد.

^۱ Support vector regression



شکل ۴ - شکل خط رگرسیون بردار پشتیبان

برای چنین کاری این الگوریتم از تابع هزینه زیر استفاده می کند:

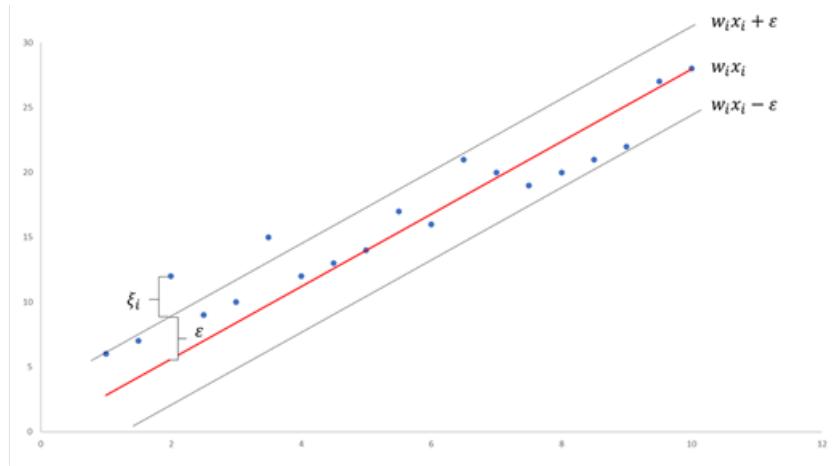
$$\min \frac{1}{2} \| w \|^2$$

$$y_i - w x_i - b \leq \epsilon$$

$$w x_i + b - y_i \leq \epsilon$$

در مسائل رگرسیون نیز همانند شکل زیر ممکن است خروجی داده‌ها یک رفتار خطی داشته باشند، ولی برخی از خروجی‌ها دارای دورافتادگی باشند، یعنی نمونه‌هایی وجود داشته باشد که خروجی آنها مقدار خیلی متفاوتی با سایر خروجی‌ها داشته باشد و از خط بین نمونه‌ها فاصله خیلی زیادی داشته باشد. اگر طبق رویکرد اول عمل کنیم و رابطه‌ی خطی پیدا کنیم که در آن مارجین طوری بددست بباید که خروجی همه نمونه‌ها حتی داده‌های دور افتاده در مارجین باشند، در این صورت رابطه خطی مناسبی برای رگرسیون بددست نخواهد آمد. چرا که داده‌های دور افتاده تاثیر منفی خود را خواهند گذاشت.

برای حل این مسئله به تابع هزینه یک متغیر مجازی اضافه می‌کنند. متغیر مجازی در مسائل رگرسیون به رگرسیون بردار پشتیبان این اجازه را میدهد که چندین نمونه خارج از مارجین قرار بگیرند. و با اینکار اثر آن‌ها را خنثی می‌کند.

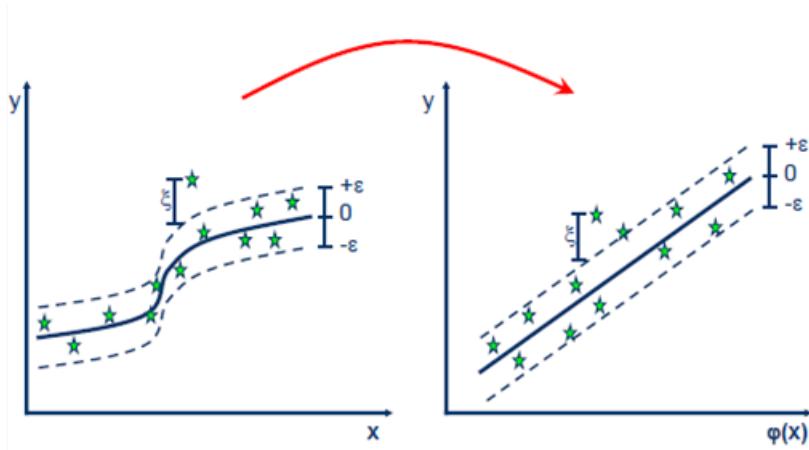


شکل ۵ - منطق شهودی رگرسیون بردار پشتیبان در فضای دوبعدی به همراه داده‌های دورافتاده

$$\min \frac{1}{2} \|w\|^2 + C \sum_{i=1}^N |\xi_i|$$

$$|y_i - w x_i - b| \leq \epsilon + |\xi_i|$$

جهت تعمیم رگرسیون بردار پشتیبان برای حل مسائل غیرخطی نیز از چنین رویکرده استفاده می‌شود. یعنی در ابتدا داده‌ها با کمک یک نگاشت غیرخطی به فضای خطی نگاشت پیدا می‌کنند، و سپس در فضای جدید رابطه خطی بین داده‌ها در فضای ویژگی و خروجی بدست می‌آید. که این رابطه خطی در فضای ویژگی معادل رابطه غیرخطی بین ورودی و خروجی در فضای اصلی است.



شکل ۶- تعمیم رگرسیون بردار پشتیبان

```

from sklearn.svm import SVR
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
import numpy as np

n_samples, n_features = 10, 5
rng = np.random.RandomState(0)
y = rng.randn(n_samples)
X = rng.randn(n_samples, n_features)

svr = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2))
svr.fit(X, y)

print("Coefficient: ", svr['svr'].dual_coef_)
print("Intercept: " , svr['svr'].intercept_)

Coefficient: [[ 1.          -0.29598093  1.          -1.          -0.70401907
   -1.          ]]
Intercept: [0.8805015]

```

شکل ۷- پیاده‌سازی رگرسیون بردار پشتیبان با زبان پایتون با یک مثال ساده

۳-۲-۱-۴ رگرسیون درخت تصمیم^۱

درخت تصمیم از جمله الگوریتم‌های یادگیری ماشین است که هم برای مسائل دسته‌بندی و هم مسائل رگرسیون مورد استفاده قرار می‌گیرد. البته، الگوریتم درخت تصمیم بیشتر در مسائل دسته‌بندی استفاده می‌شود. این الگوریتم یک نمونه را دریافت می‌کند، درخت را می‌پیماید و ویژگی‌های مهم را با یک عبارت شرطی تعیین شده، مقایسه می‌کند. اینکه آیا در شاخه فرزند سمت چپ قرار بگیرد و یا سمت راست، بستگی به نتیجه حاصل از بررسی عبارت شرطی دارد. معمولاً، یک ویژگی مهم‌تر به ریشه نزدیک‌تر است. تنها تفاوتی که رگرسیون درخت تصمیم با طبقه‌بندی درخت تصمیم دارد این میباشد که در برگ‌های درخت مقدار میانگین داده‌ها به عنوان مقدار نهایی در نظر گرفته می‌شود. با ویژگی‌ها و پارامتر‌های درخت تصمیم به صورت جامع‌تر در طبقه‌بندی با درخت تصمیم بیشتر آشنا می‌شویم.

¹ Decision tree regression



شکل ۸- نمودار درخت تصمیم یک مثال

۲-۲-۴ طبقه‌بندی (دسته‌بندی)

همانگونه که در بخش های قبلی گفته شد، یک روش یادگیری در علم شناسایی الگو یادگیری با نظارت می باشد. در این روش هر الگو یک برچسب دارد که خروجی مطلوب آن الگو می باشد. هدف این روش یادگیری تابعی است که بردارهای ویژگی ورودی را به برچسب های متناظرشان نگاشت می کند. در واقع این کار در فاز آموزش انجام می گیرد. در فاز تست الگوهایی که برچسب آنها مشخص نیست به سیستم داده می شوند و سیستم طراحی شده به کمک تابع یادگرفته شده می خود خروجی یا برچسب آنها را پیش بینی می کند. حال اگر خروجی سیستم یادگیری گسسته در نظر گرفته شود مساله طبقه بندی نام می گیرد و تابعی که ورودی را به خروجی نگاشت می کند طبقه بندی کننده امیده می شود.

اصولاً از تکنیک های طبقه بندی برای دسته بندی هر داده در مجموعه ای از داده ها و اختصاص به یکی از مجموعه های از پیش تعیین شده کلاس ها یا گروه ها استفاده می شود. روش طبقه بندی از تکنیک های ریاضی مانند درخت تصمیم، برنامه ریزی خطی، شبکه عصبی و آمار برای طبقه بندی استفاده می کند. به عبارتی طبقه بندی، فرآیند یافتن مدلی که توصیف کننده کلاس ها و مفاهیم داده است و داده ها را به گروه های مشخص تفکیک می کند. الگوریتم های طبقه بندی، قادر به یادگیری از تجربیات گذشته هستند و این یادگیری بر اساس تجربه نشان دهنده یک گام اساسی در تقلید از توانایی های مغز انسان است که براساس این توانایی مغز می تواند مسئله شناسایی یک گروه از دسته ها را انجام دهد.

شاید بتوان مسئله طبقه بندی را به چهار دسته تقسیم کرد:

- طبقه بندی دودویی

- طبقه بندی چند کلاسه

- طبقه بندی چند برچسبی

¹ Classification

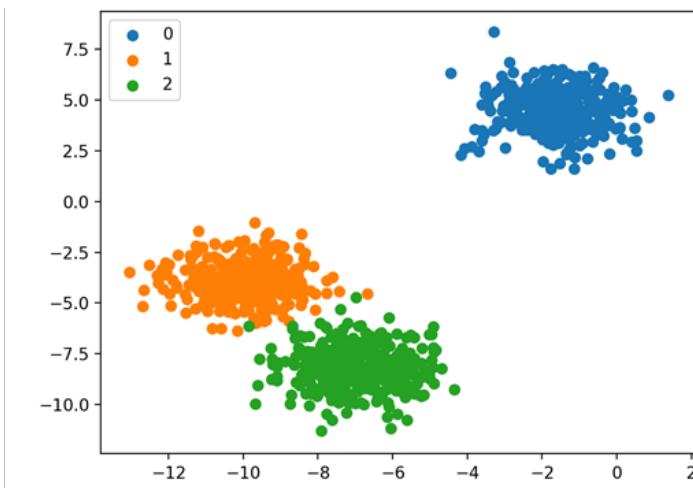
- طبقه‌بندی نامتوازن

طبقه‌بندی دودویی:

مسائل طبقه‌بندی که دارای دو برچسب کلاس هستند. مانند مسئله شناسایی ایمیل اسپم که دارای دو برچسب اسپم یا غیر اسپم است یا در آزمایشات پزشکی، مشخص می‌شود یک بیمار دارای بیماری خاصی است یا خیر، بنابراین دارای دو برچسب بیمار یا غیر بیمار هستیم. در واقع در طبقه‌بندی دودویی، همانطور که در شکل زیر می‌بینید، یک کلاس حالت نرمال و کلاس دیگر حالت غیر نرمال را نشان می‌دهد.

طبقه‌بندی چندکلاسه:

طبقه‌بندی چندکلاسه، وظایف طبقه‌بندی هستند که دارای بیش از دو برچسب کلاس هستند. به طور مثال، در شکل زیر دارای سه کلاس مختلف هستیم. مانند طبقه‌بندی چهره، طبقه‌بندی گونه‌های گیاهی و شناسایی کاراکترهای نوری. برخلاف طبقه‌بندی دودویی، نمونه‌ها متعلق به طیف وسیعی از کلاس‌های شناخته شده می‌باشند. تعداد برچسب کلاس‌ها در بعضی از مسائل ممکن است بسیار زیاد باشند. برای مثال در سیستم تشخیص چهره، مدل پیش‌بینی می‌کند عکسی به یکی از ده‌ها هزار چهره موجود در سیستم تعلق دارد یا خیر.



شکل ۹- طبقه‌بندی چندکلاسه در فضای دوبعدی

طبقه‌بندی چند برچسبی:

طبقه‌بندی چند برچسبی، وظایفی هستند که در آن برای هر نمونه دو یا چند برچسب کلاس قابل پیش‌بینی است. در مثال طبقه‌بندی عکس، زمانی که یک عکس می‌تواند شامل چند جزء در تصویر باشد، یک مدل می‌تواند به پیش‌بینی چندین

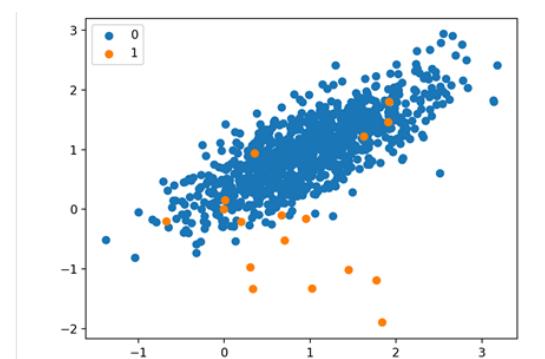
برچسب در عکس بپردازد مانند افراد، دوچرخه، سیب و غیره. در شکل زیر تفاوت بین طبقه‌بندی چندکلاسه و چند برچسبی را مشاهده می‌کنید.

Multi-Class		Multi-Label	
C = 3	Samples	Samples	
	Labels (t)	Labels (t)	
	[0 0 1]	[1 0 0]	[0 1 0]
			[1 1 1]

شکل ۱۰- تفاوت طبقه‌بندی چندکلاسه و چند برچسبی

طبقه‌بندی نامتوازن:

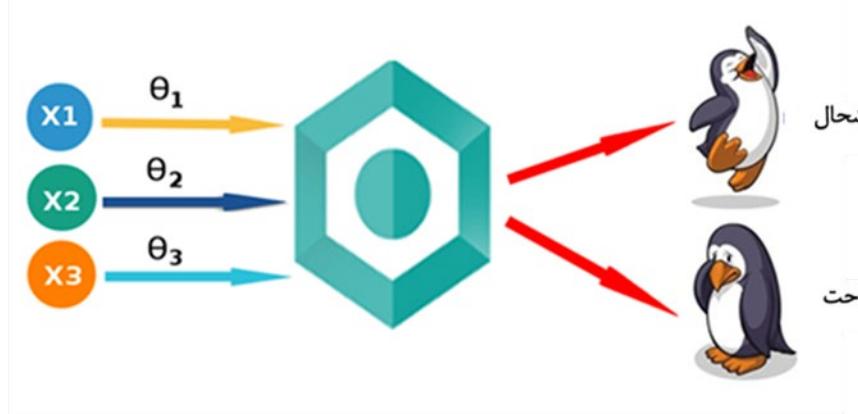
طبقه‌بندی نامتوازن، وظایف طبقه‌بندی هستند که در آن تعداد نمونه‌ها در هر کلاس به صورت نابرابر توزیع شده‌اند. معمولاً وظایف طبقه‌بندی نامتوازن، وظایف طبقه‌بندی دودویی هستند که اکثریت نمونه‌ها در مجموعه آموزشی متعلق به کلاس نرمال هستند و حداقل نمونه‌ها متعلق به کلاس غیر نرمال‌اند. همانطور که در شکل زیر می‌بینید، اکثریت نقاط به رنگ آبی و تعداد اندکی نقطه به رنگ زرد وجود دارند. مانند تشخیص تقلب، تشخیص داده پرت و تست‌های تشخیصی پزشکی. مثلاً در تست‌های تشخیص سرطان تعداد بسیار زیادی از افراد سالم و تعداد اندکی دارای بیماری سرطان هستند. این مسائل به عنوان مسائل طبقه‌بندی دودویی مدل سازی می‌شوند اما نیاز به تکنیک‌های خاصی دارد.



شکل ۱۱- طبقه‌بندی نامتوازن در فضای دوبعدی

۱-۲-۲-۴ رگرسیون لجیستیک^۱

رگرسیون لجستیک یکی از پرکاربردترین الگوریتم های حوزه‌ی یادگیری ماشین است و این الگوریتم در اوایل قرن بیستم در علوم زیستی مورد مطالعه قرار گرفت و پس از آن در بسیاری از کاربردهای علوم اجتماعی مورد استفاده واقع شد. این تکنیک یک روش یادگیری با نظارت است و داده‌ها دارای برچسب مشخص هستند و پروسه‌ی یادگیری به منظور طبقه‌بندی بر اساس این داده‌ها و برچسب‌های آن‌ها صورت می‌گیرد. رگرسیون لجستیک هنگامی استفاده می‌شود که متغیر وابسته طبقه‌ای باشد. تصویر زیر نشان دهنده‌ی سه ورودی است که مدل لجستیک رگرسیون با تخصیص وزن قرار است دو دسته خوشحال و ناراحت را پیش‌بینی کند.



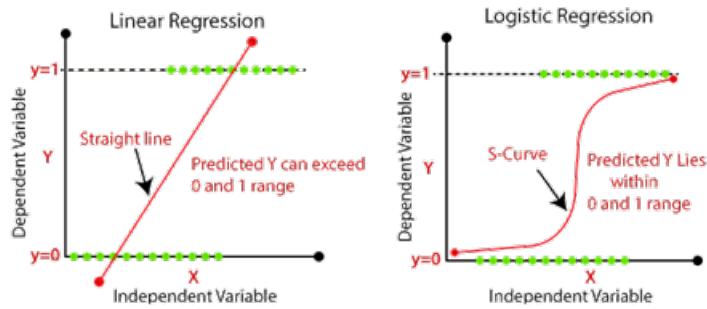
شکل ۱۲- مثال شهودی از ورودی خروجی رگرسیون لجیستیک

در رگرسیون لجستیک خروجی یک متغیر وابسته طبقه‌ای را پیش‌بینی می‌کند. بنابراین نتیجه باید یک مقدار طبقه‌ای یا گسسته باشد. می‌تواند بله یا خیر، صفر یک، درست یا نادرست و غیره باشد. اما به جای دادن مقدار دقیق صفر و یک مقادیر احتمالی بین صفر تا یک را می‌دهد. ساده‌ترین حالت یک طبقه‌بندی باینری است. ما فقط دو کلاس داریم: یک کلاس مثبت و یک کلاس منفی. معمولاً یک کلاس مثبت به وجود برخی موجودات اشاره دارد در حالی که کلاس منفی به نبود آن اشاره دارد. در این حالت، ما باید یک مقدار واحد را پیش‌بینی کنیم که آن احتمال وجود موجودیت. برای انجام این کار، خوب است که ما تابعی داشته باشیم که هر مقدار واقعی را برای تعیین در فاصله بین صفر و یک ترسیم کند.

تفاوت بین رگرسیون خطی و رگرسیون لجیستیک:

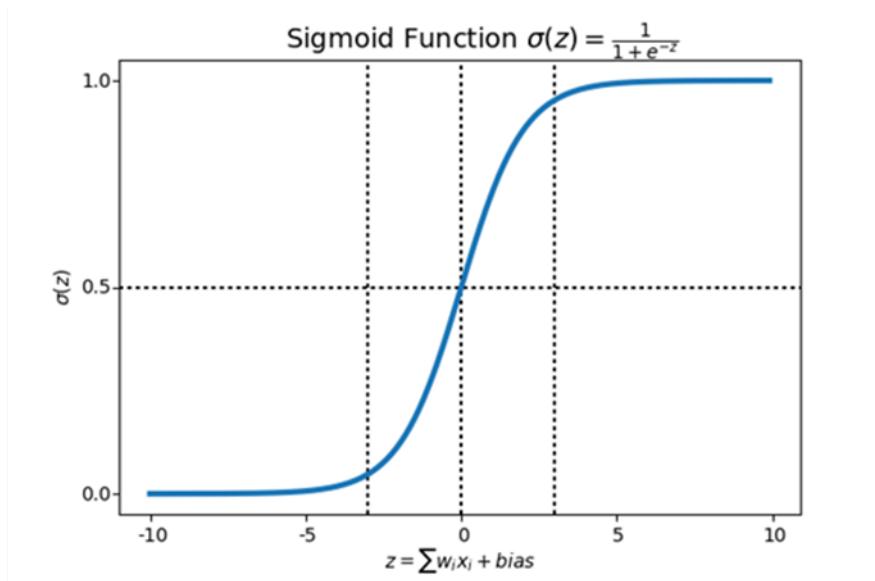
تفاوت اصلی بین آنها نحوه استفاده از آنهاست. رگرسیون خطی برای حل مشکلات رگرسیون استفاده می‌شود در حالی که رگرسیون لجستیک برای حل مشکلات طبقه‌بندی استفاده می‌شود. در شکل زیر هر دو روش به صورت نمودار مشخص است.

^۱ Logistic regression



شکل ۱۳- تفاوت رگرسیون خطی و لجیستیک

در رگرسیون خطی ما باید به بهترین معادله‌ی خطی می‌رسیدیم تا داده‌ها را به درستی نمایش دهد. این را در نظر بگیریم که در رگرسیون خطی معادله‌ی خطی می‌تواند هر مقداری داشته باشد، در حالی که در طبقه‌بندی و رگرسیون لجستیک مقداری که در خروجی داریم احتمال تعلق داده به یکی از دو کلاس صفر یا یک است؛ بنابراین مقداری باید میان صفر و یک باشد که این موضوع با معادله‌ی خطی امکان‌پذیر نیست. پس ما به معادله‌ی دیگری احتیاج داریم که آن تابع سیگموید یا همان لجیستیک است.



شکل ۱۴- نمودار تابع سیگموید

در رگرسیون لجستیک معادله‌ی ما به این شکل خواهد بود:

$$h_{\theta}(X) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 X)}}$$

مرز تصمیم‌گیری^۱

هنگامی که ورودی‌ها از یک تابع پیش‌بینی عبور می‌دهیم و در خروجی احتمالی میان صفر تا یک را داریم، همچنین انتظار داریم طبقه‌بندی کننده‌ی ما براساس احتمال، مجموعه‌ای از کلاس‌ها را به ما بدهد. یک آستانه در نظر می‌گیریم که هر داده‌ای بالای آن باشد، متعلق به کلاس یک و هر داده‌ای زیر آن باشد متعلق به کلاس صفر در نظر گرفته شود. همانطور که در نمودار بالا نشان داده شده است ما آستانه را 0.5 ، انتخاب کرده‌ایم. در واقع این مرز به ما کمک می‌کند دو کلاس را از هم متمایز کنیم. این مرز تصمیم‌گیری را تابع لجیستیک مشخص می‌کند. حال لازم است بدانیم که پارامترهای تابع لجیستیک یا همان سیگموید چطور به دست می‌آید.

تابع هزینه

در رگرسیون لجستیک نیز، مانند رگرسیون خطی، ابتدا ما یک تابع با پارامترهای رندوم داریم که درنهایت با استفاده از تابع هزینه این پارامترها بهینه می‌شود تا در خروجی بهترین نتیجه را داشته باشیم؛ به عبارت دیگر، داده‌ها به درستی طبقه‌بندی شوند. تابع هزینه میزان خطای موجود را میان کلاسی که پیش‌بینی شده است و کلاسی که واقعاً داده به آن تعلق دارد مشخص می‌کند. در رگرسیون لجستیک این خطای تابع هزینه لگاریتمی به این صورت مشخص می‌کند:

$$Cost(h_\theta(X).y) = \begin{cases} -\log(h_\theta(X)). & \text{if } y = 1 \\ -\log(1 - h_\theta(X)). & \text{if } y = 0 \end{cases}$$

شکل خلاصه‌شده‌ی تابع هزینه‌ای فوق به این صورت خواهد بود:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y_i \log(h_\theta(X_i)) + (1 - y_i) \log(1 - h_\theta(X_i))]$$

بعد از اینکه خطای تابع هزینه را بدست آوردمی‌باید پارامترهای را به گونه‌ای تعیین کنیم که مقدار آن به حداقل برسد که از روش گرادیان نزولی استفاده می‌کنیم:

$$\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)$$

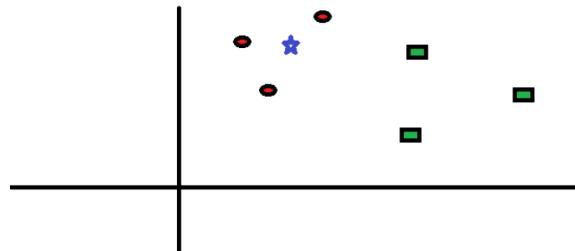
که در آن مشتق زنجیره‌ای هر پارامتر را نسبت به خطای بدست آمده می‌گیریم و درنهایت مقدار مشتص را از مقدار پارامتر قبلی کم می‌کنیم تا آن را به روزرسانی کنیم. این کار چندین بار تکرار می‌شود تا زمانی که پارامترهای ما بهینه شوند و خروجی مدل ما نتیجه چشمگیری داشته باشد.

۴-۲-۲-۲ طبقه‌بندی کا-نژدیک‌ترین همسایه

این الگوریتم از الگوریتم‌های یادگیری ماشین تحت نظرارت است که هم در مسائل طبقه‌بندی و هم در مسائل رگرسیون مورد استفاده قرار می‌گیرد. اگرچه، غالباً در مسائل طبقه‌بندی، در صنعت از آن استفاده می‌شود.

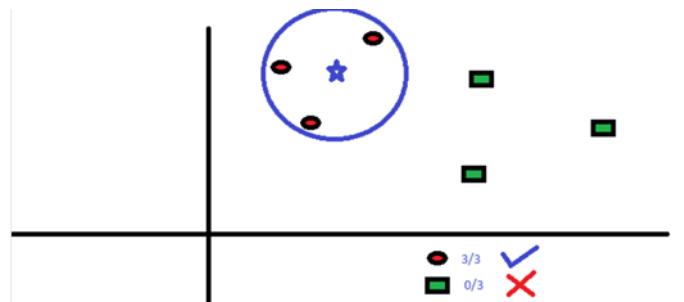
¹ Decision boundary

برای درک بهتر شیوه کار این الگوریتم، عملکرد آن را با یک مثال ساده مورد بررسی قرار می‌دهیم. در شکل زیر نحوه توزیع داده‌های قرمز و مربع‌های سبز دیده می‌شود.



شکل ۱۵- توزیع داده‌ها روی نمودار

فرض کنید قصد پیدا کردن کلاسی که ستاره آبی متعلق به آن است را داریم (یعنی دایره‌های قرمز). در این مثال در کل دو کلاس دایره‌های قرمز و مربع‌های سبز وجود دارند و ستاره آبی بر اساس منطق کلاسیک می‌تواند تنها متعلق به یکی از این دو کلاس باشد. کی در الگوریتم کی-نzdیک ترین همسایه، تعداد نزدیک‌ترین همسایه‌هایی است که براساس آن‌ها رای‌گیری درباره وضعیت تعلق یک نمونه داده به کلاس‌های موجود انجام می‌شود. فرض کنید این عدد را برابر سه در نظر بگیریم. یک دایره آبی دور سه تا از نزدیک‌ترین همسایه‌های ستاره آبی (شکل ۲) ترسیم شده است.



شکل ۱۶- تعیین کلاس نمونه جدید

سه نقطه نزدیک‌تر به ستاره آبی، همه قرمز هستند. با میزان اطمینان خوبی می‌توان گفت که ستاره آبی به کلاس دایره‌های قرمز تعلق دارد. چون هر سه نزدیک‌ترین همسایه ستاره آبی، دایره‌های قرمز هستند. فاصله مدنظر را می‌توان با کمک هر یک از روش‌های اقلیدسی، فاصله همینگ، یا فاصله منهتن محاسبه کرد که متداول‌ترین روش، روش اقلیدسی است.

پیاده‌سازی

برای پیاده‌سازی این الگوریتم از کتابخانه سایکیت‌لرن استفاده می‌کنیم که کمک بسیاری در پیاده‌سازی الگوریتم‌های مختلف ماشین‌لرنینگ به ما می‌کند. برای پیاده‌سازی این الگوریتم مشابه آنچه در پیاده‌سازی الگوریتم رگرسیون لاجیستیک هم دیدیم، مراحل اولیه شامل اضافه کردن کتابخانه‌ها، خواندن داده از روی مسیر، مشخص کردن ماتریس ویژگی و ماتریس پاسخ،

و تقسیم کردن داده به دو قسمت آموزشی و تست، و در صورت نیاز مقیاس‌بندی ویژگی‌ها را انجام می‌دهیم. مرحله‌ی بعدی آموزش مدل است که به صورت زیر انجام می‌دهیم.

```
from sklearn.neighbors import KNeighborsClassifier  
knn = KNeighborsClassifier()  
  
knn.fit(scaled_X_train, y_train)
```

شكل ۱۷-آموزش مدل کا - همسایه نزدیک

در مرحله‌ی بعدی هم، مدل را با سنجه‌ی مد نظر می‌سنجیم.

```
y_pred = knn.predict(scaled_X_test)  
  
from sklearn.metrics import confusion_matrix, classification_report  
  
print(classification_report(y_test, y_pred))
```

شكل ۱۸-سنجیدن مدل کا - همسایه نزدیک

برای مثال در مورد مجموعه داده بیماری قلبی، نتیجه‌ی پیاده‌سازی فوق، به صورت زیر بود.

	precision	recall	f1-score	support
0.0	0.92	0.97	0.94	68908
1.0	0.37	0.16	0.22	7196
accuracy			0.90	76104
macro avg	0.64	0.56	0.58	76104
weighted avg	0.87	0.90	0.88	76104

شكل ۱۹-خروجی سنجش مدل کا - همسایه نزدیک

۴-۲-۳-۱ ساده بیز

در حوزه‌ی یادگیری ماشین، تکنیک و روش بیز ساده با بکارگیری قضیه بیز و فرض استقلال بین متغیرها به عنوان عضوی از خانواده دسته‌بندی بر مبنای احتمال قرار می‌گیرد. در سال‌های ۱۹۶۰ تحقیق و بررسی‌های زیادی پیرامون بیز ساده به خصوص در زمینه بازیابی متن صورت گرفت و حتی امروز هم به عنوان ابزاری برای دسته‌بندی متن برای حل مسائلی مانند تشخیص هرزنامه‌ها به کار می‌رود. معمولاً این کار به کمک برآورد تابع احتمال و از طریق فراوانی یا فراوانی نسبی کلمات در اسناد متنی صورت می‌گیرد.

¹ Naïve bayes

بیز ساده از الگوریتم‌های با نظارت یادگیری ماشین است که می‌توانیم روی داده‌های خود اعمال کنیم. همانطور که از نامش پیداست، این الگوریتم فرض می‌کند همهٔ متغیرهای مجموعه داده ساده هستند، یعنی با یکدیگر ارتباط ندارند. این الگوریتم از نظریه بیز استفاده می‌کند که فرمول آن به این شکل است:

$$P(c|x) = \frac{P(x|c) \times P(c)}{p(x)}$$

همانطور که دیده می‌شود رابطه بالا همان قضیه بیز است. به عنوان یادآوری قضیه بیز را براساس احتمالات پیشامدهای پیشین، پسین، درستنمایی و شواهد در رابطه زیر بازنویسی می‌کنیم.

$$posterior = \frac{prior \times likelihood}{evidence}$$

C نشان دهنده کلاس مدنظر و ایکس نشان دهنده ویژگی هاست که هر یک به طور جداگانه باید محاسبه شوند.

فرض کنید (x_1, \dots, x_n) برداری از n ویژگی را بیان کند که به صورت متغیرهای مستقل هستند. به این ترتیب می‌توان احتمال رخداد C_k یعنی $P(C_k|x_1, \dots, x_n)$ را به عنوان یکی از حالت‌های کلاس رخدادهای مختلف به ازاء k های مختلف به شکل زیر نمایش داد:

$$P(C_k|X) = \frac{P(X|C_k) \times P(C_k)}{p(X)}$$

برای محاسبه احتمال $P(C_k|x_1, \dots, x_n)$ کافی است از «احتمال توام» کمک بگیریم و به کمک احتمال شرطی با توجه به استقلال متغیرها، آن را ساده کنیم.

$$\begin{aligned} P(C_k, x_1, \dots, x_n) &= P(x_1, \dots, x_n, C_k) \\ &= P(x_1|x_2, \dots, x_n, C_k)P(x_2, \dots, x_n, C_k) \\ &= P(x_1|x_2, \dots, x_n, C_k)P(x_2|x_3, \dots, x_n, C_k)P(x_3, \dots, x_n, C_k) \\ &= \dots = P(x_1|x_2, \dots, x_n, C_k)P(x_2|x_3, \dots, x_n, C_k) \dots P(x_n|C_k)P(C_k) \end{aligned}$$

با فرض استقلال مولفه‌ها یا ویژگی‌های x_i ها از یکدیگر می‌توان احتمالات را به شکل ساده‌تری نوشت. کافی است رابطه زیر را در نظر بگیریم.

$$P(x_i|x_{i+1}, \dots, x_n, C_k) \approx p(x_i|C_k)$$

به این ترتیب احتمال توام را به صورت حاصل ضرب احتمال شرطی می‌توان نوشت.

$$P(C_k|x_1, \dots, x_n) = \frac{P(C_k) \prod_{i=1}^n p(x_i|C_k)}{P(x_1, \dots, x_n)}$$

به کمک قواعد تصمیم، دسته‌بندی بیز را ایجاد و کامل می‌کنیم. یکی از اساسی‌ترین قواعد تصمیم، انتخاب فرضیه محتمل‌تر است. به این ترتیب از بین تصمیمات مختلف، آن کاری را انجام می‌دهیم که براساس شوه‌د جمع‌آوری شده، بیشترین احتمال رخداد را دارد. این قاعده را حداکثر پسین می‌نامند. این الگوریتم بیزساده را می‌توان به صورت تابعی از تصمیمات در نظر گرفت که بوسیله تابع ایگرگ تخمین زده می‌شود. حداکثرسازی این تابع را به صورت زیر نشان می‌دهیم.

$$\arg \max_{k \in 1, \dots, K} P(C_k) \prod_{i=1}^n p(x_i | C_k)$$

در نتیجه با توجه به توزیع‌های مختلفی که ممکن است نمونه تصادفی داشته باشد (نمونه از آن جامعه آمده باشد (یعنی $P(x_i | C_k)$ می‌توان پارامترها را محاسبه و یا برآورد کرد.

بیز ساده گاوی

اگر مشاهدات و داده‌ها از نوع پیوسته باشند، از مدل احتمالی با توزیع گاوی یا نرمال برای متغیرهای مربوط به شواهد می‌توانید استفاده کنید. در این حالت هر دسته یا گروه دارای توزیع گاوی است. به این ترتیب اگر k دسته یا کلاس داشته باشیم می‌توانیم برای هر دسته میانگین و واریانس را محاسبه کرده و پارامترهای توزیع نرمال را برای آن‌ها برآورد کنیم. فرض کنید که μ_k میانگین و σ_k^2 واریانس دسته C_k است. همچنین v را مشاهدات حاصل از متغیرهای تصادفی X در نظر بگیرید. از آنجایی که توزیع X در هر توزیع گاوی نرمال فرض شده است، خواهیم داشت:

$$p(x = v | C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

مدل‌های احتمالی با توزیع چندجمله‌ای و برنولی هم می‌توانیم برای بدست آوردن $P(x_i | C_k)$ استفاده کنیم.

پیاده سازی

مجددا برای پیاده سازی بعد از انجام کارهای اولیه که در الگوریتم‌های بالا هم ذکر کردیم، مراحل آموزش و سنجش مدل را انجام می‌دهیم. مراحل آموزش و سنجش مدل به صورت زیر خواهد بود.

```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb.fit(X_train, y_train)
```

شکل ۲۰- آموزش مدل بیز ساده

```
def report(model):
    preds = model.predict(X_test)
    print(classification_report(y_test, preds))
    plot_confusion_matrix(model, X_test, y_test)
```

شکل ۲۱- سنجش مدل بیز ساده

	precision	recall	f1-score	support
0.0	0.93	0.85	0.89	68908
1.0	0.21	0.39	0.27	7196
accuracy			0.81	76104
macro avg	0.57	0.62	0.58	76104
weighted avg	0.86	0.81	0.83	76104

شکل ۲۲ - خروجی سنجش مدل بیز ساده

۴-۲-۴ درخت تصمیم‌گیری

درخت تصمیم، روشی در یادگیری ماشین برای ساختاربندی الگوریتم است. یک الگوریتم درخت تصمیم برای تقسیم ویژگی‌های مجموعه داده از طریق تابع هزینه مورد استفاده قرار می‌گیرد. این الگوریتم قبل از انجام بهینه سازی و حذف شاخه‌های اضافه، به گونه‌ای رشد می‌کند که دارای ویژگی‌های نامرتب با مسئله است؛ به هین دلیل عملیات هرس کردن برای حذف این شاخه‌های اضافه در آن انجام می‌شود. در الگوریتم درخت تصمیم، پارامترهایی از جمله عمق درخت تصمیم را نیز می‌توان تنظیم کرد تا از بیش برآش^۱ یا پیچیدگی بیش از حد درخت تا جای امکان جلوگیری شود. انواع بسیاری از درخت‌های تصمیم در یادگیری ماشین برای مسئله‌های دسته‌بندی و همچنین در مسائل رگرسیون مورد استفاده قرار می‌گیرند.

بنابراین درخت تصمیم، مجموعه داده را به زیرمجموعه‌های کوچکتر تجزیه می‌کند و یک درخت تصمیم مرتبط به صورت تدریجی توسعه می‌یابد. نتیجه نهایی یک درخت با گره‌های تصمیم‌گیری و گره‌های برگ است. یگ گره تصمیم، دارای دو یا چند شاخه است. گره برگ، یک طبقه بندی یا تصمیم را نشان می‌دهد. بالاترین گره تصمیم‌گیری در یک درخت که مطابق با بهترین پیش‌بینی‌کننده است، گره ریشه می‌باشد.

در زمان پیاده‌سازی درخت تصمیم، یکی از مهم‌ترین و اساسی‌ترین مسائلی که پیش می‌آید این است که بهترین ویژگی^۲ برای گره ریشه و گره‌های فرعی دیگر چگونه باید انتخاب شود؟ بنابراین برای حل پنین مسائلی، روشی وجود دارد که به آن، معیار یا سنجش انتخاب ویژگی آگفته می‌شود. با این روش می‌توان به راحتی بهترین ویژگی را برای گره ریشه و دیگر گره‌های درخت انتخاب کرد. روش سنجش انتخاب ویژگی دارای دو رویکرد رایج به نام‌های زیر است:

- به دست آوردن اطلاعات^۳
- شاخص جینی^۴

در ادامه به بررسی هر کدام از این دومورد می‌پردازیم.

¹ overfitting

² Attribute selection measure

³ Information gain

⁴ Gini index

به دست آوردن اطلاعات

به دست آوردن اطلاعات به وسیله سنجش تغییرات آنتروپی پس از تقسیم بندی یک مجموعه داده بر اساس ویژگی‌ها انجام می‌شود. در این روش محاسبه می‌شود که یک ویژگی چه مقدار اطلاعات درباره یک کلاس می‌دهد. طبق مقادیر اطلاعات به دست آمده، گره‌ها تقسیم می‌شوند و درخت تصمیم ساخته خواهد شد. الگوریتم درخت تصمیم همیشه تا حد امکان سعی خود را در به حداکثر رساندن اطلاعات به دست آمده می‌کند و سپس، ابتدا آن گره ویژگی را تقسیم می‌کند که بیشترین اطلاعات را دارد. رابطه این روش، یعنی فرمول به دست آوردن اطلاعات در ادامه آورده شده است:

$$InformationGain = Entropy(S) - [(WeightedAvg) * Entropy(eachfeature)]$$

آنتروپی معیاری برای اندازه‌گیری ناچالصی در یک ویژگی مشخص است. همچنین تصادفی بودن داده‌ها را نیز مشخص می‌کند. این معیار با استفاده از رابطه زیر برای مثال فوق محاسبه می‌شود.

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

که در رابطه فوق، اس تعداد همه نمونه‌ها را در مسئله نشان می‌دهد و پی نشان دهد احتمال است.

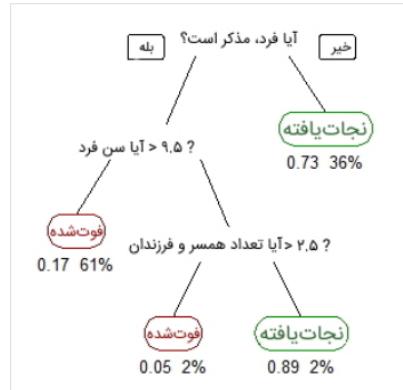
شاخص جینی

شاخص جینی، معیاری از ناچالصی به خلوص ویژگی‌ها است که هنگام ایجاد درخت تصمیم در الگوریتم کارت^۱ یا همان درخت دسته بندی و رگرسیون استفاده می‌شود. ویژگی با میزان شاخص جینی پایین در مقایسه با شاخص جینی بالا در درخت تصمیم ترجیح داده و انتخاب خواهد شد. از این شاخص فقط برای ایجاد دسته و گروه‌های باینری استفاده می‌شود. شاخص جینی به وسیله رابطه زیر محاسبه می‌شود:

$$Gini\ Index = 1 - \sum_j P_j^2$$

به عنوان مثالی برای توصیف نحوه کار الگوریتم درخت تصمیم‌گیری، مثال ساده‌ای در نظر گرفته شده است. در این مثال از مجموعه داده مسافران کشتی تایتانیک برای پیش‌بینی اینکه آیا فردی زنده می‌ماند یا خیر استفاده می‌شود. در مدل زیر از سه ویژگی یا ستون مجموعه داده با نام‌های جنسیت، سن، تعداد همسر و فرزندان استفاده شده است.

¹ Classification and regression tree(CART)



شکل ۲۳- درخت تصمیم مجموعه داده تایتانیک

بر اساس شرط‌های نوشته شده، درخت به شاخه‌های مختلف تقسیم می‌شود. انتهای شاخه‌ای که دیگر تقسیم نمی‌شود تصمیم یا برگ وجود دارد. درخت تصویر فوق یک درخت دسته بندی به حساب می‌آید؛ زیرا هدف این مسئله دسته بندی مسافران در دو گروه مرد و زن است. درخت رگرسیون نیز با همین روش فوق نشان داده می‌شود. با این تفاوت که مقادیر پیوسته از جمله قیمت مسکن را نشان می‌دهد.

پیاده‌سازی

مشابه قبل، بعضی از کارهای ابتدایی شامل اضافه کردن کتابخانه‌ها، خواندن داده‌ها، جدا کردن ماتریس ویژگی و پاسخ، تقسیم بندی داده به دو قسمت آموزش و تست و در صورت نیاز مقیاس‌بندی داده‌ها را باید انجام دهیم. بعد از آن برای پیاده سازی از کتابخانه سایکیت لرن مشابه زیر استفاده می‌کنیم. مجدداً مرحله آموزش مدل به صورت زیر خواهد بود:

```

from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

```

شکل ۲۴- آموزش مدل درخت تصمیم

مرحله سنجیدن داده‌ها هم به صورت زیر می‌باشد.

```

base_preds = model.predict(X_test)
base_preds
array([0., 0., 0., ..., 1., 0., 0.])
from sklearn.metrics import classification_report, confusion_matrix, plot_confusion_matrix
print(classification_report(y_test, base_preds))

```

شکل ۲۵- سنجیدن مدل درخت تصمیم

که نتیجه این بررسی برای مجموعه داده بیماری‌های قلبی به صورت زیر خواهد بود

	precision	recall	f1-score	support
0.0	0.92	0.92	0.92	68908
1.0	0.25	0.25	0.25	7196
accuracy			0.86	76104
macro avg	0.58	0.59	0.59	76104
weighted avg	0.86	0.86	0.86	76104

شکل ۲۶- خروجی سنجیدن مدل درخت تصمیم

۵-۲-۴ جنگل تصادفی^۱

جنگل تصادفی یک الگوریتم یادگیری ماشین با قابلیت استفاده آسان است که هم برای دسته‌بندی و هم رگرسیون استفاده می‌شود. درخت تصمیم بلوک سازنده جنگل تصادفی است. یک الگوریتم جنگل تصادفی، از چندین درخت تصمیم تشکیل شده است که نتیجه انتخاب شده توسط اکثر درختان تصمیم، انتخاب نهایی خواهد بود. برای مثال اگر سه درخت، خرید را پیش‌بینی کنند و یک درخت نخریدن را، پیش‌بینی نهایی خرید خواهد بود و در این صورت پیش‌بینی می‌شود که مشتری گوشی بخرد.

پیاده سازی

برای پیاده سازی بعد از انجام کارهای اولیه که در الگوریتم‌های بالا هم ذکر کردیم، مراحل آموزش و سنجش مدل را انجام می‌دهیم. هایپرپارامترهای مهم که از آن‌ها استفاده می‌کنیم تعداد درختان، تعداد پردازنده‌ها و نمره او-او بی ۰ هستند. کارکرد اولی و دومی که مشخص است اما درمورد نمره او-او بی باید گفت که این هایپرپارامتر روشی برای اعتبارسنجی متقابل جنگل تصادفی است. در این نمونه‌برداری، حدود یک سوم از داده‌ها برای آموزش مدل استفاده نمی‌شوند و برای ارزیابی کارآیی آن مورد استفاده قرار می‌گیرند. به این نمونه‌ها، «نمونه‌های کیسه» گفته می‌شود.

مراحل آموزش و سنجش مدل در زیر آورده شدند.

```
from sklearn.ensemble import RandomForestClassifier  
  
rfc = RandomForestClassifier()  
  
rfc.fit(X_train, y_train)
```

شکل ۲۷- آموزش مدل جنگل تصادفی

¹ Random forest

² oob_score

³ Bag samples

```

base_preds = rfc.predict(X_test)

print(classification_report(y_test, base_preds))

```

شکل -۲۸- سنجش مدل جنگل تصادفی

و نتیجه سنجش فوق برای مجموعه داده بیماری قلبی به صورت زیر است.

	precision	recall	f1-score	support
0.0	0.92	0.98	0.95	68908
1.0	0.38	0.14	0.21	7196
accuracy			0.90	76104
macro avg	0.65	0.56	0.58	76104
weighted avg	0.87	0.90	0.88	76104

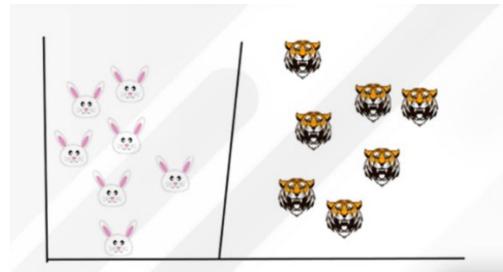
شکل -۲۹- خروجی سنجش مدل جنگل تصادفی

۶-۲-۶- ماشین بردار پشتیبان^۱

ماشین بردار پشتیبان یکی از الگوریتم‌های نظارت شده یادگیری ماشین است. بردارهای پشتیبان در واقع مختصات یک مشاهده منحفرد هستند. ماشین بردار پشتیبان نمونه داده‌هایی که به صورت نقاطی در فضای نشان داده شده است با استفاده از یک خط یا هایپرپلین از هم جدا می‌کند. این جداسازی به گونه‌ای است که نقاط داده‌ای که در یک طرف خط هستند مشابه به هم و در یک گروه قرار می‌گیرند. نمونه داده‌های جدید هم بعد از اضافه شدن به همان فضای در یکی از دسته‌های موجود قرار خواهند گرفت. برای درک نحوه عملکرد ماشین بردار پشتیبان، مثال زیر را در نظر می‌گیریم. فرض کنید صاحب مزرعه‌ای هستید و بنا به دلایلی می‌خواهید حصاری برای محافظت از خرگوش‌ها در برابر ببرها ایجاد کنیم.

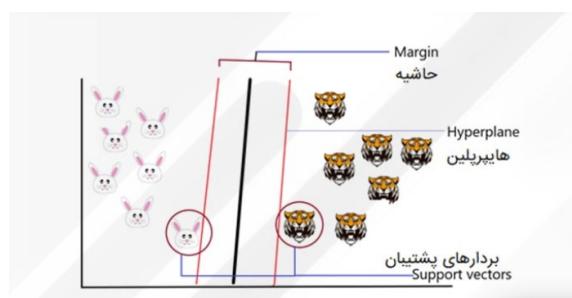
یکی از راه حل‌های این مشکل این است که یک طبقه‌بندی براساس موقعیت خرگوش‌ها و ببرها ایجاد کنیم. می‌توانیم گروه خرگوش‌ها را به عنوان یک گروه و گروه ببرها را به عنوان گروه دیگر طبقه‌بندی کنیم. در این حالت اگر سعی کنیم یک مرز میان خرگوش‌ها و ببرها بکشیم، یک خط مستقیم خواهد شد. ماشین بردار پشتیبان نیز دقیقاً به این شکل عمل می‌کند؛ یک مرز تصمیم‌گیری ترسیم می‌کند که در واقع یک هایپرپلین میان دو کلاس است تا آن‌ها را از هم جدا و طبقه‌بندی کند.

¹ Support vector machine



شکل ۳۰- شکل مرزبندی بین دو کلاس خرگوش و ببر

اصل اساسی ماشین بردار پشتیبان این است که یک هایپرپلین ترسیم کنیم که دو کلاس را به بهترین شکل از یکدیگر جدا کند. حال درمورد مثال ما دو کلاس خرگوش و ببر هستند؛ بنابراین ما با ترسیم یک هایپرپلین رندوم شروع می‌کنیم و سپس فاصله‌ی میان هایپرپلین و نزدیک‌ترین نقاط داده‌ی هر کلاس را بررسی می‌کنیم.



شکل ۳۱- شکل معرفی مفاهیم مهم ماشین بردار پشتیبان

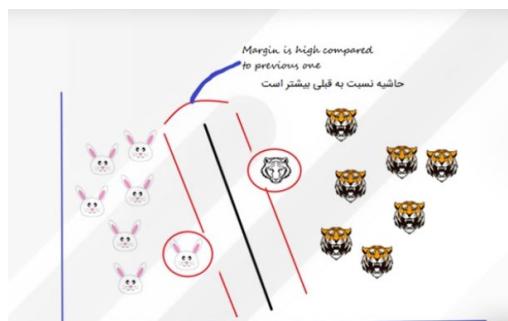
این نقاط داده نزدیک به هایپرپلین بردارهای پشتیبان نامیده می‌شوند؛ به همین دلیل است که به این الگوریتم ماشین بردار پشتیبان گفته می‌شود. اساساً هایپرپلین براساس این بردارهای پشتیبان ترسیم می‌شود. به طور معمول هایپرپلینی که بیشترین فاصله از بردارهای پشتیبان را داشته باشد بهینه‌ترین هایپرپلین است. این فاصله‌ی میان هایپرپلین حاشیه^۱ نامیده می‌شود.

¹ Margin



شکل ۳۲- شکل داده جدید نمودار

حال باید فرض کنیم یک نقطه داده جدید اضافه کنیم. اکنون می‌خواهیم یک هایپرپلین بکشیم تا این دو کلاس را به بهترین شکل از هم جدا کند؛ بنابراین، با ترسیم هایپرپلین همانطور که در تصویر بالا نشان داده شده است، شروع می‌کنیم؛ سپس فاصله‌ی میان این هایپرپلین و بردارهای پشتیبان را بررسی می‌کنیم که آیا حاشیه‌ی این هایپرپلین حداکثر است یا خیر. در این شکل حاشیه خیلی هم زیاد نیست. در سناریوی دوم، یک هایپرپلین متفاوت، مانند تصویر زیر ترسیم می‌کنیم و سپس فاصله‌ی میان هایپرپلین و بردارهای پشتیبان را بررسی می‌کنیم که آیا حاشیه‌ی این هایپرپلین حداکثر است یا خیر. می‌بینیم که حاشیه در مقایسه با هایپرپلین قبلی بسیار زیاد است؛ بنابراین ما این هایپرپلین را انتخاب می‌کنیم.



شکل ۳۳- شکل مرزبندی نهایی داده جدید

ریاضیات این الگوریتم نیز به این صورت است که ما مجموعه داده‌های آزمایش D شامل n عضو (نقطه) (را در اختیار داریم که به صورت زیر تعریف می‌شود:

$$D = \{(x_i, y_i) | x_i \in IR, y_i \in \{1, -1\}\}_{i=1}^n$$

جایی که مقدار y برابر 1 یا -1 و هر یک بردار حقیقی p بعدی است. هدف پیدا کردن ابرصفحه جداکننده با بیشترین فاصله از نقاط حاشیه‌ای است که نقاط با $y = 1$ را از نقاط با $y = -1$ جدا کند. هر ابرصفحه می‌تواند به صورت مجموعه‌ای از نقاط x که شرط زیر را ارضاء می‌کنند نوشته شود:

$$\langle w \cdot x_n \rangle + b = 0$$

ما می خواهیم w و b را طوری انتخاب کنیم که بیشترین فاصله بین ابر صفحه های موازی که داده ها را از هم جدا می کنند، ایجاد شود. این ابر صفحه ها با استفاده از رابطه زیر توصیف می شوند.

$$\langle w \cdot x_n \rangle + b = 1$$

$$\langle w \cdot x_n \rangle + b = -1$$

برای اینکه از ورود نقاط به حاشیه جلوگیری کنیم، شرایط زیر را اضافه می کنیم: برای هر

$$\begin{cases} \langle w \cdot x_n \rangle + b \geq 1, & \text{if } y_n = 1 \\ \langle w \cdot x_n \rangle + b \leq -1, & \text{if } y_n = -1 \end{cases}$$

این می تواند به صورت زیر نوشته شود:

$$y_n(\langle w \cdot x_n \rangle + b) \geq 1. \forall 1 \leq n \leq N$$

با کتاب هم قرار دادن این دو یک مسئله بهینه سازی به دست می آید:

$$\min \frac{1}{2} \|w\|^2$$

$$y_n(\langle w \cdot x_n \rangle + b) \geq 1. \forall 1 \leq n \leq N$$

می توان نشان داد که دو گان ماشین بردار پشتیبان به مسئله بهینه سازی زیر ساده می شود:

$$\max_{\forall \alpha_i} (L(\alpha)) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j). s.t \forall i. \alpha_i \geq 0. and \sum_{i=1}^N \alpha_i y_i = 0$$

عبارت α تشکیل یک دو گان برای بردار وزن ها مجموعه آموزشی می دهد:

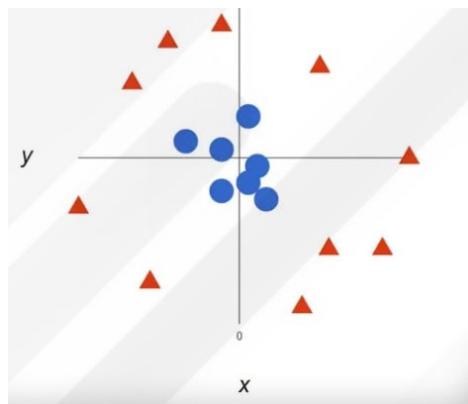
$$w = \sum_i \alpha_i y_i x_i$$

برای داده هایی که به صورت خطی جدایی ناپذیر هستند ماشین بردار پشتیبان از روشی که به آن ترفند هسته (کرنل (گفته می شود، استفاده می کند. در این روش در واقع توابعی وجود دارند که فضای ورودی بُعد پایین را دریافت کرده و آن را به فضای بُعد بالاتر تبدیل می کنند. این تبدیل، یک مسئله غیر قابل جداسازی را به مسئله قابل جداسازی مبدل می کند. به این توابع، تابع های هسته گفته می شود. توابع کرنل بیشتر در مسائل جداسازی غیرخطی مفید هستند. این توابع برخی از داده های فوق العاده پیچیده را تبدیل می کنند و سپس فرآیندی را می یابند که با استفاده از آن بتوانند این داده ها را بر اساس برچسب هایی که کاربر تعریف کرده، جداسازی کنند.

ماشین بردار پشتیبان و داده های غیرخطی

در مثال قبل دیدیم که داده های ما تاکنون به صورت خطی تفکیک پذیر بودند، یعنی می توانستیم یک خط مستقیم برای جدا کردن دو کلاس بکشیم اما اگر نقاط داده ای ما به شکل زیر باشد، چه کنیم؟ این دو کلاس با یک خط مستقیم از هم جدا نمی شوند. هنگامی که نقاط داده را نمی توان با یک خط مستقیم یا یک هایپرپلین مستقیم جدا کرد، مسئله غیرخطی نامیده می شود. در چنین شرایطی کرنل های ماشین بردار پشتیبان وارد عمل می شوند و بعد از این افزایش می دهند تا نقاط داده به

صورت خطی تفکیک پذیر شوند. توابع کرنل ، ضرب داخلی بین دو نقطه در یک فضای ویژگی مناسب را برمی گردانند. بنابراین ، با هزینه محاسباتی کم، حتی در فضاهای با ابعاد بالا، مفهومی از شباهت را تعریف می کنند.

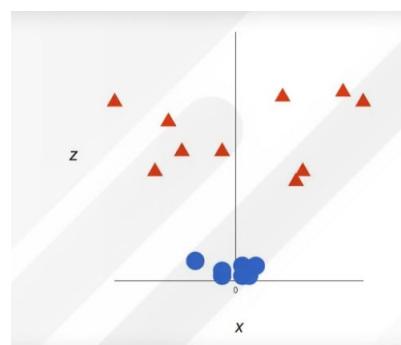


شکل ۳۴- شکل داده‌های غیرخطی

پس اگر داده‌های اولیه ما به شکل بالا باشند، لازم است که ما بعد سوم را اضافه کنیم. تابهحال ما دو بعد داشتیم: ایکس و ایگرگ؛ حال یک بعد زد جدید ایجاد می‌کنیم که به این شکل محاسبه می‌شود:

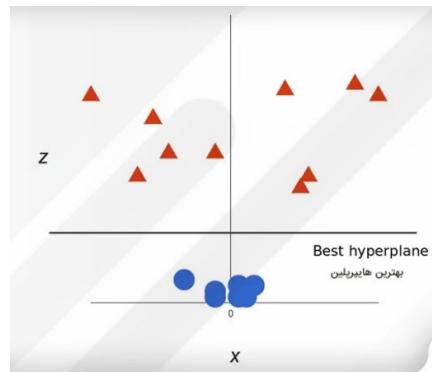
$$z = x^2 + y^2$$

این کار به ما فضایی سهبعدی می‌دهد که به این شکل می‌توان آن را در اینجا نشان داد:

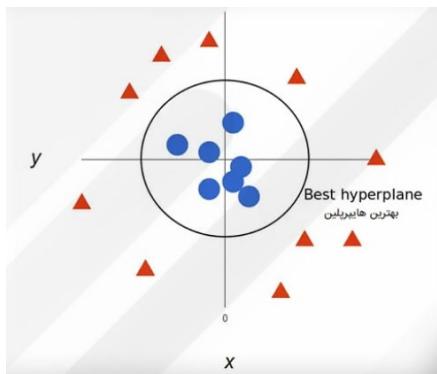


شکل ۳۵- شکل داده‌ها بعد از انتقال به فضای سه بعدی

ماشین بردار پشتیبان تفکیک را به صورت زیر انجام می‌دهد:



شکل -۳۶- تفکیک داده توسط ماشین بردار پشتیبان در فضای جدید



شکل -۳۷- تفکیک داده توسط ماشین بردار پشتیبان در فضای قبلی

نمونه‌هایی از کرنل‌های ماشین بردار پشتیبان

- کرنل چندجمله‌ای: این کرنل در پردازش تصویر پرکاربرد است و معادله‌ی آن به صورت زیر است:

$$k(x_i \cdot x_j) = (x_i \cdot x_j + 1)^d$$

- کرنل گاوسی: این یک کرنل برای اهداف عمومی است و هنگامی که هیچ دانش پشتیبانی درمورد داده‌ها وجود ندارد استفاده می‌شود و معادله آن به صورت زیر است:

$$k(x \cdot y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

- تابع پایه شعاعی گاوسی: این کرنل برای اهداف عمومی کاربرد دارد و هنگامی که هیچ دانش پشتیبانی درمورد داده‌ها وجود نداشته باشد، مورد استفاده قرار می‌گیرد؛ معادله آن به صورت زیر است:

$$k(x, y) = \exp(-\gamma \|x_i - x_j\|^2)$$

پیاده سازی نمونه در اینجا آورده شده است:

```
import numpy as np
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
y = np.array([1, 1, 2, 2])

model = make_pipeline(StandardScaler(), SVC(gamma='auto'))
model.fit(X, y)

print("Test Prediction: ", model.predict([-0.8, -1]))
Test Prediction:  [1]
```

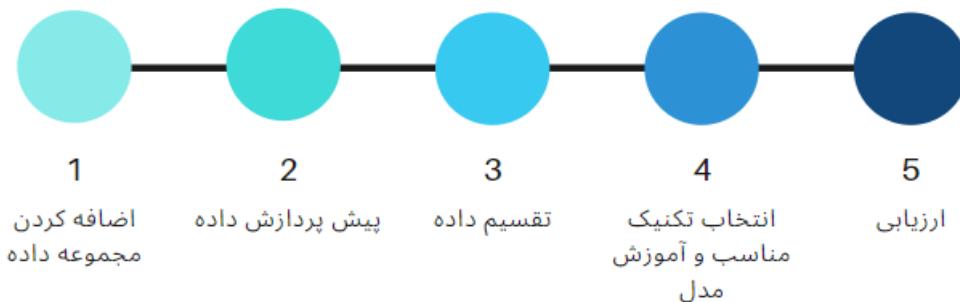
شکل ۳۸- پیاده سازی ماشین بردار پشتیبان با زبان پایتون با یک مثال ساده

۴-۳ پیاده سازی

در این بخش الگوریتم های مطالعه شده را روی پنج مجموعه داده پیاده سازی خواهیم کرد. این پنج مجموعه داده عبارت اند از:

- مجموعه داده شاخص سلامت افراد (طبقه بندی)
- مجموعه داده کلیک روی تبلیغات (طبقه بندی)
- مجموعه داده تبلیغات بانکی (طبقه بندی)
- مجموعه داده مسافران کشتی تایتانیک (طبقه بندی)
- مجموعه داده خانه های سنگاپور (رگرسیون)

مسیر کلی فرآیند مدل سازی در سایکیت لرن، به صورت شکل زیر می باشد



شکل ۳۹- مسیر کلی فرآیند مدل سازی در سایکیت لرن

که در ادامه به پیاده‌سازی الگوریتم‌های مطالعه شده مطابق روند فوق روی مجموعه داده‌های ذکر شده خواهیم پرداخت.

۴-۳-۱ طبقه‌بندی مجموعه داده شاخص سلامت افراد

سیستم نظارت بر عوامل خطر رفتاری، یک نظرسنجی تلفنی مرتبط با سلامت است که سالانه توسط مرکز کنترل و پیشگیری بیماری جمع‌آوری می‌شود. این مجموعه داده جمع‌آوری شده شامل پاسخ‌های تعداد زیادی از افراد به همراه تعدادی ویژگی است. این ویژگی‌ها یا سوالاتی هستند که مستقیماً از شرکت‌کنندگان پرسیده می‌شود یا متغیرهای محاسبه شده براساس پاسخ‌های فردی شرکت‌کنندگان هستند.

۴-۱-۱-۱ فرآیند مدل‌سازی سایکیت‌لرن روی مجموعه داده کلیک روی تبلیغات

در اولین قدم کتابخانه‌های مورد استفاده را به کد اضافه می‌کنیم.

```
#import the necessary packages
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score, StratifiedKFold, train_test_split, KFold
from sklearn.metrics import accuracy_score, recall_score, precision_score, classification_report, confusion_matrix
import warnings
from sklearn.preprocessing import StandardScaler
warnings.filterwarnings(action='ignore')

%matplotlib inline
```

مرحله اول: بارگذاری داده

در این مرحله داده مورد نظر را بارگذاری کرده و اطلاعاتی پیرامون آن بدست می‌آوریم. برای مثال، سعی می‌کنیم مقادیر خالی ستون‌ها را پیدا کنیم، اطلاعات عددی مهم از ستون‌های حاوی مقادیر عددی بدست آوریم و داده را مصورسازی کنیم. در قطعه کد زیر عمل بارگذاری داده را انجام داده‌ایم.

	HeartDiseaseorAttack	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	Diabetes	PhysActivity	Fruits	... AnyHealthcare	NoDocbcCost	GenH
0	0.0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.0	0.0	0.0	0.0	25.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
2	0.0	1.0	1.0	1.0	28.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0
3	0.0	1.0	0.0	1.0	27.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0
4	0.0	1.0	1.0	1.0	24.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0
...
253675	0.0	1.0	1.0	1.0	45.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
253676	0.0	1.0	1.0	1.0	18.0	0.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0
253677	0.0	0.0	0.0	1.0	28.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0
253678	0.0	1.0	0.0	1.0	23.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
253679	1.0	1.0	1.0	1.0	25.0	0.0	0.0	2.0	1.0	1.0	1.0	0.0	0.0

با دستور زیر نیز سعی کردیم اطلاعاتی عددی از ستون‌هایی که حاوی مقادیر کمی هستند بدست آوریم.

dataset.describe()									
	HeartDiseaseorAttack	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	Diabetes	PhysActivity
count	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000
mean	0.094186	0.429001	0.424121	0.962670	28.382364	0.443169	0.040571	0.296921	0.756544
std	0.292087	0.494934	0.494210	0.189571	6.608694	0.496761	0.197294	0.698160	0.429169
min	0.000000	0.000000	0.000000	0.000000	12.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	1.000000	24.000000	0.000000	0.000000	0.000000	1.000000
50%	0.000000	0.000000	0.000000	1.000000	27.000000	0.000000	0.000000	0.000000	1.000000
75%	0.000000	1.000000	1.000000	1.000000	31.000000	1.000000	0.000000	0.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	98.000000	1.000000	1.000000	2.000000	1.000000

و با دستور زیر نیز، تعداد مقادیر گمشده در ستون‌ها را محاسبه کردیم که همانطور که در تصویر زیر دیده می‌شود مقادیر گمشده‌ای نداشتیم.

```
#number of null values in each column  
dataset.isnull().sum()
```

HeartDiseaseorAttack	0
HighBP	0
HighChol	0
CholCheck	0
BMI	0
Smoker	0
Stroke	0
Diabetes	0
PhysActivity	0
Fruits	0
Veggies	0
HvyAlcoholConsump	0
AnyHealthcare	0
NoDocbcCost	0
GenHlth	0
MentHlth	0
PhysHlth	0
DiffWalk	0
Sex	0
Age	0
Education	0
Income	0

مرحله دوم: پیش‌پردازش داده

به طور کلی، این مرحله خود شامل چهار عمل اصلی زیر است:

- پاکسازی داده: فرآیند حذف کردن داده نادرست، داده ناتمام و یا نامناسب از مجموعه داده و جایگزینی داده گمشده.

- یکپارچه‌سازی داده: یکی کردن چندین منبع به یک مجموعه داده.
- کاهش داده: کاهش حجم داده.
- تبدیل داده: مقیاس‌دهی، گسسته‌سازی و غیره.

کتابخانه سایکیت لرن از پکیج مربوط به پیش پردازش^۱ برای این منظور استفاده می‌کند. این پکیج چندین تابع و کلاس تبدیل کننده‌ی رایج برای تبدیل بردار ویژگی و داده‌ی خام به داده‌های قابل استفاده توسط تخمین‌گرها را فراهم می‌کند. در این مرحله به صورت زیر عمل می‌کنیم:

- استاندارد سازی (مقیاس‌دهی استاندارد)^۲

الگوریتم‌های یادگیری با استفاده از استانداردسازی داده‌ها، می‌توانند عملکرد بهتری داشته باشند. اگر برخی داده‌های پرت در مجموعه داده وجود داشته باشد، مقیاس‌دهی داده‌ها و استفاده از توابع تبدیل کننده‌ی قوی تر مناسب است. برای الگوریتم‌های تخمین‌گر تعریف شده در سایکیتلرن، استانداردسازی مجموعه داده، عمل رایج و مورد نیاز محاسبه می‌شود. اگر مقادیر ویژگی‌های ورودی الگوریتم تخمین‌گر شباهت زیادی به توزیع نرمال استاندارد داده (منظور از توزیع نرمال: داده‌هایی با توزیع گوسی با میانگین صفر و واریانس واحد) نداشته باشد، باعث می‌شود که تخمین‌گر رفتار خوب و خروجی مناسبی روی داده‌ها نداشته باشد.

در عمل، اکثراً شکل و نحوه‌ی توزیع داده‌ها را در نظر نمی‌گیریم و ابتدا مقدار میانگین داده‌ها را از آن‌ها کم می‌کنیم و سپس حاصل را بر انحراف معیار تقسیم کرده و به این صورت داده‌ها را، متتمرکز می‌کنیم. به عنوان مثال بسیاری از عناصر مورد استفاده در تابع هدف الگوریتم یادگیری ماشین مانند کرنل ار بی اف^۳ در الگوریتم اس وی ام^۴، فرض می‌کنند که مرکز تمام ویژگی‌ها حول صفر است و دارای پراکندگی و واریانس یکسان‌اند. اگر ویژگی دارای واریانس بزرگ‌تر از سایرین باشد، ممکن است بر عملکرد تابع هدف مسلط شود و سبب شود تا تخمین‌گر به درستی از مقادیر سایر ویژگی‌ها برای یادگیری خود استفاده نکند. در ادامه یکی از این تکنیک‌های مهم را معرفی می‌کنیم.

ماژول پیش پردازش کلاس مقیاس‌دهی استاندارد را دارد که حذف میانگین و تقسیم واریانس را بر روی داده‌های آموزشی اجرا می‌کند و توسط رابط خود اعمال حذف همین مقدار میانگین و تقسیم همان واریانس محاسبه شده‌ی حاصل از داده‌های آموزشی را بر روی داده‌های تست فراهم می‌کند. اگر به هر دلیلی بخواهیم امكان حذف میانگین و یا تقسیم شدن داده به انحراف معیار را حذف کنیم، با ارجومان‌های ورودی می‌توانیم این کار را انجام

¹ Sklearn.preprocessing

² Standard scaling

³ Radial basis function

⁴ Support vector machine

دهیم. نکته حائز اهمیت این است که عمل استاندارد سازی بعد از مرحله تقسیم داده انجام می شود. زیرا پیش پردازش داده تست و داده آموزشی نباید با هم انجام شود. برای همین پیاده سازی این مرحله، پس از تقسیم داده خواهد بود.

مرحله سوم: تقسیم^۱ داده

برای کنترل دقیق مدل استخراج شده، می توان مجموعه داده را به دو قسمت تقسیم کرد: مجموعه آموزشی^۲ و مجموعه آزمایشی^۳ یا تست؛ از داده‌ی آموزشی برای آموزش روش یادگیری ماشین و استخراج مدل اموزشی استفاده می شود. مدل استخراجی حاصل روی مجموعه‌ی تست اعمال می شود تا دقیقیت مدل بررسی شود. در کد زیر داده را به نسبت ۷۰ به ۳۰ تقسیم کردیده‌ایم. یعنی ۷۰ درصد داده‌ها، داده‌ی آموزشی در نظر گرفته شده‌اند و ۳۰ درصد به داده‌های تست اختصاص یافته‌اند.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

همان طور که در کد بالا می‌بینید تابع `train_test_split()` برای تقسیم مجموعه داده به کار گرفته می‌شود. آرگومان‌های این تابع به شرح زیر هستند:

- ورودی اول یا ایکس ماتریس ویژگی و ورودی دوم یا ایگرگ بردار پاسخ است.
- اندازه تست یا همان ورودی سوم بیانگر نسبت داده‌های تست به کل مجموعه داده است. در مثال بالا، ۰,۳ از ۱۵۰ سطر موجود را به عنوان داده‌ی تست در نظر گرفتیم. پس ۴۵ سطر متعلق به داده‌ی تست و بقیه متعلق به داده‌های آموزشی است.

ورودی چهارم یا همان رندوم استیت پارامتری است که اگر مقدار این پارامتر را برابر هر عددی صحیحی در نظر بگیریم در آن صورت در هر مرحله‌ای که الگوریتم برای تولید مدل تکرار می‌شود، داده‌ها به نسبت مساوی با مرحله‌ی قبل به داده‌های آموزشی و تست تقسیم می‌شوند. به طور مثال در هر مرحله به طور تصادفی ۰,۷ داده‌ها به داده‌ی آموزشی و ۰,۳ آن به داده‌ی تست اختصاص می‌یابد.

مرحله دوم*: استاندارد سازی

همانطور که گفته شد، عمل استانداردسازی را بعد از تقسیم داده و به صورت زیر انجام می‌دهیم.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

¹Splitting

²Training set

³Test set

مرحله چهارم؛ آموزش و آزمایش مدل

بعد از تقسیم داده‌ها، می‌توانیم داده‌ها را برای یادگیری مدل‌های پیش‌بینی به کار بگیریم. همان طور که گفته شد کتابخانه سایکیتلرن، الگوریتم‌های یادگیری ماشین گسترده‌ای دارد که رابط ثابتی برای مدل کردن، پیش‌بینی دقیق و فراخوانی برای تمام الگوریتم‌ها ارائه می‌دهد. در این قسمت طبقه‌بندی داده با کمک الگوریتم‌های رگرسیون لجیستیک، کا-همسایه نزدیک، درخت تصمیم، بیز ساده و جنگل تصادفی انجام شده است. در اینجا ابتدا، مدل‌های مورد نظر را ایجاد کردیم و برای کم کردن تعداد خطوط کد، همه را داخل آرایه‌ای قرار داریم تا بعدا از آن‌ها استفاده کنیم.

```
models = {
    'LogisticRegression': LogisticRegression(),
    'GaussianNB': GaussianNB(),
    'DecisionTreeClassifier': DecisionTreeClassifier(),
    'RandomForestClassifier': RandomForestClassifier(),
    'KNeighborsClassifier': KNeighborsClassifier()
}
```

پس از آن، آرایه‌ای ایجاد کردیم تا نتایج ارزیابی الگوریتم‌ها را داخل آن‌ها نگهداری کنیم و بلاfacسله نیز عمل برآش روی مدل و پیش‌بینی را انجام دادیم. نتیجه پیش‌بینی را داخل آرایه قرار دادیم.

```
scores = {}

for i, model in models.items():
    print("model: ", model)

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

    print(classification_report(y_test, y_pred))
    print("-" * 50)
    scores[i] = model.score(X_test, y_test)
```

نتایج پیاده‌سازی فوق در زیر آورده شده‌اند.

model:	LogisticRegression()	precision	recall	f1-score	support	
		0.0	0.92	0.99	0.95	68970
		1.0	0.54	0.13	0.21	7134
		accuracy			0.91	76104
		macro avg	0.73	0.56	0.58	76104
		weighted avg	0.88	0.91	0.88	76104

```

model: GaussianNB()
      precision    recall  f1-score   support

       0.0      0.95     0.84     0.89    68970
       1.0      0.27     0.55     0.36    7134

   accuracy                           0.82    76104
  macro avg      0.61     0.70     0.63    76104
weighted avg      0.88     0.82     0.84    76104
-----


model: DecisionTreeClassifier()
      precision    recall  f1-score   support

       0.0      0.92     0.91     0.92    68970
       1.0      0.24     0.27     0.26    7134

   accuracy                           0.85    76104
  macro avg      0.58     0.59     0.59    76104
weighted avg      0.86     0.85     0.86    76104
-----


model: RandomForestClassifier()
      precision    recall  f1-score   support

       0.0      0.91     0.99     0.95    68970
       1.0      0.44     0.11     0.17    7134

   accuracy                           0.90    76104
  macro avg      0.68     0.55     0.56    76104
weighted avg      0.87     0.90     0.88    76104
-----

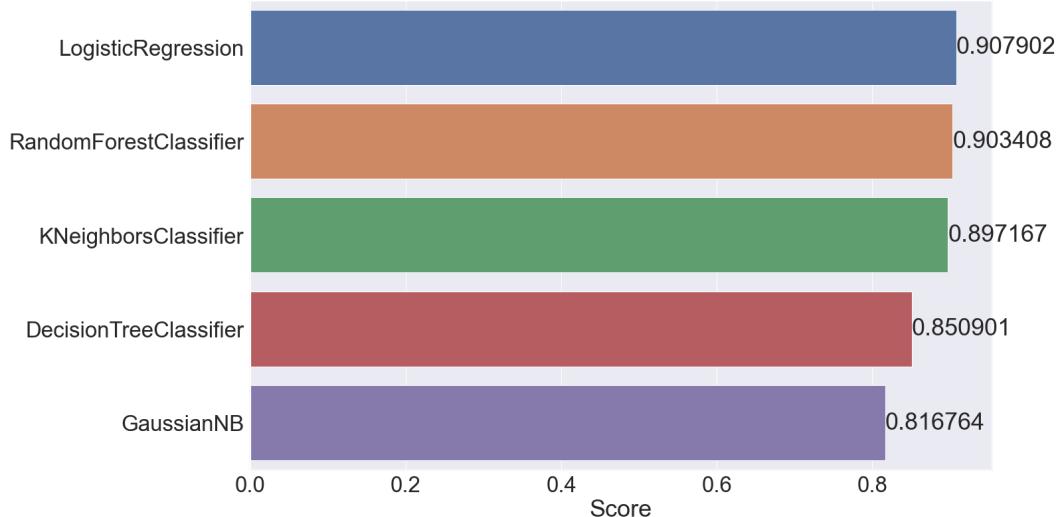

model: KNeighborsClassifier()
      precision    recall  f1-score   support

       0.0      0.92     0.97     0.94    68970
       1.0      0.38     0.16     0.22    7134

   accuracy                           0.90    76104
  macro avg      0.65     0.57     0.58    76104
weighted avg      0.87     0.90     0.88    76104
-----
```

در پایان نیز، نتایج تمام این پیاده‌سازی‌ها را داخل آرایه‌ای قرار داده و با هم مقایسه کردیم؛ معیار ارزیابی را در اینجا، دقت مدل¹ قرار دادیم که این نتایج به صورت زیر بودند.

¹ Accuracy



۲-۳-۴ طبقه‌بندی مجموعه داده کلیک روی تبلیغات

این مجموعه داده دربرگیرنده‌ی اطلاعاتی آماری درباره مشتری و اینکه آیا آن مشتری روی تبلیغی کلیک کرده و خریداری کرده است یا خیر، می‌باشد که با استفاده از الگوریتم طبقه‌بندی و داده‌های داده شده باید این پیش‌بینی را انجام دهیم. ستون‌های ویژگی این مجموعه داده شامل آیدی کاربر، سن، حقوق تخمینی، جنسیت، و ستون خروجی، اینکه خرید انجام داده یا خیر می‌باشند. ستون‌های مربوط به سن مشتری، حقوق تخمینی، آیدی کاربر، خریداری شده (صفر نشان دهنده کلیک نشده و یک نشان دهنده کلیک شده) مقدارهای عددی یا کمی و ستون جنسیت، مقدار غیرعددی یا کیفی دارند. از الگوریتم‌های طبقه‌بندی روی این مجموعه داده استفاده می‌کنیم تا این پیش‌بینی را انجام دهیم. در آخر نیز، مدل‌های بدست‌آمده را با سنجه‌های مختلف ارزیابی خواهیم کرد.

۱-۲-۳ فرآیند مدل‌سازی سایکیت‌لرن روی مجموعه داده کلیک روی تبلیغات

در ابتدا ما کتابخانه‌های مورد نیاز برای این پیاده‌سازی را اضافه می‌کنیم.

```
#import the necessary packages
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score, StratifiedKFold, train_test_split, KFold
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score, classification_report, confusion_matrix
import warnings
warnings.filterwarnings(action='ignore')

%matplotlib inline
```

مرحله اول: اضافه کردن مجموعه داده

در این مرحله مجموعه داده جمع آوری شده را با دستور زیر بارگذاری می‌کنیم.

```
dataset = pd.read_csv('Social_Network_Ads.csv')  
dataset
```

که در این مثال خروجی آن به صورت زیر خواهد بود.

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

پس از آنکه مجموعه داده را بارگذاری کردیم، سعی می‌کنیم اطلاعاتی کلی درباره این مجموعه بدست آوریم. برای مثال با دستور زیر، برای ستون‌های دارای مقادیر کمی، کمترین و بیشترین مقدار، تعداد کل، میانگین و از این دست اطلاعات را می‌توانیم بدست آوریم که به همراه خروجی در زیر آورده شده است.

```
dataset.describe()
```

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

پس از آن اطلاعات دیگری نظری تعداد سلول‌های خالی در یک ستون را نیز با دستور زیر بدست آوردم که در این مجموعه داده سلول خالی در ستون‌ها وجود ندارد.

```
dataset.isnull().sum()
```

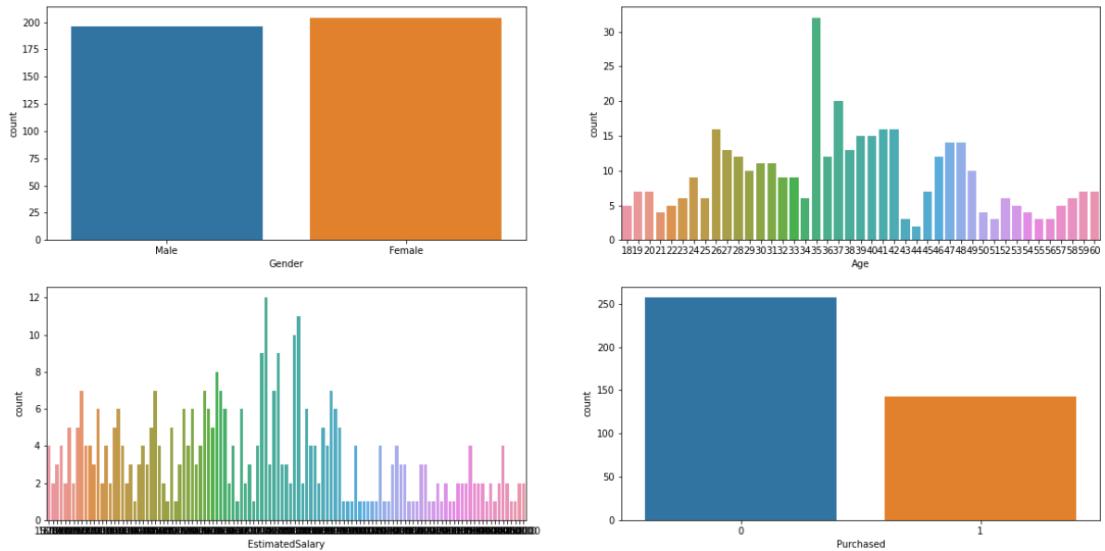
```
User ID          0  
Gender          0  
Age             0  
EstimatedSalary 0  
Purchased       0  
dtype: int64
```

همچنین، نمودارهای مختلفی رسم می‌کنیم تا اطلاعات بیشتری درمورد این داده‌ها بدست آوریم. این کار را با دو کتابخانه مت‌پلات و سیبیورن انجام می‌دهیم. کد مربوط به تعدادی از این مصورسازی‌ها به همراه خروجی در تصاویر زیر آمده است. برای مثال در زیر ما نمودار شمارشی را برای ستون‌های ویژگی رسم کردیم. نمودار بالا سمت چپ، نشان می‌دهد که تعداد آقایان و خانم‌ها تقریباً با هم برابرند. نمودار مربوط به سن نشان می‌دهد که بیشتر افراد در این مجموعه داده ۳۵ ساله اند و ستون مربوط به خریداری شده هم که ستون خروجی ما می‌باشد نشان‌دهنده این است که تعداد افراد خرید نکرده بیشتر از خرید کرده‌ها هستند.

```

fg, ax = plt.subplots(2 , 2, figsize=(20,10))
for i in range(1 , len(dataset.columns)):
    x = i - 1;
    sns.countplot(x = dataset.columns[i], data = dataset, ax=ax[(int)(x / 2)][(int)(x % 2)]);

```

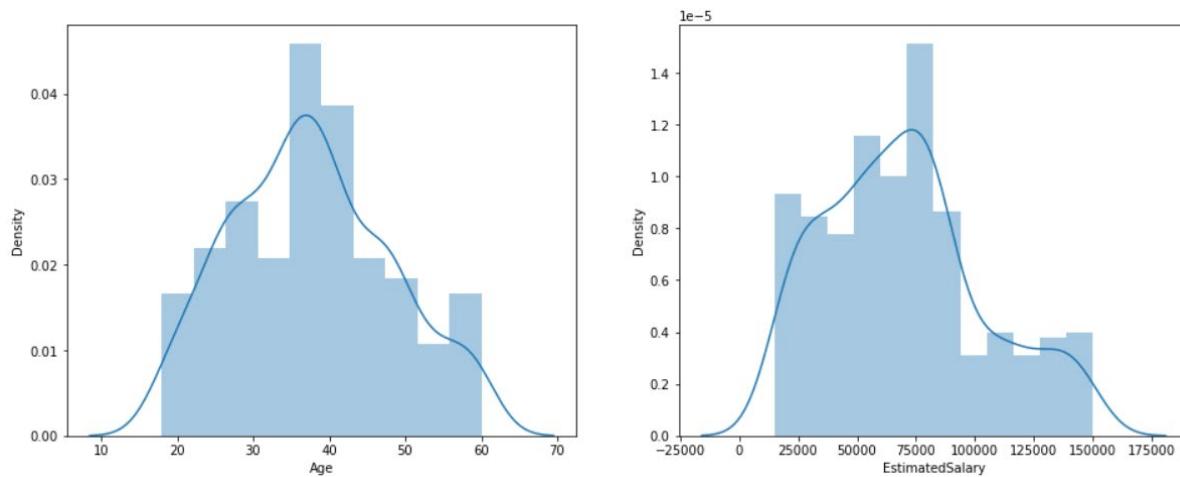


در نمونه‌ای دیگر، سعی می‌کنیم نمودار توزیعی این داده برای ستون‌های سن و ستوان حقوق تخمینی را رسم کنیم. همانطور که در شکل زیر نشان داده شده در نمودار چگالی تعداد افراد بر حسب سن، چگالی در بازه سی تا چهل بیشترین مقدار و در نمودار چگالی بر حسب حقوق تخمینی هم بیشترین چگالی، در حدود ۷۵ هزار می‌باشد.

```

fg,ax = plt.subplots(1, 2, figsize = (16, 6))
sns.distplot(dataset.Age, ax = ax[0])
sns.distplot(dataset.EstimatedSalary, ax = ax[1])

```



مرحله دوم: پیش‌پردازش داده

برای پیش‌پردازش داده در این مجموعه به صورت زیر عمل می‌کنیم.

- تبدیل داده‌های کیفی به کمی:

در این مثال نیاز است تا ستون‌های کیفی به ستون‌های کمی تبدیل شوند. این کار با دستور زیر انجام می‌شود که در حقیقت برای هر یک از مقادیر ممکنی که این ستون می‌تواند داشته باشد، یک ستون جدید ایجاد می‌کند که مقدار صفر یا یک دارد. به این معنی که هر نمونه داده، در یکی از آن ستون‌های تازه ایجاد شده می‌تواند مقدار یک داشته باشد. در اینجا چون جنسیت تنها دو مقدار دارد می‌توانیم به جای ایجاد دو ستون، تنها یک ستون ایجاد کنیم که یک بودن در این ستون نشان دهنده مرد بودن، و صفر نشان دهنده خانم بودن است.

```
Gender = pd.get_dummies(dataset['Gender'], drop_first = True)  
Gender
```

Male	
0	1
1	1
2	0
3	0
4	1
...	...
395	0
396	1
397	0
398	1
399	0

400 rows × 1 columns

- حذف ستون‌های بدون کاربرد:

سپس ستون آیدی کاربر را حذف می‌کنیم زیرا آیدی کاربر هیچ تاثیری در تخمین نهایی اینکه او خرید کرده است یا خیر ندارد و بدتر ممکن است باعث ایجاد خطا شود.

```
dataset.drop(['UserID'], axis=1, inplace=True)  
dataset
```

	Gender	Age	EstimatedSalary	Purchased
0	1	19	19000	0
1	1	35	20000	0
2	0	26	43000	0
3	0	27	57000	0
4	1	19	76000	0
...
395	0	46	41000	1
396	1	51	23000	1
397	0	50	20000	1
398	1	36	33000	0
399	0	49	36000	1

- جدا کردن ستون های ویژگی و ستون خروجی:

در انتهای این مرحله نیز، ستون های ویژگی و ستون خروجی را به صورت زیر از هم جدا می کنیم.

```
X = dataset.iloc[:, :3]  
y = dataset['Purchased']
```

- استاندارد سازی (مقیاس دهی استاندارد)

پیاده سازی این مرحله برای این مثال، پس از مرحله تقسیم داده آورده شده است.

مرحله سوم: تقسیم¹ داده

در کد زیر داده را به نسب ۷۰ به ۳۰ تقسیم کردہ ایم. یعنی ۷۰ درصد داده ها، داده های آموزشی در نظر گرفته شده اند و ۳۰ درصد به داده های تست اختصاص یافته اند.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

¹ Splitting

مرحله دوم*: استاندارد سازی

همانطور که گفته شد، عمل استاندارد سازی باید بعد از مرحله تقسیم داده انجام شود. پیاده‌سازی آن به صورت زیر خواهد بود.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

مرحله چهارم: آموزش و آزمایش مدل

در این قسمت نیز طبقه‌بندی داده با کمک الگوریتم‌های رگرسیون لجیستیک، کا-همسایه نزدیک، درخت تصمیم، بیز ساده و جنگل تصادفی انجام شده است. همچنین طبقه‌بندی داده را با چهار کرنل ماشین بردار پشتیبان و چهار کرنل از پکیج او-آر-اس وی ام نیز انجام دادیم که پیاده‌سازی و نتایج آن‌ها را در ادامه خواهیم دید. شکل زیر پیاده‌سازی با کمک پنج الگوریتم اول است که به همراه نتیجه آورده شده است. ایجاد کردن مدل مشابه قبل به صورت زیر بود.

```
models = {
    'LogisticRegression': LogisticRegression(),
    'KNeighborsClassifier': KNeighborsClassifier(),
    'DecisionTreeClassifier': DecisionTreeClassifier(),
    'GaussianNB': GaussianNB(),
    'RandomForestClassifier': RandomForestClassifier()
}
```

همچنین آرایه‌ای خالی بوجود می‌آوریم تا نتایج ارزیابی را داخل آن قرار بrizیم.

```
scores = {}
```

پس از آن عمل برآش مدل را روی داده آموزشی انجام می‌دهیم و پیش‌بینی مدل از داده آزمایشی را داخل متغیری ریخته تا نتایج را ارزیابی کنیم.

```
for i, model in models.items():
    print("model: ", model)

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

پس از آن ارزیابی‌های بدست آمده را داخل آرایه ای ریخته و آن را چاپ می‌کنیم که به همراه نتایج در زیر آورده شده‌اند.

```
print(classification_report(y_test , y_pred))
print("-" * 50)
scores[i] = model.score(X_test,y_test)
```

model:	LogisticRegression()	precision	recall	f1-score	support
	-1	0.83	0.97	0.89	73
	1	0.94	0.68	0.79	47
	accuracy			0.86	120
	macro avg	0.88	0.83	0.84	120
	weighted avg	0.87	0.86	0.85	120

model:	KNeighborsClassifier()	precision	recall	f1-score	support
	-1	0.93	0.93	0.93	73
	1	0.89	0.89	0.89	47
	accuracy			0.92	120
	macro avg	0.91	0.91	0.91	120
	weighted avg	0.92	0.92	0.92	120

model:	DecisionTreeClassifier()	precision	recall	f1-score	support
	-1	0.88	0.86	0.87	73
	1	0.79	0.81	0.80	47
	accuracy			0.84	120
	macro avg	0.83	0.84	0.83	120
	weighted avg	0.84	0.84	0.84	120

model: GaussianNB()				
	precision	recall	f1-score	support
-1	0.89	0.97	0.93	73
1	0.95	0.81	0.87	47
accuracy			0.91	120
macro avg	0.92	0.89	0.90	120
weighted avg	0.91	0.91	0.91	120

model: RandomForestClassifier()				
	precision	recall	f1-score	support
-1	0.91	0.93	0.92	73
1	0.89	0.85	0.87	47
accuracy			0.90	120
macro avg	0.90	0.89	0.89	120
weighted avg	0.90	0.90	0.90	120

حال به سراغ پیاده‌سازی با کمک کرنل‌های ماشین بردار پشتیبان می‌رویم و کارهای فوق را برای این کرنل‌ها نیز انجام می‌دهیم. که پیاده‌سازی مدل ماشین بردار پشتیبان به همراه خروجی در اینجا آورده شده است.

```

kernels = {'SVC_linear':'linear','SVC_poly':'poly','SVC_rbf':'rbf','SVC_sigmoid':'sigmoid'}
```

```

svc = []
```

```

def SVM_fit_score(kernels, X_train, X_test, y_train, y_test):
    for i, kernel in kernels.items():
        print('kernel: ', kernel)
        svc[i] = SVC(kernel=kernel).fit(X_train, y_train)
        y_pred = svc[i].predict(X_test)

        print(classification_report(y_test , y_pred))
        print('*'*60)
        scores[i] = svc[i].score(X_test,y_test)
```

```

SVM_fit_score(kernels, X_train, X_test, y_train, y_test)
```

kernel: linear					
	precision	recall	f1-score	support	
-1	0.81	0.99	0.89	73	
1	0.97	0.64	0.77	47	
accuracy			0.85	120	
macro avg	0.89	0.81	0.83	120	
weighted avg	0.87	0.85	0.84	120	

kernel: poly					
	precision	recall	f1-score	support	
-1	0.84	1.00	0.91	73	
1	1.00	0.70	0.82	47	
accuracy			0.88	120	
macro avg	0.92	0.85	0.87	120	
weighted avg	0.90	0.88	0.88	120	

kernel: rbf					
	precision	recall	f1-score	support	
-1	0.96	0.93	0.94	73	
1	0.90	0.94	0.92	47	
accuracy			0.93	120	
macro avg	0.93	0.93	0.93	120	
weighted avg	0.93	0.93	0.93	120	

kernel: sigmoid					
	precision	recall	f1-score	support	
-1	0.84	0.95	0.89	73	
1	0.89	0.72	0.80	47	
accuracy			0.86	120	
macro avg	0.87	0.83	0.85	120	
weighted avg	0.86	0.86	0.85	120	

۴-۳-۲-۲ فرآیند مدلسازی با پکیج او-آر-اس وی ام روی مجموعه داده کلیک روی تبلیغات

همچنین این عمل را برای کرنل‌های پکیج او-آر-اس وی ام هم انجام می‌دهیم. اما نیاز به چند تغییر دیگر نیز برای اینکه بتوانیم داده را به کرنل‌های او-آر-اس وی ام بدهیم وجود دارد. اول از همه نیاز به دانلود و نصب این پکیج و سپس اضافه کردن این کتابخانه است. با دستور زیر این کتابخانه را بعد از نصب می‌توانیم به کد اضافه کنیم.

```
import orsvm
```

بعد از آن نیز، به دلایل مربوط به پیاده‌سازی داخلی این پکیج، اندازه داده تست و آموزشی در این مدل با قبلی‌ها متفاوت است و همچنین به همان دلیل فوق لازم بود تا داده آموزشی و تست به آرایه نامپای برگردانده شود.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8, random_state=42)

X_train = X_train.to_numpy()
X_test = X_test.to_numpy()
y_test = y_test.to_numpy()
y_train = y_train.to_numpy()
```

پس از آن با چهار کرنل چبیشو، لگندر، گن بایر و جاکوبی چهار مدل ایجاد می‌کنیم و عمل برآش و پیش‌بینی را انجام می‌دهیم. پیاده‌سازی این چهار کرنل به همراه خروجی، به ترتیب در زیر آورده شده‌اند.

```

model = orsvm.Model(kernel="Chebyshev",order=7,T=0.85,form='r', c = 3000)

Weights, SupportVectors, Bias, KernelInstance = model.ModelFit(X_train,y_train)

***** 17/08/2022 18:14:09 *****
** OSVM kernel: Chebyshev
** Order: 7
** Fractional mode, transition : 0.85
** Average support vector determiner selected!
** sv threshold: 10^ -2

scores['*ORSVM_Chebyshev*'] = model.ModelPredict(X_test, y_test, Bias, KernelInstance)

** Accuracy score: 0.9125
** Classification Report:
      precision    recall  f1-score   support
-1       0.91     0.95     0.93     202
 1       0.91     0.85     0.88     118

accuracy                           0.91     320
macro avg       0.91     0.90     0.90     320
weighted avg    0.91     0.91     0.91     320

** Confusion Matrix:
[[192  10]
 [ 18 100]]

```

```
model2 = orsvm.Model(kernel="Legendre",order=9,T = 0.85,form='r', c = 2000)
Weights, SupportVectors, Bias, KernelInstance = model2.ModelFit(X_train,y_train)
```

***** 17/08/2022 18:14:14 *****

** OSVM kernel: Legendre
** Order: 9
** Fractional mode, transition : 0.85
** Avegage support vector determiner selected!
** sv threshold: 10^ -3

```
scores['*ORSVM_Legendre*'] = model2.ModelPredict(X_test, y_test, Bias, KernelInstance)
```

** Accuracy score: 0.896875
** Classification Report:

	precision	recall	f1-score	support
-1	0.89	0.96	0.92	202
1	0.91	0.80	0.85	118
accuracy			0.90	320
macro avg	0.90	0.88	0.89	320
weighted avg	0.90	0.90	0.90	320

** Confusion Matrix:

```
[[193  9]
 [ 24 94]]
```

```
model3 = orsvm.Model(kernel="Gegenbauer",order=8,param1=0.5,T=0.85,form='r', c = 2000)
Weights, SupportVectors, Bias, KernelInstance = model3.ModelFit(X_train,y_train)
```

***** 17/08/2022 18:14:32 *****

** OSVM kernel: Gegenbauer
** Order: 8
** Fractional mode, transition : 0.85
** Avegage support vector determiner selected!
** sv threshold: 10^ -1

```
scores['*ORSVM_Gegenbauer*'] = model3.ModelPredict(X_test, y_test, Bias, KernelInstance)
```

** Accuracy score: 0.878125
** Classification Report:

	precision	recall	f1-score	support
-1	0.88	0.94	0.91	202
1	0.88	0.77	0.82	118
accuracy			0.88	320
macro avg	0.88	0.86	0.87	320
weighted avg	0.88	0.88	0.88	320

** Confusion Matrix:

```
[[190 12]
 [ 27 91]]
```

```

model4 = orsvm.Model(kernel="Jacobi",order=7,param1=-0.8,param2=0.2,T=0.85,noise=0.1,form='r', c = 2000)
Weights, SupportVectors, Bias, KernelInstance = model4.ModelFit(X_train,y_train)

***** 17/08/2022 18:14:42 *****
** OSVM kernel: Jacobi
** Order: 7
** Fractional mode, transition : 0.85
** Avegage support vector determiner selected!
** sv threshold: 10^-3

scores['*ORSVM_Jacobi*'] = model4.ModelPredict(X_test, y_test, Bias, KernelInstance)

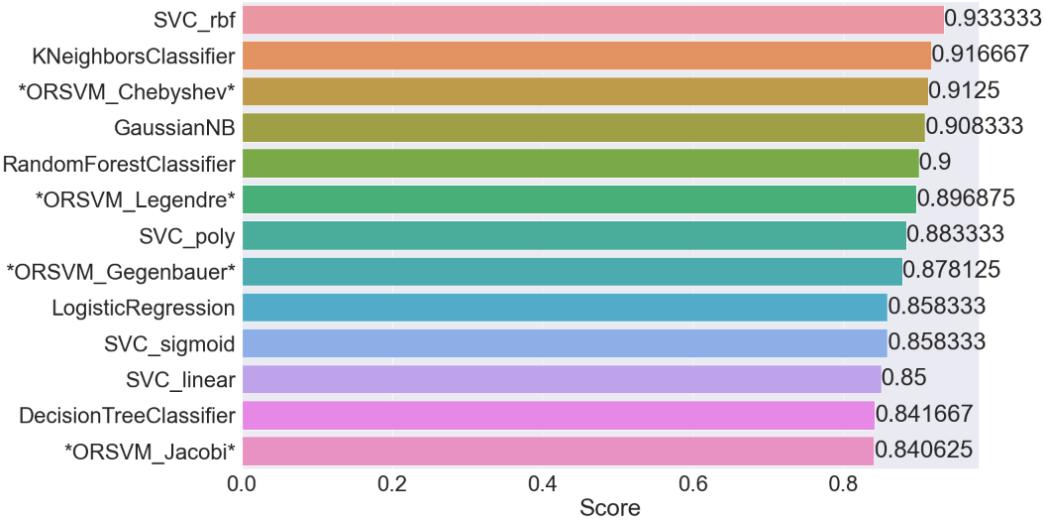
** Accuracy score: 0.840625
** Classification Report:
      precision    recall   f1-score   support
      -1       0.86     0.89     0.88     202
       1       0.80     0.76     0.78     118

      accuracy          0.84     320
      macro avg       0.83     0.82     0.83     320
  weighted avg       0.84     0.84     0.84     320

** Confusion Matrix:
[[179  23]
 [ 28  90]]

```

نتیجه مقایسه الگوریتم‌های فوق در نمودار زیر آورده شده‌اند.



که طبق نمودار فوق، بر اساس دقت مدل، کرنل آر-بی-اف ماشین بردار پشتیبان از همه بهتر کار کرد.

۳-۳-۴ طبقه‌بندی مجموعه داده تبلیغات بانکی

این مجموعه داده، داده‌های مرتبط با کمپین‌های بازاریابی مستقیم (تماس‌های تلفنی) یک موسسه بانکی پرتغالی است که این کمپین‌ها مبتنی بر تماس‌های تلفنی بود. هدف طبقه‌بندی این است که پیش‌بینی کند آیا مشتری یک سپرده مدت دار را ثبت می‌کند یا خیر. این بار نیز، با الگوریتم‌های طبقه‌بندی پیاده‌سازی شده در کتابخانه سایکیت‌لرن، مدل‌سازی خواهیم کرد. همچنین در این مجموعه داده، ستون ایگرگ، ستون خروجی و باقی ستون‌های ویژگی اند.

۱-۳-۳-۴ فرآیند مدل‌سازی سایکیت‌لرن روی مجموعه داده تبلیغات بانکی

ابتدا کتابخانه‌های مورد استفاده در پیاده‌سازی را اضافه می‌کنیم.

```
#import the necessary packages
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.model_selection import train_test_split, KFold, cross_val_score
import warnings
warnings.filterwarnings(action='ignore')
%matplotlib inline
```

مرحله اول: بارگذاری داده

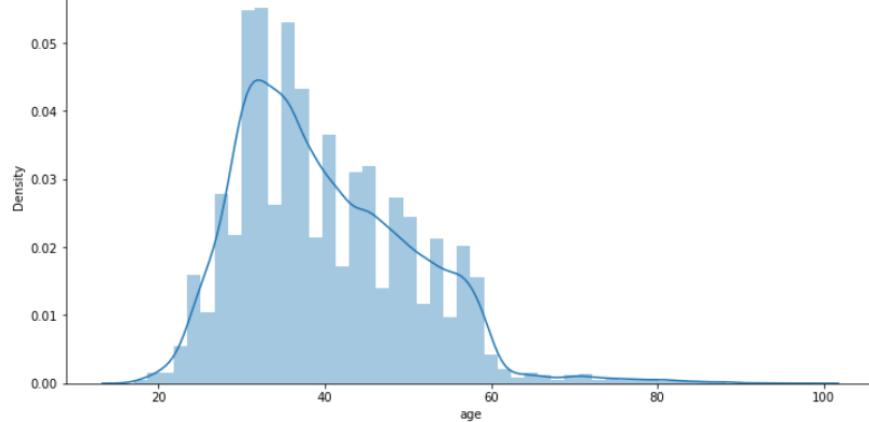
سپس داده مورد استفاده را اضافه می‌کنیم و سعی می‌کنیم با دستوراتی مشابه قبل، اطلاعاتی درمورد داده بدست آوریم. به علت تکراری بودن دستورات در اینجا ذکر نمی‌کنیم. اما پس از بررسی متوجه شدیم که این مجموعه داده مقدار گمشده‌ای ندارد.

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	er
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	
...
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	1	999	0	nonexistent	
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	nonexistent	
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	...	2	999	0	nonexistent	
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	...	1	999	0	nonexistent	
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	3	999	1	failure	

بعد از آن نوبت به مصورسازی داده می‌رسد در اینجا با دستورات مختلف مصورسازی‌هایی انجام دادیم. برای مثال، نمودار سن و سال افراد را رسم کردیم تا ببینیم بیشتر افراد در چه بازه سنی اند. نتیجه این دستور در شکل زیر آمده است. همانطور که در تصویر دیده می‌شود بیشتر افراد در بازه بیست تا چهل سال قرار دارند.

```
#Age distplot
fig, ax = plt.subplots(1, 1, figsize = (12, 6))
sns.distplot(dataset.age)
```

<AxesSubplot:xlabel='age', ylabel='Density'>



مرحله دوم: پیش‌پردازش داده

همانطور که در بخش قبل توضیح دادیم، زمانی که با داده کیفی روپرتو هستیم باید آن را به نوعی به مقادیر عددی تبدیل کنیم. در بخش قبل این کار را با یکی از توابع کتابخانه پانداز آنجام دادیم. اما داده‌هایی که ترتیب آن‌ها اهمیت داشته باشند باید با احتیاط و توجه به ماهیتشون به مقادیر عددی تبدیل بشن. به عنوان اندازه لباس ممکنه به صورت کوچک، متوسط و یا بزرگ در داده‌ها وجود داشته باشند و این مورد احتمالاً در الگوریتم یادگیری ماشین ما تأثیر گذار خواهد بود. این بار برای اینکه با نوع دیگر تبدیل داده هم مواجه شده باشیم از رمزگذاری برچسب استفاده کردیم. البته، ماهیت داده اینجا به گونه‌ای است که مشابه قبل می‌توان از تابع مورد استفاده در بخش قبل هم استفاده کرد.

sing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	0	1	6	1	...	1	999	0	1	1.1	93.994	-36.4	4.857	5191.0	0
0	0	1	6	1	...	1	999	0	1	1.1	93.994	-36.4	4.857	5191.0	0
2	0	1	6	1	...	1	999	0	1	1.1	93.994	-36.4	4.857	5191.0	0
0	0	1	6	1	...	1	999	0	1	1.1	93.994	-36.4	4.857	5191.0	0
0	2	1	6	1	...	1	999	0	1	1.1	93.994	-36.4	4.857	5191.0	0
...
2	0	0	7	0	...	1	999	0	1	-1.1	94.767	-50.8	1.028	4963.6	1
0	0	0	7	0	...	1	999	0	1	-1.1	94.767	-50.8	1.028	4963.6	0
2	0	0	7	0	...	2	999	0	1	-1.1	94.767	-50.8	1.028	4963.6	0
0	0	0	7	0	...	1	999	0	1	-1.1	94.767	-50.8	1.028	4963.6	1
2	0	0	7	0	...	3	999	1	0	-1.1	94.767	-50.8	1.028	4963.6	0

و پس از آن نیز ستون‌های ویژگی و خروجی را با دستورات زیر، از هم جدا می‌کنیم.

```
#feature variable  
X = dataset.iloc[:, :-1]  
X
```

```
#Target variable  
y = dataset['y']  
y
```

مرحله سوم: تقسیم داده

در این مرحله داده آزمایشی و آموزشی را از هم جدا می‌کنیم

¹ Pd.get_dummies()

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

و در ادامه مرحله پیش‌پردازش عمل استاندارد سازی را انجام می‌دهیم.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

مرحله چهارم: آموزش و تست مدل

در اینجا نیز، به ترتیب مدل‌ها را بوجود آورده، مجدداً آرایه‌ای خالی بوجود می‌آوریم تا نتایج ارزیابی را داخل آن قرار ببریزیم. پس از آن عمل برآذش مدل را روی داده آموزشی انجام می‌دهیم و پیش‌بینی مدل از داده آزمایشی را داخل متغیری ریخته تا نتایج را ارزیابی کنیم. اعمال فوق به ترتیب در زیر آورده شده‌اند.

```
models = [
    'LR': LogisticRegression(),
    'KNN': KNeighborsClassifier(),
    'DT': DecisionTreeClassifier(),
    'NB': GaussianNB(),
    'RF': RandomForestClassifier()
]
```

```
scores = {}

for i, model in models.items():
    print("model: ", model)

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

    print(classification_report(y_test, y_pred))
    print("-" * 50)
    scores[i] = model.score(X_test, y_test)
```

نتایج پیاده‌سازی فوق به صورت زیر بود

```
model: LogisticRegression()
      precision    recall  f1-score   support

         0       0.93     0.97    0.95    10968
         1       0.66     0.41    0.50    1389

    accuracy                           0.91    12357
   macro avg       0.80     0.69    0.73    12357
weighted avg       0.90     0.91    0.90    12357


model: KNeighborsClassifier()
      precision    recall  f1-score   support

         0       0.93     0.97    0.95    10968
         1       0.60     0.40    0.48    1389

    accuracy                           0.90    12357
   macro avg       0.76     0.68    0.71    12357
weighted avg       0.89     0.90    0.89    12357

-----
model: DecisionTreeClassifier()
      precision    recall  f1-score   support

         0       0.94     0.94    0.94    10968
         1       0.51     0.52    0.52    1389

    accuracy                           0.89    12357
   macro avg       0.73     0.73    0.73    12357
weighted avg       0.89     0.89    0.89    12357
```

model: GaussianNB()		precision	recall	f1-score	support
	0	0.95	0.88	0.91	10968
	1	0.39	0.61	0.48	1389
accuracy				0.85	12357
macro avg		0.67	0.75	0.70	12357
weighted avg		0.88	0.85	0.86	12357
model: RandomForestClassifier()		precision	recall	f1-score	support
	0	0.94	0.97	0.95	10968
	1	0.65	0.50	0.57	1389
accuracy				0.91	12357
macro avg		0.79	0.74	0.76	12357
weighted avg		0.91	0.91	0.91	12357

حال نوبت به پیاده‌سازی با الگوریتم ماشین بردار پشتیبان می‌رسد. مجدداً فرآیند ساخت مدل، برازش و پیش‌بینی را انجام می‌دهیم و نتایج پیش‌بینی را داخل متغیری نگه داشته تا بعداً با هم مقایسه کنیم.

```

kernels = {'SVC_linear':'linear','SVC_poly':'poly','SVC_rbf':'rbf','SVC_sigmoid':'sigmoid'}
```

```

svc = {}
```

```

def SVM_fit_score(kernels, X_train, X_test, y_train, y_test):
    for i, kernel in kernels.items():
        print('kernel: ', kernel)
        svc[i] = SVC(kernel=kernel).fit(X_train, y_train)
        y_pred = svc[i].predict(X_test)
        print(classification_report(y_test , y_pred))
        print('-'*60)
        scores[i] = svc[i].score(X_test,y_test)
```

نتایج پیاده‌سازی نیز به صورت زیر بود.

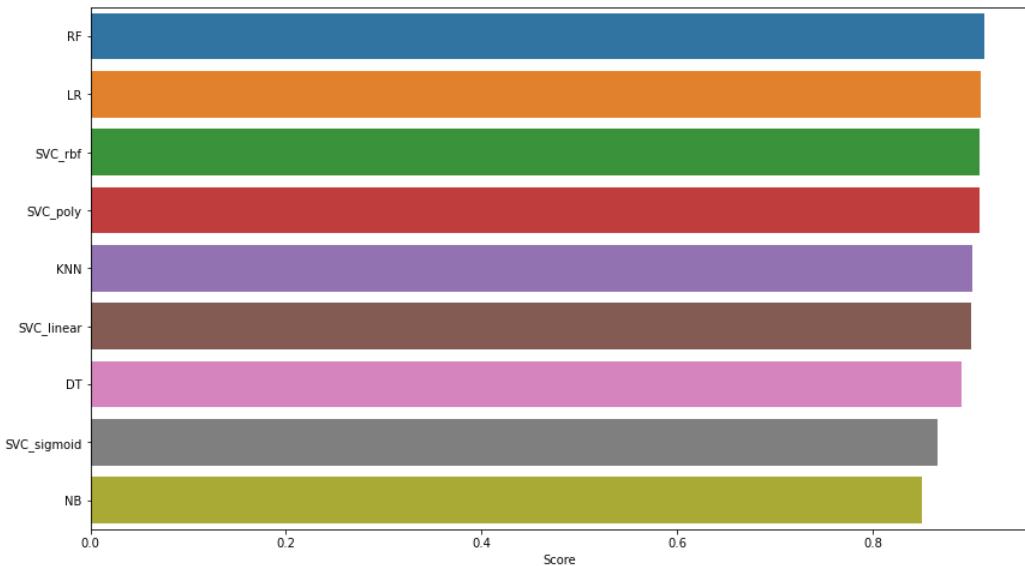
	kernel: linear			
	precision	recall	f1-score	support
0	0.91	0.98	0.95	10968
1	0.63	0.28	0.38	1389
accuracy			0.90	12357
macro avg	0.77	0.63	0.67	12357
weighted avg	0.88	0.90	0.88	12357

	kernel: poly			
	precision	recall	f1-score	support
0	0.92	0.98	0.95	10968
1	0.68	0.37	0.48	1389
accuracy			0.91	12357
macro avg	0.80	0.67	0.71	12357
weighted avg	0.90	0.91	0.90	12357

	kernel: rbf			
	precision	recall	f1-score	support
0	0.93	0.98	0.95	10968
1	0.67	0.38	0.49	1389
accuracy			0.91	12357
macro avg	0.80	0.68	0.72	12357
weighted avg	0.90	0.91	0.90	12357

	kernel: sigmoid			
	precision	recall	f1-score	support
0	0.92	0.93	0.92	10968
1	0.41	0.40	0.40	1389
accuracy			0.87	12357
macro avg	0.66	0.66	0.66	12357
weighted avg	0.87	0.87	0.87	12357

در پایان نیز مجددا، نتایج تمام این پیاده‌سازی‌ها را داخل آرایه‌ای قرار داده و با هم مقایسه کردیم؛ معیار ارزیابی را در اینجا، دقیق مدل اقرار دادیم که این نتایج به صورت زیر بودند.



و همانطور که از نمودار بالا مشخص است، کرنل تابع پایه شعاعی ماشین بردار پشتیبان بهتر از همه کار کرد اما به صورت کلی، نتایج بسیار به هم نزدیک بودند.

۴-۳-۴ طبقه‌بندی مجموعه داده مسافران کشتی تایتانیک

این مجموعه داده نیز شامل دو مجموعه آموزشی و مجموعه تست مربوط به مسافران کشتی تایتانیک می‌باشد. مجموعه آموزشی باید برای ساخت مدل‌های یادگیری ماشین استفاده شود. مجموعه دوم به عنوان داده آزمایشی باید برای مشاهده عملکر مدل بر روی داده دیده نشده استفاده شود؛ به این معنی که در این مجموعه داده، ستونی به عنوان خروجی نداریم. در این طبقه‌بند، باید براساس یک سری از ویژگی‌ها پیش‌بینی کنیم که آیا این افراد از غرق شدن کشتی تایتانیک جان سالم به در برده اند یا خیر. مجددا این پیاده‌سازی را هم با کرنل‌های ماشین بردار پشتیبان و هم با کرنل‌های پکیج او-آر-اس وی ام انجام خواهیم داد. در ادامه فرآیند مدل‌سازی آورده شده است.

¹ Accuracy

۱-۴-۳-۴ فرآیند مدلسازی سایکیتلرن روی مجموعه داده مسافران کشتی تایتانیک

مرحله اول: بارگذاری داده

دقیقا مشابه قبل، پس از اضافه کردن کتابخانه، داده مورد نظر را بارگذاری می‌کنیم؛ تنها تفاوت این مجموعه داده این است این بار دو مجموعه داده آموزشی و آزمایشی را باید بارگذاری کنیم.

```
train_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')
```

همچنین مشابه روند قبل، سعی می‌کنیم اطلاعاتی پیرامون داده بدست آوریم. برای مثال اطلاعات عددی بدست آوریم؛ تعداد مقادیر گمشده در هر ستون را بدست آوریم یا مصورسازی انجام دهیم. این موارد در زیر آورده شده اند که در ادامه به توضیح آن‌ها خواهیم پرداخت.

اولین نکته مهم در این مجموعه داده، این است که برخلاف نمونه‌های قبلی در اینجا با سلول‌های گمشده در هر ستون مواجهیم و باید سعی کنیم بهترین عمل را در مواجهه با آن‌ها انجام دهیم. در مرحله پیش‌پردازش، به این قسمت خواهیم پرداخت.

```
#number of null values in each column
train_data.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked        2
dtype: int64
```

```
#number of null values in each column
test_data.isnull().sum()
```

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age            86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked        0
dtype: int64
```

سپس هم درمورد داده کیفی و هم برای داده کمی سعی می‌کنیم اطلاعاتی بدست آوریم؛ برای مثال پس از اجرای دستور فوق، اطلاعاتی از قبیل تعداد مقادیر یکتا، تعداد کل، بیشترین تکرار را برای ستون‌های دارای مقادیر کیفی بدست آوردهیم. برای مقادیر عددی و کیفی هر دو مجموعه داده آموزشی و آزمایشی همین کار را تکرار می‌کنیم؛ اما برای جلوگیری از طولانی تر شدن گزارش، تنها به دستور مربوط به مقادیر کیفی داده آموزشی بسته می‌کنیم.

#Categorical columns					
	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Braund, Mr. Owen Harris	male	347082	B96 B98	S
freq	1	577	7	4	644

مرحله دوم: پیش‌پردازش داده

در مرحله قبل متوجه شدیم که در برخی از این ستون‌ها، مقادیر گمشده‌ای وجود دارد. در این مرحله، نوبت به حساب‌رسی به داده‌های گمشده می‌رسد؛ یعنی یا با مقادیری جایگزین شوند و یا حذف شوند. بعضی ستون‌هایی که در دستور زیر آمده اند را حذف می‌کنیم. این به این علت است که این ستون‌ها ارتباطی با مدل و پیش‌بینی ما ندارند و حتی در صورت باقی ماندن ممکن است باعث ایجاد خطا در مدل شوند. برای مثال نام مسافران، آیدی آن‌ها، شماره بلیط یا کابین تاثیری در مدل ندارد و برای همین حذف می‌شوند. گاهی اوقات هم آن چنان تعداد مقادیر گمشده در یک ستون زیاد است که حذف آن انتخاب بهتری نسبت به باقی ماندن آن می‌باشد. در ادامه، برای بعضی از مقادیر گمشده مانند مقادیر گمشده در ستون سن مقدار میانگین را جایگزین کردیم و برای بعضی نیز مانند ستون مکان سوار شدن، سطرهای حاوی مقادیر گمشده را حذف کردیم. کد مربوط به این دو دستور، به ترتیب در دو تصویر زیر آمده است.

```
for data in dataset:
    data['Age'].fillna(data['Age'].median(), inplace = True)
    data['Fare'].fillna(data['Fare'].median(), inplace = True)
    data['Embarked'].fillna(data['Embarked'].mode()[0], inplace = True)
```

همچنین دستور حذف این ستون‌ها در تصویر زیر آمده است. عمل زیر را برای داده‌های آزمایشی نیز انجام می‌دهیم که برای جلوگیری از تکرار، تنها حذف ستون‌های داده آموزشی آمده است.

```
train.drop(['Name', 'Ticket', 'PassengerId', 'Cabin'], axis= 1, inplace= True)
```

پس از رسیدگی به مقادیر گمشده در داده تست و داده آموزشی نوبت به داده‌های کیفی می‌رسد که داده‌های عددی نیستند. با دستور زیر، ستون‌های مناسبی برای آن‌ها بوجود آورده و تبدیل به مقادیر عددی می‌کنیم.

X = pd.get_dummies(X, drop_first=True)									
	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embarked_S
0	0	3	22.000000	1	0	7.2500	1	0	1
1	1	1	38.000000	1	0	71.2833	0	0	0
2	1	3	26.000000	0	0	7.9250	0	0	1
3	1	1	35.000000	1	0	53.1000	0	0	1
4	0	3	35.000000	0	0	8.0500	1	0	1
...
886	0	2	27.000000	0	0	13.0000	1	0	1
887	1	1	19.000000	0	0	30.0000	0	0	1
888	0	3	29.699118	1	2	23.4500	0	0	1
889	1	1	26.000000	0	0	30.0000	1	0	0
890	0	3	32.000000	0	0	7.7500	1	1	0

889 rows × 9 columns

بعد از آن ستون برچسب را از ستون‌های ویژگی را از هم جدا می‌کنیم.

```
#feature variable
X = train.iloc[:, 1:]

#Target variable
y = train['Survived']
```

مرحله سوم: تقسیم داده

در این مرحله نیز مشابه قبل، داده تست و آموزشی را از هم جدا می‌کنیم و پس از آن عمل استاندارسازی را انجام می‌دهیم.

```
scaler = StandardScaler()

scaled_X_train = scaler.fit_transform(X_train)
scaled_X_test = scaler.transform(X_test)
```

مرحله چهارم: آموزش مدل

مرحله ایجاد، برآش و پیش‌بینی با کمک الگوریتم‌های رگرسیون لجیستیک، کا-همساخه نزدیک، درخت تصمیم، بیز ساده، جنگل تصادفی و ماشین بردار پشتیبان را مجدداً انجام می‌دهیم که این روند دقیقاً مشابه قبل می‌باشد. برای همین صرفاً به قرار دادن نتیجه آن‌ها اکتفا می‌کنیم.

```
model: LogisticRegression()
      precision    recall  f1-score   support

       -1      0.82     0.87     0.85     157
        1      0.80     0.73     0.76     111

   accuracy                           0.81      268
  macro avg      0.81     0.80     0.80      268
weighted avg      0.81     0.81     0.81      268
-----
```

```
model: KNeighborsClassifier()
      precision    recall  f1-score   support

       -1      0.78     0.88     0.83     157
        1      0.79     0.66     0.72     111

   accuracy                           0.79      268
  macro avg      0.79     0.77     0.77      268
weighted avg      0.79     0.79     0.78      268
-----
```

```
model: DecisionTreeClassifier()
      precision    recall  f1-score   support

       -1      0.80     0.82     0.81     157
        1      0.73     0.70     0.72     111

   accuracy                           0.77      268
  macro avg      0.76     0.76     0.76      268
weighted avg      0.77     0.77     0.77      268
-----
```

```
model: GaussianNB()
      precision    recall  f1-score   support

       -1      0.84     0.82     0.83     157
        1      0.75     0.77     0.76     111

   accuracy                           0.80      268
  macro avg      0.79     0.80     0.79      268
weighted avg      0.80     0.80     0.80      268
-----
```

```
-----  
model: RandomForestClassifier()  
      precision    recall   f1-score   support  
  
      -1       0.81      0.82      0.81      157  
       1       0.73      0.72      0.73      111  
  
accuracy                           0.78      268  
macro avg       0.77      0.77      0.77      268  
weighted avg     0.78      0.78      0.78      268  
  
-----  
  
kernel: linear  
      precision    recall   f1-score   support  
  
      -1       0.80      0.85      0.83      157  
       1       0.77      0.70      0.74      111  
  
accuracy                           0.79      268  
macro avg       0.79      0.78      0.78      268  
weighted avg     0.79      0.79      0.79      268  
  
-----  
  
kernel: poly  
      precision    recall   f1-score   support  
  
      -1       0.79      0.87      0.83      157  
       1       0.79      0.68      0.73      111  
  
accuracy                           0.79      268  
macro avg       0.79      0.77      0.78      268  
weighted avg     0.79      0.79      0.79      268  
  
-----  
  
kernel: rbf  
      precision    recall   f1-score   support  
  
      -1       0.80      0.93      0.86      157  
       1       0.87      0.67      0.76      111  
  
accuracy                           0.82      268  
macro avg       0.83      0.80      0.81      268  
weighted avg     0.83      0.82      0.82      268
```

kernel: sigmoid		precision	recall	f1-score	support
-1	0.77	0.80	0.78	157	
1	0.70	0.66	0.68	111	
accuracy				0.74	268
macro avg		0.73	0.73	0.73	268
weighted avg		0.74	0.74	0.74	268

۴-۳-۲-۴ فرآیند مدلسازی با پکیج او-آر-اس وی ام روی مجموعه داده مسافران کشتی تایتانیک برای این مجموعه داده نیز با کمک پکیج او-آر-اس وی ام، عمل طبقه‌بندی را انجام می‌دهیم. همانطور که در مجموعه داده دوم هم ذکر کردیم، درصد داده‌های آزمایشی باید برای این پکیج تغییر کند و داده‌های آموزشی و آزمایشی نیز به آرایه نامپای تبدیل شوند.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8, random_state=42)

X_train = X_train.to_numpy()
X_test = X_test.to_numpy()
y_test = y_test.to_numpy()
y_train = y_train.to_numpy()
```

مدلسازی با چهار کرنل این پکیج به همراه خروجی در زیر آورده شده است.

```

model = orsvm.Model(kernel="Chebyshev",order=6,T=0.15,form='r', c = 3000)

Weights, SupportVectors, Bias, KernelInstance = model.ModelFit(X_train,y_train)

***** 20/08/2022 11:48:02 *****
** OSVM kernel: Chebyshev
** Order: 6
** Fractional mode, transition : 0.15
** Avegage support vector determiner selected!
** sv threshold: 10^ -1

scores['*ORSVM_Chebyshev*'] = model.ModelPredict(X_test, y_test, Bias, KernelInstance)

** Accuracy score: 0.729312762973352
** Classification Report:
      precision    recall   f1-score   support
      -1       0.81     0.72     0.77      436
        1       0.63     0.74     0.68      277

      accuracy          0.73      713
      macro avg       0.72     0.73     0.72      713
  weighted avg       0.74     0.73     0.73      713

** Confusion Matrix:
[[316 120]
 [ 73 204]]
```

```

model2 = orsvm.Model(kernel="Legendre",order=5,T=0.15, c = 3000)
Weights, SupportVectors, Bias, KernelInstance = model2.ModelFit(X_train,y_train)

***** 20/08/2022 11:48:20 *****
** OSVM kernel: Legendre
** Order: 5
** Fractional mode, transition : 0.15
** Avegage support vector determiner selected!
** sv threshold: 10^ -4

scores['*ORSVM_Legendre*'] = model2.ModelPredict(X_test, y_test, Bias, KernelInstance)

** Accuracy score: 0.7166900420757363
** Classification Report:
      precision    recall   f1-score   support
      -1       0.75     0.80     0.78      436
        1       0.65     0.58     0.61      277

      accuracy          0.72      713
      macro avg       0.70     0.69     0.70      713
  weighted avg       0.71     0.72     0.71      713

** Confusion Matrix:
[[350  86]
 [116 161]]
```

```

model3 = orsvm.Model(kernel="Gegenbauer",order=6,param1=0.5,T=0.15, c = 20000)
Weights, SupportVectors, Bias, KernelInstance = model3.ModelFit(X_train,y_train)

***** 20/08/2022 11:48:30 *****
** OSVM kernel: Gegenbauer
** Order: 6
** Fractional mode, transition : 0.15
** Avegage support vector determiner selected!
** sv threshold: 10^ -3

scores['*ORSVM_Gegenbauer*'] = model3.ModelPredict(X_test, y_test, Bias, KernelInstance)

** Accuracy score: 0.758765778401122
** Classification Report:
      precision    recall   f1-score   support
      -1       0.81     0.80     0.80      436
        1       0.69     0.70     0.69      277

      accuracy                           0.76      713
      macro avg       0.75     0.75     0.75      713
  weighted avg       0.76     0.76     0.76      713

** Confusion Matrix:
[[348  88]
 [ 84 193]]

```

```

model4 = orsvm.Model(kernel="Jacobi",order=6,param1=0.8,param2=0.2,T=0.15,nois=0.1, c = 30000)
Weights, SupportVectors, Bias, KernelInstance = model4.ModelFit(X_train,y_train)

***** 20/08/2022 11:48:52 *****
** OSVM kernel: Jacobi
** Order: 6
** Fractional mode, transition : 0.15
** Avegage support vector determiner selected!
** sv threshold: 10^ -2

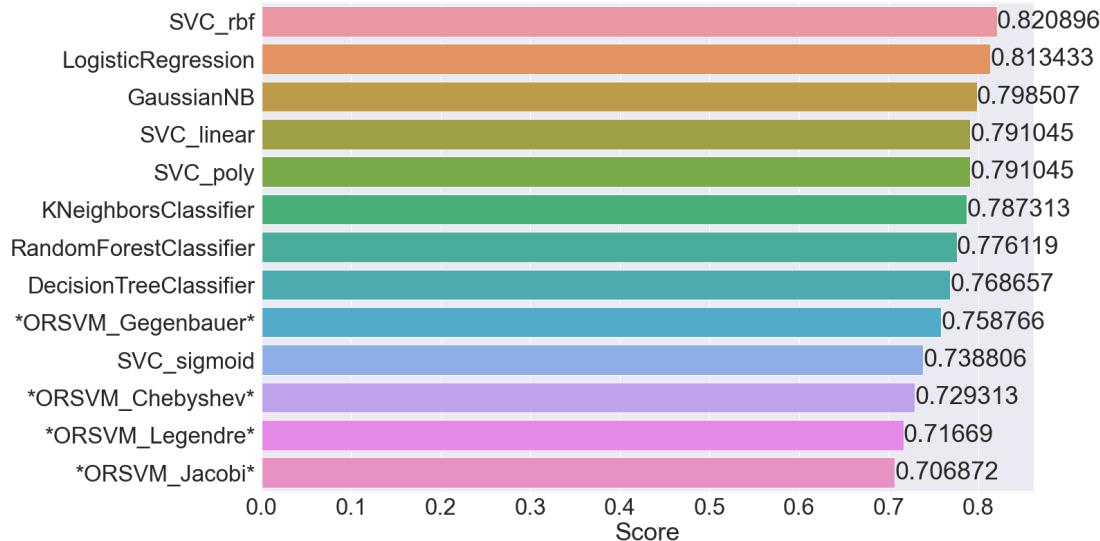
scores['*ORSVM_Jacobi*'] = model4.ModelPredict(X_test, y_test, Bias, KernelInstance)

** Accuracy score: 0.7068723702664796
** Classification Report:
      precision    recall   f1-score   support
      -1       0.76     0.75     0.76      436
        1       0.62     0.63     0.63      277

      accuracy                           0.71      713
      macro avg       0.69     0.69     0.69      713
  weighted avg       0.71     0.71     0.71      713

```

در انتها با معیار دقت، نتایج تمام این الگوریتم‌ها را در کنار یکدیگر آورده‌یم که در نمودار زیر آورده شده است. همانطور که از این نمودار مشخص است همچنان بهترین عملکرد را الگوریتم ماشین بردار پشتیبان با کرنل تابع پایه شعاعی داشته است.



۴-۳-۵ طبقه‌بندی مجموعه داده تخمین قیمت خانه‌های سنگاپور

این مجموعه داده حاوی بیش از ۸۰۰۰۰ رکورد تراکنش آپارتمان‌های هیئت توسعه مسکن سنگاپور است. در این مجموعه داده، برخی از متغیرها تاثیر بیشتری نسبت به بقیه دارند. این مجموعه داده شامل ۲۸ ستون کلی است اما فقط تعدادی از آن‌ها برای رگرسیون استفاده می‌شوند. این موارد شامل سال، شهر، تعداد اتاق‌ها، طول جغرافیایی، عرض جغرافیایی، مساحت طبقاتی، اجاره شروع، تعداد سال‌های اجاره باقی‌مانده، قیمت هر متر مربع به عنوان ستون‌های ویژگی و موثر بر قیمت و قیمت فروش به عنوان ستون خروجی است. توضیحات پیاده‌سازی در زیر آورده شده‌اند.

۴-۳-۶ فرآیند مدلسازی سایکیت‌لرن روی مجموعه داده خانه‌های سنگاپور

جمع آوری داده

اضافه کردن کتابخانه‌های مورد نظر:

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso,
ElasticNet
from sklearn.preprocessing import LabelEncoder, PolynomialFeatures
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error
import warnings
warnings.filterwarnings(action='ignore')

%matplotlib inline
```

```
dataset = pd.read_csv('ALL Prices 1990-2021 mar.csv')
dataset
```

	month	year	town	town_dummy	flat_type	block	street_name	address	latitude	longitude	...	price_psm_yearly	Core CPI	price cpi_adj
0	1990-01	1990	ANG MO KIO		2 4 ROOM	308	ANG MO KIO AVE 1	308 ANG MO KIO AVE 1 SINGAPORE	1.365485	103.844025	...	10.997442	61.59	139633.05730
1	1990-01	1990	ANG MO KIO		2 3 ROOM	308	ANG MO KIO AVE 1	308 ANG MO KIO AVE 1 SINGAPORE	1.365485	103.844025	...	8.464849	61.59	95794.77188
2	1990-01	1990	ANG MO KIO		2 3 ROOM	216	ANG MO KIO AVE 1	216 ANG MO KIO AVE 1 SINGAPORE	1.366272	103.841465	...	7.606769	61.59	76635.81750
3	1990-01	1990	ANG MO KIO		2 3 ROOM	308	ANG MO KIO AVE 1	308 ANG MO KIO AVE 1 SINGAPORE	1.365485	103.844025	...	8.287972	61.59	76635.81750
4	1990-01	1990	ANG MO KIO		2 4 ROOM	211	ANG MO KIO AVE 3	211 ANG MO KIO AVE 3 SINGAPORE	1.369226	103.841652	...	11.484353	61.59	129891.21610
...
840913	2021-03	2021	YISHUN		5 3 ROOM	110	YISHUN RING RD	110 YISHUN RING RD SINGAPORE	1.433340	103.829168	...	66.650391	100.40	271912.35060
840914	2021-03	2021	YISHUN		5 3 ROOM	712	YISHUN AVE 5	712 YISHUN AVE 5 SINGAPORE	1.430248	103.828862	...	63.567362	100.40	266932.27090
840915	2021-03	2021	YISHUN		5 2 ROOM	424B	YISHUN AVE 11	424B YISHUN AVE 11 SINGAPORE	1.423215	103.848264	...	59.482956	100.40	258964.14340
840916	2021-03	2021	YISHUN		5 2 ROOM	459	YISHUN AVE 11	459 YISHUN AVE 11 SINGAPORE	1.421059	103.846357	...	57.234432	100.40	249003.98410
840917	2021-03	2021	YISHUN		5 2 ROOM	459	YISHUN AVE 11	459 YISHUN AVE 11 SINGAPORE	1.421059	103.846357	...	48.076923	100.40	209163.34660

اطلاعات داده از جمله ویژگی ها، تعداد داده های معلوم در هر ستون و نوع آن ها:

```

shape = dataset.shape
print('Dataset Shape:', shape)

Dataset Shape: (840918, 28)

Info of Dataset
dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 840918 entries, 0 to 840917
Data columns (total 28 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   month            840918 non-null   object  
 1   year             840918 non-null   int64  
 2   town              840918 non-null   object  
 3   town_dummy        840918 non-null   int64  
 4   flat_type         840918 non-null   object  
 5   block             840918 non-null   object  
 6   street_name       840918 non-null   object  
 7   address            840918 non-null   object  
 8   latitude           840918 non-null   float64
 9   longitude          840918 non-null   float64
 10  storey_range      840918 non-null   object  
 11  storey            840918 non-null   int64  
 12  area_sqm          840918 non-null   float64
 13  flat_model         840918 non-null   object  
 14  lease_start        840918 non-null   int64  
 15  lease_rem          840918 non-null   int64  
 16  resale_price       840918 non-null   float64
 17  price_psm          840918 non-null   float64
 18  price_psm_yearly  840918 non-null   float64
 19  Core CPI           840918 non-null   float64
 20  price_cpi_adj     840918 non-null   float64
 21  price_psm_cpi_adj 840918 non-null   float64
 22  bala_lease_pct    840918 non-null   float64
 23  price_lease_adj_implied 840918 non-null   float64
 24  price_psm_lease_adj_implied 840918 non-null   float64
 25  price_cpi_lease_adj_implied 840918 non-null   float64
 26  price_psm_cpi_lease_adj_implied 840918 non-null   float64
 27  year_gni           840918 non-null   int64  
dtypes: float64(14), int64(6), object(8)
memory usage: 179.6+ MB

```

داده گمشده‌ای هم نداشتیم

```

#number of null values in each column
dataset.isnull().sum()

month 0
year 0
town 0
town_dummy 0
flat_type 0
block 0
street_name 0
address 0
latitude 0
longitude 0
storey_range 0
storey 0
area_sqm 0
flat_model 0
lease_start 0
lease_rem 0
resale_price 0
price_psm 0
price_psm_yearly 0
Core CPI 0
price_cpi_adj 0
price_psm_cpi_adj 0
bala_lease_pct 0
price_lease_adj_implied 0
price_psm_lease_adj_implied 0
price_cpi_lease_adj_implied 0
price_psm_cpi_lease_adj_implied 0
year_gni 0
dtype: int64

```

اطلاعات درمورد داده های کیفی و کمی:

اطلاعاتی از قبیل تعداد مقادیر یکتا، تعداد کل، بیشترین تکرار را برای ستون های دارای مقادیر کیفی بدست آوردهیم. برای مقادیر عددی و کیفی هر دو مجموعه داده آموزشی و آزمایشی همین کار را تکرار می کنیم.

`dataset.describe()`

	year	town_dummy	latitude	longitude	storey	area_sqm	lease_start	lease_rem	resale_price	price_p
count	840918.000000	840918.000000	840918.000000	840918.000000	840918.000000	840918.000000	840918.000000	840918.000000	8.409180e+05	840918.0000
mean	2004.512196	3.892261	1.361102	103.839049	7.567124	95.608704	1987.313952	81.780300	2.968002e+05	3062.2329
std	8.135802	1.403197	0.041576	0.073889	4.657776	26.026266	9.637392	9.959341	1.509764e+05	1299.3108
min	1990.000000	1.000000	1.266682	103.685206	2.000000	28.000000	1966.000000	45.000000	5.000000e+03	161.2903
25%	1998.000000	2.000000	1.333346	103.772844	5.000000	73.000000	1980.000000	75.000000	1.830000e+05	2238.0952
50%	2003.000000	4.000000	1.354131	103.842792	8.000000	93.000000	1986.000000	83.000000	2.780000e+05	2797.6190
75%	2010.000000	5.000000	1.380824	103.897752	11.000000	114.000000	1995.000000	90.000000	3.870000e+05	3818.1818
max	2021.000000	6.000000	1.457052	103.988777	50.000000	307.000000	2019.000000	101.000000	1.258000e+06	12762.2365

تعداد مقادیر مختلف برای ستون ویژگی:

```

#number of unique values in each column
dataset.nunique()

month                      375
year                       32
town                        27
town_dummy                  6
flat_type                   7
block                      2510
street_name                 572
address                     9381
latitude                    9145
longitude                   9137
storey_range                25
storey                      22
area_sqm                    209
flat_model                  22
lease_start                 54
lease_rem                   57
resale_price                8600
price_psm                   64743
price_psm_yearly            249007
Core CPI                     373
price_cpi_adj                178155
price_psm_cpi_adj            557518
bala_lease_pct                55
price_lease_adj_implied       61561
price_psm_lease_adj_implied   312711
price_cpi_lease_adj_implied    618477
price_psm_cpi_lease_adj_implied 761810
year_gni                     32
dtype: int64

```

تجسم داده‌ها:

Visualizing Data

Count plot:

```

fg, ax = plt.subplots(1 , 2,figsize=(25,8))

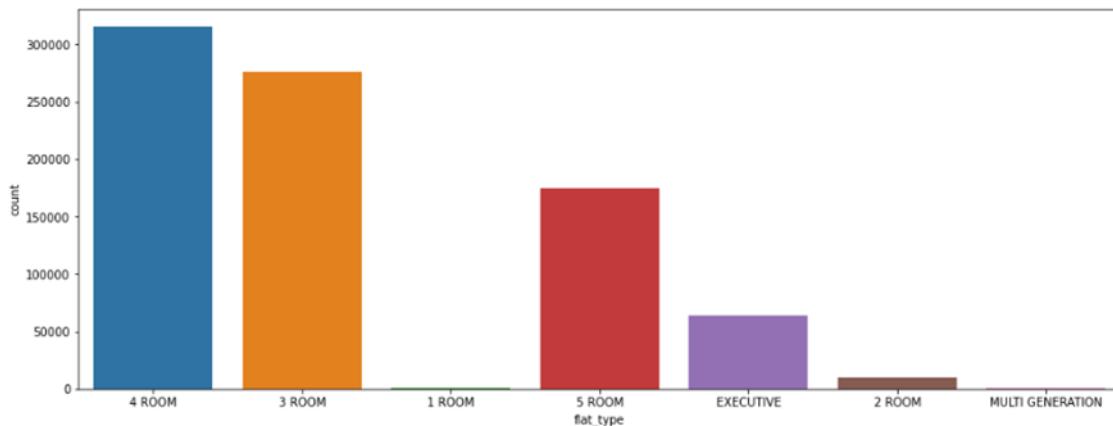
sns.countplot(data = dataset, x = 'year', ax = ax[0]);
sns.countplot(data = dataset, x = 'month', ax = ax[1]);

```

```

fig = plt.figure()
fig.set_size_inches(16, 6)
sns.countplot(data = dataset, x = 'flat_type');

```



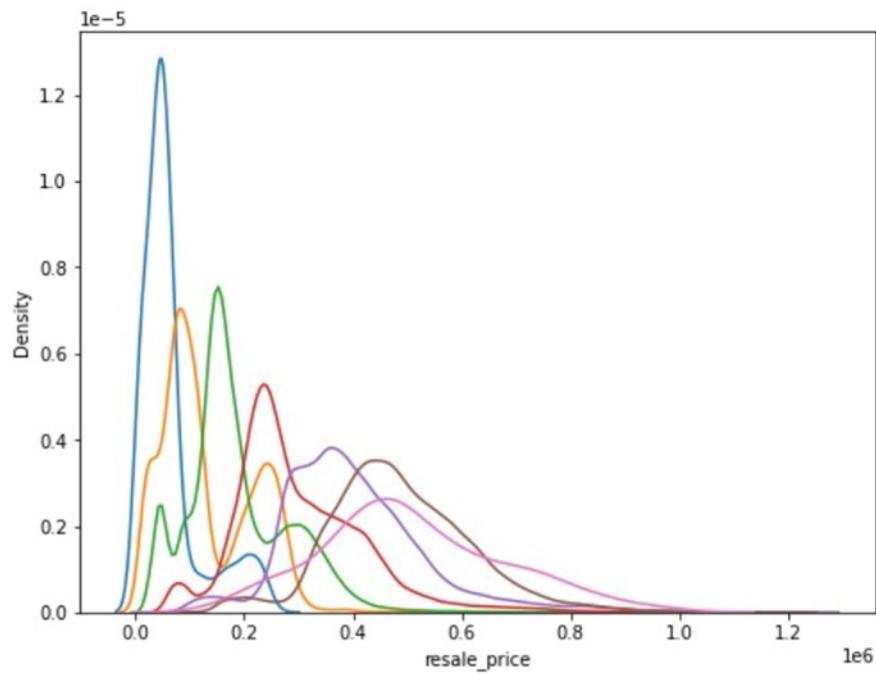
نمودار تابع چگالی احتمال قیمت خانه برای تعداد اتاق‌های مختلف:

```

fig = plt.figure()
fig.set_size_inches(8, 6)

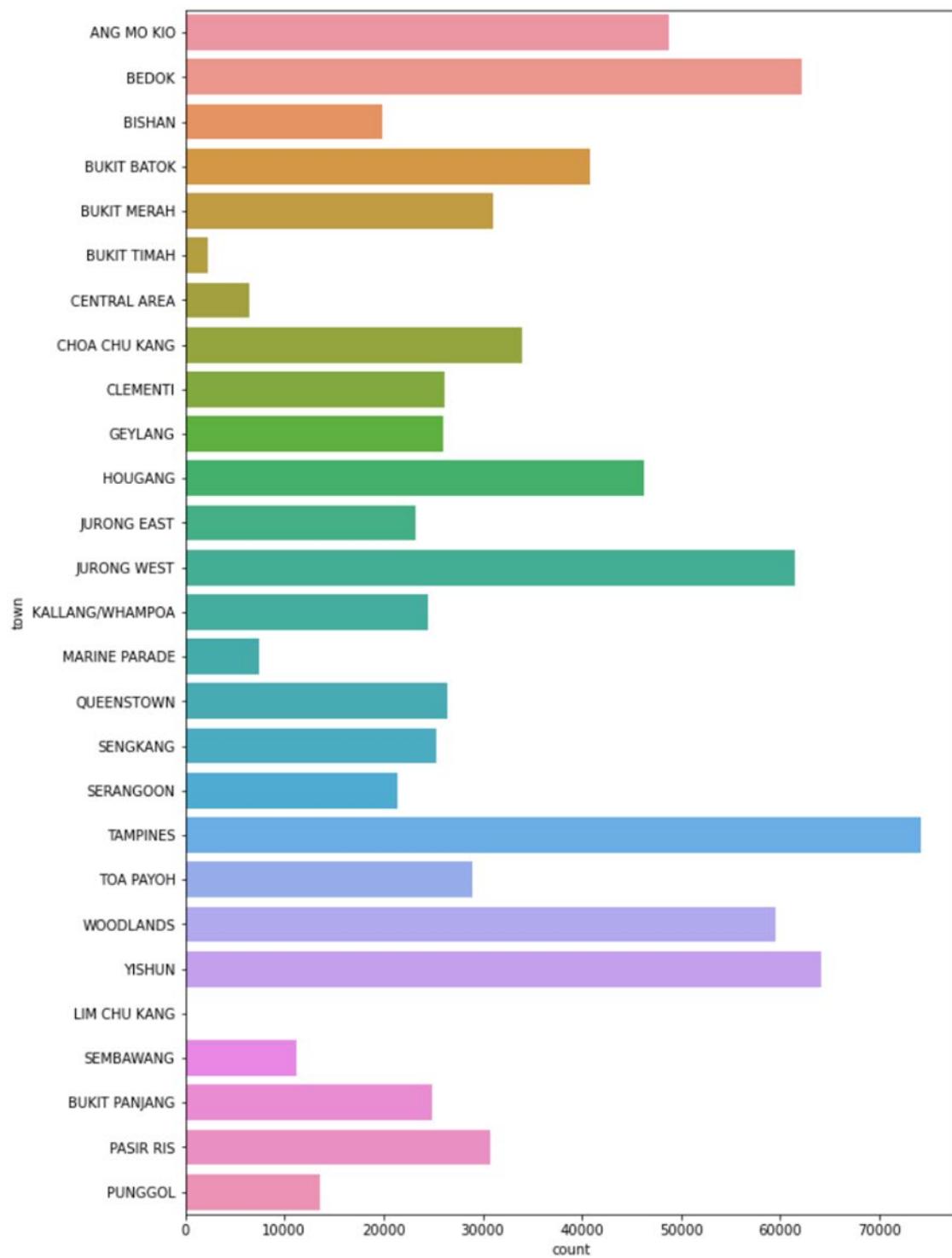
sns.kdeplot(dataset[dataset['flat_type'] == '1 ROOM']
            ['resale_price']);
sns.kdeplot(dataset[dataset['flat_type'] == '2 ROOM']
            ['resale_price']);
sns.kdeplot(dataset[dataset['flat_type'] == '3 ROOM']
            ['resale_price']);
sns.kdeplot(dataset[dataset['flat_type'] == '4 ROOM']
            ['resale_price']);
sns.kdeplot(dataset[dataset['flat_type'] == '5 ROOM']
            ['resale_price']);
sns.kdeplot(dataset[dataset['flat_type'] == 'EXECUTIVE']
            ['resale_price']);
sns.kdeplot(dataset[dataset['flat_type'] == 'MULTI GENERATION']
            ['resale_price']);

```



نمودار شمارشی برای شهرها:

```
fig = plt.figure()
fig.set_size_inches(10, 16)
sns.countplot(data = dataset, y = 'town');
```



پیش‌پردازش داده

همانطور که در بخش قبل توضیح دادیم، زمانی که با داده کیفی روپرتو هستیم باید آن را به نوعی به مقادیر عددی تبدیل کنیم. در بخش قبل این کار را با یکی از توابع کتابخانه پانداز انجام دادیم. اما داده‌هایی که ترتیب آن‌ها اهمیت داشته باشد باید با احتیاط و توجه به مقدار عددی تبدیل شوند. به عنوان اندازه لباس ممکن است به صورت کوچک، متوسط و یا بزرگ در داده‌ها وجود داشته باشند و این مورد احتمالاً در الگوریتم یادگیری ماشین متأثیر گذار خواهد بود. همچنین ستون‌های ویژگی که کمتر اهمیت دارند را حذف می‌کنیم:

Changing the Dataset

```
dataset = dataset.drop(columns=['month', 'street_name', 'address',
'price_psm_yearly', 'Core CPI',
'year_gni', 'flat_model', 'price_cpi_adj',
'price_psm_cpi_adj', 'bala_lease_pct',
'price_psm_cpi_lease_adj_implied'])
dataset = dataset.drop(columns=['price_lease_adj_implied', 'price_psm_lease_adj_implied',
'price_cpi_lease_adj_implied'])

dataset = dataset.drop(columns=['block', 'town_dummy',
'storey_range'])

encoder = LabelEncoder()
for i in range(len(dataset.columns)):
    c = dataset.columns[i]
    if(dataset.dtypes[c] == 'object'):

        dataset[c] = encoder.fit_transform(dataset[c])

dataset
```

	year	town	flat_type	latitude	longitude	storey	area_sqm	lease_start	lease_rem	resale_price	price_psm
0	1990	0	3	1.365485	103.844025	11	92.0	1976	85	86000.0	934.782609
1	1990	0	2	1.365485	103.844025	8	82.0	1976	85	59000.0	719.512195
2	1990	0	2	1.366272	103.841465	5	73.0	1976	85	47200.0	646.575343
3	1990	0	2	1.365485	103.844025	11	67.0	1976	85	47200.0	704.477612
4	1990	0	3	1.369226	103.841652	5	81.0	1977	86	80000.0	987.654321
...
840913	2021	26	2	1.433340	103.829168	8	64.0	1986	64	273000.0	4265.625000
840914	2021	26	2	1.430248	103.828862	2	68.0	1984	62	268000.0	3941.176471
840915	2021	26	1	1.423215	103.848264	8	47.0	2015	93	260000.0	5531.914894
840916	2021	26	1	1.421059	103.846357	8	48.0	2013	91	250000.0	5208.333333
840917	2021	26	1	1.421059	103.846357	2	48.0	2013	91	210000.0	4375.000000

840918 rows × 11 columns

جدا کردن ستون‌های ویژگی و ستون‌های هدف:

در انتهای این مرحله نیز، ستون‌های ویژگی و ستون هدف را به صورت زیر از هم جدا می‌کنیم.

Splitting Data to X, y

```
X = dataset.drop(columns=['resale_price'])
y = dataset['resale_price']
```

استانداردسازی: عمل استانداردسازی را روی ستون‌های سال، ماه، تعداد اتاق، طبقه، متراژ، زمان، شروع قرارداد، مدت زمان مانده تا پایان قرارداد و قیمت به ازای هر متر مربع انجام می‌دهیم.

Scaling

```
Standard_Scaler = StandardScaler()
```

```
X[['year']] = Standard_Scaler.fit_transform(X[['year']])
X[['town']] = Standard_Scaler.fit_transform(X[['town']])
X[['flat_type']] = Standard_Scaler.fit_transform(X[['flat_type']])
X[['storey']] = Standard_Scaler.fit_transform(X[['storey']])
X[['area_sqm']] = Standard_Scaler.fit_transform(X[['area_sqm']])
X[['lease_start']] = Standard_Scaler.fit_transform(X[['lease_start']])
X[['lease_rem']] = Standard_Scaler.fit_transform(X[['lease_rem']])
X[['price_psm']] = Standard_Scaler.fit_transform(X[['price_psm']])
```

تقسیم داده

در کد زیر داده را به نسبت ۷۰ به ۳۰ تقسیم کردۀایم. یعنی ۷۰ درصد داده‌ها، داده‌ی آموزشی در نظر گرفته شده‌اند و ۳۰ درصد به داده‌های تست اختصاص یافته‌اند.

Split Dataset to Train , Test

```
X_train , X_test, y_train, y_test = train_test_split(X, y,
test_size=0.4, random_state=111)

print('X train shape', X_train.shape)
print('y train shape', y_train.shape)

X train shape (504550, 10)
y train shape (504550,)

print('X test shape', X_test.shape)
print('y test shape', y_test.shape)

X test shape (336368, 10)
y test shape (336368,)
```

آموزش و آزمایش مدل

بعد از تقسیم داده‌ها می‌توانیم داده‌ها را برای یادگیری مدل‌های پیش‌بینی به کار بگیریم. همان‌طور که گفته شد کتابخانه سایکیتلرن، الگوریتم‌های یادگیری ماشین گسترده‌ای دارد که رابط ثابتی برای مدل کردن، پیش‌بینی دقیق و فراخوانی برای تمام الگوریتم‌ها ارائه می‌دهد. در این قسمت رگرسیون داده با کمک الگوریتم‌های رگرسیون خطی، لاسو، ستیغی، شبکه الاستیک، کا-همساخه نزدیک و درخت تصمیم انجام می‌دهیم.

در اینجا ابتدا، مدل‌های مورد نظر را ایجاد کردیم و برای کم کردن تعداد خطوط کد، همه را داخل آرایه‌ای قرار داریم تا بعداً از آن‌ها استفاده کنیم. پس از آن عمل برازش مدل را روی داده آموزشی انجام می‌دهیم و پیش‌بینی مدل از داده آزمایشی را داخل متغیری ریخته تا نتایج را ارزیابی کنیم.

Model Building (Regression)

```
models={
    'LinearRegression':LinearRegression(),
    'Ridge':Ridge(),
    'Lasso':Lasso(),
    'ElasticNet':ElasticNet(),
    'KNeighborsRegressor':KNeighborsRegressor(),
    'DecisionTreeRegressor':DecisionTreeRegressor(),
}
```

```
scores = {}

for i, model in models.items():

    print("model: ", model)

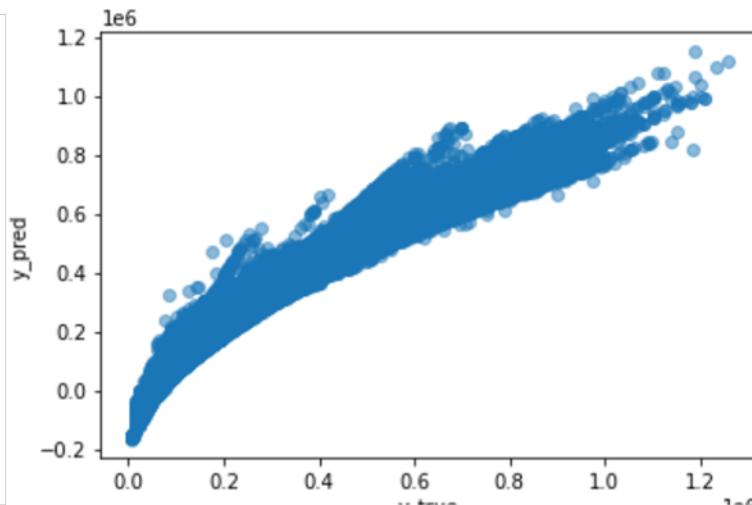
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    print("r2 score: ", r2_score(y_test, y_pred))
    print("MAE score: ", mean_absolute_error(y_test, y_pred))
    print("MSE score: ", mean_squared_error(y_test, y_pred))

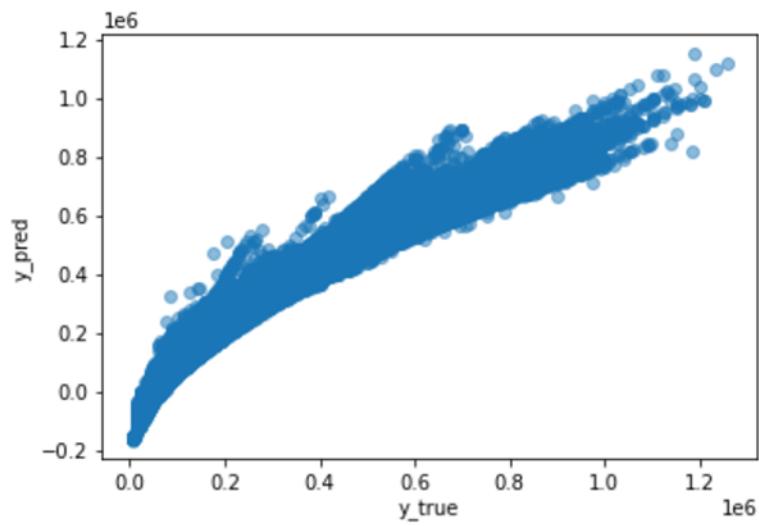
    plt.figure()
    fig.set_size_inches(8, 8)
    plt.scatter(y_test, y_pred, alpha=0.5)
    plt.xlabel('y_true')
    plt.ylabel('y_pred')
    plt.show()

    print("-" * 50)
    scores[i] = model.score(X_test,y_test)
```

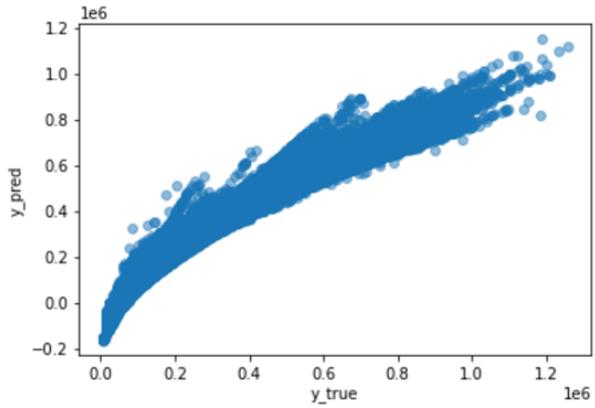
```
model: LinearRegression()
r2 score: 0.959769454845847
MAE score: 20442.539553131395
MSE score: 916006579.2470516
```



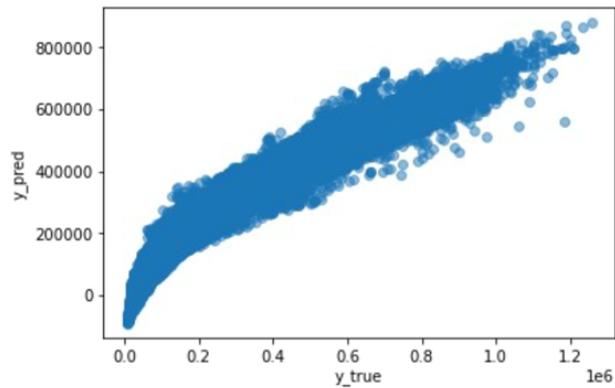
```
model: Ridge()
r2 score: 0.9597694554295702
MAE score: 20442.472858249843
MSE score: 916006565.9562976
```



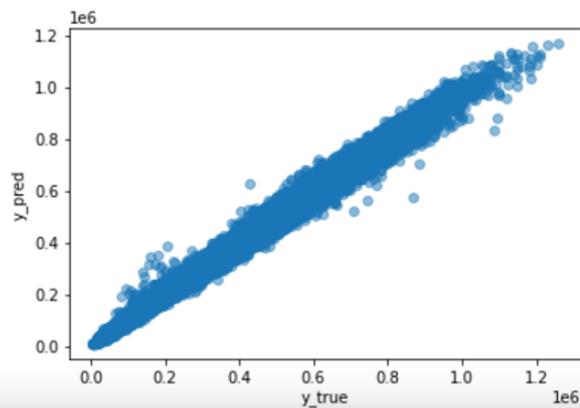
```
model: Lasso()  
r2 score: 0.9597671588219513  
MAE score: 20443.372370074256  
MSE score: 916058857.2608467
```

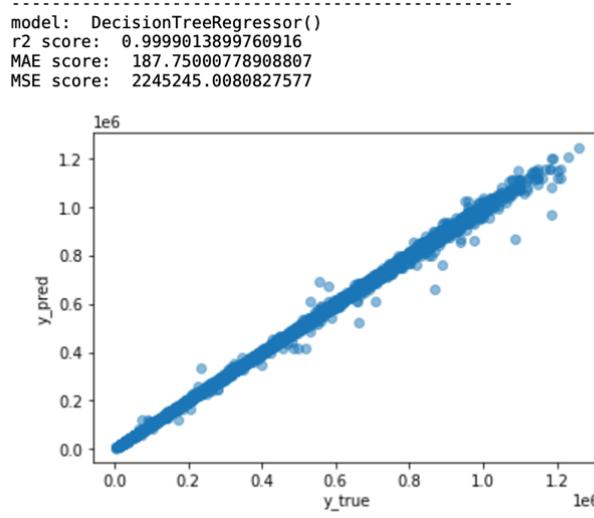


```
model: ElasticNet()  
r2 score: 0.8928021058251749  
MAE score: 34508.08357039292  
MSE score: 2440781649.100579
```



```
model: KNeighborsRegressor()  
r2 score: 0.9956733694866835  
MAE score: 6191.382381992341  
MSE score: 98512759.4215531
```





برای هر مدل سه پارامتر ارزیابی مدل رگرسیون شامل خطای میانگین مربعات، میانگین خطای مطلق و ضریب تعیین را بدست آوردیم. همچنین نمودارها نشان‌دهنده میزان تطبیق جواب‌های پیش‌بینی شده و جواب‌های واقعی هستند.

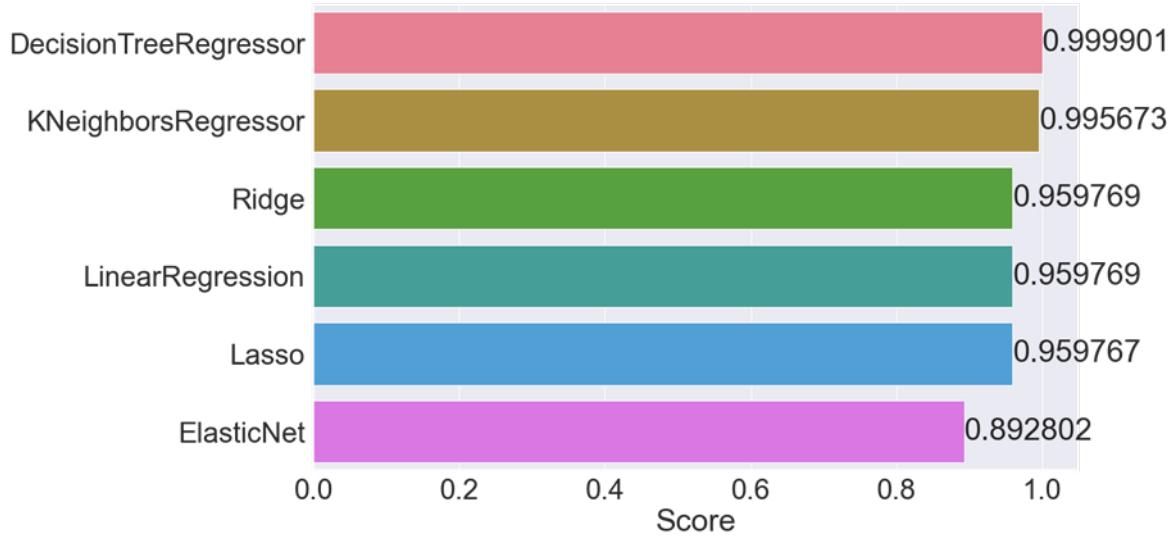
```

scores
{'LinearRegression': 0.959769454845847,
'Ridge': 0.9597694554295702,
'Lasso': 0.9597671588219513,
'ElasticNet': 0.8928021058251749,
'KNeighborsRegressor': 0.9956733694866835,
'DecisionTreeRegressor': 0.9999013899760916}

scores = pd.DataFrame(scores, index=[ 'Score']).transpose()
scores.sort_values(by = [ 'Score'], ascending = False, inplace = True)

plt.figure(figsize = (16,10))
sns.set(font_scale = 3)
s = sns.barplot(x=scores.Score, y = scores.index, data = scores,
palette="husl")
s.bar_label(s.containers[0]);

```



همانطور که مشاهده می‌کنیم الگوریتم درخت تصمیم بیشترین دقت را به ما داده است و نسبت به مدل‌های دیگر توانسته عملکرد بهتری در پیش‌بینی داده‌ها داشته باشد.

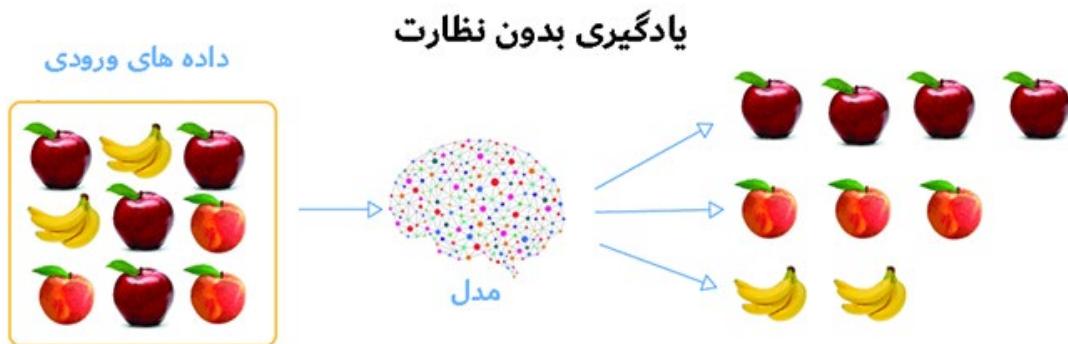
فصل پنجم

یادگیری ماشین نظارت‌نشده

یادگیری ناظارت‌نشده

۱-۵ تعاریف اولیه

یادگیری بدون ناظر یک دسته از روش‌های یادگیری ماشین برای کشف الگوهای موجود در میان داده‌ها است. داده‌های ارائه شده به الگوریتم ناظارت‌نشده دارای برچسب نیستند بدین معنا که متغیر ورودی بدون هیچ متغیر خروجی متناظری داشته است. در یادگیری ناظارت‌نشده، الگوریتم‌ها به حال خود رها می‌شوند تا ساختارهای جالب موجود در میان داده‌ها را کشف کنند. یان لیوکن، دانشمند فرانسوی کامپیوتر و پدر بنیان‌گذار شبکه عصبی پیچشی، یادگیری ماشین ناظارت‌نشده را چنین تعریف کرده است: «آموزش دادن ماشین‌ها برای یادگیری برای خودشان بدون آنکه به آن‌ها صراحتاً گفته شود کاری که انجام می‌دهند درست محسوب می‌شود یا غلط.



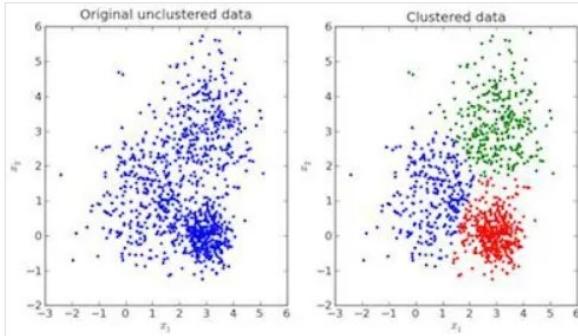
۲-۵ انواع مسائل یادگیری ماشین بدون ناظر

یادگیری ناظارت‌نشده راهی به سوی هوش مصنوعی حقیقی است. الگوریتم‌های یادگیری ناظارت‌نشده به دو دسته خوشه‌بندی و کاهش بعد تقسیم می‌شوند که در ادامه به بررسی آن‌ها می‌پردازیم.

۱-۲-۵ خوشه‌بندی^۱

در خوشه‌بندی، داده‌ها به چندین گروه تقسیم بندی می‌شوند. به بیان ساد، هدف جداسازی گروه‌هایی با صفات مشترک و تخصیص آن‌ها به خوشه‌ها است. در تصویر زیر، نقاط داده در نمودار سمت چپ داده‌های خامی هستند که خوشه‌بندی نشده‌اند. تصویر سمت راست، داده‌های خوشه‌بندی شده‌است.

^۱ clustering



شکل نمودارهای خوشبندی شده و خوشبندی نشده

۱-۱-۲-۵ الگوریتم خوشبندی میانگین کا
الگوریتم میانگین کا یک سلسله مرحل تکراری دارد که در زیر آورده شده اند.

مرحله صفر: تعیین تعداد خوش

این است که تعداد خوش باید توسط کاربر تعیین شود. ما باید تعیین کنیم که می خواهیم داده ها به چند خوش تقسیم شوند. برای مثالی که مطرح می کنیم می خواهیم سه خوش داشته باشیم.

مرحله یک: تعیین مرکز خوش

در این رحله باید به اندازه تعداد خوش، مرکز خوش تعیین کنیم. مرکز خوش، به نوعی نماینده خوش خود است. این مرکز خوشها را به صورت تصادفی انتخاب می کنیم. چون سه خوش داریم پس به صورت تصادفی سه نقطه به عنوان مرکز خوش انتخاب می کنیم. در تصویر زیر مراکز خوش را با ستاره نشان داده ایم.

مرحله دو: فاصله سنجی و خوشبندی

این مرحله شامل دو قسمت فاصله سنجی و خوشبندی است. ابتدا فاصله سنجی، باید فاصله هر داده با مراکز خوش محاسبه شود که این فاصله سنجی براساس فاصله اقلیدسی انجام می شود که فاصله نقطه داده، با نقطه مرکز محاسبه می شود. بعد از اینکه فاصله داده با مراکز خوش محاسبه شد باید مشخص کنیم که این داده به کدام خوش تعلق دارد. یک نمونه داده به خوشه ای تعلق دارد که تا مرکز آن خوش کمترین فاصله را دارد. این روال باید برای تک تک داده های موجود انجام شود.

مرحله سوم: آپدیت مرکز خوشها

قبل تر گفتیم که مرکز هر خوش نماینده آن خوش هست اما چون در ابتدا این نماینده ها به صورت تصادفی انتخاب می شوند، پس خوب نیست که این مراکز ثابت باقی بمانند. بهترین راه ایجاد یک مرکز خوش جدید، میانگین گیری از اعضای خوش است. اگر میانگین اعضای موجود در هر خوش را حساب کنیم، یک مرکز خوش جدید خواهیم داشت.

پس از تکرار مراحل ذکر شده، خوشه هر کدام از دادهها ممکن است عوض شود. آنقدر مرحله دوم و سوم را تکرار میکنیم تا بالاخره به مرحله پایدار برسیم و دیگر داده، خوشههای خود را عوض نکنند. اگر مهاجرت نداشته باشیم، مراکز خوشه هم تغییر نمیکنند و الگوریتم به پایان می‌رسد. بلوک دیاگرام این الگوریتم در تصویر زیر آورده شده است.



بلوک دیاگرام الگوریتم کی میانگین

منظور از تابع هدف آورده شده در لوزی موجود در شکل فوق، یک تابع کلی است که می‌تواند میزان خطای خطا را در خوشبندی به ما نشان دهد. ما آنقدر ادامه می‌دهیم تا تابع خطا به حداقل مقدار برسد؛ یعنی صفر شود یا مقدار خیلی کمی داشته باشد. فرمول تابع هدف به صورت زیر است:

$$J = \sum_{k=1}^K \sum_{p_i \in s_k} (p_i - c_k)^2$$

در فرمول فوق، سیگمای داخلی داده‌های یک خوشه را با مرکز همان خوشه محاسبه می‌کند. زمانی که این عبارت به کمترین مقدار می‌رسد که هر داده ای دقیقاً در خوشه درست خودش قرار بگیرد.

پیاده‌سازی

نحوه پیاده‌سازی این الگوریتم با کمک پکیج سایکیت لرن به صورت زیر است که بعد از بارگذاری داده‌ها مدل را تعریف کرده، آن را فیت می‌کنیم؛ و به مدل به دست آمده داده‌های جدید می‌دهیم تا آن‌ها را پیش بینی کند

```

from sklearn.cluster import KMeans
import numpy as np
X = np.array([[1, 2], [1, 4], [1, 0],
              [10, 2], [10, 4], [10, 0]])
kmeans = KMeans(n_clusters=2, random_state=0).fit(X)

print("Labels: \n", kmeans.labels_)

print("Test Prediction: \n", kmeans.predict([[0, 0], [12, 3]]))

print("Cluster Centers: \n", kmeans.cluster_centers_)

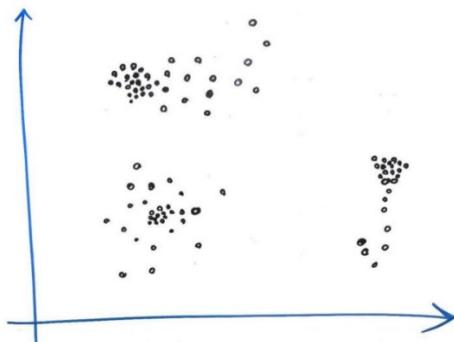
Labels:
[1 1 1 0 0 0]
Test Prediction:
[1 0]
Cluster Centers:
[[10.  2.]
 [ 1.  2.]]

```

شکل ۳-۵ پیاده سازی K میانگین با زبان پایتون برای یک مثال ساده

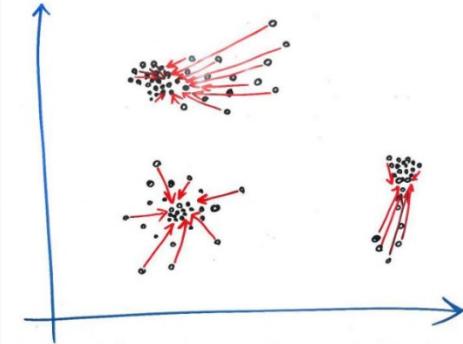
۲-۱-۲-۵ الگوریتم خوشبندی انتقال میانگین

الگوریتم خوشبندی میانگین کا، با وجود اینکه الگوریتم بسیار خوبی برای خوشبندی است، یک مشکل اساسی دارد و آن این است که کاربر، در ابتدا باید تعداد خوشبدهای موجود را به این الگوریتم بدهد. باری غلبه به این مشکل می توان از الگوریتم انتقال میانگین استفاده کرد. شکل زیر را در نظر بگیرید.



شکل نقاط آغازین قبل از خوشبندی انتقال میانگین

فرض کنید می خواهیم این شکل را بدون دانستن تعداد نقاط آغازین، خوشبندی کنیم. همان طور که می بینید، در هر گروه یک سری نقاط وجود دارند که تجمع بالاتری در آنها وجود دارد. کار اصلی الگوریتم انتقال میانگین، پیدا کردن این نقاط با تجمع بالا در هر خوشه است تا با کمک آن بتواند خوشبدهای مختلف را پیدا کند. در واقع الگوریتم انتقال میانگین مانند این است که هر نمونه در فضا را به آرامی و در تکرارهای مختلف به سمت نقطه ای با تجمع بیشتر در نزدیکی خود حرکت می دهد تا سرانجام همه نقاط تقریبا در یک جا جمع شوند. برای مثال شکل زیر را در نظر بگیرید.



نشان دادن حرکت کردن الگوریتم به سمت نقاط با تجمع بیشتر

هر کدام از نقاط در هر گروه تمایل دارند به نقطه‌ای که تجمع بیشتری در آن قرار دارد (یعنی غلیظ‌تر است) حرکت کنند. اگر بخواهیم آماری صحبت کنیم، در نقطه که تجمع بیشتری وجود دارد مقدار تابع چگالی احتمال بالاتری داریم و الگوریتم می‌خواهد نمونه‌ها را آرام آرام به این نقطه همگرا کند. برای مثال یک خوش‌خاص در شکل فوق را تصور کنید؛ یک نقطه سیاه بالا را در نظر بگیرید. نقطه سیاه نشان داده شده با عدد یک، در دور اول ابتدا به نقطه قرمز شماره دو می‌رود و در دور بعد به نقطه قرمز رنگ شماره سه انتقال پیدا می‌کند. این نقطه در هر دور به مکان غلیظ‌تر که تجمع بیشتری از نقاط آن جا هستند حرکت می‌کند. پس به صورت کلی مراحل این الگوریتم به این صورت است:

- با نقاط داده تخصیص یافته به خوش‌خود، شروع می‌کنیم.
- سپس این الگوریتم نقاط مرکزی را محاسبه خواهد کرد
- مکان نقاط مرکزی جدید به روزرسانی خواهد شد.
- حال پردازش تکرار خواهد شد و به ناحیه با تراکم بالاتر منتقل خواهد شد
- تنها، زمانی که نقاط مرکزی به مکانی برسند که نتوانند پیش بروند متوقف خواهند شد.

این الگوریتم عملیات خوش‌بندی را با استفاده از میانگین وزنی انجام می‌دهد. که به جای استفاده از وزن ساده، می‌تواند از کرنل گاوی هم استفاده کند.

پیاده‌سازی

نحوه پیاده‌سازی این الگوریتم با کمک پکیج سایکیت لرن به صورت زیر است؛ که البته پارامترهای ورودی را می‌توانیم خودمان تنظیم کنیم.

```

from sklearn import datasets
from sklearn.cluster import MeanShift

iris_df = datasets.load_iris()

model = MeanShift()

model.fit(iris_df.data)

MeanShift()

pred_label = model.predict([[7.2,3.5,0.8,1.6]])

all_pred = model.predict(iris_df.data)

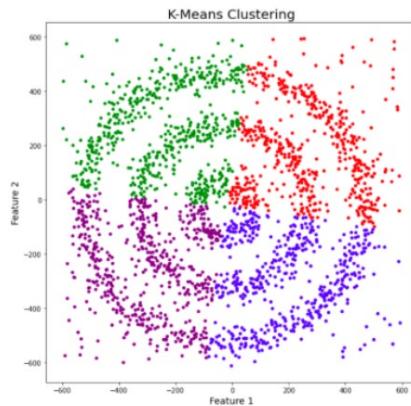
print(all_pred)

```

کد پیاده‌سازی الگوریتم انتقال میانگین با کمک پکیج سایکیت لرن

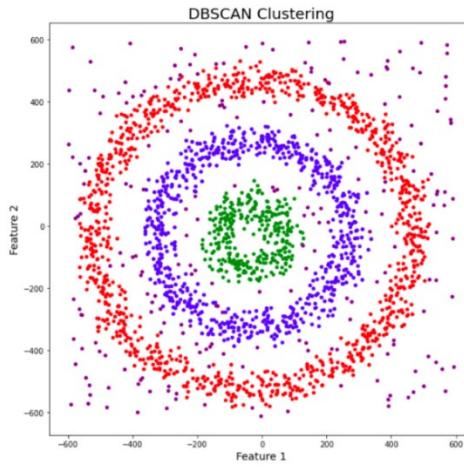
۳-۱-۲-۵ الگوریتم خوشبندی فضایی مبتنی بر چگالی کاربردهای دارای نویز (دی‌بی‌اسکن)

الگوریتم‌های فوق در خوشبندی شکل‌های پیچیده و پیچ در پیچ شکست می‌خورند و همچنین توانایی برخورد مناسب با داده‌های نویز را هم ندارد. برای مثال خوشبندی الگوریتم کی میانگین برای شکل پیچیده زیر را می‌بینیم.



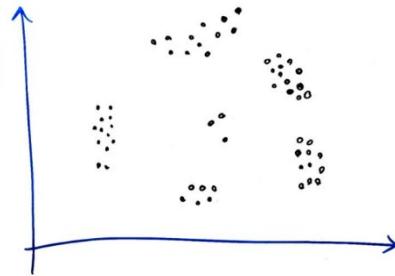
خوشبندی شکلی پیچیده با الگوریتم میانگین کا

الگوریتم دی‌بی‌اسکن این نقطه ضعف‌ها را پوشش می‌دهد. هم برای شکل‌های پیچیده خوب جواب می‌دهد و هم نویزها را تشخیص می‌دهد. خوشبندی داده مربوط به شکل فوق با کمک الگوریتم دی‌بی‌اسکن هم انجام شده که نتیجه آن به صورت زیر است.



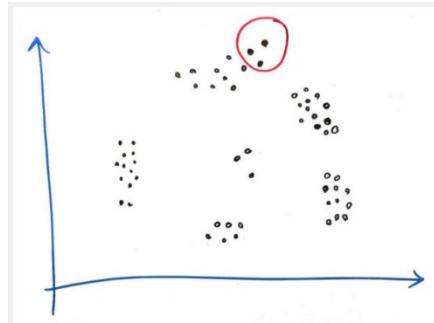
شکل مربوط به خوشه‌بندی داده‌ای با شکل پیچیده توسط الگوریتم DBSCAN

الگوریتم DBSCAN دو پارامتر بسیار مهم دارد؛ مینیمم تعداد نقاط و اپسیلون که به توضیح آن خواهیم پرداخت. پارامتر اپسیلون نشان‌دهنده شعاع دایره‌ای است که دور هر نقطه برای تعیین چگالی آن به کار می‌رود. منظور از مینیمم نقاط هم، تعداد داده‌ای است که باید در دایره یک نقطه باشند تا آن نقطه هست به حساب آید؛ یعنی حداقل نقاطی که باید در شعاع باشد تا یک خوشه تشکیل شود. اگر ابعاد دایره بیشتر شوند، دایره به ابر کره تبدیل می‌شود. شکل زیر را در نظر بگیرید.

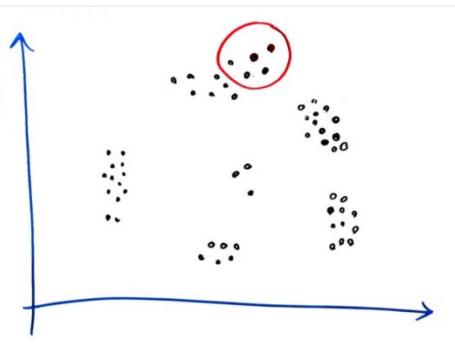


فرض کنید می‌خواهیم داده‌های بالا را خوشه‌بندی کنیم. الگوریتم دور هر نقطه دایره‌ای با شعاع اپسیلون ترسیم می‌کند که مرکز هر دایره همان داده است و هر نقطه را هسته یا مرز یا نویز در نظر می‌گیرد. منظور از هسته، داده‌ای است که حداقل به تعداد پارامتر مینیمم نقاط دایره آن را محاصره کرده باشد. منظور از مرز داده‌ای است که کمتر از پارامتر تعداد نقاط مینیمم (بیشتر از یکی) دایره آن را محاصره کرده باشند و منظور از نویز، داده‌ای است که یک دایره آن را محاصره کرده باشد. این الگوریتم را روی مثال زیر به این صورت پیاده می‌کنیم

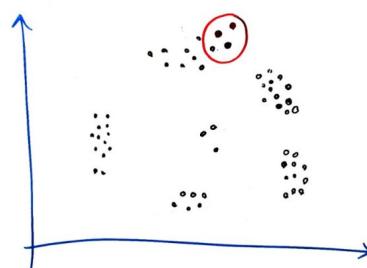
ابتدا یک نقطه را به صور تصادفی در فضای انتخاب می‌کنیم. در این فرآیند، مقدار مینیمم نقاط را برابر سه و مقدار شعاع را برابر یک قرار می‌دهیم. شکل زیر انتخاب یک نقطه تصادفی و شعاع آن را نشان می‌دهد.



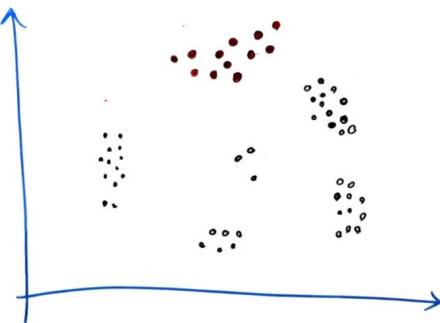
همان طور که می‌بینید، در یک شعاع مشخص، حداقل سه نمونه در شعاع نمونه‌ی مرکزی قرار دارند. پس این نمونه می‌تواند به عنوان یک خوشه انتخاب شود. مثلاً رنگ قرمز را برای این خوشه انتخاب می‌کنیم. حال به سراغ نمونه همسایه‌ای نقطه می‌رویم و آن را مانند شکل زیر بررسی می‌کنیم:



همانطور که نمونه قبلی قرمز رنگ شده بود، با توجه به اینکه این نمونه هم حداقل سه نقطه در شعاع دور خود دارد، پس این نمونه هم به خوشه قبلی ما می‌پیوندد و قرمز رنگ می‌شود. حال برای نقطه همسایه‌ی نیز این کار را انجام می‌دهیم:



این کار ادامه پیدا می کند تا زمانی که دیگر نقطه ای در میان خوشه های نزدیک همسایه ها نباشد. مانند شکل زیر، که تمامی نمونه های بالای شکل را به خوشه های قرمز نسبت داده ایم.



الگوریتم به همین ترتیب ادامه پیدا می کند. هنگامی که یک خوشه تشکیل شد، یک نقطه تصادفی دیگر را انتخاب می کنیم و این روند را ادامه می دهیم. این الگوریتم به دو پارامتر گفته شده بسیار حساس است، هر چه شعاع را کوچکتر در نظر بگیریم، احتمال ایجاد خوشه های بیشتر و کوچکتری تشکیل می شود و هر چقدر پارامتر مینیمم نقاط را بزرگ تر در نظر بگیریم، خوشه ها کمتر می شود؛ زیرا الگوریتم باید تعداد نمونه های بیشتری را در یک شعاع خاص ببیند تا خوشه را تشکیل دهد.

پیاده سازی

پیاده سازی این الگوریتم برای مجموعه داده آیریس به صورت زیر است.

```
from sklearn.cluster import DBSCAN
import numpy as np

X = np.array([[1, 2], [2, 2], [2, 3],
              [8, 7], [8, 8], [25, 80]])

dbscan = DBSCAN(eps=3, min_samples=2).fit(X)

print("Labels: \n", dbscan.labels_)
```

```
Labels:
[ 0  0  0  1  1 -1]
```

شکل ۱۰-۵ پیاده سازی DBSCAN با زبان پایتون برای یک مثال ساده

۲-۲-۵ الگوریتم‌های کاهش ابعاد

۱-۲-۲-۵ الگوریتم تحلیل مولفه اساسی^۱

گاهای ممکن است در کاربردهای مختلف با حالت‌هایی مواجه شویم که در آن‌ها ابعاد مجموعه داده بسیار بزرگ باشد. به عبارت دیگر، تعداد ویژگی‌ها بسیار زیاد است. این بزرگ بودن سبب می‌شود که پیچیدگی داده‌ها و در نتیجه پیچیده‌تر شدن مدل‌های ما می‌شود که این امر باعث افزایش حجم محاسبات می‌گردد. هم‌چنین قابلیت تفسیرپذیری ما نیز کاهش می‌یابد. لذا با توجه به تمامی موارد گفته‌شده نیاز به روش‌هایی جهت کاهش ابعاد داریم.

به صورت کلی در تحلیل مولفه اصلی، ما به دنبال یافتن یک فضای با ابعاد کوچک‌تر می‌باشیم به طوری که هنگامی که ویژگی‌های ما در اصلی اولیه روی این فضا توصیر می‌شوند از دست رفتن اطلاعات کمینه باشد. اگر بخواهیم این عبارت را به صورت ریاضی بنویسیم به صورت زیر اقدام می‌کنیم. X ویژگی‌های ما در فضای اولیه باشند و W فضای مورد نظر باشد که می‌خواهیم آن را به دست بیاوریم و تصویر X بر روی فضای W را با Z می‌توانیم بنویسیم.

$$Z = W^T X$$

هدف ما یافتن W است به گونه‌ای که واریانس Z بیشینه گردد. براساس تعریف واریانس و امید ریاضی می‌دانیم که:

$$Var(z) = E[(x - \mu)(x - \mu)^T] = \Sigma$$

علامت سیگما در اینجا بیان‌گر کوواریانس می‌باشد. حال خواهیم داشت:

$$\begin{aligned} Var(z) &= Var(W^T X) \\ &= E[(W^T x - W^T \mu)^2] \\ &= E[(W^T x - W^T \mu)(W^T x - W^T \mu)^T] \\ &= E[W^T(x - \mu)(x - \mu)^T W] \\ &= W^T E[(x - \mu)(x - \mu)^T] W \\ &= W^T \Sigma W \end{aligned}$$

پس هدف ما یافتن W است به طوری که مقدار واریانس بیشینه گردد. برای W جواب‌های مختلفی ممکن است یافت شود لذا به جهت اینکه جواب یکتا باشد فرض می‌کنیم نرم W برابر یک می‌باشد. حال مسئله به فرم ریاضی بازنویسی می‌کنیم.

$$\max_{w_1} w_1^T \Sigma w_1 \text{ subject to } \|w_1\| = 1$$

¹ Principal Component Analysis

با استفاده از روش لاگرانژ می‌توان قیدها را در معادله بالا وارد کرد که معادله زیر را نتیجه می‌دهد.

$$\max_{w_1} w_1^T \Sigma w_1 - \alpha_1 (w_1^T w_1 - 1)$$

برای یافتن نقطه بهینه از معادله فوق نسبت به ویژگی‌ها مشتق می‌گیریم که خواهیم داشت:

$$\Sigma w_1 = \alpha_1 w_1$$

معادله فوق ما را یاد مفاهیم بردار ویژه و مقدار ویژه می‌اندازد. در معادله فوق w_1 همان بردار ویژه برای سیگما که کوواریانس است می‌باشد. در اصل سوال یافتن یک فضای ابعاد کوچکتر مانند W به یافتن بردار ویژه و مقدار ویژه نبدیل می‌گردد. به W_1 اولین مولفه اساسی می‌گوییم. برای یافتن دومین مولفه اساسی W_2 مانند قبلی عمل می‌کنیم. یعنی هدف یافتن W_2 به گونه‌ای است که واریانس بیشینه شود. علاوه بر قرار دادن نرم این مولفه دوم برابر یک می‌خواهیم این مولفه بر مولفه اول نیز عمود باشد. حال قیدها را وارد معادله کرده و دوباره معادله را حل می‌کنیم برای این مولفه نیز W_2 مجدداً یک بردار ویژه برای کوواریانس می‌گردد. به همین ترتیب برای d ویژگی خود یک بردار ویژه پیدا می‌کنیم و از بین این‌ها به انتخاب می‌پردازیم. نمونه‌ای از پیاده‌سازی در زیر آورده شده است.

```
import numpy as np
from sklearn.decomposition import PCA

X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
pca = PCA(n_components=2)
pca.fit(X)

print("Variance ratio: ", pca.explained_variance_ratio_)
print("\nSingular values: ", pca.singular_values_)

Variance ratio:  [0.99244289  0.00755711]
Singular values:  [6.30061232  0.54980396]
```

۳-۵ پیاده‌سازی

۱-۳-۵ خوشبندی مجموعه داده خرده فروشی آنلاین

این مجموعه داده، مجموعه داده‌ای فراملیتی است که شامل تمام تراکنش‌های انجام شده بین تاریخ ۲۰۱۰/۱۲/۱ و ۲۰۱۱/۱۲/۹ برای خرده‌فروش آنلاینی مستقر در بریتانیا می‌باشد. این مجموعه داده به طور کلی شامل هشت ستون است. این هشت ستون عبارت اند از:

- شماره فاکتور: عددی ۶ رقمی است که به طور منحصر به قدر به هر تراکنش اختصاص داده می‌شود. اگر این کد با حرف انگلیسی سی شروع شود نشان دهنده لغو است.

- کد محصول: عددی ۵ رقمی یکتا برای هر محصول
- توضیحات محصول
- مقدار یا تعداد هر محصول در هر تراکنش
- تاریخ فاکتور: تاریخ و زمان صورتحساب.
- قیمت واحد
- شناسه مشتری: عددی یکتا منحصر به مشتری
- کشور: نام کشوری که مشتری در آن ساکن است

ما می‌خواهیم عمل خوشبندی را روی این داده انجام دهیم تا مشتریانی که در ویژگی‌های مشخصی که جلوتر تعریف می‌کنیم مشاهده دارند را در یک دسته قرار دهیم. توضیحات پیاده‌سازی در ادامه آورده شده است.

۱-۳-۵ فرآیند مدل‌سازی سایکیت‌لرن روی مجموعه داده خرده‌فروشی آنلاین مرحله اول: بارگذاری داده

برای این مدل‌سازی مراحلی تکراری مانند اضافه کردن کتابخانه‌ها، بارگذاری داده، و دیدن اطلاعاتی درمورد داده و مصورسازی را انجام می‌دهیم که برای جلوگیری از تکرار به توضیح آن نمی‌پردازیم. اما مرحله متفاوت و مهم در پیاده‌سازی این داده مرحله پاکسازی و پیش‌پردازش داده است که در ادامه به توضیح آن خواهیم پرداخت.

Importing Libraries

```
#import the necessary packages
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings

from matplotlib.gridspec import GridSpec
import plotly.express as px
from sklearn.cluster import KMeans, MeanShift, DBSCAN
from sklearn.preprocessing import StandardScaler

import datetime as dt
from sklearn.metrics import silhouette_score,
adjusted_mutual_info_score, calinski_harabasz_score

warnings.filterwarnings(action='ignore')

%matplotlib inline
```

Load Dataset

```
dataset = pd.read_excel('Online Retail.xlsx')
dataset
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

541909 rows × 8 columns

Info of Dataset

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   InvoiceNo   541909 non-null   object 
 1   StockCode    541909 non-null   object 
 2   Description  540455 non-null   object 
 3   Quantity     541909 non-null   int64  
 4   InvoiceDate  541909 non-null   datetime64[ns]
 5   UnitPrice    541909 non-null   float64
 6   CustomerID   406829 non-null   float64
 7   Country      541909 non-null   object 
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

```
dataset.describe()
```

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

مرحله دوم: پیش پردازش داده

در ابتدا، تعداد سطرهای تکراری را پیدا می کنیم و آنها را حذف می کنیم. پس از آن نیز، تراکنش هایی که لغو شده‌اند را پیدا و حذف می کنیم. همچنین داده‌هایی که ستون مقدار آن‌ها دارای مقدار کوچکتر از صفر است را حذف می کنیم چون مقدار منفی برای این ستون بی معنی است. سطرهای بدون شناسه کاربر را نیز حذف می کنیم.

Data Cleaning

```
#number of duplicated rows
dataset.duplicated().sum()
5268

#remove duplicated rows
dataset = dataset.drop_duplicates()
dataset.duplicated().sum()
0

dataset = dataset[dataset['InvoiceNo'].str.startswith('C')!=True]
dataset.shape
(527390, 8)

#remove negative quantity
dataset = dataset[dataset['Quantity']>=0]
dataset.shape
(526054, 8)

#minimum and maximum Invoice Date
print('The minimum date is:',dataset.InvoiceDate.min())
print('The maximum date is:',dataset.InvoiceDate.max())

The minimum date is: 2010-12-01 08:26:00
The maximum date is: 2011-12-09 12:50:00
```

پس از آن ستون جدیدی به اسم مقدار کل ایجاد می کنیم که این ستون نشان‌دهنده مجموع کل آن جنس است که حاصل ضرب ستون مقدار در قیمت واحد است. پس از آن روی این ستون مقادیر مجموع کل برای هر مشتری را جمع می‌زنیم تا مقدار کلی خرید برای هر مشتری مشخص شود که این را در جدولی جدید نگهداری می کنیم. سپس جدول جدید دیگری نیز ایجاد کرده که دوباره برای هر مشتری تعداد فاکتورهایش را محاسبه می کنیم. این کار را برای این انجام می‌دهیم تا تعداد دفعات خرید هر مشتری را داشته باشیم. پس از آن دو جدول فوق را در یکدیگر ضرب و یکی می کنیم تا برای هر مشتری هر دو ستون جدید ایجاد کرده را داشته باشیم. همچنین زمان آخرین خرید انجام شده در این جدول را پیدا کرده و اختلاف آن با تاریخ هر فاکتوری را بدست می آوریم و در ستونی جدید نگهداری می کنیم. این ستون نشان‌دهنده فاصله هر تراکنش با آخرین تراکنش انجام شده است.

```

#Dropping the rows where customerID is missing
dataset = dataset[pd.notnull(dataset['CustomerID'])]

dataset.isnull().sum()

InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID    0
Country        0
dtype: int64

dataset['Amount'] = dataset['Quantity'] * dataset['UnitPrice']

each_customer = dataset.groupby('CustomerID')['Amount'].sum()
each_customer = each_customer.reset_index()
each_customer

   CustomerID      Amount
0     12346.0    77183.60
1     12347.0    4310.00
2     12348.0    1797.24
3     12349.0    1757.55
4     12350.0    334.40
...
4334    ...      ...
4335    18280.0    180.60
4335    18281.0    80.82
4336    18282.0    178.05
4337    18283.0    2045.53
4338    18287.0    1837.28

[4339 rows x 2 columns]

each_customer_freq = dataset.groupby('CustomerID')
['InvoiceNo'].count()

each_customer_freq = each_customer_freq.reset_index()

```

```

each_customer_freq = dataset.groupby('CustomerID')
['InvoiceNo'].count()

each_customer_freq = each_customer_freq.reset_index()

each_customer_freq

      CustomerID  InvoiceNo
0        12346.0         1
1        12347.0        182
2        12348.0         31
3        12349.0         73
4        12350.0         17
...
4334     18280.0         10
4335     18281.0          7
4336     18282.0         12
4337     18283.0        721
4338     18287.0         70

[4339 rows x 2 columns]

```

```

customer_details = pd.merge(each_customer_freq, each_customer, on =
'CustomerID')
customer_details

      CustomerID  InvoiceNo    Amount
0        12346.0         1  77183.60
1        12347.0        182  4310.00
2        12348.0         31  1797.24
3        12349.0         73  1757.55
4        12350.0         17  334.40
...
4334     18280.0         10  180.60
4335     18281.0          7  80.82
4336     18282.0         12  178.05
4337     18283.0        721 2045.53
4338     18287.0         70  1837.28

[4339 rows x 3 columns]

```

```

max_data = max(dataset['InvoiceDate'])
max_data

Timestamp('2011-12-09 12:50:00')

dataset['recency'] = max_data - dataset['InvoiceDate']

```

سپس برای هر مشتری کمترین مقداری که در این ستون دارند را محاسبه کرده و در ستونی در جدولی جدید نگهداری می‌کنیم. این مقدار فاصله آخرین خرید هر مشتری با آخرین خرید انجام شدهی کلی است.

```

dates = dataset.groupby('CustomerID')['recency'].min()

dates = dates.reset_index()
dates['recency'] = dates['recency'].dt.days
dates

   CustomerID  recency
0      12346.0      325
1      12347.0        1
2      12348.0       74
3      12349.0       18
4      12350.0      309
..          ...
4334    18280.0      277
4335    18281.0      180
4336    18282.0        7
4337    18283.0        3
4338    18287.0      42

[4339 rows x 2 columns]

```

سپس این جدول را با جدول جدیدی که ایجاد کردیم یکی می‌کنیم تا جدولی جدید شامل سه ستون داشته باشیم که این ستون‌ها در بر گیرنده تعدادی ویژگی برای هر مشتری است. این ویژگی‌ها شامل مقدار کل تراکنش‌ها، تعداد تراکنش‌ها، و فاصله آخرین تراکنش آن مشتری با تاریخ آخرین تراکنش انجام شده است که در حقیقت این ستون نشان‌دهنده میزان تازگی فعالیت مشتری است.

```

customer_details = pd.merge(customer_details, dates, on =
'CustomerID')
customer_details

   CustomerID  InvoiceNo    Amount  recency
0      12346.0         1  77183.60      325
1      12347.0        182  4310.00        1
2      12348.0         31  1797.24       74
3      12349.0         73  1757.55       18
4      12350.0         17   334.40      309
..          ...
4334    18280.0        10   180.60      277
4335    18281.0         7   80.82      180
4336    18282.0        12   178.05        7
4337    18283.0       721  2045.53        3
4338    18287.0        70  1837.28       42

[4339 rows x 4 columns]

```

پس از آن که جدول فوق را ایجاد کردیم، داده‌های پرت را حذف می‌کنیم که این کار با حذف داده‌های کمتر از چارک اول و بیشتر از چارک سوم برای هرستون انجام می‌دهیم.

```

q1 = customer_details.Amount.quantile(0.05)
q3 = customer_details.Amount.quantile(0.95)
iqr = q3 - q1

customer_details = customer_details[(customer_details['Amount'] >= q1
- 1.5 * iqr) &
                                      (customer_details['Amount'] <= q3
+ 1.5 * iqr) ]

customer_details

   CustomerID  InvoiceNo    Amount  recency
1      12347.0       182  4310.00        1
2      12348.0        31  1797.24       74
3      12349.0        73  1757.55       18
4      12350.0        17   334.40      309
5      12352.0        85  2506.04       35
...
4334     18280.0       10   180.60      277
4335     18281.0        7   80.82      180
4336     18282.0       12   178.05        7
4337     18283.0       721  2045.53        3
4338     18287.0       70  1837.28       42

[4276 rows x 4 columns]

q1 = customer_details.Amount.quantile(0.05)
q3 = customer_details.Amount.quantile(0.95)
iqr = q3 - q1

customer_details = customer_details[(customer_details['InvoiceNo'] >=
q1 - 1.5 * iqr) &
                                      (customer_details['InvoiceNo'] <=
q3 + 1.5 * iqr) ]

q1 = customer_details.Amount.quantile(0.05)
q3 = customer_details.Amount.quantile(0.95)
iqr = q3 - q1

customer_details = customer_details[(customer_details['recency'] >= q1
- 1.5 * iqr) &
                                      (customer_details['recency'] <= q3
+ 1.5 * iqr) ]

customer_details

```

که خروجی آن به صورت زیر می باشد:

```

customer_details

   CustomerID  InvoiceNo    Amount  recency
1      12347.0       182  4310.00        1
2      12348.0        31  1797.24       74
3      12349.0        73  1757.55       18
4      12350.0        17   334.40      309
5      12352.0        85  2506.04       35
...
4334     18280.0       10   180.60      277
4335     18281.0        7   80.82      180
4336     18282.0       12   178.05        7
4337     18283.0       721  2045.53        3
4338     18287.0       70  1837.28       42

[4276 rows x 4 columns]

```

مراحل ذکر شده بالا، روند مهمی بود که مشابه آن را قبلا در جایی ندیده بودیم. این کار مدلی معروف به اسم آر-اف-ام است که برای دسته‌بندی مشتریان کاربرد دارد. این مدل شامل زمان خرید، تعداد خرید، مبلغ خرید است که در بالا هم ذکر کردیم. این کار را انجام می‌دهیم تا مشتریانی که رفتار نزدیک به هم دارند را پیدا کنیم و از این مشابهت‌ها برای مثال در ارائه پیشنهادات به آن‌ها استفاده کنیم. به گروهی از مشتریان که نیازها و خواسته‌های مشترک یا در یک کلمه رفتار مشترک دارند بخش یا سگمنت گفته می‌شود.

پس از انجام مراحل فوق عمل استاندارسازی را انجام می‌دهیم و مدل کا-میانگین را ایجاد می‌کنیم. این کار را نیز اینگونه انجام می‌دهیم که حلقه‌ای از دو تا ده ایجاد می‌کنیم و مدل کا-میانگین را داخل این حلقه هر بار با تعداد خوش‌هایی برابر با متغیر حلقه ایجاد می‌کنیم و امتیاز این مدل ایجاد شده را داخل متغیری نگهداری می‌کنیم. سپس نمودار elbow^۱ این مدل‌های ایجاد شده را رسم می‌کنیم تا بینیم بهترین مقدار برای تعداد خوش‌ها چقدر است. همچنین بهترین مدل از نظر امتیاز سیلوئت و کالینسکی را نیز پیدا می‌کنیم. در نهایت با توجه به مقایسه نتایج بدست آمده تعداد خوش‌ها را برابر سه قرار داده و مدل کا-میانگین را با این تعداد خوش‌های ایجاد کرده و روی داده برآشش می‌کنیم که این گونه خوش‌های مشتری مشخص می‌شود.

Scaling

```
scaler = StandardScaler()

X_scaled = scaler.fit_transform(customer_details[['InvoiceNo',
    'Amount', 'recency']])

X_scaled_values = pd.DataFrame(X_scaled)
X_scaled_values.columns = ['InvoiceNo', 'Amount', 'recency']
X_scaled_values

      InvoiceNo      Amount      recency
0       0.832627   1.616030  -0.914622
1      -0.402114   0.249982  -0.185589
2      -0.058676   0.228405  -0.744847
3      -0.516593   -0.545282   2.161298
4       0.039449   0.635318  -0.575072
...
4271   -0.573832  -0.628895   1.841722
4272   -0.598364  -0.683140   0.873007
4273   -0.557478  -0.630281  -0.854701
4274    5.240078   0.384964  -0.894648
4275   -0.083207   0.271750  -0.505165

[4276 rows x 3 columns]
```

¹ elbow

Clustering

K-means

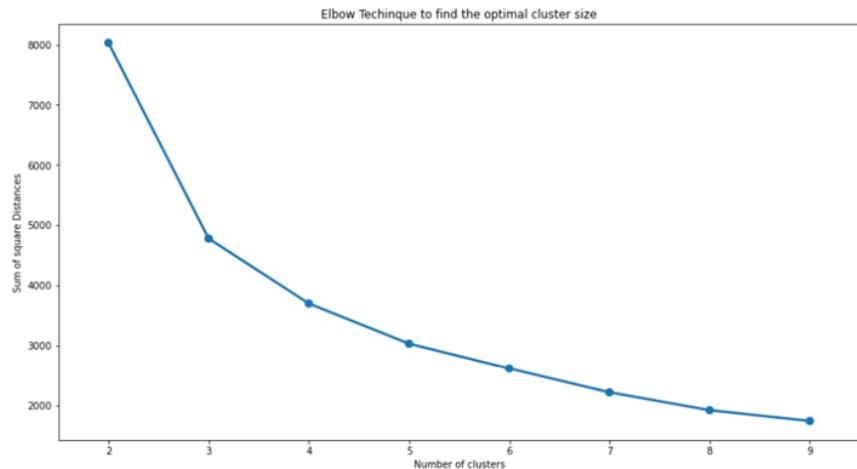
```
scores = {}

for i in range(2, 10):

    kmeans = KMeans(n_clusters=i, max_iter=40).fit(X_scaled_values)
    scores[i] = kmeans.inertia_

plt.figure(figsize = (15,8))
sns.pointplot(x = list(scores.keys()), y = list(scores.values()))
plt.xlabel("Number of clusters")
plt.ylabel("Sum of square Distances")
plt.title("Elbow Technique to find the optimal cluster size")

Text(0.5, 1.0, 'Elbow Techinque to find the optimal cluster size')
```



```
for i in range(2, 10):

    kmeans = KMeans(n_clusters=i, max_iter=40).fit(X_scaled_values)
    cluster_labels = kmeans.labels_
    silhouette_avg = silhouette_score(X_scaled_values, cluster_labels)
    print("for n_clusters = {} the silhaoutte_score is {}".format(i,
silhouette_avg))

for n_clusters = 2 the silhaoutte_score is 0.5660503710164054
for n_clusters = 3 the silhaoutte_score is 0.521729149850061
for n_clusters = 4 the silhaoutte_score is 0.49434437992239655
for n_clusters = 5 the silhaoutte_score is 0.49510532718507716
for n_clusters = 6 the silhaoutte_score is 0.45456194261954247
for n_clusters = 7 the silhaoutte_score is 0.4279274260773397
for n_clusters = 8 the silhaoutte_score is 0.42357793019212303
for n_clusters = 9 the silhaoutte_score is 0.42198732380644227
```

```

for i in range(2, 10):

    kmeans = KMeans(n_clusters=i, max_iter=40).fit(X_scaled_values)
    cluster_labels = kmeans.labels_
    calinski_avg = calinski_harabasz_score(X_scaled_values,
    cluster_labels)
    print("for n_clusters = {} the calinski harabasz score is
    {}".format(i, calinski_avg))

for n_clusters = 2 the calinski harabasz score is 2553.854435096958
for n_clusters = 3 the calinski harabasz score is 3599.705746749419
for n_clusters = 4 the calinski harabasz score is 3516.235440719681
for n_clusters = 5 the calinski harabasz score is 3454.1016899705905
for n_clusters = 6 the calinski harabasz score is 3334.199129698623
for n_clusters = 7 the calinski harabasz score is 3413.382973430448
for n_clusters = 8 the calinski harabasz score is 3469.7512384064207
for n_clusters = 9 the calinski harabasz score is 3402.6427931987378

```

همانطور که بالاتر هم ذکر کردیم و از معیارهای ارزیابی مشخص می‌باشد (بالاترین مقدار کالینسکی و سیلوئت) بهترین مقدار کا برای خوشبندی سه می‌باشد.

```

kmeans = KMeans(n_clusters=3).fit(X_scaled_values)

customer_details['clusters'] = kmeans.labels_
customer_details.head()

   CustomerID  InvoiceNo   Amount  recency  clusters
1      12347.0       182  4310.00        1          0
2      12348.0        31  1797.24       74          1
3      12349.0        73  1757.55       18          1
4      12350.0        17   334.40      309          2
5      12352.0        85  2506.04       35          1

column = ['InvoiceNo', 'Amount', 'recency']

```

```

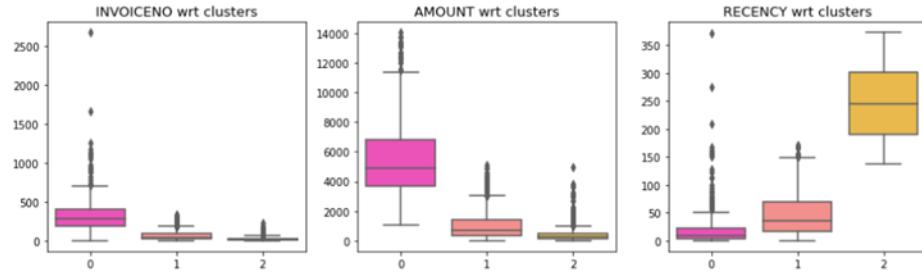
plt.figure(figsize=(15,4))

for i,j in enumerate(column):

    plt.subplot(1,3,i+1)
    sns.boxplot(y=customer_details[j], x=customer_details['clusters'],
    palette='spring')
    plt.title('{} wrt clusters'.format(j.upper()), size=13)
    plt.ylabel('')
    plt.xlabel('')

plt.show()

```



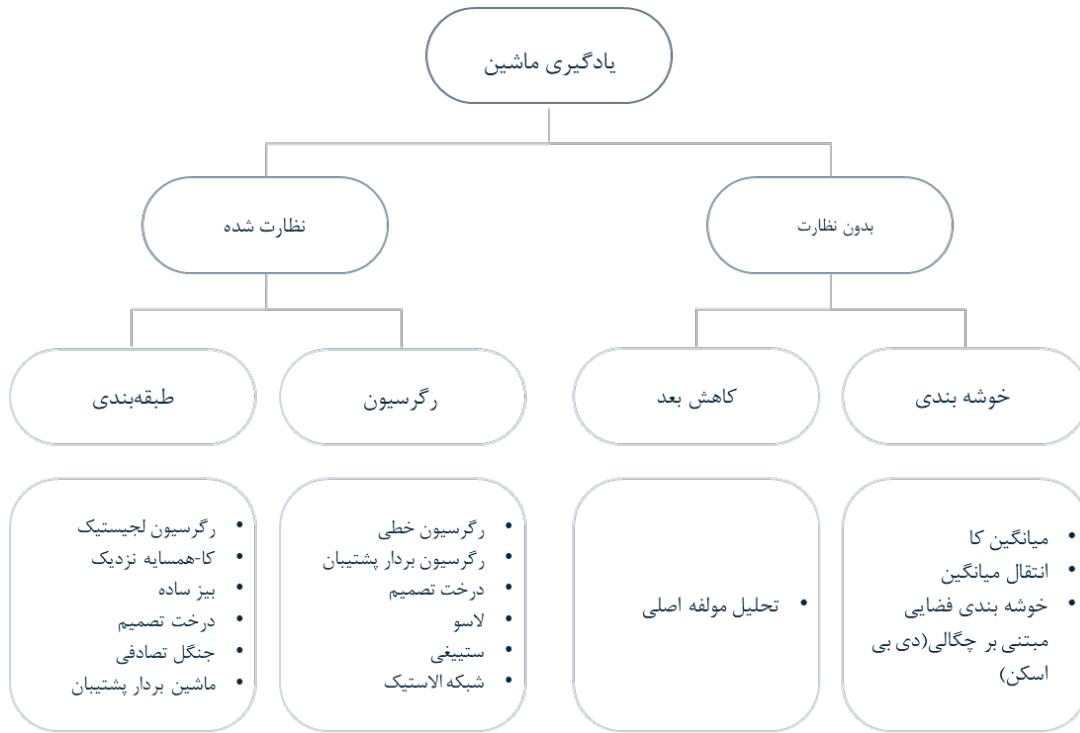
فصل ششم

نتیجه گیری و پیشنهادها

نتیجه گیری و پیشنهادها

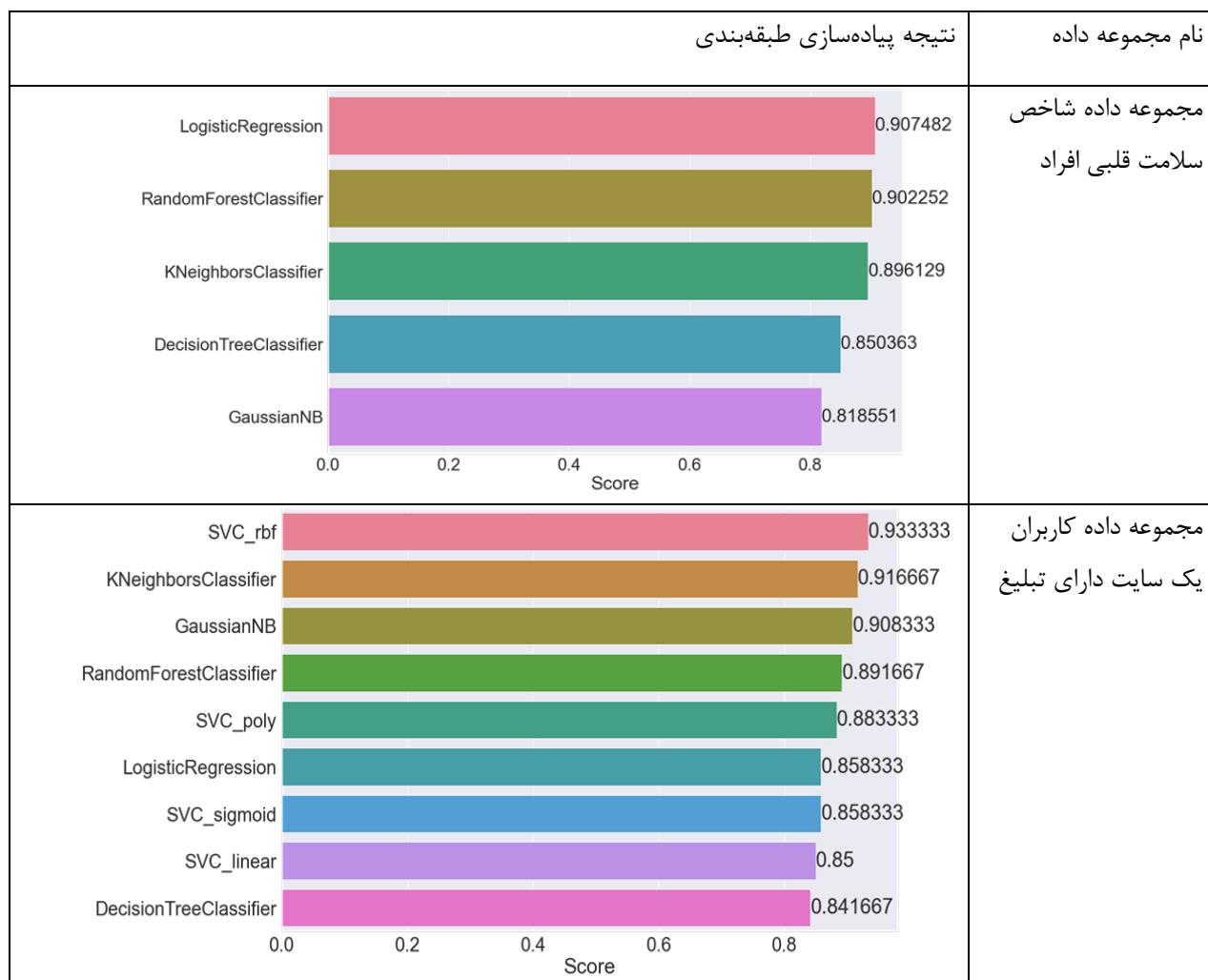
نتیجه گیری

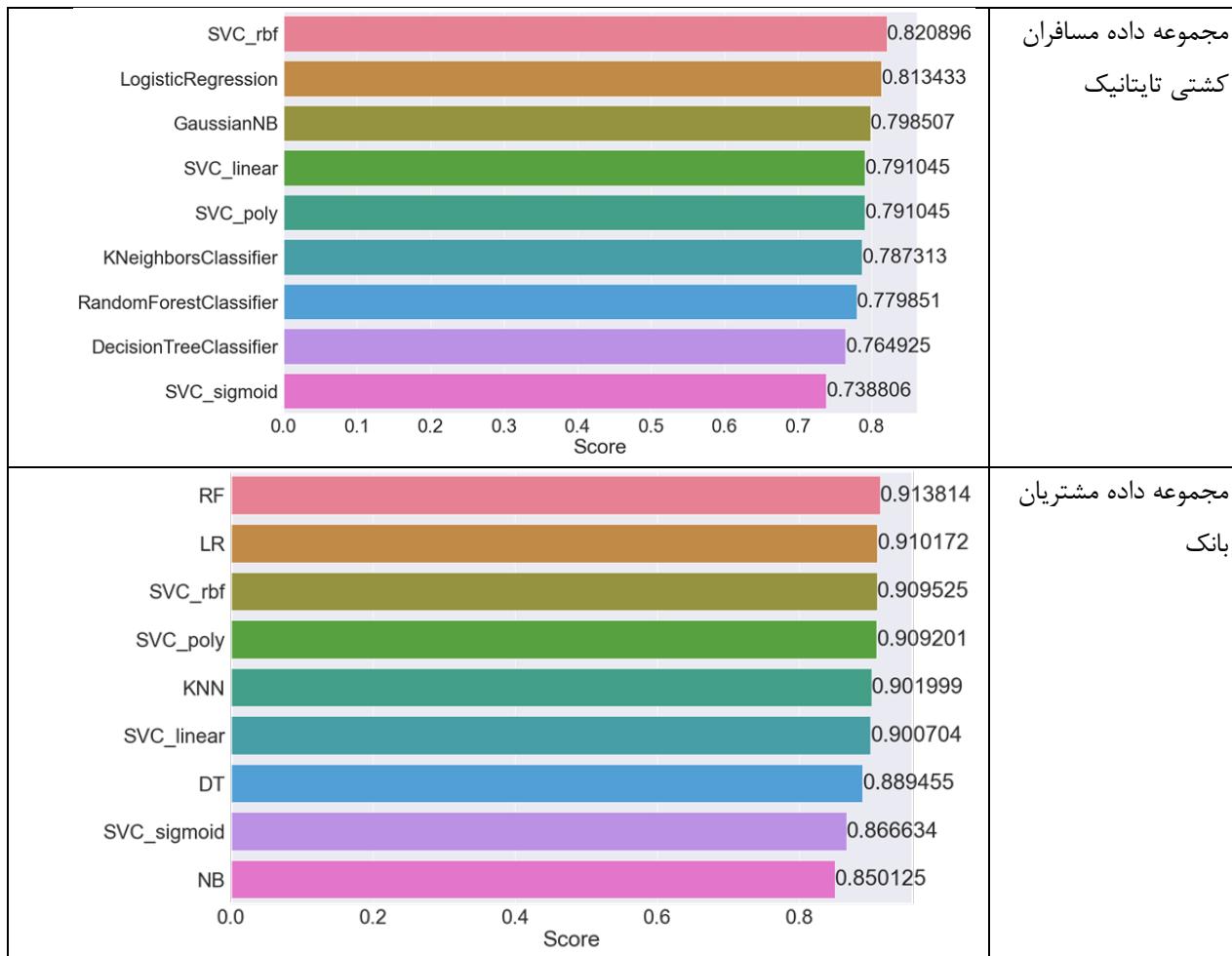
در این گزارش ما مجموعی جامع بر الگوریتم‌های یادگیری ماشین داشتیم و برخی از این الگوریتم‌ها را بررسی کردیم. ابتدا ما یک تعریف کلی از یادگیری ماشین ارائه دادیم. سپس به بررسی الگوریتم‌های مختلف یادگیری ماشین پرداختیم و بیشتر وارد جزئیات شدیم. دیدیم که یادگیری ماشین به چهار نوع اصلی تقسیم می‌شود: یادگیری ماشین ناظارت شده، ناظارت نشده، نیمه ناظارتی و تقویتی که در این دوره کارآموزی تمرکز بر دو دسته اصلی بود. همچنین پس از یادگیری تعدادی از این الگوریتم‌ها، به پیاده‌سازی آن‌ها روی تعدادی از الگوریتم‌ها پرداختیم. تقسیم‌بندی کلی الگوریتم‌های یادگیری ماشین در شکل زیر آورده شده است.



در یادگیری ناظارت شده، مجموعه داده ما دارای ستون‌های ویژگی و ستون خروجی یا برچسب است و براساس این مجموعه داده‌ای که داریم برای داده ورودی جدید باید برچسب تعیین کنیم. یادگیری ماشین ناظارت شده به دو دسته کلی رگرسیون و طبقه‌بندی تقسیم می‌شود که رگرسیون برای داده‌هایی که خروجی آن‌ها مقادیری پیوسته باشند و طبقه‌بندی زمانی است که خروجی مقادیر گسسته باشند می‌باشد. در دسته الگوریتم‌های رگرسیون به بررسی الگوریتم‌های رگرسیون خطی ساده، چندگانه، لاسو، ستیغی، شبکه الاستیک، کا-همسایه نزدیک، درخت تصمیم و رگرسیون بردار پشتیبان پرداختیم.

در دسته الگوریتم‌های طبقه‌بندی به بررسی الگوریتم‌های کا-همساخه نزدیک، ماشین بردار پشتیبان، درخت تصمیم، جنگل تصادفی و بیز ساده پرداختیم. همچنین الگوریتم‌های طبقه‌بندی مطالعه شده روی مجموعه داده‌های شاخص سلامت قلبی افراد، کاربران یک سایت دارای تبلیغ، مسافران کشتی تایتانیک و مشتریان بانک و الگوریتم‌های رگرسیون روی مجموعه داده قیمت خانه در سنگاپور پیاده‌سازی شد. نتیجه پیاده‌سازی این الگوریتم‌ها در شکل زیر آورده شده است.





نتیجه پیاده‌سازی الگوریتم‌های رگرسیون نیز در شکل زیر آورده شده است.



اما در یادگیری نظارت نشده داده‌ها بدون برچسب اند و برحسب شباهت نقاط داده، باید آن‌ها را برچسب گذاری کرد. این الگوریتم‌ها خود به دو دسته کاهش بعد و خوشبندی تقسیم می‌شوند. در قسمت خوشبندی ما الگوریتم‌های کای میانگین، انتقال میانگین، و خوشبندی فضایی مبتنی بر چگالی کاربردهای دارای نویز را بررسی کردیم و پیاده‌سازی آن روی مجموعه داده خرده فروشی آنلاین را دیدیم. همچنین الگوریتم تحلیل مولفه اساسی را نیز به عنوان نمونه‌ای از الگوریتم‌های کاهش ابعاد آموختیم.

پیشنهادها

اگر چه تحقیقات در زمینه الگوریتم‌های یادگیری ماشین در سال‌های اخیر در گستره بالایی در حال صورت گرفتن است، اما همچنان این علم بسیار نوپاست و جای پیشرفت فراوانی دارد. یادگیری ماشین کاربردهای گسترده‌ای در مسائل ضروری زندگی از جمله پزشکی و سلامت، کشاورزی، امنیت و همچنین جنبه‌های سرگرمی مختلف دارد. راه حل‌های یادگیری ماشین همچنان در فرایندهای اصلی کسب و کارها تغییرات مهمی را ایجاد می‌کنند و در زندگی روزمره ما رایج‌تر می‌شوند. در حال حاضر بسیاری از شرکت‌ها استفاده از یادگیری ماشین را به دلیل پتانسیل بالای آن برای پیش‌بینی‌های دقیق‌تر و تصمیم‌گیری‌های تجاری آغاز کرده‌اند.

با ادامه‌ی گسترش فناوری‌های جدید، می‌توان از الگوریتم‌های یادگیری ماشین بهره‌ورتری استفاده کرد. به عنوان مثال با پیشرفت پردازنده‌های گرافیکی در آینده، محاسبه داده‌های بزرگ‌تر با سرعت بیشتری امکان‌پذیر خواهد بود.

یادگیری ماشین همچنین پتانسیل تحقیقاتی بالایی دارد. دحال حاضر یکی از موضوعات داغ در مقالات تحقیقاتی حوزه‌ی علوم کامپیوتر، یادگیری ماشین است و از آن در صنایع و زمینه‌های تحقیقاتی مختلفی استفاده می‌شود. با این سرعت و افزایش نفوذ در بازار، یادگیری ماشین آینده درخشنانی خواهد داشت.

منابع و مراجع

- [1] Deisenroth, M.P., Faisal, A.A. and Ong, C.S., 2020. *Mathematics for machine learning*. Cambridge University Press.
- [2] James, G., Witten, D., Hastie, T. and Tibshirani, R., 2013. *An introduction to statistical learning* (Vol. 112, p. 18). New York: springer.
- [3] Bishop, C.M. and Nasrabadi, N.M., 2006. *Pattern recognition and machine learning* (Vol. 4, No. 4, p. 738). New York: springer.
- [4] Chen, L.P., 2019. Mehryar mohri, afshin rostamizadeh, and ameet talwalkar: Foundations of machine learning.