به نام خدا

تکلیف دوم- یادگیری ماشین کاربردی

استاد مربوطه: دکتر ناظرفرد

نام: زهرا اخلاقی

شماره دانشجویی: ۴۰۱۱۳۱۰۶۴

ایمیل: zahra.akhlaghi@aut.ac.ir

بهار ۱۴۰۲

فهرست مطالب

f) 28

g) 29

Q1

(a

مجموعه داده های نامتعادل مجموعه داده هایی هستند که در آنها توزیع کلاس ها برابر نیست، به این معنا که یک یا چند کلاس تعداد اعضای کمتری نسبت به بقیه دارند. چالش اصلی آن عبارتند از:

- یک مدل ممکن است به سمت کلاس اکثریت سوگیری داشته باشد و عملکرد ضعیفی در کلاس اقلیت داشته باشد.
- تعداد نمونه ها در کلاس اقلیت آموزش برای آموزش یک مدل بسیار کم است. این می تواند به بیش از حد برازش تبدیل شود، جایی که مدل نویز را به جای الگوهای اساسی یاد می گیرد.
- تعمیم ضعیف در مدل های جدید و دیده نشده.
- معیارهای ارزیابی غیرمعمول ممکن است برای مجموعه داده های نامتعادل مناسب نباشند، زیرا عدم تعادل کلاس را در نظر نمی گیرند. برای مثال، مدلی که همیشه کلاسیک اکثریت را پیش‌بینی می‌کند، ممکن است دقت بالایی داشته باشد، اما مدل مفیدی نیست.
- اگر مجموعه داده نامتعادل نماینده توزیع واقعی کلاس‌ها نباشد، ممکن است مدل روی داده‌های جدید و دیده نشده عملکرد خوبی نداشته باشد.

مجموعه داده های نامتعادل می توانند بر مدل تاثیرات، سوگیری نسبت به کلاس اکثریت، تطبیق بیش از حد، معیارهای عملکرد ضعیف، مشکل در آموزش را داشته باشد.

برای رفع این مشکل تکنیک های زیر وجود دارد:

- نمونه‌گیری مجدد: این روش یا کلاس اقلیت را بیش از حد نمونه‌گیری می‌کنند یا از کلاس اکثریت کمتر نمونه‌برداری می‌کنند تا توزیع طبقات متعادل شود.
- یادگیری حساس به هزینه: تخصیص هزینه بالاتر برای طبقه بندی اشتباه طبقه اقلیت است و مدل را نسبت به آن حساس تر می کند.
- روش مجموعه: ترکیب چندین مدل برای بهبود عملکرد کلی است، با تمرکز هر مدل بر جنبه‌های مختلف داده‌ها.
- تولید داده مصنوعی: تولید نمونه های مصنوعی برای کلاس اقلیت برای افزایش آن در مجموعه داده است.
- استفاده از الگوریتم: برخی از الگوریتم‌ها، مانند درخت‌های تصمیم‌گیری یا SVM، روش‌های خاص خود را برای مقابله با داده‌های نامتعادل دارند.

مجموعه داده های نامتعادل می تواند به طور قابل توجهی بر عملکرد مدل های یادگیری ماشین تأثیر بگذارد و استفاده از تکنیک های مناسب برای رسیدگی به این مسئله ضروری است.

(b

نرمال سازی داده ها می تواند تاثیر قابل توجهی بر عملکرد مدل های یادگیری ماشین داشته باشد که عبارتند از:

۱- میتواند به همگرایی سریعتر الگوریتم بهینه سازی کمک کند.
۲- احتمال خطاهای عددی را هنگام آموزش مدل کاهش دهد.
۳- دقت مدل را بهبود بخشد، به خصوص زمانی که ویژگی های ورودی مقیاس ها یا واحدهای متفاوتی داشته باشند.
۴- عملکرد تعمیم مدل را با کاهش تأثیر عوامل پرت و حساسیت کمتر مدل به تغییرات در مقیاس ویژگی های ورودی بهبود بخشد.
از تکنیک های نرمال سازی داده ها عبارتند از: Min-max scaling، Z-score scaling، Log transformation، Robust scaling، Power transformation، Unit vector scaling.
انتخاب تکنیک نرمال سازی می تواند تاثیر قابل توجهی بر نتایج یک مدل یادگیری ماشین داشته باشد. تکنیک‌های نرمال‌سازی مختلف می‌تواند منجر به دامنه‌ها و توزیع‌های متفاوتی از داده‌ها شود که می‌تواند روی همگرایی، سوگیری، تعمیم، حساسیت و تفسیرپذیری مدل تاثیر داشته باشد.
هنگام انتخاب یک تکنیک نرمال سازی برای یک مجموعه داده خاص، عواملی نظیر: مقیاس داده ها، توزیع داده ها، موارد پرت، تفسیرپذیری، حساسیت، پیچیدگی محاسباتی، دانش دامنه باید در نظر گرفته شود.

(c

در مرحله پیش پردازش داده ها میتوان تبدیل داده های دسته بندی به عددی را انجام داد. کلینیک های زیر برای encoding داده ها استفاده میشود، که عبارتند از:
One-Hot Encoding: این تکنیک زمانی استفاده می شود که یک متغیر تعداد کمی از مقادیر یا دسته های منحصر به فرد داشته باشد. در این تکنیک، یک ویژگی باینری جدید برای هر مقدار منحصر به فرد متغیر ایجاد می شود و در صورتی که مشاهده متعلق به آن دسته باشد، مقدار ویژگی باینری برابر با 1 و در غیر این صورت 0 تنظیم می شود.
این روش می تواند ابعاد فضای ویژگی را به میزان قابل توجهی افزایش دهد که می تواند منجر به "curse of dimensionality" شود،ا این حال، این روش می‌تواند روابط غیرخطی بین متغیر طبقه‌بندی و متغیر هدف را ثبت کند، که می‌تواند دقت مدل را بهبود بخشد.
Label Encoding: این تکنیک زمانی استفاده می شود که یک متغیر طبقه بندی دارای تعداد زیادی مقادیر یا دسته های منحصر به فرد باشد. در این تکنیک، به هر مقدار، یک مقدار صحیح منحصر به فرد اختصاص داده می شود. این روش می‌تواند منجر به پیش‌بینی‌های نادرست شود، به‌ویژه اگر سوگیری ترتیب ماهیت واقعی داده‌ها را منعکس نکند.
Target Encoding: این روش هر دسته از متغیرهای طبقه بندی را با میانگین مقدار هدف برای آن دسته جایگزین می کند. اگر تعداد مشاهدات برای هر دسته کم باشد، یا اگر متغیر هدف دارای نویز باشد، رمزگذاری هدف می‌تواند منجر به بیش از حد برازش شود.
انتخاب روش رمزگذاری برای متغیرهای طبقه بندی باید بر اساس مساله، داده های موجود و همچنین بر اساس نوع مدل مورد استفاده باشد و براساس آن از روش مناسب استفاده شود.
انتخاب تکنیک رمزگذاری برای متغیرهای طبقه‌بندی می‌تواند بر پیچیدگی محاسباتی و تفسیرپذیری یک مدل یادگیری ماشینی به روش‌های مختلف تأثیر بگذارد.

پیچیدگی محاسباتی: One-Hot Encoding می تواند ابعاد فضای ویژگی را به میزان قابل توجهی افزایش دهد، که می تواند منجر به افزایش پیچیدگی محاسباتی و زمان آموزش شود. با این حال، پیچیدگی محاسباتی ممکن است همچنان به اندازه داده ها و الگوریتم های خاص مورد استفاده بستگی داشته باشد.
تفسیرپذیری: One-Hot Encoding می تواند مدل را کمتر قابل تفسیر کند زیرا فضای ویژگی حاصل می تواند بسیار بزرگ و پراکنده باشد، که ممکن است درک اهمیت ویژگی های فردی را دشوار کند. رمزگذاری برچسب و رمزگذاری هدف می‌توانند تفسیرپذیرتر باشند، زیرا مقادیر عددی حاصل را می‌توان به راحتی به مقادیر طبقه‌بندی اصلی نگاشت و تفسیر روابط بین ویژگی‌ها و متغیر هدف را آسان‌تر می‌کند.
انتخاب تکنیک رمزگذاری باید بر اساس ویژگی‌های خاص مجموعه داده، مدل یادگیری ماشینی مورد استفاده و الزامات تفسیرپذیری مدل باشد. مهم است که هنگام انتخاب یک تکنیک تاثیرات آن به دقت در نظر گرفته شود.

(d

درخت تصمیم یک مدل ناپارامتریک است زیرا هیچ فرضی در مورد توزیع اساسی داده ها یا شکل عملکردی رابطه بین متغیرهای ورودی و خروجی ندارد و درخت تصمیم، هیچ شکل عملکردی خاصی را به خود نمی‌گیرند و در برازش روابط پیچیده بین متغیرهای ورودی و متغیر خروجی انعطاف‌پذیر است،
در یک درخت تصمیم، ساختار درخت با تقسیم‌بندی بازگشتی داده‌ها بر اساس مقادیر متغیرهای ورودی تعیین می‌شود و تقسیم‌بندی‌های حاصل بر اساس معیارهایی از افزایش اطلاعات یا کاهش ناخالصی انتخاب می‌شوند. درخت حاصل را می توان رابطه بین متغیرهای ورودی و متغیر خروجی در نظر گرفت.

(e

انتخاب معیارهای تقسیم برای درخت تصمیم بستگی به مساله و نوع داده مورد استفاده دارد. چندین معیار تقسیم رایج وجود دارد که در الگوریتم های درخت تصمیم استفاده می شود، از جمله:
Gini impurity: این معیار اندازه‌گیری می‌کند که اگر یک عنصر به‌طور تصادفی بر اساس توزیع برچسب‌ها در زیرمجموعه برچسب‌گذاری شود، چند بار به اشتباه برچسب‌گذاری می‌شود.
Entropy: میزان عدم قطعیت یا تصادفی بودن توزیع کلاس نمونه ها را در گره اندازه گیری می کند.
Classification error: این معیار میزان ناخالصی یک گره بر اساس نرخ طبقه‌بندی اشتباه نمونه‌های موجود در گره است که نسبت نمونه هایی را در گره که به اشتباه طبقه بندی شده اند اندازه گیری می کند.
انتخاب معیار تقسیم می تواند به ویژگی های داده ها و مسئله بستگی داشته باشد. به طور کلی، دو مورد اول بیشتر استفاده می شوند، زیرا نسبت به تغییرات توزیع کلاس حساس‌تر هستند و می‌توانند منجر به درخت‌های تصمیم گیری دقیق‌تر و قوی‌تر شوند.

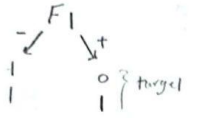
(f

F) Entropy(S) = $-\sum p_i * \log p_i$ 4011310 64 زهرا اخلاقی

information gain (F) = Entropy(S) − $\sum \frac{|Sa(v)|}{|S|} * Entropy(Sa(V))$

target برای Entropy : $Entropy(3,1) = -\frac{3}{4}\log(\frac{3}{4}) - \frac{1}{4}\log(\frac{1}{4}) = ,5623$

→ information gain for F1 :



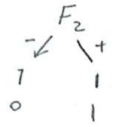
F_1 :− برای زیرشاخه entropy : entropy(0/2, 2/2) = 0

F_1 :+ " " " : entropy(1/2, 1/2) = −1/2 log(1/2) − 1/2 log(1/2) = 1

Averge info of subtree (weighted) = 1/2 × 0 + 1/2 × 1 = 1/2

information gain(S, F_1) = ,5623 − ,5 = ,0623

→ information gain for F_2 :



F_2 = + برای زیرشاخه entropy : entropy(2/2, 2/2) = 0

F_2 = − " " " " : entop(1/2, 1/2) = 1

Averge inform → 1/2 × 0 + 1/2 × 1 = 1/2

information gain(S, F_2) = ,5623 − ,5 = ,0623

→ information gain for F_3 :

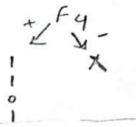
F_3 = + برای زیرشاخه entropy : 0

F_3 = − برای زیرشاخه entropy : entropy(2/3, 1/3) = −1/3 log(1/3) − 2/3 log(2/3) = 9b3

Averge information = 3/4 × ,63 + 1/4 × 0 = ,4725

information gain (S, F_3) = ,5623 − ,4725 = ,0898

information gain For F_4 :

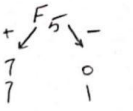


F_4 :+ برای زیرشاخه entropy : entropy(3/4, 1/4) = ,5623

Averge information = 4/4 × ,5623 + 0 × 0 = ,5623

information gain(S, F_4) = 0

→ information gain For F_5 :



F_5 = + entropy → entropy(2/2, 2/2) = 0

F_5 = − " → entropy(1/2, 1/2) = 1

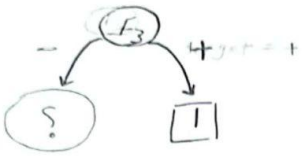
Averge infor of subtree = 1/2

information gain (S, F_5) = ,56 − ,5 = ,06

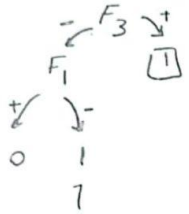
$\arg\max(0.6, 0.6, 0, 0.6, 0.8, 0.6) = 0.8 \rightarrow F_3$

ماکزیمم مقدار بین information gain ها برای ستون های مختلف مربوط به ستون F_3 است.

پس شاخه‌ی اول با F_3 تقریب می‌گردد



همین مراحل را برای شاخه‌ی $F_3 = -$ انجام می‌دهیم



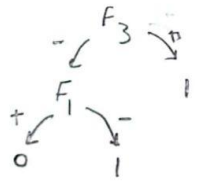
$F_1 = -$ برای زیرشاخه entropy $\rightarrow 0$

$F_1 = +$ " " " $\rightarrow 0$ $\Big\} \rightarrow$ Average information $\rightarrow 0$

information gain $(S_1, F_3) = 1.5623 - 0 = 1.5623$

** با توجه به اینکه این شاخه خالص‌ترین تقسیم‌بندی ممکن را دارد نیازی به بررسی تقسیم‌بندی ستون‌های باقیست.

نتیجه‌ی نهایی به صورت زیر است.



(g

g) تکنیک pruning در الگوریتم decision tree استفاده می‌شود که برای کاهش اندازه‌ی درخت به وسیله‌ی کاهش شاخه‌های اضافی به کار می‌رود به طوری که تاثیری زیادی روی دقت نهایی نداشته باشد.

همینطور تکنیک pruning می‌تواند نتایج نهایی decision tree را با کاهش overfit شدن، بهبود دهد.

overfit در دقت زیادی رخ می‌دهد که دیتا بسیار پیچیده باشد و مدل کاملاً با دیتا fit شده باشد. در این حالت مدل decision tree تمامی دیتای مورد نظر را به نوعی حفظ کرده است اما اگر ورودی دیتای دیگری آن را آزمایش کنیم ممکن است نتیجه‌ی مطلوبی به ما ندهد و pruning به ما کمک می‌کند که از overfit شدن جلوگیری کنیم. البته pruning همیشه باعث افزایش دقت نتیجه‌گیری نهایی نمی‌شود در برخی موارد حذف کردن برخی شاخه‌ها باعث کاهش دقت مدل decision tree می‌شود.

ما در قسمت f از این تکنیک pruning استفاده کرده‌ایم، به این دلیل که زمانی که با F_3 شاخه‌بندی می‌کنیم اگر F_3 + باشد target همواره یک خواهد بود درنتیجه نیازی به ادامه‌ی این شاخه دیگر نیازی نیست. همین طور در شاخه‌بندی‌های بعد از F_1 دیگر نیازی به ادامه‌ی شاخه‌بندی نیست زیرا با F_1 که منفی باشد به یک می‌رسیم و با F_1 + همواره به صفری می‌رسیم. و دیگر نیازی به ادامه دادن در شاخه درخت نیست. همینطور اگر درخت بزرگتر و شاخه‌ها بیشتر و دیتا بزرگتر باشد، شاخه‌هایی که خلوص آن‌ها از یک مقدار معین بیشتر دیگر نیازی به ادامه دادن شاخه و رسیدن به خلوص صد درصدی در برگ درخت نیست.

(h

درخت های تصمیم را می توان برای مسائل طبقه بندی چند طبقه استفاده کرد. در واقع درخت های تصمیم یکی از محبوب ترین و موثرترین روش ها برای multi-class classification هستند. در یک درخت تصمیم برای طبقه بندی

چند کلاسه با پارتیشن بندی بازگشتی داده ها بر اساس مقادیر متغیرهای ورودی ساخته می شود، به طوری که هر گره داخلی یک قانون تصمیم را بر اساس یکی از متغیرهای ورودی نشان می دهد و هر برگ نشان دهنده یک برچسب کلاس است.

چندین استراتژی برای ساخت درخت‌های تصمیم برای multi-class classification ه وجود دارد، از جمله رویکرد one-vs-all، که در آن یک درخت تصمیم باینری جداگانه برای هر کلاس ساخته می‌شود، و رویکرد one-vs-one. که در آن یک درخت تصمیم باینری جداگانه برای هر جفت کلاس ساخته شده است.

(i

بهره اطلاعات و نسبت بهره دو معیار تقسیم رایج هستند که در الگوریتم های درخت تصمیم برای تعیین بهترین راه برای تقسیم داده ها در هر گره استفاده می شوند. تفاوت اصلی بین افزایش اطلاعات و نسبت بهره در این است که بهره اطلاعات فقط کاهش آنتروپی حاصل از تقسیم داده ها بر اساس یک ویژگی خاص را در نظر می گیرد، در حالی که نسبت بهره اطلاعات ذاتی ویژگی و تعداد شاخه هایی را که ایجاد می کند در نظر می گیرد.

در عمل، هم نسبت به دست آوردن اطلاعات و هم نسبت بهره می توانند به عنوان معیارهای تقسیم در الگوریتم های درخت تصمیم مورد استفاده قرار گیرند و انتخاب بین آنها به مشکل خاص و ویژگی های داده ها بستگی دارد.

(j

درخت های تصمیم را می توان برای مسائل رگرسیون چند خروجی با ساختن یک درخت تصمیم جداگانه برای هر متغیر خروجی استفاده کرد. چندین استراتژی برای ساخت درخت تصمیم برای مسائل رگرسیون چند خروجی وجود دارد، که عبارتند از:

رویکرد مستقل: هر درخت تصمیم با استفاده از یک الگوریتم رگرسیون استاندارد، مانند CART یا C4.5 ساخته می‌شود و پیش‌بینی‌های هر متغیر خروجی با دنبال کردن مسیری در درخت تصمیم مربوطه که به گره برگ منتهی می‌شود، به دست می‌آید.

رویکرد مشترک: از یک الگوریتم رگرسیون اصلاح‌شده استفاده می‌شود که می‌تواند چندین متغیر خروجی را به طور همزمان مدیریت کند، مانند Random Forest یا Gradient Boosting. در این مورد، یک درخت تصمیم واحد برای پیش‌بینی همه متغیرهای خروجی با هم، با استفاده از یک معیار تقسیم مشترک که همبستگی بین متغیرهای خروجی را در نظر می‌گیرد، ساخته می‌شود.

(k

CART و C4.5 از جهات زیر با یکدیگر متفاوت هستند:

نوع درخت: CARTیک درخت باینری است، به این معنی که هر گره داخلی دقیقاً دو شاخه دارد، در حالی که C4.5 هر گره داخلی می تواند بیش از دو شاخه داشته باشد.

معیار تقسیم: CART از شاخص جینی به عنوان معیار تقسیم خود برای مسائل طبقه بندی و میانگین مربعات خطا به عنوان معیار تقسیم خود برای مسائل رگرسیونی استفاده می کند، در حالی که C4.5 از نسبت به دست آوردن اطلاعات به عنوان معیار تقسیم خود برای مسائل طبقه بندی و رگرسیون استفاده می کند.

مدیریت مقادیر از دست رفته: CART می‌تواند مقادیر از دست رفته را با تخصیص آن‌ها به پرتکرارترین کلاس در داده‌های آموزشی یا با تخصیص تصادفی آنها به کلاس‌های موجود، مدیریت کند.

هرس: CART از تکنیکی به نام هرس هزینه-پیچیدگی برای جلوگیری از تطبیق بیش از حد درخت تصمیم استفاده می کند، در حالی که C4.5 از روشی به نام "هرس خطای کاهش یافته" استفاده می کند که دقت درخت تصمیم را در مجموعه اعتبارسنجی ارزیابی می کند و شاخه هایی را که این کار را نمی کنند حذف می کند. دقت را بهبود بخشد

مدیریت متغیرهای پیوسته: CART می تواند متغیرهای پیوسته را با انتخاب مقدار آستانه و تقسیم داده ها به دو گروه بر اساس اینکه مقدار متغیر بالاتر یا پایین تر از آستانه است، مدیریت کند، در حالی که C4.5 از روشی به نام "گسسته سازی" استفاده می کند.

(l

درخت‌های تصمیم و سیستم‌های مبتنی بر قانون هر دو مدل‌های یادگیری ماشینی هستند که برای کارهای طبقه‌بندی و پیش‌بینی استفاده می‌شوند، اما در رویکردشان به مدل‌سازی و نمایش فرآیند تصمیم‌گیری متفاوت هستند.

درخت‌های تصمیم‌گیری فرآیند تصمیم‌گیری را به‌عنوان ساختاری درخت‌مانند نشان می‌دهند، که در آن هر گره داخلی یک تصمیم را بر اساس ویژگی یا ویژگی خاصی از داده‌ها نشان می‌دهد و هر گره برگ نشان‌دهنده یک کلاس یا مقدار پیش‌بینی‌شده است. درخت با تقسیم بازگشتی داده ها بر اساس ویژگی انتخاب شده که به بهترین وجه داده ها را به کلاس ها جدا می کند، ساخته می شود تا زمانی که یک معیار توقف برآورده شود. درختان تصمیم اغلب برای تفسیرپذیری استفاده می شوند.

در مقابل، سیستم‌های مبتنی بر قانون، فرآیند تصمیم‌گیری را به‌عنوان مجموعه‌ای از قوانین if-then نشان می‌دهند، که در آن هر قانون یک شرط را در متغیرهای ورودی و یک اقدام یا خروجی مربوطه را مشخص می‌کند. قوانین با تجزیه و تحلیل داده ها و استخراج الگوهایی که نشان دهنده کلاس ها یا نتایج مختلف هستند ساخته می شوند. سیستم‌های مبتنی بر قانون اغلب برای انعطاف‌پذیری‌شان استفاده می‌شوند.

(m

در درخت‌های تصمیم، عمق درخت می‌تواند تأثیر قابل‌توجهی بر مبادله بایاس واریانس داشته باشد. مبادله بایاس-واریانس به تعادل بین توانایی مدل برای گرفتن الگوهای اساسی در داده ها (بایاس) و حساسیت آن به نوسانات در داده های آموزشی (واریانس) اشاره دارد.

هنگامی که یک درخت تصمیم خیلی کم عمق باشد، ممکن است سوگیری بالایی داشته باشد، به این معنی که ممکن است نتواند الگوهای اساسی در داده ها را به خوبی ثبت کند. با این حال، ممکن است واریانس کمی داشته باشد، به این معنی که ممکن است نسبت به نوسانات در داده های آموزشی حساسیت کمتری داشته باشد. برعکس، وقتی درخت تصمیم

خیلی عمیق است، ممکن است بایاس کم داشته باشد، به این معنی که ممکن است بتواند الگوهای اساسی در داده ها را به خوبی ثبت کند. با این حال، ممکن است واریانس بالایی داشته باشد، به این معنی که ممکن است به نوسانات در داده های آموزشی بسیار حساس باشد.

یکی از روش‌های رایج برای یافتن عمق درخت بهینه، استفاده از تکنیکی به نام «هرس» است که شامل رشد درخت بزرگ‌تر و سپس حذف شاخه‌هایی است که عملکرد درخت را در مجموعه اعتبارسنجی بهبود نمی‌بخشند. با انجام این کار، درخت هرس شده به دست آمده ممکن است واریانس کمتر و سوگیری کمی بالاتر داشته باشد و در نتیجه عملکرد کلی بهتری در داده‌های جدید و دیده نشده داشته باشد.

Q2

(a

برای دسته بندی داده ها در prob_cancar ابتدا مقادیر null در این ستون را با مقدار مناسب جایگذاری کردم، سپس توزیع داده را بررسی کرده و داده ها را به بازه هایی ۰.۲ تقسیم کردم که ۵ دسته ایجاد شد(داده ها بین ۰و۱ قرار داشتند)و به ترتیب برای آنها لیبل های زیر را در نظر گرفتم:

"Very Low", "Low", "Moderate", "High", "Very High"

	weight	height	salads_per_week	veggies_fruits_per_day	healthy_diet	aerobic_per_week	sports_per_week	current_smoking	survey.month	category_cancer
5	140.0	69.0	0.0	NaN	Below average	2.0	0.0	Never	2008.09	Very Low
6	150.0	67.0	2.0	1.0	Below average	3.0	3.0	Never	2008.09	Low
9	105.0	66.0	0.0	2.0	Average	1.0	0.0	Never	2008.09	Very High
7	220.0	77.0	2.0	5.0	Very healthy	5.0	5.0	Never	2008.09	Moderate
2	135.0	62.0	0.0	1.0	Unhealthy	0.0	0.0	Never	2008.09	Very Low

(b

یکی از مشکلات ستون 'height' این است که شامل تعداد زیادی null است، همانطور که قبلا دیدیم که می تواند استفاده موثر از این ستون را دشوار کند. مسئله دیگر این است که مقادیر null در ستون "height" را نمی توان به راحتی با استفاده از یک استراتژی ساده مانند میانه و میانگین نسبت داد.

یکی از راه‌حل‌های ممکن برای این مشکل این است که با استفاده از ستون‌های دیگر در مجموعه داده‌ها که ممکن است با ارتفاع مرتبط باشند، مقادیر null را در ستون «height»نتیجه گرفت. به عنوان مثال، وزن اغلب با قد همبستگی زیادی دارد و ما می توانیم با استفاده از رگرسیون خطی از این رابطه برای تخمین مقادیر قد از دست رفته استفاده کنیم.

```
df['height'].isnull().sum()
df['height']
```

```
0      69.0
1      67.0
2      66.0
3      77.0
4      62.0
       ...
379    75.0
380    77.0
382    70.0
383    66.0
384    71.0
Name: height, Length: 375, dtype: float64
```

(c

با مشاهده مقادیر current_smoking مقادیر پرت و null را حذف کردم

```
Never             355
Once in a while    10
0                   1
5                   1
2                   1
Some days           1
Name: current_smoking, dtype: int64
```

با توجه به مقادیر بالا به نظر میرسد current_smoking داده توصیف پذیر است و مقادیر ۰و۵و۲ پرت هستند و باید پاک شوند

```
df['current_smoking'].value_counts()
```

```
Never             355
Once in a while    10
Some days           1
Name: current_smoking, dtype: int64
```

(d

مقادیر از نوع object زا به int با استفاده از map تبدیل کردم.

```
]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 366 entries, 0 to 384
Data columns (total 11 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   user_id                 366 non-null    int64
 1   weight                  366 non-null    float64
 2   height                  366 non-null    float64
 3   salads_per_week         359 non-null    float64
 4   veggies_fruits_per_day  358 non-null    float64
 5   healthy_diet            360 non-null    float64
 6   aerobic_per_week        365 non-null    float64
 7   sports_per_week         365 non-null    float64
 8   current_smoking         366 non-null    int64
 9   survey.month            366 non-null    float64
 10  category_cancer         366 non-null    category
dtypes: category(1), float64(8), int64(2)
memory usage: 32.0 KB
```

```
df.fillna(df.mean(), inplace=True)
```

```
/tmp/ipykernel_12029/820435583.py:1: FutureWarning: The
ted. In a future version, it will default to False. In
ect only valid columns or specify the value of numeric_
  df.fillna(df.mean(), inplace=True)
```

```
df.isnull().sum()
```

```
user_id                   0
weight                    0
height                    0
salads_per_week           0
veggies_fruits_per_day    0
healthy_diet              0
aerobic_per_week          0
sports_per_week           0
current_smoking           0
survey.month              0
category_cancer           0
dtype: int64
```

(e

با مشاهده corr و وابستگی میان داده ها، ستون های user_id, survey.month را به عنوان ستون های اضافه حذف کردم.

(f

```
train: (256, 8)
test: (110, 8)
```

(h

```
Accuracy: 0.20909090909090908
Confusion matrix:
[[ 7  6  7  0  3]
 [12  6  5  1  3]
 [ 4  6  6  1  1]
 [ 6  5  5  1  1]
 [ 9  6  5  1  3]]
R2 score: -0.9608303037410582
```

(i

برای پیش پردازش داده ها:

۱. داده ها در prob_cancer رو به ۵ دسته تقسیم کردم و مقادیر null را در این دسته حذف کردم.

۲. ستون های user_id,survey.month را حذف کردم.

۳. مقادیر nullدر ستون height را به میانگین آنها جایگذاری کردم.

۴. ستون های healthy_diet و current_smoking را به مقدار عددی تبدیل کردم.

۵. مقادیر null باقی مانده در بقیه ستون ها را با میانگین جایگذاری کردم.

۶. از MinMaxScaler روی همه داده ها استفاده کردم

```
Accuracy: 0.7699115044247787
Confusion matrix:
[[23  5  0  0  0]
 [ 7 15  2  0  0]
 [ 0  4 19  0  0]
 [ 0  0  2 12  3]
 [ 0  0  0  3 18]]
R2 score: 0.8884586180713743
```

Q3

زمینه پژوهش من الگوریتمی برای بررسی ترافیک شهری با استفاده از پیش بینی حرکات رانندگان می باشد بنابراین دیتاست پیشنهادی من به صورت زیر است:

(a

برای این سوال از دیتاست Dataset behavior Driving در سایت data mendeley استفاده شده است و حاوی 1114 سطر و 7 ستون است. که ستون های آن شامل شتاب ماشین درسه محور x, y, z, Accz, Accy) Accx,)است و جهت گیری و سرعت زاویه ماشین که می تواندچرخش ماشین را تشخیص دهد حول سه محور zy, x, .(Gyrox,Gyroz به Gyroy, است.) همراه یک ستون(class)که در آن یکی از چهار حالت زیر را نشان مي دهد:

1. شتاب ماشین به صورت ناگهاني افزایش پیدا کرده و سرعت آن زیاد می شود.
2. به سمت چپ چرخش ناگهاني دارد .
3. به سمت راست چرخش ناگهاني دارد .
4. یا به صورت ناگهاني شتاب آن کاهش پیدا میکند و ماشین به مانع برخورد میکند.

این اطلاعات در شرایطی جمع آوری شده که وضعیت بارندگي نبوده و سطح جاده خشک بوده است. همینطور حسگرها برای سنجش شتاب و سرعت زاویه ایی بر روی داشبورد ماشین قرار داشته و توسط 3 راننده جمع آوری شده است. قسمتي از داده ها به شکل زیر است :

(b

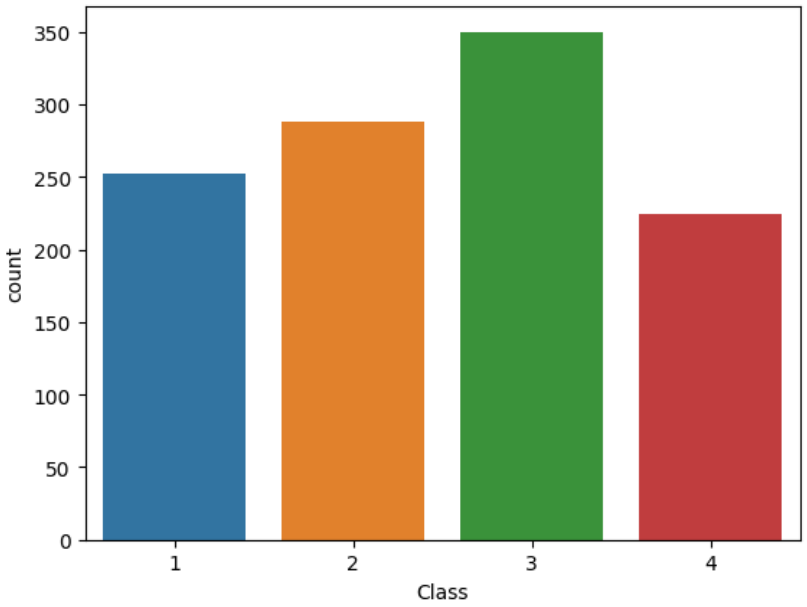
```
df = pd.read_csv('sensors.csv')
df.head(7)
```

	Class	GyroX	GyroY	GyroZ	AccX	AccY	AccZ
0	1	-0.923664	3.694656	0.824427	0.162598	-0.086670	-0.969482
1	1	-0.908397	4.534351	0.832061	0.175781	-0.100586	-1.013184
2	1	0.786260	3.969466	0.587786	0.322754	-0.140381	-0.911621
3	1	0.335878	4.564885	-0.251908	0.480225	-0.226807	-0.936768
4	1	3.351145	2.694656	-0.106870	0.426025	-0.253906	-0.950195
5	1	-1.503817	3.183206	-1.656489	0.383789	-0.141602	-0.919678
6	1	1.358779	8.725191	-0.946565	0.404785	-0.257324	-0.862549

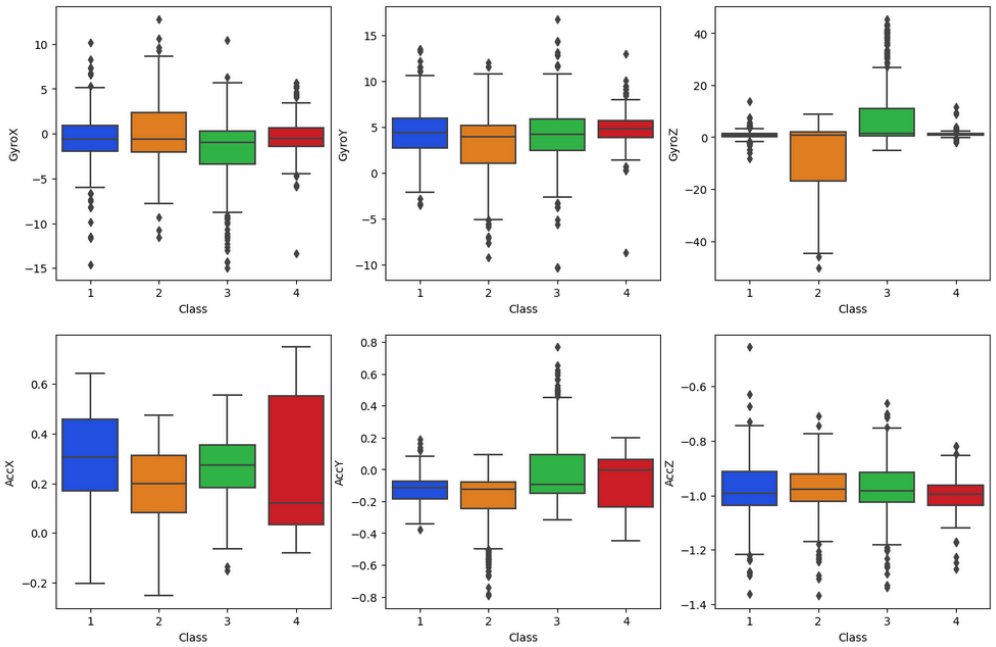
(c

برای آشنا شدن بیشتر با دیتاست و درک بهتر آن سراغ مصور سازی میرویم شکل زیرنشان میدهد که هر یک از

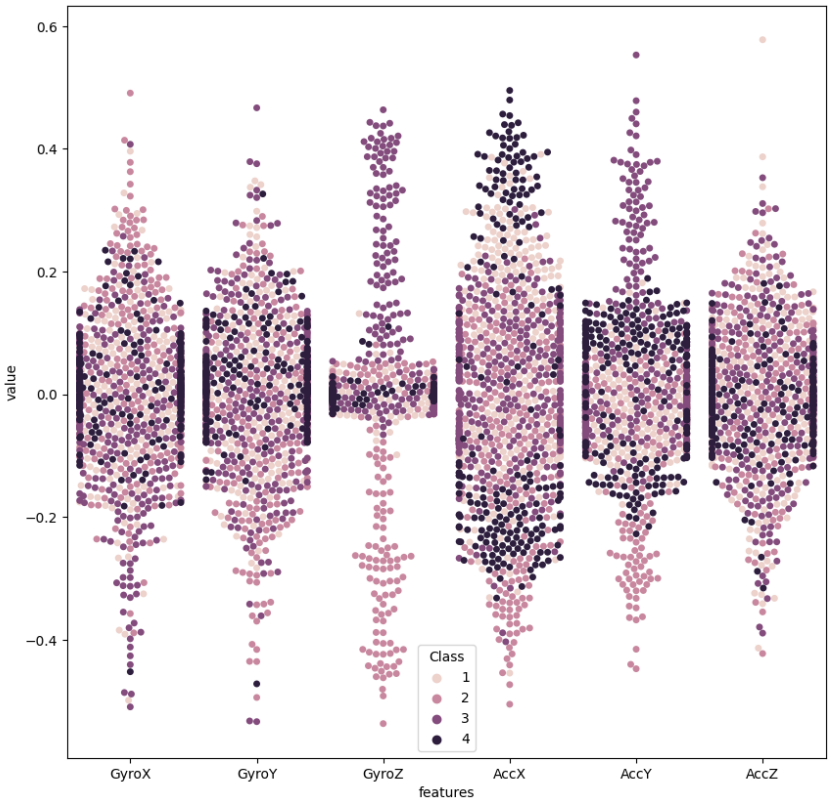
چهارکلاس گفته شده چه تعداد نمونه داریم .



شکل‌زیر نمودار باکس پلات هر کلاس را به تفکیک نشان میدهد و در هر کدام به تفکیک داده هایoutlier آن مشخص است مثال برای ستون accx فقط کلاس 3 دارای داده پرت است.



شکل زیر نمونه های مربوط به هر کلاس را به تفکیک نشان میدهد.نمونه هایي باعث ایجاد مشکل در مراحل بعدی و طبقه بندی ما میشود پس در مرحله پیش پردازش داده یک سری داده های outlier را با میانگین داده های همان کلاس در ستون جایگزین میکنیم تا در طبقه بندی ما دچار مشکل نشود.



همانطورکه در قسمت قبل گفته شد برخي از داده های پرت در طبقه بندی ما باعث ایجاد مشکل میشوند مثلا باتوجه به شکل بالا کلاس 3 فقط یک نمونه از آن در ستون ACCX کمتر از -دارد 0.4 و ... این داده ها را بامیانگین نمونه های دیگر کلاس 3 در ستون Accx جایگزین می کنیم بخشي از خروجي کد به شکل زیر است:

```
1 GyroX
Q1 , Q2 , Q3 , IQR :  -2.030534351 -0.618320611 0.8969465645 2.9274809155
outlier data : [10.15267176, 6.732824427, 7.290076336, 5.351145038, 6.58778626, 7.34351145, 8.267175573, -7.37404580
2, -8.167938931, -11.66412214, -6.709923664, -7.541984733, -11.48091603, -9.839694656, -14.64885496, -6.641221374, -
8.221374046]


4 GyroX
Q1 , Q2 , Q3 , IQR :  -1.4694656485 -0.561068702 0.6145038165 2.083969465
outlier data : [5.160305344, 4.503816794, 5.160305344, 5.290076336, 4.06870229, 5.679389313, 5.312977099, 4.11450381
7, 4.641221374, 5.633587786, 4.305343511, -13.35877863, -4.702290076, -5.877862595, -5.847328244, -5.717557252, -4.6
94656489]


4 GyroY
Q1 , Q2 , Q3 , IQR :  3.8358778625 4.847328244 5.6679389315 1.8320610690000003
outlier data : [10.13740458, 9.091603053, 9.122137405, 8.465648855, 12.98473282, 8.580152672, 9.480916031, 8.6564885
5, 8.770992366, 9.442748092, 0.351145038, -8.679389313, 0.671755725, 0.290076336, 0.702290076]


4 GyroZ
Q1 , Q2 , Q3 , IQR :  0.667938931 0.977099237 1.3435114505 0.6755725195000001
outlier data : [9.335877863, 2.72519084, 3.213740458, 2.732824427, 2.389312977, 11.6259542, 3.152671756, 2.78625954
2, 2.763358779, 3.847328244, 2.664122137, 2.816793893, 3.763358779, 8.893129771, -1.938931298, -0.404580153, -1.2519
08397, -1.916030534, -1.145038168, -0.458015267, -0.58778626]


3 AccX
Q1 , Q2 , Q3 , IQR :  0.18359375 0.27465820350000003 0.3524169925 0.16882324250000003
outlier data : [-0.136230469, -0.150878906]
```

نرمالسازی داده ها باعث میشود که همه‌ی داده ها در یک رنج قرار بگیرند و بتوان ستون های مختلف رابا هم مقایسه کرد و StandardScaler در کتابخانه sklearn برای نرمال سازی همه ی ستون ها به جزستون class استفاده شده است.

```
array([[-0.03170665, -0.13015507, -0.01861359, -0.49672404,  0.04434692,
         0.14009952],
       [-0.02685281,  0.13389953, -0.01797916, -0.42440636, -0.02877161,
        -0.3057532 ],
       [ 0.51192402, -0.0437372 , -0.03828114,  0.38180183, -0.23786493,
         0.73041848],
       [ 0.36873558,  0.14350152, -0.10806921,  1.24559632, -0.69196946,
         0.47386636],
       [ 1.32737003, -0.4446201 , -0.09601491,  0.94829031, -0.83435816,
         0.33687251]])
```

(d

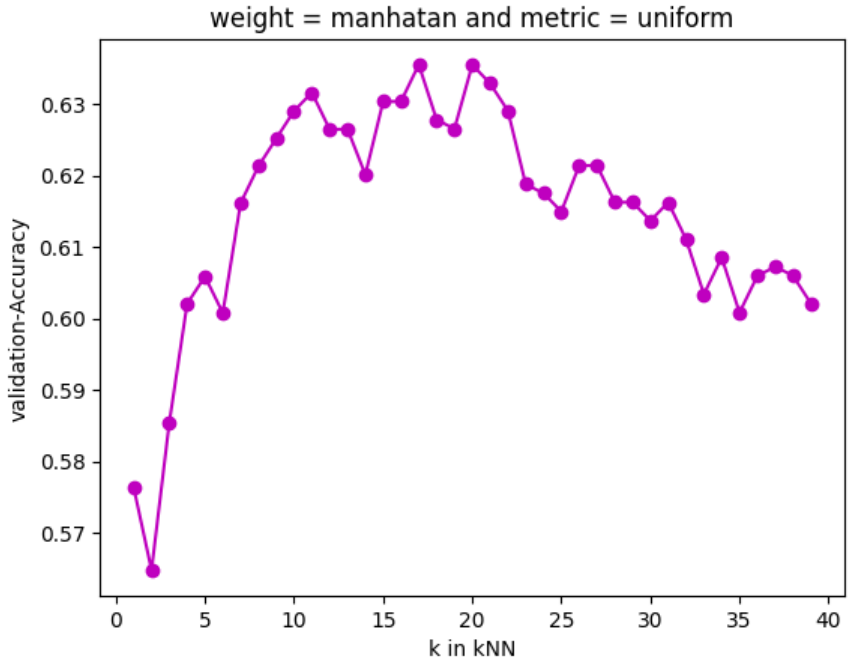
برای اجرای الگوریتم های طبقه بندی نیاز است که داده ها را به سه دسته validation, test, train, دسته بندی کرد داده هایtrain را برای آموزش و داده یvalidation را برای ارزیابي مدل استفاده میکنیم، و در انتها مدل را با داده تست می سنجیم.

برای این کار از متد train_test_split در کتابخانه sklearn استفاده شده است و دیتاست را به نسبت 70– 30 تقسیم بندی میکنیم:

```
Train set: (779, 6) (779,)
Test set: (335, 6) (335,)
```

پیاده سازی الگوریتم KNN

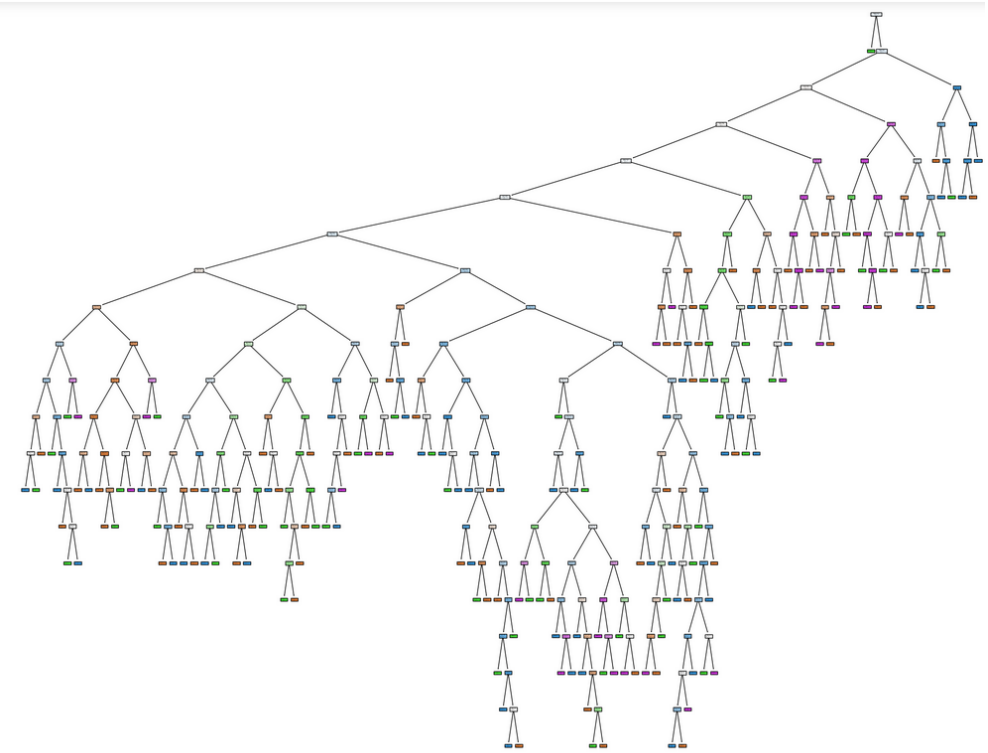
یکی از الگوریتم های پیاده سازی شده روی این مجموعه داده knn می باشد، در این مسئله الگوریتم knn را به ازای پارامتر های مختلف امتحان کردم و در نهایت بهترین آن پارامتر ها را انتخاب کردم و روی مجموعه داده آزمایش استفاده کردم.



```
Accuracy: 0.5880597014925373
Confusion matrix:
[[26 17 21 12]
 [10 50 20  7]
 [15 21 65  4]
 [ 6  2  3 56]]
R2 score: -0.12797345089869672
```

پیاده سازی الگوریتم Decision Tree

یکی دیگر از متد های به کار گرفته شده روی این مجموعه داده dessision tree میباشد.



```
Accuracy: 0.564179104477612
Confusion matrix:
[[31 17 15 13]
 [11 51 21  4]
 [26 22 52  5]
 [ 8  3  1 55]]
R2 score: -0.23887731537554702
```

Q4

(a

Out[132]:

	f1	f2	f3	f4	f5	f6	target
0	M01	A	20	N	N	Y	+
1	M01	A	20	?	?	?	+
2	M01	A	30	Y	Y	Y	+
3	M01	A	50	N	Y	Y	+
4	M01	A	55	Y	Y	N	+

برای پیش پردازش داده ها ستون هایی که شامل مقدار ؟ میشدند را حذف کردم، مقادیر از نوع object در هر ستون را با مشاهده مقادیر آنها از طریق value_count با مقدار عددی جایگذاری کردم.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51 entries, 0 to 64
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   f1      51 non-null     int64
 1   f2      51 non-null     int64
 2   f3      51 non-null     int64
 3   f4      51 non-null     int64
 4   f5      51 non-null     int64
 5   f6      51 non-null     int64
 6   target  51 non-null     int64
dtypes: int64(7)
memory usage: 3.2 KB
```

(b

نتخاب روش برای تقسیم داده ها به مجموعه های آموزشی و تست به مجموعه داده های خاص و مشکل موجود بستگی دارد. هر دو روش hold-out و cross-validation مزایا و معایب خود را دارند و انتخاب باید بر اساس اندازه مجموعه داده، پیچیدگی و هدف تجزیه و تحلیل انجام شود. به طور کلی، زمانی که مجموعه داده بزرگ است و منابع محاسباتی محدود است، روش hold-out برای پیاده سازی ساده است و می تواند تخمین خوبی از عملکرد مدل ارائه دهد. با این حال، ممکن است برای مجموعه داده های کوچک بهینه نباشد، زیرا مجموعه تست حاصل ممکن است از نظر آماری معنی دار نباشد. از سوی دیگر، cross-validation انتخاب بهتری برای مجموعه داده‌های کوچک است، جایی که خطر بیش از حد برازش یا عدم تناسب مدل وجود دارد. این روش تخمین دقیق تری از عملکرد مدل ارائه می دهد و

می تواند به شناسایی مسائل بالقوه مدل، مانند سوگیری یا واریانس کمک کند. با این حال، می تواند از نظر محاسباتی گران باشد، به خصوص زمانی که مجموعه داده بزرگ باشد.

برای این مجموعه داده با توجه به اینکه تعداد محدودی دارد استفاده از cross validation میتواند انتخاب بهتری باشد.

(e

```
Accuracy: 0.8181818181818182
Precision: 0.9
Recall: 0.9
F1 score: 0.9
```

(f

(g

```
Accuracy: 0.8181818181818182
Precision: 0.9
Recall: 0.9
F1 score: 0.9
```

نتیجه در مقایسه با حالت قبل تغییری ایجاد نشد.

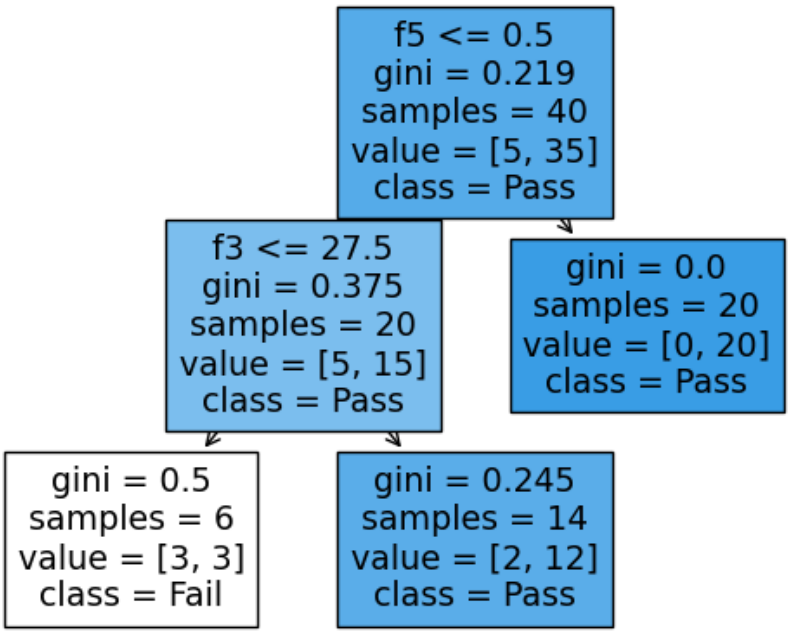
(h

```
{'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 10}
```

(i

```
Accuracy: 0.8181818181818182
Precision: 0.9
Recall: 0.9
```

```
_ = tree.plot_tree(ct3,
                   feature_names= X.columns,
                   class_names=['Fail', 'Pass'],
                   filled=True)
```




```
from sklearn.metrics import  classification_report

print(classification_report(y_test, y_pred, target_names=['fail', 'pass']
```

```
              precision    recall  f1-score   support

        fail       0.00      0.00      0.00         1
        pass       0.90      0.90      0.90        10

    accuracy                           0.82        11
   macro avg       0.45      0.45      0.45        11
weighted avg       0.82      0.82      0.82        11
```

ارزیابی در بالا نشان داده شده است، که دارای دقت ۰.۸ میباشد.

Q5

(a

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

برای پیش پردازش داده ها :

در ابتدا ستون های Ticketو Name و PassengerId و Name که تاثیری روی خروجی ندارد را حذف کردم.

مقادیر null ، در ستون age برابر ۱۱۷ می باشد، با استفاده از میانگین و انحراف معیار با مقدار مناسبی جایگذاری کردم و دیتای خالی در ستون Embarked با S که دارای بیشترین تعداد میباشد جایگذاری کردم. (فقط همین دو ستون دارای مقدار null بودند).

نتیجه دیدن تعداد داده های null بعد از این پردازش:

```
: Survived    0
  Pclass      0
  Sex         0
  Age         0
  SibSp       0
  Parch       0
  Fare        0
  Embarked    0
  dtype: int64
```

- ستون Embarked , Sex را به مفدار عددی با استفاده از map جایگذاری کردم.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    int64
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64
 5   Parch     891 non-null    int64
 6   Fare      891 non-null    float64
 7   Embarked  891 non-null    int64
dtypes: float64(2), int64(6)
memory usage: 55.8 KB
```

(b

```
train: (712, 7)
test: (179, 7)
```

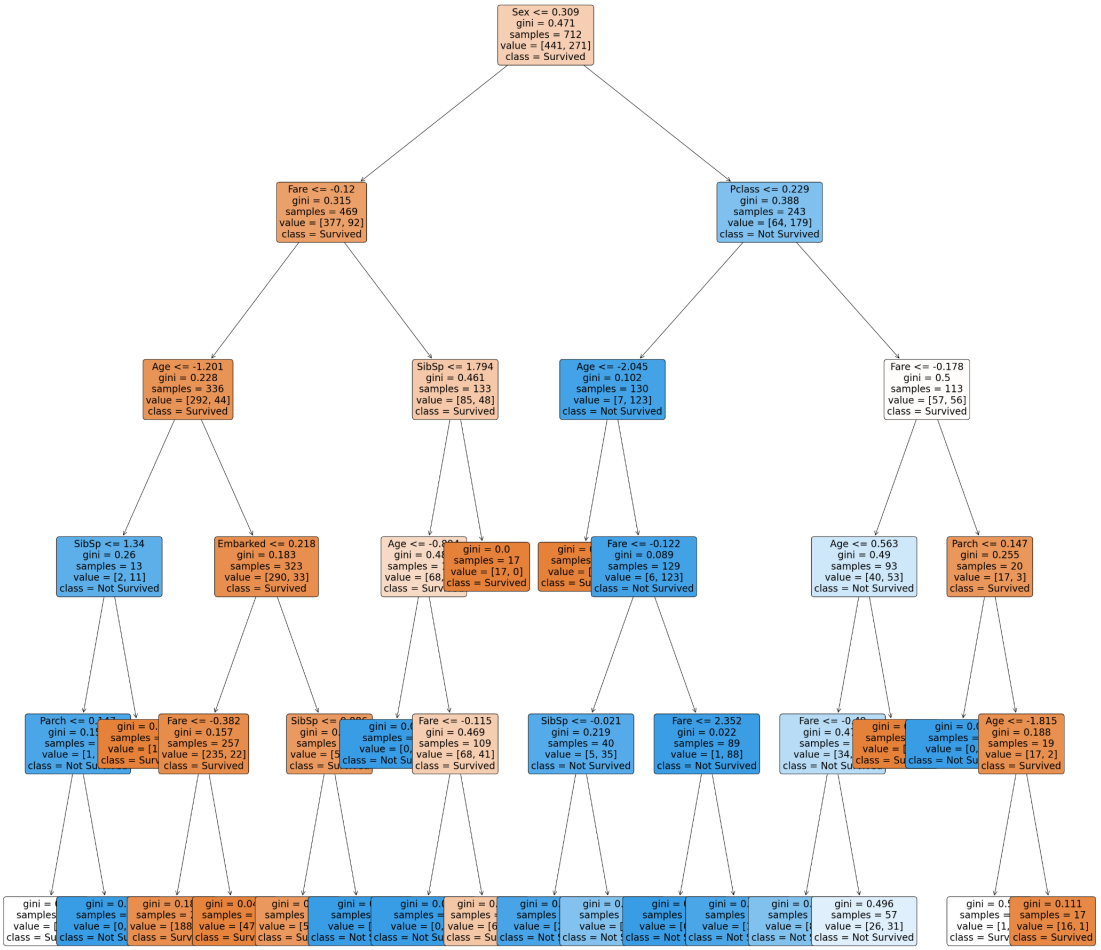
(c

```
{'max_depth': 5,
 'max_features': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2}
```

(d

```
Accuracy: 0.8435754189944135
Precision: 0.8208955223880597
Recall: 0.7746478873239436
F1 score: 0.7971014492753623
```

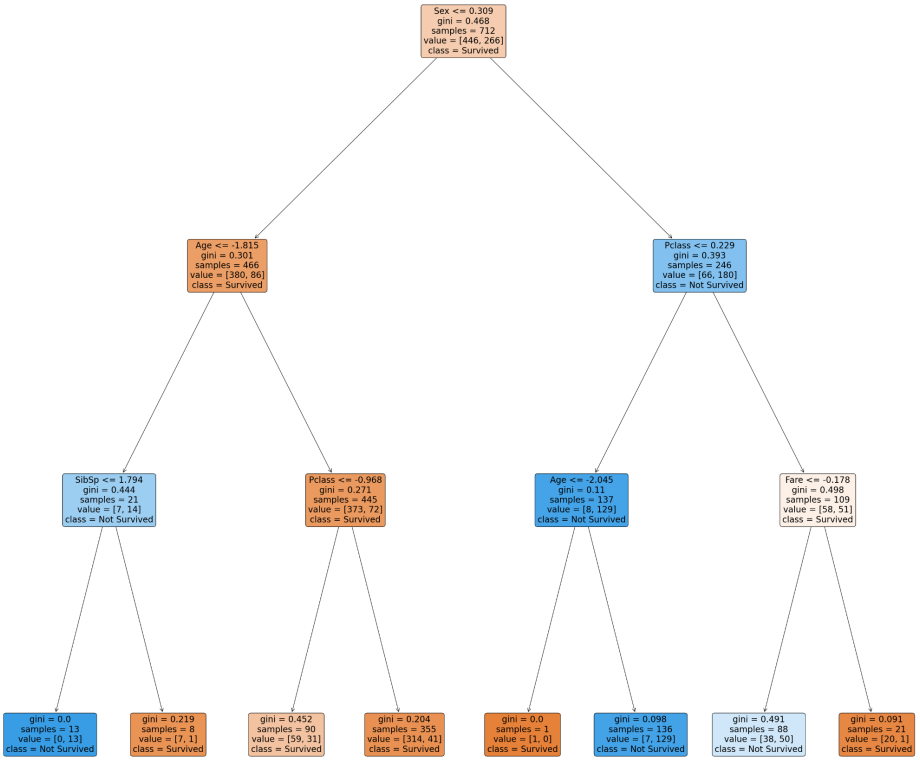
(e



(f

```
DecisionTreeClassifier(ccp_alpha=0.005851278782500746, max_depth=5)
```

(g




```
Accuracy: 0.8603351955307262
Precision: 0.8382352941176471
Recall: 0.8028169014084507
```

مدل نسبت به حالت قبل عملکرد بهتری دارد با مقایسه عملکرد و نتایج نسبت به حالت قبل