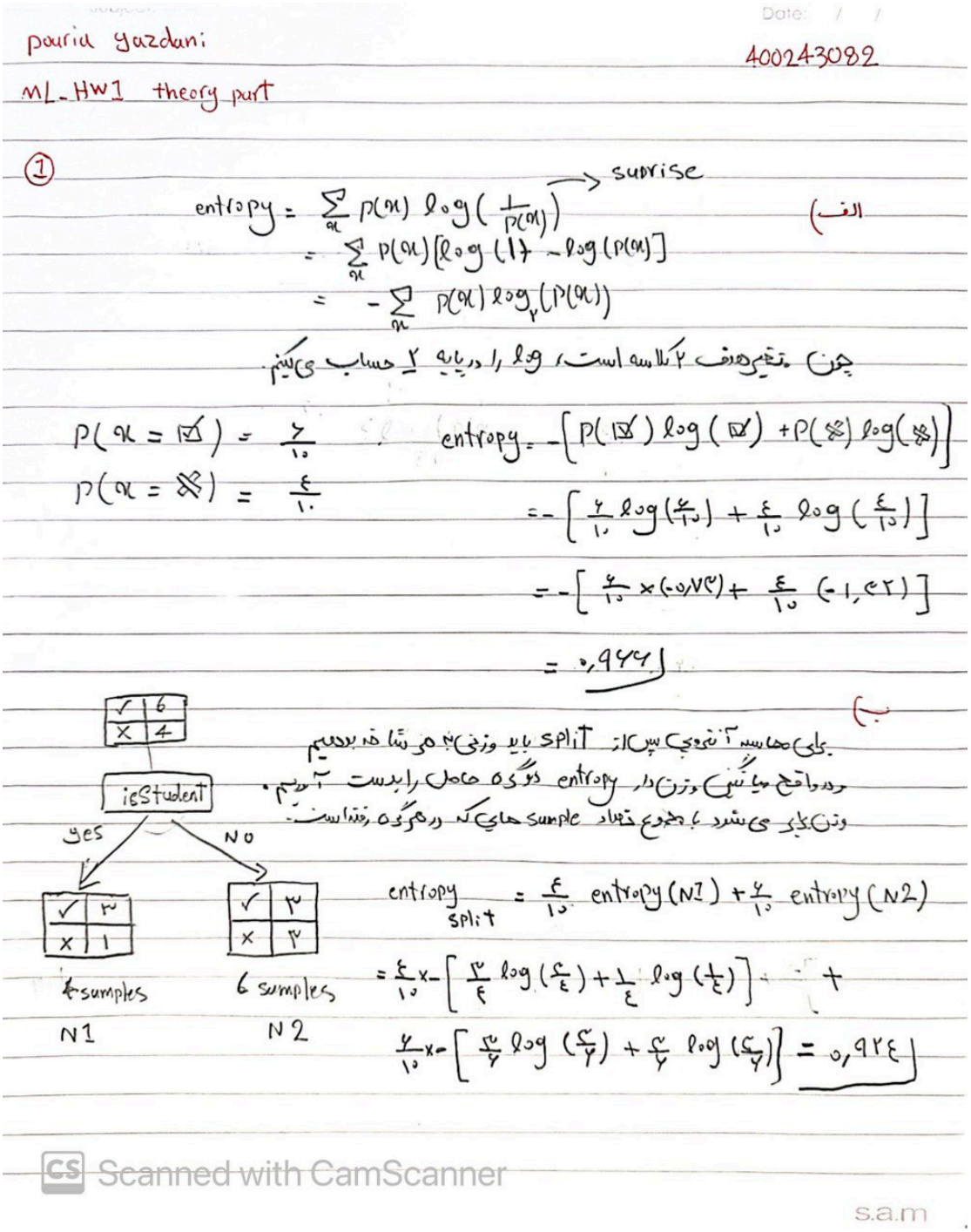
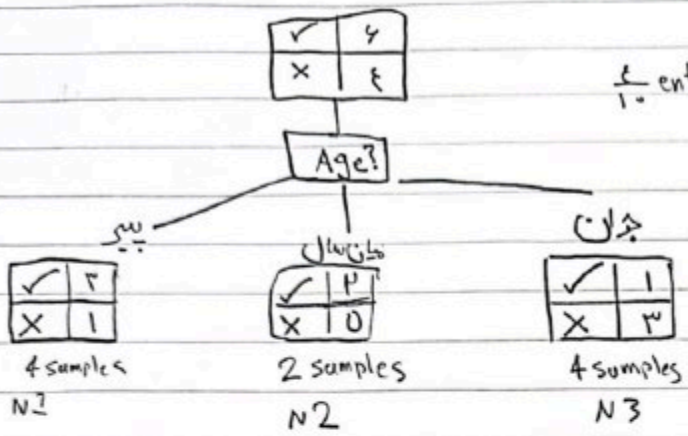
پوریا یزدانی ۴۰۰۲۴۳۰۸۲

تمرین اول - بخش تئوری

سوال اول)



ب) سن = { پیر ، جوان سال ، جوان }



$\frac{4}{10}$ entropy(N1) + $\frac{2}{10}$ entropy(N2) + $\frac{4}{10}$ entropy(N3)

$$\Rightarrow -\frac{4}{10}\left[\frac{2}{4}\log\left(\frac{2}{4}\right)+\frac{2}{4}\log\left(\frac{2}{4}\right)\right]-\frac{2}{10}\left[1\log 1+0\log 0\right]-\frac{4}{10}\left[\frac{3}{4}\log\left(\frac{3}{4}\right)+\frac{1}{4}\log\left(\frac{1}{4}\right)\right]$$

$$= -\frac{8}{10}\left[\frac{2}{4}\log\frac{2}{4}+\frac{1}{4}\log\frac{1}{4}\right]-0$$

$$= 0,469$$

gain حس ← $\text{entropy(root)} - \text{entropy}_{\text{split}} = 0,996 - 0,649 = 0,347$

ت) در حالتی که سن را برای split انتخاب کردیم IG برابر با ۰,۳۴۷ شد

اگر split را براساس دانشجو بودن بگیریم ← IG = ۰,۹۹۷ - ۰,۹۲۴

= ۰,۰۴۲

می شود سن ویژگی تقسیم بندی با سن را انتخاب کنیم چون داده ها را بهتر پخش کرده است و IG بیشتری دارد

سوال دوم)

چالش تعداد زیاد ابعاد: با توجه به اینکه الگوریتم KNN الگوریتمی بر پایه فاصله است و در آن معیار شباهت بین نمونه‌ها بر اساس این فواصل که می‌تواند اقلیدسی یا کسینوسی (با توجه به جنس مسئله مشخص می‌شود) باشند، زمانی که ابعاد و در واقع همان تعداد ویژگی‌های ما بسیار زیاد می‌شود عملا معنای فاصله نیز از بین می‌رود زیرا همه نمونه‌ها در فاصله بسیار زیاد از هم قرار می‌گیرند و در آن صورت نمی‌توان از فاصله به عنوان معیار شباهت استفاده کرد زیرا تعداد عناصری که در محاسبه فاصله به کار می‌روند بسیار زیاد می‌شوند و این موجب زیاد شدن عدد بدست آمده به عنوان فاصله می‌شود.

راه حل: با توجه به جنس مجموعه داده می‌توان از روش‌های کاهش ابعاد مانند PCA برای داده‌های خطی و UMAP برای داده‌های غیرخطی استفاده کرد. در این روش‌ها ویژگی‌ها با نقش بیش‌تر در **تفسیر داده** بیشتر خود را نشان داده و ویژگی‌هایی که اهمیت کمتری از این حیث دارند حذف می‌شوند.(البته در روشی مانند PCA ترکیبی از این ویژگی‌ها مورد نظر است) البته نکته حائز اهمیت این است که انرژی یا واریانس توضیح داده شده توسط این ابعاد کاهش یافته از حد مشخصی کم‌تر نشود.

چالش مقیاس‌بندی ویژگی‌ها: همان‌طور که گفتیم الگوریتم KNN بر اساس فواصل بین نمونه‌ها می‌باشد بنابراین اندازه ویژگی‌ها در این روش خیلی مهم می‌شوند، به عنوان مثال فرض کنید دارای ۲ ویژگی در دیتاست خانه‌ها هستیم و X_1 تعداد اتاق‌ها و X_2 مساحت خانه را به متر مربع نشان می‌دهد. بدیهی است که X_1 نهایتا بتواند تعداد ۱۰ را بپذیرد در صورتی که X_2 در مقیاس دیگری است و به عنوان مثال می‌تواند از ۵۰ تا ۵۰۰ باشد. اتفاقی که در هنگام محاسبه فاصله بین داده‌ها می‌افتد این است که فاصله حساب شده توسط X_2 اصطلاحا dominate می‌شود و تفاوت ویژگی‌های مانند X_1 دیگر به چشم نمی‌آید که این اتفاق خوبی نیست. زیرا توجهی بیش از اندازه به ویژگی‌های نظیر X_2 خواهد شد و از اهمیت آن‌ها صرف نظر می‌کند.

راه حل: برای بی‌اثر کردن این تفاوت مقیاس راه‌های مختلفی وجود دارد که پرکاربردترین آن‌ها **نرمال‌سازی** و **استانداردسازی یا z-score normalization** داده می‌باشد. در روش اول داده به بازه‌ای فیکس (معمولا بین [1,1-]) منتقل می‌شود و در روش دوم داده‌ها حول ۰ و با واریانس ۱

rescale می‌شوند در واقع با اینکار توزیع داده‌ها گوسی شده و با هم قابل مقایسه خواهند شد. با این راهکارها ویژگی‌ها از جنس‌های مختلف با هم قابل مقایسه می‌شوند.

$$X' = \frac{X - min(X)}{max(X) - min(X)}$$

$$X' = \frac{X - \mu}{\sigma}$$

سوال سوم)

تفاوت‌های اصلی جنگل تصادفی با درخت تصمیم در این است که در اولی ما بر خلاف دومی اقدام به آموزش چندین درخت تصمیم می‌کنیم. در این روش ما ابتدا از طریق نمونه‌برداری با جایگزینی از دیتاست موجود B دیتاست با همان اندازه دیتاست اصلی می‌سازیم و توجه داریم که یک نمونه می‌تواند چندین بار در دیتاست‌های مختلف ظاهر شود و دیگر نمونه‌ای می‌تواند در هیچ کدام ظاهر نشود البته با بالا بردن B این احتمال کم می‌شود. دیگر تفاوت این الگوریتم با درخت تصمیم در نحوه انتخاب ویژگی‌ها در هر اسپلیت است به این صورت که برخلاف درخت تصمیم که در هر مرحله می‌توانیم از بین تمام ویژگی‌ها استفاده کنیم، اینجا زیرمجوعه‌ای از این ویژگی‌ها را در هر مرحله در اختیار داریم.(اگر n ویژگی داشته باشیم معمولا $\sqrt{n}$ ویژگی را در نظر می‌گیریم دلیل اصلی این کار کاهش همبستگی بین درخت‌ها است) در آخر نیز وقتی تمام درخت‌های آموزش داده شدند وقتی که یک نمونه جدید آمد بین درخت‌های رای‌گیری میکنیم و کلاسی را که بیشترین درخت‌ها به آن رای داده بودند را اعلام می‌کنیم و اگر مسئله رگرسیون بود باید میانگین بگیریم.

برای تشریح مزایای این الگوریتم ابتدا باید به این اشاره کنیم که یک درخت تصمیم بسیار به دیتاست وابسته است و حساس است و تغییری کوچک در آن می‌تواند ساختار درخت را به کلی عوض کند. این موضوع همچنین باعث می‌شود که یک درخت تصمیم نویزها را یاد بگیرد و به راحتی overfit شود. این شدت حساس بودن الگوریتم باعث پایدار نبودن مدل ما می‌شود اتفاق خوبی نیست. درخت تصادفی با در نظر گرفتن variation های مختلف از دیتاست این حساسیت را کم می‌کند و باعث می‌شود مدلی robust تر داشته باشیم. به طور کلی مزایای آن نسبت به درخت تصمیم شامل دقت بیشتر، واریانس کم‌تر، احتمال overfit شدن کمتر و کمتر

حساس بودن آن به دیتاست می‌شود. همینطور درخت تصمیم می‌تواند به ما کمک کند که بیابیم کدام ویژگی‌ها برای مسئله ما مهم‌تر و کاراتر هستند. این کار می‌تواند با محاسبه مقدار IG ای که توسط هر ویژگی در درخت‌های مختلف بدست آمده انجام شود.

سوال چهارم)

کی‌ان‌ان وزندار: این الگوریتم به طور مستقیم سعی می‌کند یکی از مشکلات و چالش‌هایی که معمولا در knn معمولی مطرح می‌شود را حل کند و آن مسئله تجمیع برچسب‌های نقاط همسایه یک نقطه در فضای ویژگی‌ها می‌باشد. می‌دانیم در knn این مورد به رای‌گیری گذاشته می‌شود که می‌تواند در شرایطی که نقاط همسایه دارای فاصله‌های بسیار متفاوت هستند و نقاط نزدیک‌تر هستند که می‌توانند به درستی کلاس آن نقطه را مشخص کنند. برای حل این موضوع در knn وزن‌دار ابتدا k نقطه نزدیک مانند روش پایه پیدا می‌شود و سپس توسط یک تابع کرنل یا وزن به هر نقطه همسایه به نسبت عکس فاصله‌اش تا نقطه مورد سوال وزن داده می‌شود. به این صورت که هرچه یک نقطه نزدیک‌تر باشد دارای اهمیت و وزن بیشتری خواهد بود. توابعی که بیش‌تر استفاده می‌شوند، عکس فاصله، کرنل گوسی هستند. در این الگوریتم مانند قبل رای‌گیری انجام می‌شود ولی به هر رای وزنی متناسب با فاصله آن نقطه تا نقطه مورد سوال داده می‌شود و سپس کلاس با بیشترین وزن انتخاب می‌شود. در مسئله رگرسیون نیز مقدار پیش‌بینی شده میانگین وزندار k نقطه نزدیک است.

ان‌سی‌ای: یک روش با نظارت است که برای یادگیری متریک فاصله جهت بهبود knn استفاده می‌شود. این الگوریتم یک تبدیل از داده را به طوری یاد می‌گیرد که داده‌های از یک کلاس را به هم نزدیک می‌کند و به طور همزمان نیز سعی می‌کند نقاط از کلاس‌های مختلف را از هم دور کند. به طور دقیق‌تر در این روش ماتریس تبدیلی از داده آموخته می‌شود تا داده را به فضای جدید ببرد که در آن فواصل همانطور که توضیح داده شد، به صورت بهتری و مناسب برای مسئله تعریف شوند. برای استفاده در knn در واقع هر نقطه x به $A.x$ نگاشت می‌شود. این الگوریتم یک تابع احتمالی هدف تعریف می‌کند که و سعی می‌کند p_{ij} را به ازای i و j هایی که در یک کلاس هستند **ماکسیموم** کند.

$$p_{ij} = \frac{\exp(-||Ax_i - Ax_j||^2)}{\sum_{k \neq i} \exp(-||Ax_i - Ax_k||^2)}$$

پس از یادگیری این ماتریس، در knn فاصله‌ها در این فضای جدید محاسبه می‌شوند که دارای معنی بیش‌تری هستند و حالا که فاصله‌ها بهتر و دقیق‌تر تعریف شده‌اند عملکرد knn که به کلی روی این فاصله‌ها بنا شده است نیز بهتر می‌شود.