

گروه پژوهشی ایک



دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکده مهندسی برق - گروه مهندسی کنترل

یادگیری ماشین آزمون میان ترم

نام و نام خانوادگی	علیرضا امیری
شماره دانشجویی	۴۰۲۰۲۴۱۴
تاریخ	خرداد ماه ۱۴۰۴



لینک کولب فایل

فهرست مطالب

۱	سوال ۱، سوالات هماهنگ شده	۱
۱	پاسخ سوال هماهنگ شده اول	۱.۱
۳	پاسخ سوال هماهنگ شده دوم	۲.۱
۳	پاسخ سوال ۲	۲
۳	الف	۱.۲
۳	ب	۲.۲
۳	ج	۳.۲
۴	د	۴.۲
۴	ه	۵.۲
۴	پاسخ سوال ۳	۳
۴	الف	۱.۳
۴	ب	۲.۳
۵	پاسخ سوال ۴	۴
۵	پیش پردازش و مدل لجستیک رگرسیون	۱.۴
۱۰	درخت تصمیم	۲.۴
۱۳	پاسخ سوال ۵	۵

۱ سوال ۱، سوالات هماهنگ شده

۱.۱ پاسخ سوال هماهنگ شده اول

برای کلاس بندی دو دسته داده که هر یک دارای توزیع نرمال گوسی با میانگین μ ، کوواریانس σ و ویژگی های x هستند، از روابط احتمالی این توزیع ها و تعریف توزیع نرمال به شرح زیر استفاده می کنیم.

$$p(\mathbf{x} | C_i) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)\right), \quad i = 1, 2. \quad (1)$$

حال، برای دسته بندی این دو کلاس، طبق رابطه ی زیر با استفاده از لگاریتم و، likelihood با فرض $P(C_1) = P(C_2)$ خواهیم داشت:

$$\delta(\mathbf{x}) = \ln p(\mathbf{x} | C_1) - \ln p(\mathbf{x} | C_2) <>_{C_2}^{C_1} 0. \quad (2)$$

با جایگذاری فرم گوسی داده ها در این رابطه، رابطه ی تصمیم گیری به فرم کوادراتیک به صورت زیر به دست می آید:

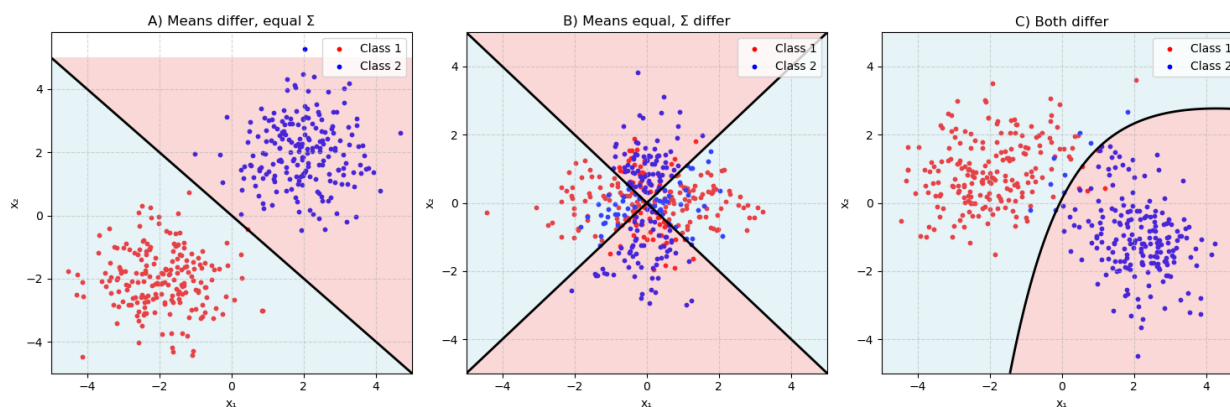
$$\begin{aligned} \delta(\mathbf{x}) &= \left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^\top \Sigma_1^{-1}(\mathbf{x} - \boldsymbol{\mu}_1) - \frac{1}{2} \ln |\Sigma_1| \right] \\ &\quad - \left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_2)^\top \Sigma_2^{-1}(\mathbf{x} - \boldsymbol{\mu}_2) - \frac{1}{2} \ln |\Sigma_2| \right] \\ &= -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^\top \Sigma_1^{-1}(\mathbf{x} - \boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_2)^\top \Sigma_2^{-1}(\mathbf{x} - \boldsymbol{\mu}_2) - \frac{1}{2} \ln \frac{|\Sigma_1|}{|\Sigma_2|}. \end{aligned} \quad (۳)$$

$$\mathbf{x}^\top (\Sigma_2^{-1} - \Sigma_1^{-1}) \mathbf{x} + 2(\boldsymbol{\mu}_1^\top \Sigma_1^{-1} - \boldsymbol{\mu}_2^\top \Sigma_2^{-1}) \mathbf{x} + c, c = \boldsymbol{\mu}_2^\top \Sigma_2^{-1} \boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^\top \Sigma_1^{-1} \boldsymbol{\mu}_1 - \ln \frac{|\Sigma_2|}{|\Sigma_1|}. \quad (۴)$$

که با در نظر داشتن متغیر ویژگی های x و عبارت $\mathbf{x}\mathbf{x}^\top = x^2$ فرمت کوادراتیک این معادله مشخص میشود. حال، با تغییر مقادیر میانگین و کوواریانس، می توانیم سطوح مختلفی را به دست بیاوریم. به عنوان مثال، در صورتی که مقادیر کوواریانس دو کلاس با یکدیگر برابر باشد، آنگاه $\Sigma_1 = \Sigma_2 = \Sigma$ و در نتیجه $\Sigma_2^{-1} - \Sigma_1^{-1} = 0$ و در نهایت با حذف ضریب بخش درجه ۲ این معادله، صفحه ی تصمیم به فرمت خطی به صورت زیر به دست می آید:

$$(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \Sigma^{-1} \mathbf{x} - \frac{1}{2}(\boldsymbol{\mu}_1^\top \Sigma^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^\top \Sigma^{-1} \boldsymbol{\mu}_2) = 0, \quad (۵)$$

به عنوان مثال، حالت های مختلف برای تغییرات میانگین و کوواریانس داده های کلاس ها در شکل ۱ رسم شده است.



شکل ۱: وضعیت های مختلف صفحه تصمیم با تغییر میانگین و کوواریانس

۲.۱ پاسخ سوال هماهنگ شده دوم

در صورتی که تابع توزیع احتمال را در اختیار داشته باشیم، اولین گام برای تولید نمونه هایی که دارای توزیعی مطابق با این تابع باشند، محاسبه ی تابع توزیع احتمال تجمعی یا به اختصار CDF است. بنابراین، ابتدا با محاسبه ی انتگرال، این تابع را محاسبه می کنیم:

$$F(x) = \int_{-\infty}^x f(t)dt \quad (۶)$$

آنگاه با استفاده از وارون تابع CDF و وارد کردن داده های تصادفی به آن، داده هایی با توزیع مورد نظر بسازیم. این فرایند به طریق زیر انجام خواهد شد:

$$x = F^{-1}(u) \Rightarrow x \sim f(x) \quad (۷)$$

برای راحتی در این بخش، می توان داده های تصادفی را با استفاده از دستور $Uniform(0, 1)$ ایجاد کرد. در این حالت، با جایگذاری این داده ها به عنوان ورودی $F^{-1}(u)$ ، می توانیم داده های مورد نظر را به دست بیاوریم. با این حال، این روش تنها در حالتی برقرار است که بتوانیم از CDF به دست آمده وارون بگیریم که این در همه ی شرایط برقرار نیست. علاوه بر این، فرایند تولید اعداد با توزیع یکنواخت نیز در اینجا توضیح داده نشده است که در ادامه به آن خواهیم پرداخت. برای ساخت اعداد تصادفی، الگوریتم های مختلفی مانند $LinearCongruentialGenerator (LCG)$ و $MersenneTwister$ تولید می شوند. در این الگوریتم ها، اعداد رندوم در بازه ی ۰ تا ۱ از تقسیم عددی بزرگ بر F^{32} یا F^{64} به دست می آیند. در اینجا، عددی که بر این مقدار تقسیم می شود با استفاده از پارامترهای تصادفی در سخت افزار کامپیوتر ایجاد می شود.

۲ پاسخ سوال ۲

۱.۲ الف

بله، اما طبق بند بیز در صورتی که تعداد داده ی بی نهایت در اختیار باشد، می تواند طبقه بند بهینه باشد. این در حالی است که شروطی مانند گوسی بودن توزیع داده ها و همچنین متعامد بودن ویژگی ها بر هم برقرار باشد. در این شرایط، به طور ایده آل طبقه بند بیز می تواند در صورت برقراری شرایط بهترین طبقه بند باشد.

۲.۲ ب

بله. به دلیل آنکه این توزیع، احتمالات پیشین برای هر پارامتر را دست می آورد که به نوعی رگولاریزیشن محسوب می شود، می تواند وابستگی به پارامتر ها را کم کرده و در نتیجه از اورفیت شدن جلوگیری کند.

۳.۲ ج

بله، به این دلیل که IG به داده هایی که تعداد بیشتری دارند و راحت تر تفکیک می شوند تمایل بیشتری دارد و به آنها در دسته بندی وزن بیشتری می دهد. در حالی که تنوع بیشتر حالت ها لزوماً به معنای تفکیک پذیری بیشتر نیست.



۴.۲ د

بله. در صورتی که هیچ خاصیت غیر خطی در توابع فعال ساز استفاده نشود، می توانیم با استفاده از رابطه ی زیر اسببات کنیم که اعمال یک تابع خطی بر تابع خطی دیگر، به تولید یک تابع خطی منجر می شود.

$$f(x) = W_2(W_1x + b_1) + b_2 = Wx + b \quad (۸)$$

۵.۲ ه

۳ پاسخ سوال ۳

۱.۳ الف

در این درخت، با فرض ورودی داده شده، ابتدا از شاخه ی ابتدایی شروع می کنیم.

$$A = 1 \Rightarrow DD = 1 \Rightarrow \forall \text{ Leaf} \quad (۹)$$

حال با استفاده از این ویژگی ها و وزن های داده شده و بایاس، مقدار نهایی را با استفاده از تابع sign برای مجموع مقادیر به دست آمده حساب می کنیم.

$$\text{Features} = [B, C, E, F] = [1, 0, 0, 1] \quad (۱۰)$$

$$\text{Weights} = [1, 0, -1, 1] \quad (۱۱)$$

$$\text{Bias} = 1 \quad (۱۲)$$

$$z = 1 \cdot 1 + 0 \cdot 0 + (-1) \cdot 0 + 1 \cdot 1 + 1 = 3 \quad (۱۳)$$

$$\hat{y} = \text{sign}(z) = \text{sign}(3) = +1$$

(۱۴)

که در نتیجه، پاسخ یک برای این مدل به دست می آید.

۲.۳ ب

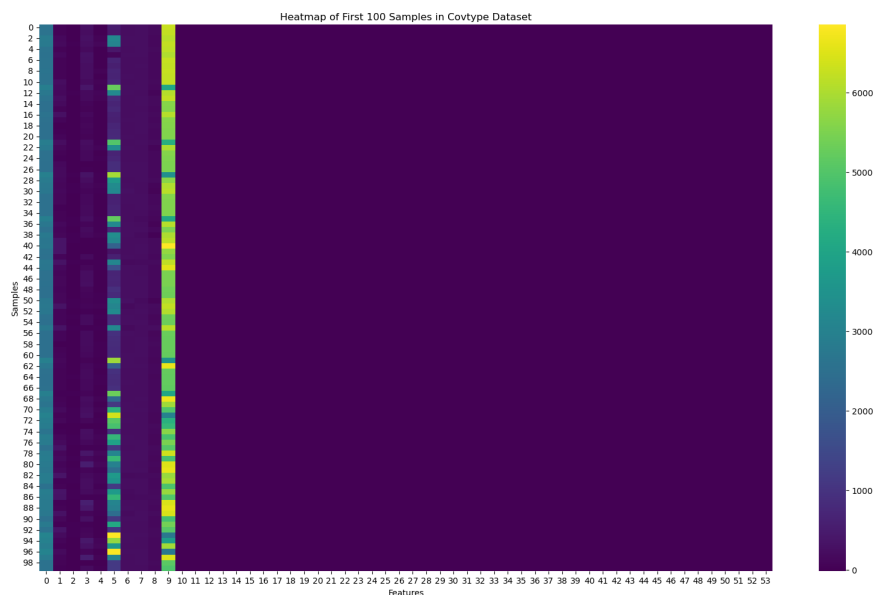
درخت تصمیم علی رغم اینکه از توابع جداساز خطی استفاده می کند، اما در هر گره، یک جداساز جدید تعریف می کند که در نهایت، به تولید یک مدل تکه ای خطی piecewise منجر می شود که یک مدل خطی محسوب نمی شود. با این حال، این مدل در حالت ساده همواره در هر بخش از دسته بند خطی استفاده می کند. در مقایسه ی درخت تصمیم و شبکه عصبی کم عمق، می توان گفت که درخت تصمیم gemneralization بهتری

نسبت به شبکه کم عمق دارد و همچنین سریع تر آموزش می بیند. با این حال، این مدل نسبت به نویز و خطا در داده ها حساسیت بیشتری دارد و در صورتی که داده ی نویزی به سیستم داده شود، دچار خاطی بیشتری می شود. و در صورتی که تفسیرپذیری مدل اهمیت داشته باشد، واضح است که درخت تصمیم مفاهیم فیزیکی و قابل تفسیری دارد که شبکه از آن بی بهره است. علاوه بر این، درخت تصمیم با فرض متعامد بودن ویژگی ها عمل می کند و تعامل آنها با هم دیگر را در نظر نمی گیرد، در حالی که مدل کم عمق همچنان می تواند این کار را انجام دهد.

۴ پاسخ سوال ۴

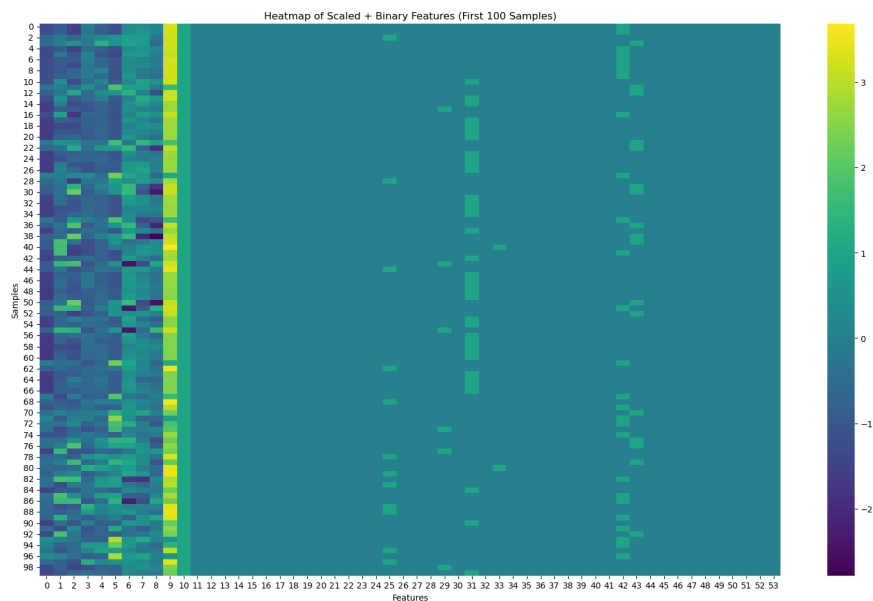
۱.۴ پیش پردازش و مدل لجستیک رگرسیون

در بخش ابتدایی با استفاده از دستور `fetch_covtype` دیتاست را به محیط پایتون لود کرده و آن را در قالب یک دیتافریم ذخیره می کنیم. با بررسی این دیتاست، مشاهده می کنیم که هیچ مقدار NAN ندارد و نکته ی مهم در این باره آن است که از ستون ۱۰ به بعد، مقدار فیچر ها به صورت Categorical هستند. این موارد در شکل ۲ مشخص شده است.



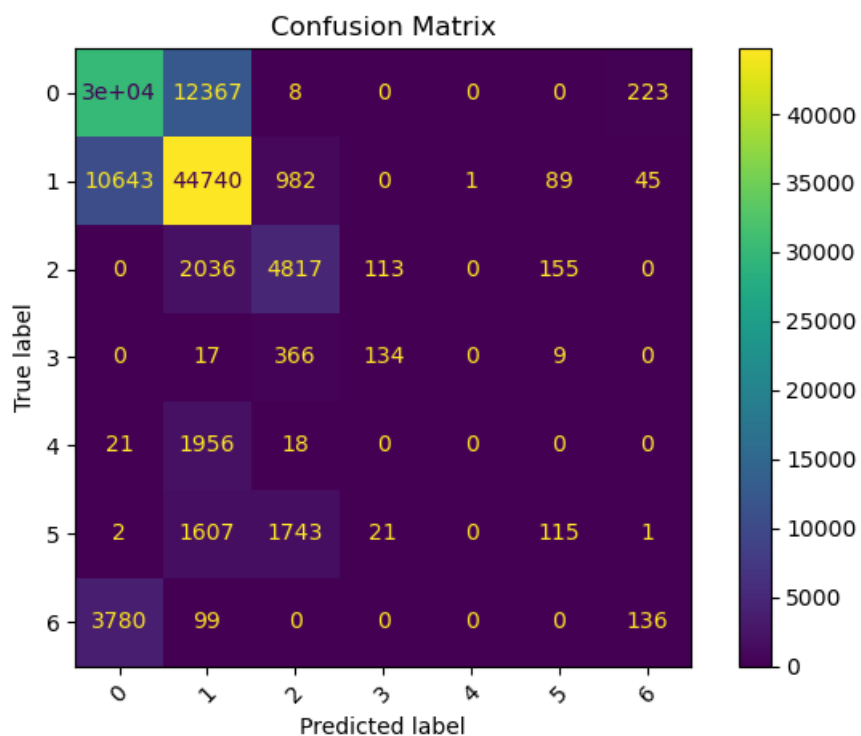
شکل ۲: نمودار حرارتی دیتاست

بر این اساس، داده‌های ستون‌های ۱۰ به بعد را همان‌طور که هستند نگه داشته و داده‌های سایر ستون‌ها را با استفاده از StandardScaler تغییر می‌دهیم. در نهایت دیتاست به صورت شکل ۳ دست خواهد آمد.



شکل ۳: نمودار حرارتی دیتاست پس از Scaling

بدون در نظر داشتن این مورد و با آموزش مدل بر دیتاست خام، پاسخ های به دست آمده به صورت زیر بود.



شکل ۴: نتایج مدل بدون در نظر داشتن ستون های دسته بندی

نتایج به دست آمده از مدل قبل از تغییر ویژگی ها به صورت زیر است:

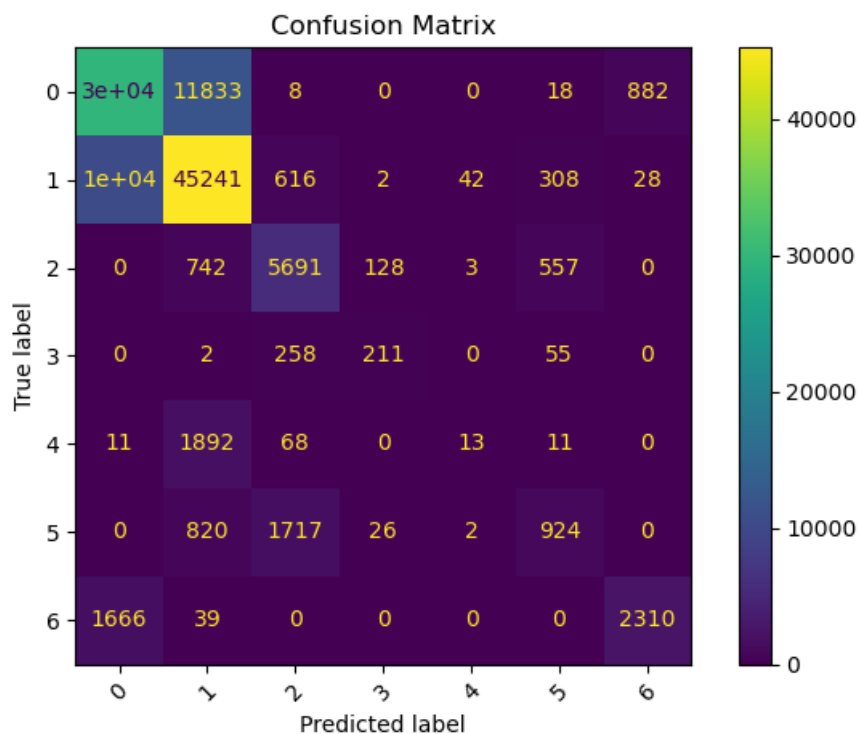
Accuracy: ۰.۸۷۲۰۸۹۰۴۲۶۸۵۹۸۷۵۶

Precision: ۰.۸۷۶۶۸۱۱۱۱۸۷۴۵۳۸

Recall: ۰.۸۷۲۰۸۹۰۴۲۶۸۵۹۸۷۵۶

F1 Score: ۰.۸۰۱۰۹۶۱۳۳۱۶۶۱

اما پس از اعمال این تغییرات، نتایج به دست آمده به صورت زیر خواهد بود که می بینیم عملکرد آن بهبود پیدا کرده است. **شکل ۵**



شکل ۵: [عملکرد مدل بعد از پردازش ها]



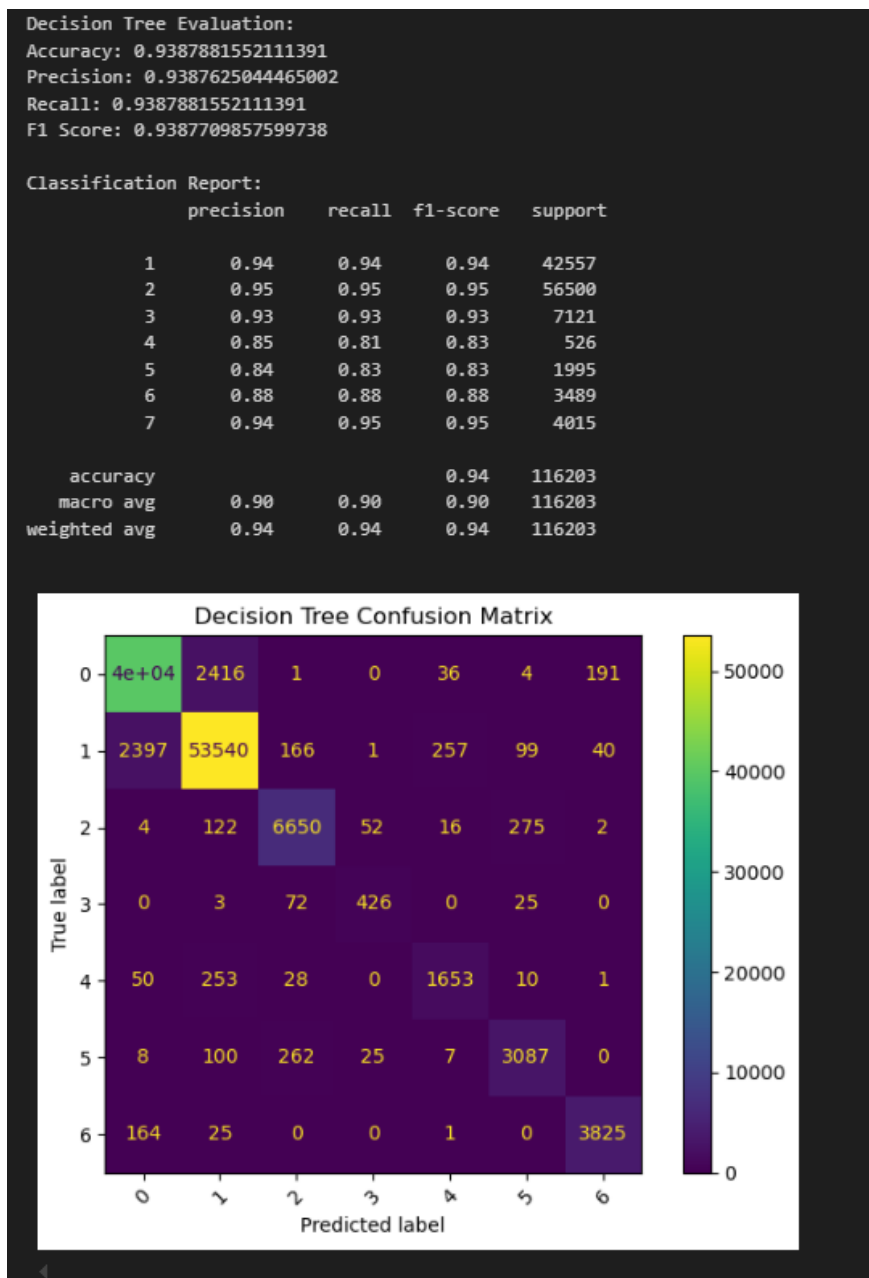
Classification Report:				
	precision	recall	f1-score	support
1	0.71	0.70	0.71	42557
2	0.75	0.80	0.77	56500
3	0.68	0.80	0.74	7121
4	0.57	0.40	0.47	526
5	0.22	0.01	0.01	1995
6	0.49	0.26	0.34	3489
7	0.72	0.58	0.64	4015
accuracy			0.72	116203
macro avg	0.59	0.51	0.53	116203
weighted avg	0.71	0.72	0.71	116203

شکل ۶: report classification

در اینجا مشاهده می شود که کلاس های ۴ به بعد به دلیل تعداد داده های بسیار کمتر و نامتوازن بودن، عملکرد بدتری دارند نسبت به سایر داده ها.

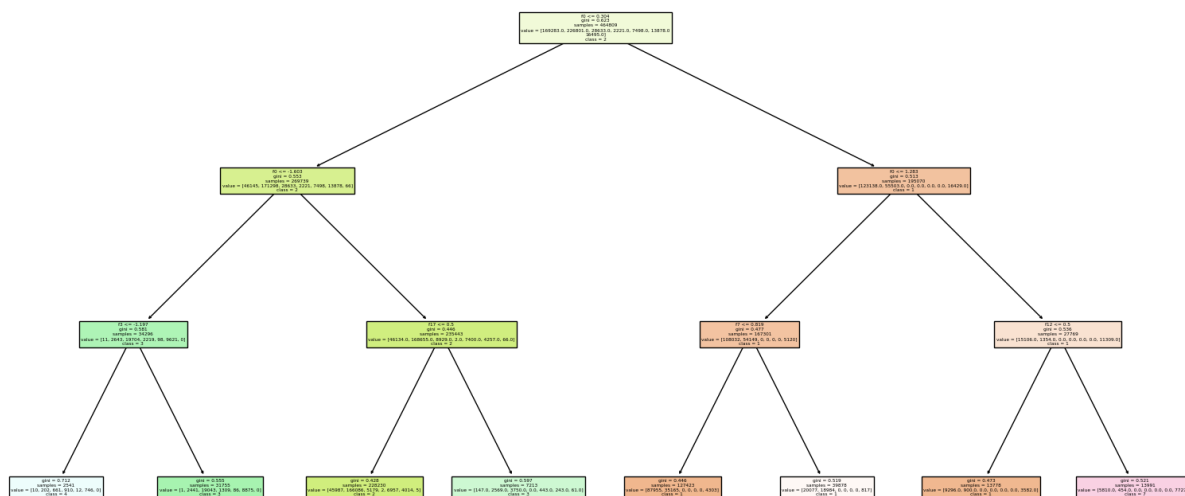
۲.۴ درخت تصمیم

در این بخش با آموزش مدل درخت تصمیم بر روی داده های پردازش شده در بخش قبل، نتایج بهتری به دست می آوریم. شکل ۷

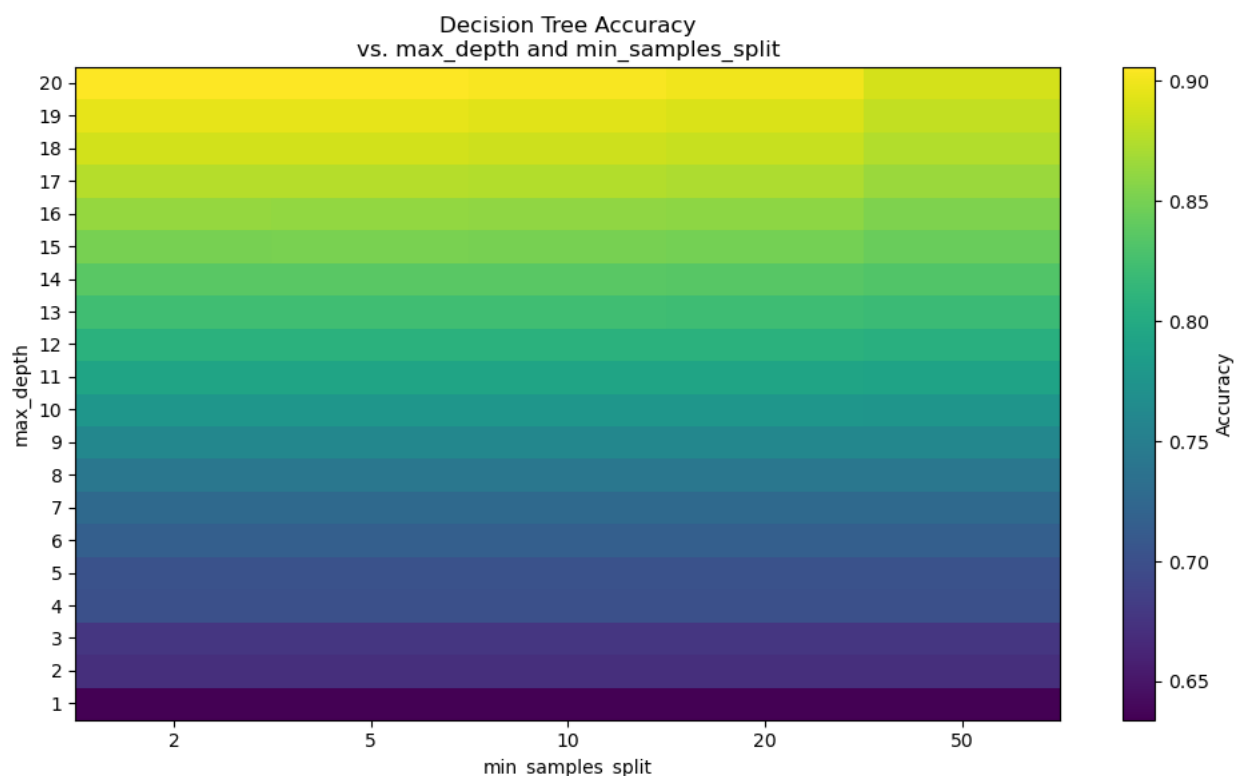


شکل ۷: نتایج درخت تصمیم

شکل ۸: نمودار درخت تصمیم



همچنان مشاهده می کنیم که برای داده هایی که تعداد کمتری داشتند، نتایج با مقدار خطای بیشتری همراه است مانند کلاس های ۴ و ۵ و ۶. اما این عملکرد در مقایسه با لجستیک رگرشن پیشرفت بهتری داشته است. در ادامه ی این بهش، به تنظیم هایپر پارامتر های max_depth و $min_samples_split$ می پردازیم و برای این مقادیر، مدل هایی را آموزش داده و نتایج هر یک را بررسی می کنیم. چنان که مشاهده می شود، مطابق



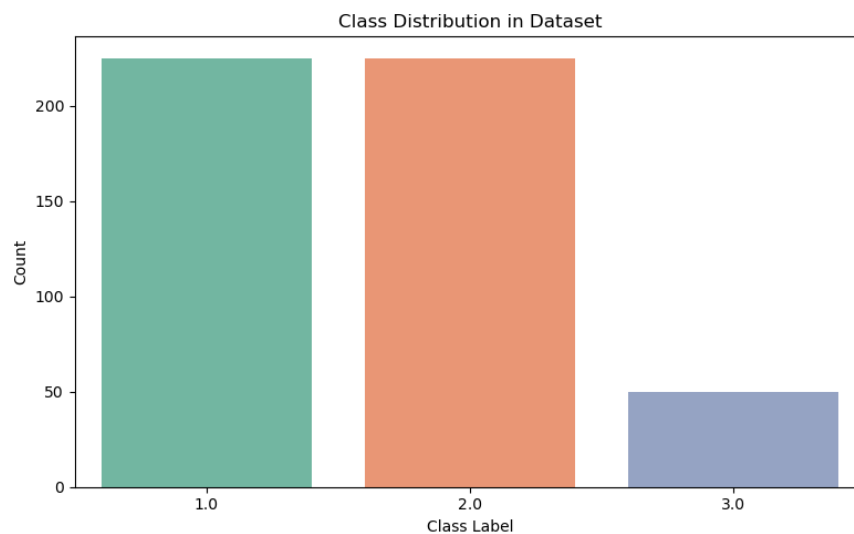
شکل ۹: نتایج تغییر پارامتر ها بر عملکرد مدل

انتظار با افزایش عمق درخت و کاهش کمتری تعداد نمونه در هر شاخه، دقت سیستم افزایش یافته است. اگرچه در حالت بیشینه، این امر به اورفیت شدن منجر می شود. با تغییر این پارامتر ها، می توان به ترکیبی دست یافت که هم به داده ها اورفیت نشده باشد و $general$ باشد، و هم دقت کافی را در دسته بتندی داشته باشد.



۵ پاسخ سوال ۵

در این بخش، ابتدا داده ها را دانلود کرده و دیتای آن را بررسی می کنیم. مشاهده می شود که داده ها در سه کلاس دسته بندی شده اند و در مجموع ۵۰۰ داده قرار دارد که ستون چهارم، لیبل ها هستند. مشاهده می شود که توزیع داده ها در دو کلاس اول برابر و در کلاس سوم کمتر است.



شکل ۱۰: توزیع داده ها

بنابراین می توان انتظار داشت که نتایج به دست آمده برای این داده ها در پایان دقت کمتری نیز داشته باشند. در این بخش، ابتدا یک تابع فاصله ی اقلیدوسی به صورت زیر برای سیستم تعریف شد:

$$d(x_1, x_2) = \sqrt{\sum (x_{1i} - x_{2i})^2} \quad (۱۵)$$

با استفاده از این تابع فاصله، الگوریتم KNN مطابق فرایند زیر تعریف شد.

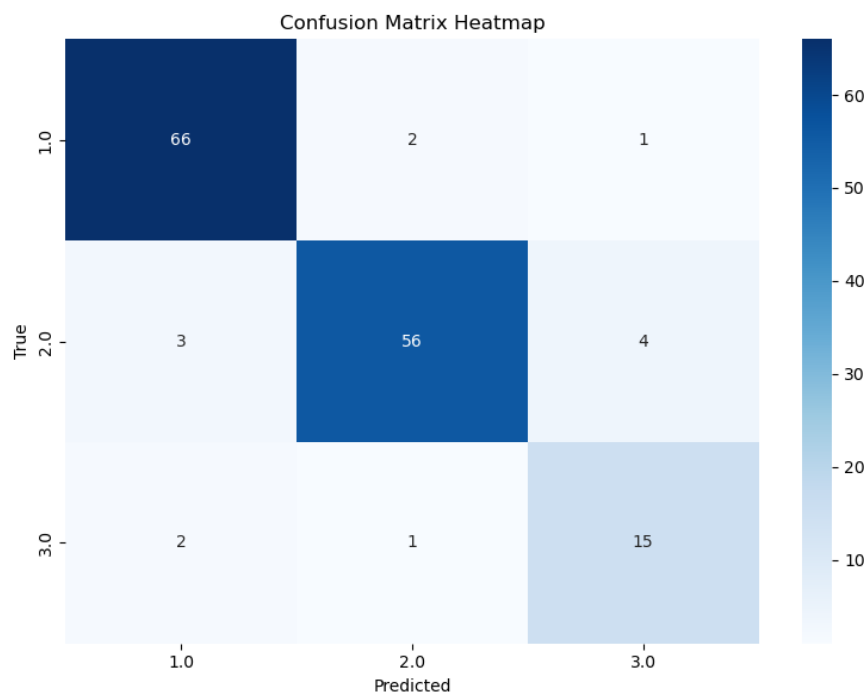
$$\mathcal{D}_{x_t} = \{(d(x_t, x_i), y_i) \mid x_i \in \text{Set Train}\} \quad (۱۶)$$

$$\mathcal{N}_k(x_t) = \text{Top-}k \text{ in elements smallest } \mathcal{D}_{x_t} \quad (۱۷)$$

$$\hat{y}_t = \arg \max_y \sum_{(d, y_i) \in \mathcal{N}_k(x_t)} 1_{[y_i=y]} \quad (۱۸)$$

$$\text{Accuracy} = \frac{1}{m} \sum_{i=1}^m 1_{[\hat{y}_i=y_i]} \quad (۱۹)$$

با آموزش این مدل، نتایج دسته بندی به شرح زیر به دست می آید. شکل ۱۱



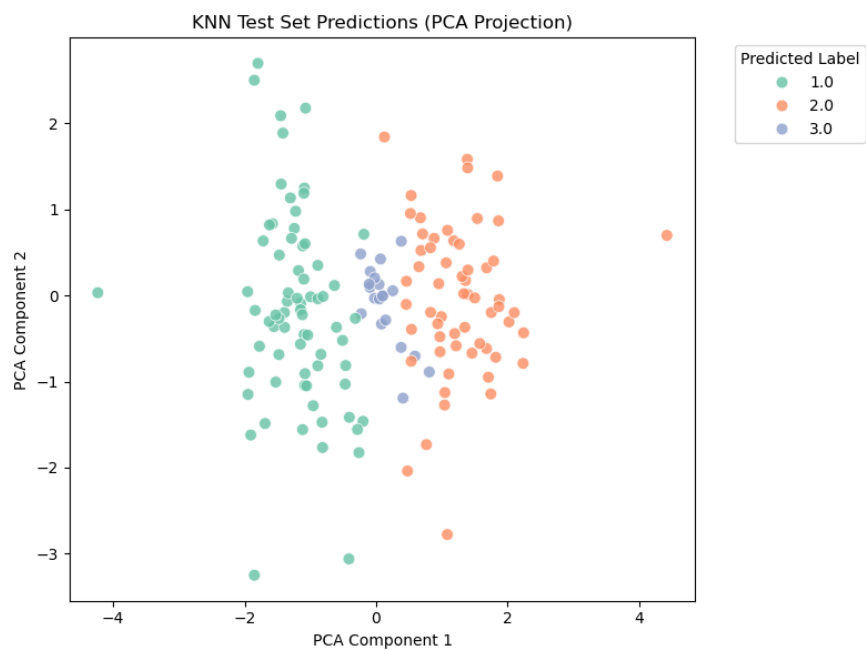
شکل ۱۱: نتایج KNN



Classification Report:				
	precision	recall	f1-score	support
1.0	0.93	0.96	0.94	69
2.0	0.95	0.89	0.92	63
3.0	0.75	0.83	0.79	18
accuracy			0.91	150
macro avg	0.88	0.89	0.88	150
weighted avg	0.92	0.91	0.91	150
Accuracy: 0.91				

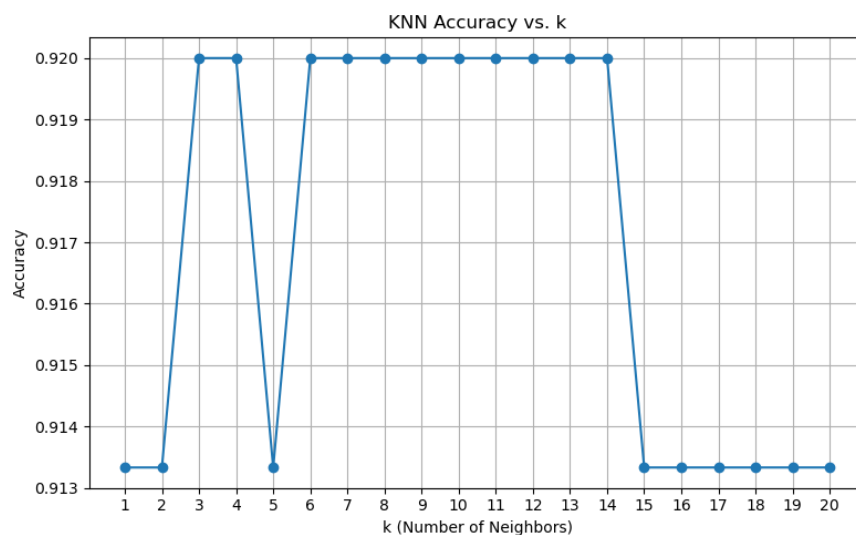
شکل ۱۲: گزارش دسته بندی [KNN]

مشاهده می شود که این مدل توانسته داده ها را به خوبی با دقت ۹۱ درصد دسته بندی کند. برای نمایش توزیع داده های نمونه با استفاده از `pca` در فضایی کاهش بعد داده شده خواهیم داشت:



شکل ۱۳: [توزیع با استفاده از `pca`]

در ادامه، به بررسی تاثیر تغییرات مقدار k در این الگوریتم می پردازیم. این مقادیر از ۱ تا ۲۱ تغییر داده شده و دقت مدل برای هر یک در رسم شده است.



شکل ۱۴: نمودار تغییرات دقت بر حسب k

مشاهده می کنیم که تغییرات k تاثیر به سزایی در دقت مدل ندارد در نتیجه استفاده از مقادیر کمتر، برای سادگی الگوریتم توصیه می شود. در بخش بعد الگوریتم KNN وزن دار تعریف و استفاده شده است.

🔍 Classification Report (Weighted KNN - Model 2):				
	precision	recall	f1-score	support
1.0	0.93	0.96	0.94	69
2.0	0.95	0.89	0.92	63
3.0	0.75	0.83	0.79	18
accuracy			0.91	150
macro avg	0.88	0.89	0.88	150
weighted avg	0.92	0.91	0.91	150

شکل ۱۵: [نتیجه KNN وزن دار]

و مشاهده می‌شود که تعداد پ‌یش‌بینی شده دقیقاً برابر با الگوریتم KNN ساده است و اعمال وزن تأثیری در آن نداشته. در بخش انتهایی با استفاده از روش validation cross به شرح زیر، عملکرد مدل‌ه را بررسی می‌کنیم. برای هر تایش (fold):

- مدل روی $K - 1$ بخش آموزش می‌بیند.
- روی بخش باقی‌مانده ارزیابی می‌شود.
- دقت برای هر بار تایش به صورت زیر محاسبه می‌شود:

$$\text{Accuracy}_i = \frac{1}{m} \sum_{j=1}^m 1[\hat{y}_j = y_j]$$

این فرآیند برای دو مدل انجام می‌شود:

- مدل ۱: KNN معمولی (با رأی‌گیری اکثریت)
- مدل ۲: KNN وزنی (با وزن‌دهی معکوس فاصله):


$$w_i = \frac{1}{d(x, x_i) + \epsilon}$$

در نهایت، دقت میانگین محاسبه می‌شود:

$$\text{Accuracy Mean} = \frac{1}{K} \sum_{i=1}^K \text{Accuracy}_i$$

آنگاه نتایج این validation cross به صورت زیر به دست می‌آید که می‌بینیم مدل دوم اندکی بهتر از مدل اول عمل کرده است.

```

 Cross-Validation Results:
Regular KNN (Model 1) - Mean Accuracy: 0.9540
Weighted KNN (Model 2) - Mean Accuracy: 0.9560
    
```

شکل ۱۶: validation cross