

1.

۱. حجم مجموعه داده:

اگر مجموعه داده بسیار کوچک باشد مدل ممکن است نتواند الگوهای کلی داده ها را درک کند و به جزئیات داده های آموزشی پاسخ دهد یعنی مدل به جای یادگیری اطلاعات عمومی، به طور مکرر الگوهای خاص داده های آموزشی را حفظ میکند و این امر باعث میشود که عملکرد مدل بر روی داده های جدید (که ممکن است الگو های متفاوتی داشته باشند) ناکارآمد باشد.

۲. تعادل داده ها (داده های بالانس و غیر بالانس):

وجود داده های نامتوازن یعنی تعداد بیشتری از یک دسته/دسته ها نسبت به دسته های دیگر وجود داشته باشد در این حالت مدل ممکن است به سرعت الگوهای داده هایی که تعدادشان بیشتر است را یاد بگیرد و اطلاعات داده های کمتر را نادیده بگیرد.

۳. ویژگی های نامربوط:

اگر ویژگی های انتخاب شده، بیش از حد جزئیات بی ارتباط را به مدل منتقل کنند (یعنی استفاده از ویژگی هایی که ارتباط مستقیم با هدف پیش بینی ندارند) ممکن است مدل الگوهای نویزی یا غیرضروری را یاد بگیرد و از قابلیت عمومی سازی برای داده های جدید دور شود.

۴. تعداد ایپاک های آموزش:

تعداد زیادی ایپاک میتواند باعث بیش برآزش شود زیرا مدل ممکن است به طور غیر منطقی به داده های آموزشی سازگار شود و به الگوهای نویزی در داده های آموزشی واکنش نشان دهد یعنی مدل بیش از حد به داده های آموزشی بپیوندد و الگوهای موجود در داده های نویزی را هم یاد بگیرد.

۵. پیچیدگی مدل:

استفاده از مدل های بسیار پیچیده و با تعداد پارامترهای زیاد میتواند باعث بیش برآزش شود به علت اینکه مدل ها ممکن است سعی کنند الگوهای پیچیده و نویزی را در داده های آموزشی یاد بگیرند که در این حالت ممکن است قابلیت عمومی سازی مدل را به شدت کاهش دهد.

۶. نشت داده ها:

وقتی که اطلاعات از داده های آزمون به داده های آموزشی به طور غیرمستقیم نفوذ کند نشت داده رخ می دهد پس در این حالت اگر مدل از اطلاعاتی که در داده های آزمون هست در فرایند آموزش استفاده کند ممکن است به نظر برسد که عملکرد بسیار خوبی دارد زیرا از داده هایی که نباید در دسترس باشند استفاده کرده است اما وقتی این مدل روی داده های واقعی جدید استفاده می شود عملکرد آن ممکن است به شدت کاهش یابد. بنابراین نشت داده می تواند منجر به بیش برآزش شود زیرا مدل از اطلاعاتی استفاده می کند که نباید در دسترس باشد و این باعث می شود که مدل به نظر بیشتر از آنچه واقعا است دقیق باشد. این موضوع میتواند منجر به اعتماد بیش از حد به مدل و در نهایت تصمیمات نادرست بر اساس پیش بینی های آن شود.

Propensity of 1	Actual
0.03	0
0.52	0
0.38	0
0.82	1
0.33	0
0.42	0
0.55	1
0.59	0
0.09	0
0.21	0
0.43	0
0.04	0
0.08	0
0.13	0
0.01	0
0.79	1
0.42	0
0.29	0
0.08	0
0.02	0

0.25		0.5		0.75	
Propensity of 1	Actual	Propensity of 1	Actual	Propensity of 1	Actual
0	0	0	0	0	0
1	0	1	0	0	0
1	0	0	0	0	0
1	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	0	0
1	1	1	1	0	1
1	0	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
1	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

For 0.25:

$$\text{error rate} = \frac{\text{Number of Incorrect Predictions}}{\text{Total Number of Predictions}} = \frac{8}{20} = 0.4$$

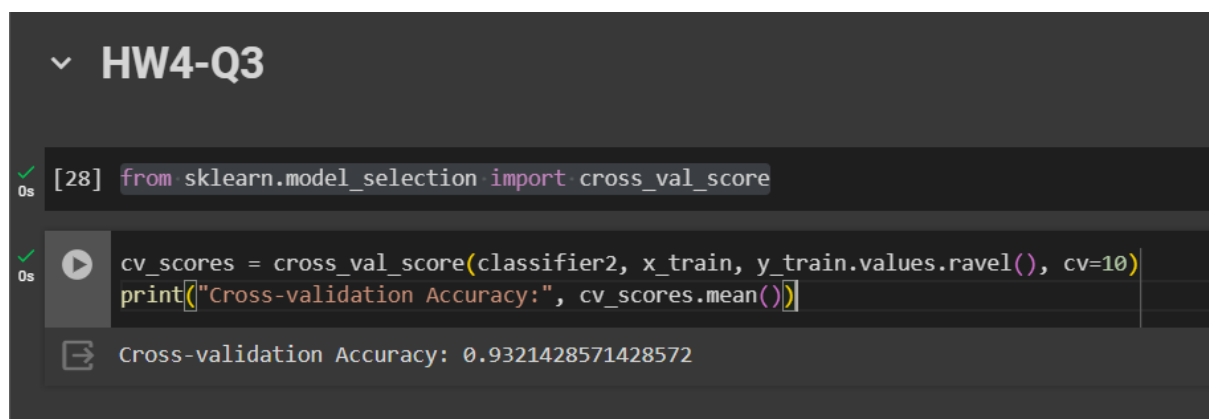
For 0.5:

$$\text{error rate} = \frac{\text{Number of Incorrect Predictions}}{\text{Total Number of Predictions}} = \frac{2}{20} = 0.1$$

For 0.75:

$$\text{error rate} = \frac{\text{Number of Incorrect Predictions}}{\text{Total Number of Predictions}} = \frac{1}{20} = 0.05$$

3.

A screenshot of a Jupyter Notebook cell titled 'HW4-Q3'. The cell contains two lines of Python code: `[28] from sklearn.model_selection import cross_val_score` and `cv_scores = cross_val_score(classifier2, x_train, y_train.values.ravel(), cv=10)` followed by `print("Cross-validation Accuracy:", cv_scores.mean())`. The output of the cell is `Cross-validation Accuracy: 0.9321428571428572`.

```
HW4-Q3

[28] from sklearn.model_selection import cross_val_score

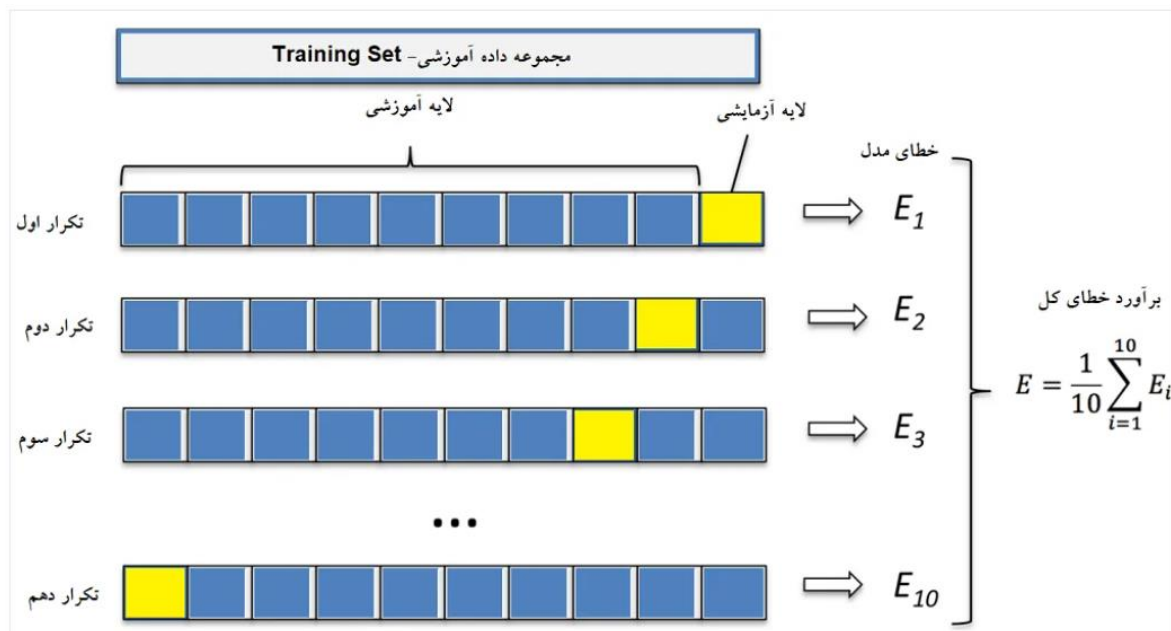
cv_scores = cross_val_score(classifier2, x_train, y_train.values.ravel(), cv=10)
print("Cross-validation Accuracy:", cv_scores.mean())

Cross-validation Accuracy: 0.9321428571428572
```

روشی که استفاده کردم روش k-Fold است به این صورت که اگر مجموعه داده های آموزشی را به طور تصادفی به k زیرنمونه یا لایه (Fold) با حجم یکسان تفکیک کنیم، می توانیم در هر مرحله از فرایند CV، تعداد k-1 از این لایه ها را به عنوان مجموعه داده آموزشی و یکی را به عنوان مجموعه داده اعتبارسنجی در نظر بگیریم.

در این کد `cross_val_score` متدی است که از `classifier2` مدل که به عنوان آرگومان اول به آن پاس داده شده استفاده می کند و عملکرد آن را با استفاده از اعتبارسنجی متقاطع ارزیابی می کند. در اینجا، داده ها به ۱۰ بخش تقسیم می شوند (به عنوان CV) و هر بار ۹ بخش برای آموزش و یک بخش برای اعتبارسنجی استفاده می شود.

شکل زیر برای درک بهتر از k-Fold است:



4.

تعداد نمونه هایی که به اشتباه به عنوان کلاهبردار تشخیص داده شده اند یا false positive :

$$88 - 30 = 58$$

تعداد نمونه هایی که به اشتباه به عنوان غیر کلاهبردار تشخیص داده شده اند یا false negative :

$$952 - 920 = 32$$

تعداد کل نمونه ها:

$$88 + 952 = 1040$$

$$\text{misclassification rate} = \frac{\text{false positive} + \text{false negative}}{\text{total predictions}} = \frac{58 + 32}{1040} \approx 0.0865 \rightarrow 8.65\%$$

5.

1.

درست است چون برای داده های با نویز بالا پس هرس یا post-pruning معمولاً بهتر از پیش هرس یا pre-pruning عمل می کند. به علت اینکه پیش هرس ممکن است درخت تصمیم را زودتر از حد لازم هرس کند و این کار باعث می شود که مدل نتواند با نویز موجود در داده ها کنار بیاید. در مقابل پس هرس اجازه می دهد که درخت تصمیم به صورت کامل رشد کند و سپس بر اساس ارزیابی داده های معتبر، برخی از شاخه ها را حذف می کند.

2.

نادرست است برای داده های غیر بالانس، هر دو روش پیش هرس و پس هرس می توانند مشکلاتی ایجاد کنند اما به طور کلی، پس هرس معمولاً بهتر از پیش هرس عمل می کند. بخاطر اینکه پیش هرس ممکن است درخت تصمیم را زودتر از حد لازم هرس کند و این کار باعث می شود که مدل نتواند با غیر بالانس بودن داده ها کنار بیاید. در مقابل پس هرس اجازه می دهد که درخت تصمیم به صورت کامل رشد کند و سپس بر اساس ارزیابی داده های معتبر، برخی از شاخه ها را حذف کند. این روش می تواند بهتر با غیر بالانس بودن داده ها مقابله کند.

همچنین در داده های غیر بالانس، احتمال بیش برآزش به داده های کلاس اقلیتی وجود دارد پس پیش هرس باعث میشود که برخی از شاخه ها قبل از رسیدن به حالت کامل حذف شوند که این ممکن است باعث از دست رفتن اطلاعات مفید شود از طرف دیگر، پس هرس میتواند به مدل اجازه دهد تا به داده ها بیشتر بپردازد و از بیش برآزش جلوگیری کند.

(نکته مهم: برای داده های غیر بالانس، هیچ یک از روش های پیش هرس یا پس هرس به طور خاصی بهتر نیست. انتخاب بین پیش هرس و پس هرس بیشتر به نوع و میزان نویز در داده ها، تعداد داده ها و پیچیدگی مدل بستگی دارد ولی ممکن است در اکثریت مواقع حرف بالا جوابگو باشد.)

3.

درست است چون وقتی مجموعه داده کوچک است احتمال بیش برآزش بالا می رود زیرا مدل ممکن است به جزئیات نویزی یادگیری کند که در داده های آموزشی وجود دارد اما در داده های تست واقعیت ندارد. وقتی مجموعه داده بزرگ است، احتمال کم برآزش بیشتر می شود زیرا داده های بیشتر به مدل اجازه می دهد تا الگوهای عمومی را یاد بگیرد.

4.

درست است چون درخت های تصمیم به خوبی می توانند با روابط غیرخطی و پیچیده در داده ها سازگاری پیدا کنند. این به دلیل این است که آنها قادرند الگوهای پیچیده تر و روابط غیر خطی را به خوبی مدل کنند، به خصوص اگر معیارهای تقسیم مناسبی انتخاب شود که بتوانند مرزهای غیرخطی را در داده ها تعیین کنند.

6.

برای ارزیابی مدل شناسایی کلاهبرداری کارت اعتباری بهتر است از F1-Score استفاده کنیم چون داده های کلاهبرداری کارت اعتباری معمولاً نامتعادل هستند یعنی تعداد موارد کلاهبرداری نسبت به تعداد معاملات معتبر بسیار کم است و استفاده از معیارهایی مانند دقت ممکن است در چنین مواردی گمراه کننده باشند زیرا حتی یک مدل ساده که همیشه پیش بینی می کند که کلاهبرداری رخ نداده است، میتواند دقت بالایی داشته باشد اما چنین مدلی برای کاربرد واقعی بی فایده است زیرا هیچگاه کلاهبرداری را شناسایی نمی کند.

همچنین معیار F1-Score یک معیار ترکیبی است که هر دو عامل Precision و Recall را در نظر می گیرد و Precision به تعداد پیش بینی های صحیح مثبت نسبت به کل پیش بینی های مثبت اشاره دارد، در حالی که Recall به تعداد پیش بینی های صحیح مثبت نسبت به کل موارد مثبت واقعی اشاره دارد. بنابراین F1-Score

می تواند یک تعادل خوب بین Precision و Recall ایجاد کند و این امکان را بدهد تا عملکرد مدل را در شناسایی کلاهبرداری های واقعی ارزیابی کند.

$$F1 - score = \frac{2 * precision * recall}{precision + recall}$$

$$precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$