آبتین ماه یار – ۹۸۲۲۲۰۸۷

در ادامه به سوالاتی که داخل داک اصلی بود ولی داخل ریپورت ۱ توضیح داده نشدند، پاسخ میدهیم.

۱. همانطور که در ریپورت اصلی توضیح داده شد، مدل انتخاب شده Logistic Regreesion CV میباشد که یک هایپر پارامتر multi_class دارد که میتوان در آن استراتژی مورد استفاده در هنگام مواجه با تسک مولتی کلاس کلاسیفیکشن را تعیین کرد که به صورت پیش فرض بر auto می باشد که از آنجایی که solver ما در این تسک lbfgs میباشد استراتژی multinomial را انتخاب میکند که در این استراتژی عملا دیگر بحث ovr و ovo را نخواهیم داشت چون با این استراتژی مدل سعی دارد تا مقدار loss را کاهش دهد که به شکل زیر است و تمام کلاس های مختلف دیتاست را در بر میگیرد. به این صورت که به ازای هر رکورد احتمال قرار گیری در هرکدام از آن کلاس هارا بدست می آورد و بزرگترین آن را انتخاب میکند.

$$L = -\frac{1}{n}\sum_{i=0}^{i=n} y_i \log(s_i)$$

Here,

L = Average cross-entropy loss for the model

y = Ground truth (1-hot encoded target variable; 1 for target outcome, 0 for all other possible outcomes)

s = probability vectors obtained from softmax function

n = Total number of feature sets

$$\Pr(Y_i = 1) = \frac{e^{\beta_1' \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\beta_k' \cdot \mathbf{X}_i}}$$

$$\cdots\cdots$$

$$\Pr(Y_i = K-1) = \frac{e^{\beta_{K-1}' \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\beta_k' \cdot \mathbf{X}_i}}$$

$$\Pr(Y_i = K) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\beta_k' \cdot \mathbf{X}_i}}$$

Ova: strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. This strategy requires the base classifiers to produce a real-valued confidence

score for its decision, rather than just a class label; discrete class labels alone can lead to ambiguities, where multiple classes are predicted for a single sample.

Ovr: one trains $K\,(K-۱)\,/\,۲$ binary classifiers for a K-way multiclass problem; each receives the samples of a pair of classes from the original training set, and must learn to distinguish these two classes. At prediction time, a voting scheme is applied: all $K\,(K-۱)\,/\,۲$ classifiers are applied to an unseen sample and the class that got the highest number of "+۱" predictions gets predicted by the combined classifier.

۲. confusion matrix در آخرین مدل بر روی دیتا ست تست به شکل زیر می باشد

$$\begin{pmatrix} ۹۹ & ۱ & ۰ & ۰ \\ ۱ & ۹۵ & ۴ & ۰ \\ ۰ & ۱ & ۹۷ & ۲ \\ ۰ & ۰ & ۴ & ۹۶ \end{pmatrix}$$

از روی این ماتریس میتوان نتیجه گرفت با توجه به سایز دیتاست تست، مدل بر روی تمامی کلاس ها پرفورمنس تقریبا یکسانی داشته است.

۳. در دیتاست اصلی دیتا متوازن می باشد و به ازای هر ۴ کلاس ما ۴۰۰ رکورد داریم.

در زمانی که دیتا imbalance باشد میتوان رویکرد های زیر را در نظر گرفت.

– در ابتدا در مواجهه با دیتای imbalanced باید توجه داشت که باید metric های مناسبی داشته باشیم تا بتوانیم بهترین evaluation را بین مدل های مختلف پیدا کنیم و بعضی از متریک هایی که میتوان نام برد به شکل زیر است :

Precision/recall/f۱-score/mcc/auc

در مرحله بعدی برای اینکه مدل یادگیری بهتر train ببیند میتوان از دیتاست های مختلفی که از دیتاست اصلی بدست می ایند استفاده کرد که چندین روش برای انجام این کار موجود است

- Under sampling

که در این روش دیتاست تبدیل به دیتایی balanced خواهد شد به صورتی که تمام رکورد ها با کلاسی که جمعیت کمتری دارند نگه داشته میشنوند ولی از رکورد هایی که از کلاسی بودند که جمعیت زیادی داشتند تعدادی (بسته به تعداد رکورد های کلاس کم تعداد) به صورت رندوم انتخاب شده و با این تعداد جمعیت رکورد های با کلاس جمعیت بالا را کم کرده ایم و دیتاستی با جمعیت های تقریبا برابری خواهیم ساخت

- Over sampling

در این روش دیتاست تبدیل به دیتایی balanced خواهد شد به صورتی که تعداد رکورد هایی که در کلاس با جمعیت کم هستند را افزایش میدهیم ولی تعداد رکورد هایی که در کلاس با جمعیت بالا هستند تغییری نمی کند. برای این کار چندین روش موجود است روش اول به این صورت است که دیتا هایی دقیقا یکسان با دیتا های قبلی دوباره در دیتاست اضافه میشوند به صورت رندوم(repetition) و یا روش دیگر به این صورت است که دیتایی مصنوعی شبیه به دیتای اصلی تولید کرد و ان را به دیتاست اصلی اضافه کرد (SMOTE)

همچنین بعد از انجام over sampling لازم است تا cross validation ای از دیتاست بوجود امده انجام شود از آنجایی که اگر این کار را نکنیم مدل ما over fit میشود به ان دیتاست تولید شده و با دیتای واقعی نمیتواند نتیجه ی خوبی داشته باشد.

همچنین میتوان sample های مختلف دیگری از دیتای اصلی ساخت و بر روی مدل هایی که داریم ان ها را train کرد تا از over fit شدن مدل ها جلوگیری شود از طرفی این سمپل ها میتوانند ratio ی متفاوتی داشته باشند به این معنی که تعداد رکورد های کلاس کم جمعیت در یک سمپل بسیار کم تر از رکورد های کلاس پر جمعیت باشد و بر عکس (مقادیر مختلف ratio برای سمپل های مختلف)

همچنین میتوان یک clustering بر روی دیتا های کلاس با جمعیت زیاد و سمپلی درست کرد از دیتا های کلاس کم جمعیت بعلاوه ی یک پروتوتایپ (مثل میانگین نقاط) و این سمپل را برای مدل ها استفاده کنیم

همچنین اگر بتوان دیتای بیشتری از منابع بدست اورد میتوان سمپل های بهتری تولید کرد.

– تسک ۱۰ داک انجام شد و داده ها نامتوازن شدند و روش های گفته شده در بالا بر روی آن ها انجام شدند که به صورت زیر می باشند.

No technique:

```
--- TEST ---
[[ 99   1]
 [  6 294]]
              precision    recall  f1-score   support

           0       0.94      0.99      0.97       100
           4       1.00      0.98      0.99       300

    accuracy                           0.98       400
   macro avg       0.97      0.98      0.98       400
weighted avg       0.98      0.98      0.98       400
```

Undersampling:

```
--- TEST ---
[[ 98   2]
 [  8 292]]
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       100
           4       0.99      0.97      0.98       300

    accuracy                           0.97       400
   macro avg       0.96      0.98      0.97       400
weighted avg       0.98      0.97      0.98       400
```

Oversampling:

```
--- TEST ---
[[100   0]
 [  9 291]]
              precision    recall  f1-score   support

           0       0.92      1.00      0.96       100
           4       1.00      0.97      0.98       300

    accuracy                           0.98       400
   macro avg       0.96      0.98      0.97       400
weighted avg       0.98      0.98      0.98       400
```

SMOTE:

```
--- TEST ---
[[100   0]
 [  8 292]]
              precision    recall  f1-score   support

           0       0.93      1.00      0.96       100
           4       1.00      0.97      0.99       300

    accuracy                           0.98       400
   macro avg       0.96      0.99      0.97       400
weighted avg       0.98      0.98      0.98       400
```

Logistic regression using class-weights:

```
--- TEST ---
[[100   0]
 [  9 291]]
              precision    recall  f1-score   support

           0       0.92      1.00      0.96       100
           4       1.00      0.97      0.98       300

    accuracy                           0.98       400
   macro avg       0.96      0.98      0.97       400
weighted avg       0.98      0.98      0.98       400
```

همانطور که از عکس های بالا قابل مشاهده است با مقایسه confusion matrix های روش های مختلف، استفاده از روش SMOTE باعث شده که پرفورمنس مدل مقداری بهتر شود ولی در کل از آنجایی که دیتاست کوچک است و دیتا هم مقدار زیادی نامتوازن نیست (۱/۴ دیتا را کلاس صفر تشکیل داده.) تمامی مدل ها پرفورمنس تقریبا یکسانی دارند.

۴. درست است که استفاده از pca در پروژه داده شده تاثیر زیادی بر روی خروجی مدل نداشته ولی دلیل بر بی استفاده بودن آن نیست زیرا اولا از pca برای تسک های ویژوالیزیشن و بررسی داده نیز استفاده میشود و دوما pca صرفا متغییر هایی که مقدار واریانس آن ها کمترین است را از دیتاست حذف میکند و تضمینی نمیکند که این حذف کردن بهترین عملی است که میتوان بر روی دیتا انجام داد تا پرفورمنس مدل را بالا برد همانطور که در اینجا از آنجایی که دیتاست کوچک بود تاثیر خاصی نداشت ولی ممکن است در پروژه ی دیگری با دیتاست بزرگتر از curse of dimensionality نیز جلوگیری کند و پرفورمنس مدل را افزایش دهد.

۵. بخش پری پروسسینگ (هندل کردن داده ها با چولگی) توسط پیکج Dask انجام شد و مقایسه نتایج به این صورت است که با استفاده از پکیج پانداس این عمل در ۰.۰۰۶ ثانیه انجام میشود ولی پکیج Dask با استفاده از ۳ partitions ۰.۰۱۵ ثانیه این عمل را طول میدهد. همچنین به ازای partitions های مختلف این عمل با استفاده از این پکیج تست شده که نتایج در نمودار زیر مشخص هستند و با استفاده از ۳ partitions ما کمترین زمان را مصرف خواهیم کرد.