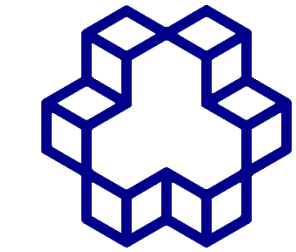
Google drive: https://drive.google.com/drive/folders/1n8uWALN39KWm7Zrq6tpUlCi1efUUQn4D?usp=sharing

Github: https://github.com/AliBagheriNejad/ML-AliYari



گزارش نیمچه پروژه سوم درس یادگیری ماشین

علی باقری نژاد

۴۰۲۰۲۸۵۴

ازمیان سوال ۱و۲، سوال دوم برای حل انتخاب شد

فهرست

سوال ۲

طرح مسئله

SVM به صورت بهینه ای می تواند مسائل طبقه بندی خطی و غیری خطی را با استفاده از support vector حل نماید. درضمن با استفاده از regularization هم می‌توان از مسئله overfitting روی داده های آموزش نیز جلوگیری کرد. به دلیلی عملکرد مناسب این الگوریتم روی طبقه‌بندی دو کلاسه، تلاش‌های بسیاری انجام شده که بتوان این الگوریتم را برای طبقه‌بندی چندکلاسه هم استفاده نمود.

به صورت کلی روش‌های پیشنهاد شده برای حل مسئله MSVM را می‌توان به ۳ بخش تقسیم نمود:

۱. **روش‌های اکتشافی**: این روش‌ها از SVM دوکلاسه به عنوان اساس روش خود استفاده کرده و مسئله طبقه‌بندی چندکلاسه را به چندین مسئله دوکلاسه تبدیل می‌کنند. از جمله این روش‌ها می‌توان به روش‌های one-vs-oneیا one-vs-all اشاره کرد.

۲. **روش‌های اصلاح خطا کد:** این روش از کدهای اصلاح خطا استفاده می‌کند. در این روش‌ها، مسئله به چندین مسئله طبقه‌بندی دوکلاسه تبدیل می‌شود که زیرمسائل طبقه‌بندی دوکلاسه توسط یک ماتریس کدینگ ساخته شده، گروه بندی می‌شوند.

۳. **روش‌های تک ماشین:** این روش‌ها صرفا با استفاده از یک تابع تبدیل و بهینه کردن آن، مرزبندی‌های همه کلاس‌ها را به صورت هم زمان انجام می‌دهند.

روش‌های کنونی از مشکلات متعددی رنج می‌برند.

- روش‌هایی که بیس طبقه‌بندی دوکلاسه دارند، باید تعداد زیادی مسئله بهینه‌سازی را انجام دهند؛ که این باعث افزایش هزینه و طولانی شدن زمان حل مسئله می‌شود.
- روش‌های تک ماشینه، بر اساس misclassificationهای کلاس مشاهده شده با کلاس‌های دیگر می‌باشند. جمع کردن خطاهای misclassification که از مشاهدات خطا چندین کلاس بدست نقش

پررنگ تری نسبت به misclassification هایی دارند که تنها از مشاهده یک کلاس بدست می‌آیند. یعنی تاثیر خطا دسته اول تاثیر بیشتری روی بهینه‌سازی می‌گذارد که این ویژگی مناسبی برای طبقه‌بندی چندکلاسه نیست.

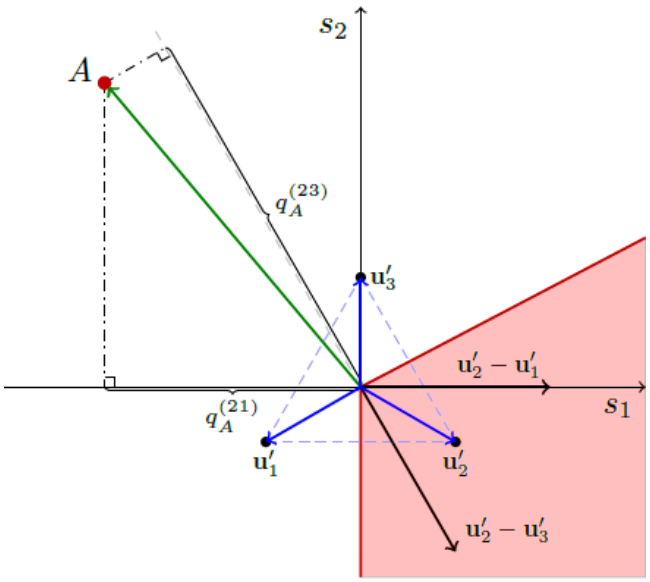
معرفی روش

مدل پیشنهاد شده یک ساختار encoding ساده را برای فرمول‌بندی کردن MSVM پیشنهاد می‌دهد که ابعاد مسئله را به یک مسئله طبقه‌بندی K-1 (K تعداد کل کلاس ها می باشد) بعدی کاهش می‌دهد و بدین صورت تنها یک مسئله بهینه‌سازی را حل می‌نماید. با استفاده از یک تابع flexible hinge و یک نُرم l_p برای خطا، تابع هزینه GenSVM سه روش MSVM را با هم ترکیب می‌کند و در عین حال آن روش‌ها را گسترش می‌دهد.

نگاشت از فضای ورودی به فضای simplex توسط minimize کردن misclassification error حل می‌شود. این خطا توسط محاسبه فاصله مختصات از مرز تصمیم‌گیری در فضای simplex محاسبه می‌شود.

محاسبه خطا

اگر نقطه x را در فضای ورودی درنظر بگیریم، در فضای simplex خواهیم داشت $\mathbf{s}_i' = \mathbf{x}_i'\mathbf{W} + \mathbf{t}'$ که درواقع نگاشت نقطه در فضای simplex می‌باشد.



تصویر بالا را در نظر بگیرید. تقطه A مربوط به کلاس ۲ می‌باشد. برای محاسبه خطا مربوط به این نقطـه، ابتـدا فاصه این نقطه نسبت به محورها $u_k - u_j$ محاسبه می نماییم(u ها تعریف خاص خودشان را دارند که در این گـزارش آورده نشده اند). $q_i^{(kj)} = (\mathbf{x}_i'\mathbf{W} + \mathbf{t}')(\mathbf{u}_k - \mathbf{u}_j).$ با استفاده از این فرمول فاصله نقطه محاسبه مـی شـود. در ایـن الگوریتم قصد بر این است که تا جای ممکن q در بیشترین مقدار خود باشد. برای این کار از تابع خطا Huber-hinge استفاده می‌شود. البته باید این نکته را ذکر کرد که خطا محاسبه شده، وزن دار می باشد.

$$h(q) = \begin{cases} 1 - q - \frac{\kappa+1}{2} & \text{if } q \leq -\kappa \\ \frac{1}{2(\kappa+1)}(1-q)^2 & \text{if } q \in (-\kappa, 1] \\ 0 & \text{if } q > 1, \end{cases}$$

تابع بالا، تابع خطا Huber-hinge را نشان می‌دهد. در این جا اگر فاصله بیشتری از مقدار ۱ نسـبت بـه بـردار محور کلاس واقعی (در این جا کلاس ۲) باشد، تابع هیچ خطایی ندارد. اما هر چه که از این خط فاصـله بیگیـریم و بـه سمت مخالف بردار کلاس ۲ حرکت کنیم، به همان نسبت خطا خواهیم داشت.

نرم p مربوط به هر خطا به صورت زیر محاسبه می شود:

$$\left(\sum_{\substack{j=1 \\ j \neq y_i}}^{K} h^p \left(q_i^{(y_i j)} \right) \right)^{1/p}.$$

نرم p به این دلیل تاثیر regularizationای که دارد، به عبارت خطا اضافه می شود .

در این الگوریتم، از p و κ به عنوان پارامتر استفاده می شود.

$$L_{\mathrm{MSVM}}(\mathbf{W}, \mathbf{t}) = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in G_k} \rho_i \left(\sum_{j \neq k} h^p \left(q_i^{(kj)} \right) \right)^{1/p} + \lambda \operatorname{tr} \mathbf{W}'\mathbf{W}$$

با استفاده از فرمول بالا، مقدار وزن مربوط به هر داده وارد بازی می شود.. در مقاله اثبات می شود که تابع هزینه بالا، یک تابع convex می باشد.

در نهایت با استفاده از $\hat{y}_{n+1} = \arg\min_k \|\mathbf{s}'_{n+1} - \mathbf{u}'_k\|^2, \quad \text{for } k = 1, \ldots, K.$ ، مقادیر لیبل update می شوند.

بهینه سازی

برای بهینه سازی و کمینه کردن خطا، از الگوریتم iterative majorization (IM) استفاده می شود.

نتیجه‌گیری

این مقاله، روش خود را با سه روش دیگر مقایسه کرد و گزارش داده که روش طراحی شده، نسبت به روش های دیگر از لحاظ پیشبینی خروجی، ۵٪ بهبود عملکرد داشته است که GenSVM را به عنوان بهترین مدل درمیان مدل های تک ماشینه تبدیل می کند. در مقایسه با ۵ مدل دیگر نیز، GenSVM دارای کمترین مدت زمان لازم برای آموزش بود. این درحالی بود که تعداد پارامترهای آموزش داده شده، ۱۸ برابر دیگر مدل ها بود.

برتری ها:

- سرعت محاسباتی بالا در عین داشتن پارامتر های بیشتر
- دارای قابلیت warm start که باعث کاهش زمان آموزش می شود (کاهش زمان آموزش در مقاطع ابتدایی آموزش مدل)
- انعطاف پذیری مدل (انعطاف پذیری تابع هزینه تعریف شده)
- قابلیت Generalizatoin

نوآوری ها

از جمله خلاقیت های بکار گرفته شده در ساختار مدل می توان به موارد زیر اشاره کرد:

- تعریف فضای simplex برای کاهش ابعاد
- استفاده از Huber-hinge و نرم p برای انفطاف‌پذیر کردن مدل و افزایش generalization

- استفاده از الگوریتم IM که قابلیت warm start را به الگوریتم می دهد

پیاده‌سازی

مدل GenSVM به صورت یک تابع از پیش آماده در کتابخانه gensvm موجود است و برای استفاده از آن کتابخانه را بر روی محیط colab نصب کردیم. البته برای ادامه دادن محاسبات این مدل، باید یک نسخه قبلی scikit-learn را هم نصب می کردیم.

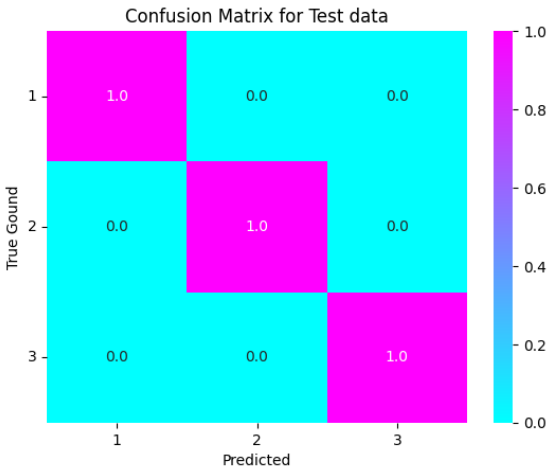
برای مقایسه عملکرد مدل در تمرین با مقاله، از دیتاست IRIS استفاده شد. در قدم اول به نسبت ۱به۹، داده تست از داده آموزش جدا شد. قبل از استفاده از داده برای آموزش مدل، داده ها با روش `standardscaler` ، نرمال شدند. سپس با استفاده از `GridsearchCV`، با استفاده از 10-fold، بهترین مقادیر پارامترهای زیر انتخاب شد:

$$p: \{1,2\}, \kappa: \{0,1,5,10\}, C: \{0.1,1,10,100\}$$

```
Best parameters found: {'coef': 10, 'kappa': 1, 'p': 2}
Best cross-validation score: 0.99
Accuracy of best parameters on Test data: 100.00
```

تصویر بالا، بهترین پارامترها، عملکرد مدل را برای داده های آموزش (10-fold CV) نمایش می دهد. از طرفی دقت بهترین مدل برای داده های تستی که از قبل جدا شده بودند گزارش شده است.

مدل	دقت گزارش شده
مقاله	٪ ۹۶.۰۰
تمرین	٪ ۹۸.۵۱



تصویر بالا، ماتریس درهم ریختگی را برای داده های تست به تصویر کشیده است. دقت بدست آمده نسبت به مقاله بیشتر می باشد. میتوان این مسئله را بدین صورت تحلیل کرد که از آنجایی که در این جا از 10-fold استفاده کردیم و مقاله از 5-fold و از طرفی در این تمرین، قبل از آموزش داده ها، یک بخش آزمون جدا گذاشته شد و بر ۹۰ درصد باقی مانده، برای K-fold CV استفاده کردیم.

سوال ۳

چالش ها و راه حل

چالش های موجود در بحث تشخیص تقلب، با توجه به مقاله، عبارت اند از:

- رفتار کلاه برداری متغییر (دینامیک): یعنی کلاه برداری ها از روش های مغییری استفاده می کنند که کار خود را قانونی جلوه دهند.
- کمبود دیتاست: دیتاست برای تراکنش های مالی حساب ها به سختی وجود دارد
- عدم تعادل دیتاست: دیتاست ها، عموما دارای unbalancing می باشند و داده های مربوط به تراکنش های قانونی بسیار بیشتر از کلاه برداری ها می باشد.
- انتخاب ویژگی مناسب
- معیار مناسب برای سنجش عملکرد مدل تشخیص کلاه برداری

- روند انتخاب نمونه

این مقاله برای مقابله با چالش ها موجود، از راه حل های زیر استفاده کرده است:

- ✓ اعمال PCA روی داده های برای حذف کردن ویژگی های بی تاثیر
- ✓ استفاده از AutoEncoder برای از بین بردن نویز در دیتاست
- ✓ بهره گیری از over sampling (روش SMOTE) برای متعادل کردن دیتاست
- ✓ استفاده از یک MLP به عنوان classifier
- ✓ درنظر گرفتن معیار recall به همراه معیار accuracy

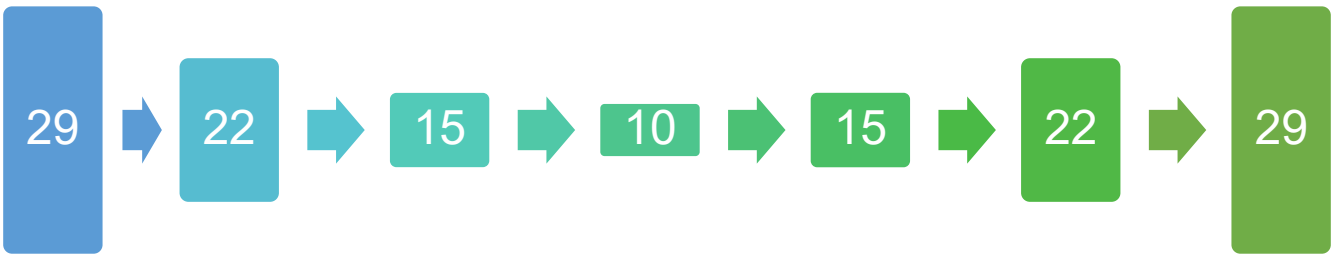
معماری شبکه ارائه شده

قبل از توضیح معماری شبکه باید بیان شود که این مقاله، قبل از دادن داده ها به شبکه برای آموزش، بر روی آن ها over sampling اعمال کرد و سپس شبکه ها را طراحی کرد.

به صورت کلی دو شبکه عصبی در این مقاله ارائه شده است:

۱. ساختار Auto Encoder
۲. ساختار MLP

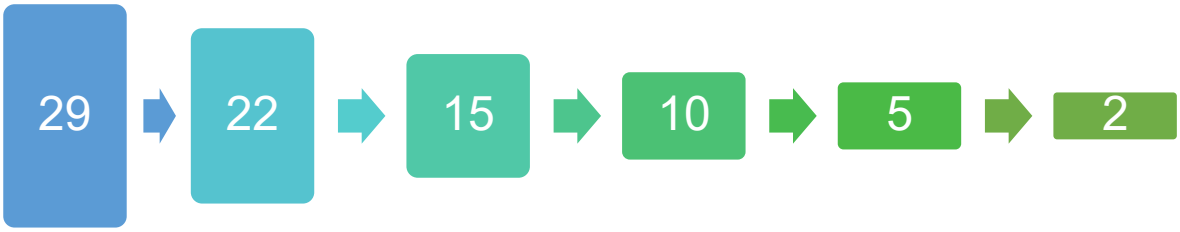
Auto Ecoder برای از بین بردن نویز در داده استفاده شده است.



شکل بالا ساختار استفاده شده برای Auto Encoder را به نمایش گذاشته است. هدف این است که با کاهش ابعاد، ویژگی های بی ربط و به درد نخور حذف شده و با دوباره ساختن داده ورودی توسط ویژگی های latent، داده هایی بدون نویز بازتولید شوند.

در این ساختار، از MSE به عنوان تابع هزینه استفاده شده است. دلیل استفاده از این تابع این است که اختلاف بین خروجی و ورودی را به کمترین مقدار ممکن برساند. یعنی خروجی ای که ایجاد می شود، شبیه ورودی شبکه باشد.

ساختار بعدی مربوط به classifier می باشد که هدف آن تشخیص تراکنش قانونی از تراکنش کلاه برداری می باشد.



شکل بالا شماتیک ساختار MLP طراحی شده برای طبقه بندی را به نمایش می گذارد. به عنوان تابع هزینه، در این ساختار، از تابع SoftMax استفاده شده است. این تابع هزینه برای طبقه بندی چند کلاسه مناسب می باشد.

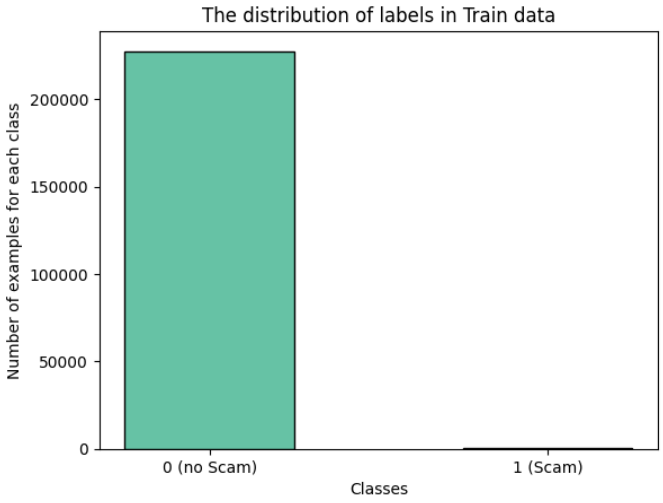
پیاده سازی مدل

دیتاست را از Kaggle دانلود کرده و با استفاده از کتابخانه panda آن را می خوانیم. سپس سطون Time را از داده حذف می کنیم چراکه مقاله هم از این داده استفاده ای نکرده است.

مرحله بعد، تقسیم داده ها به بخش های آموزش و آزمون تقسیم می کنیم. همانند مقاله، نسبت تقسیم داده ها ۲به۸ می باشد. ۲۰٪ مربوط به داده های آزمون و ۸۰٪ مربوط به داده های آموزش.

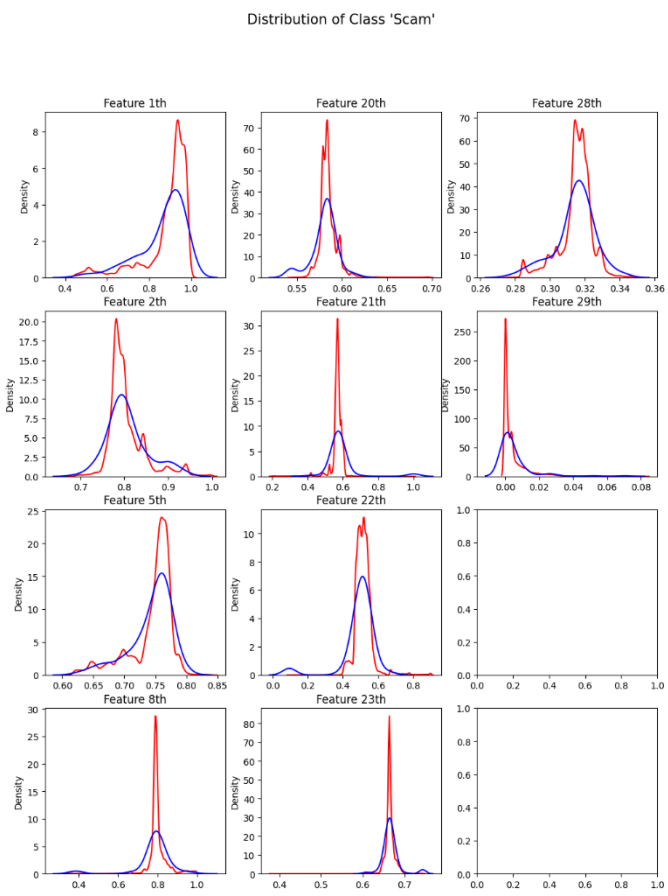
```
Shape of train input data: (227845, 29)
Shape of test input data: (56962, 29)
Data distribution of Train data for each class:
Class
0    227464
1       381
Name: count, dtype: int64
Data distribution of Test data for each class:
Class
0    56851
1      111
```

در تصویر بالا، تعداد داده های هر بخش را بعد از تقسیم ملاحضه می نمایید. همانطور که ملاحظه می شـود، در بخش آموزش، تعداد داده مربروط به هر کلاس یکسان نیست و دیتاست دچار عدم تعادل می باشد.



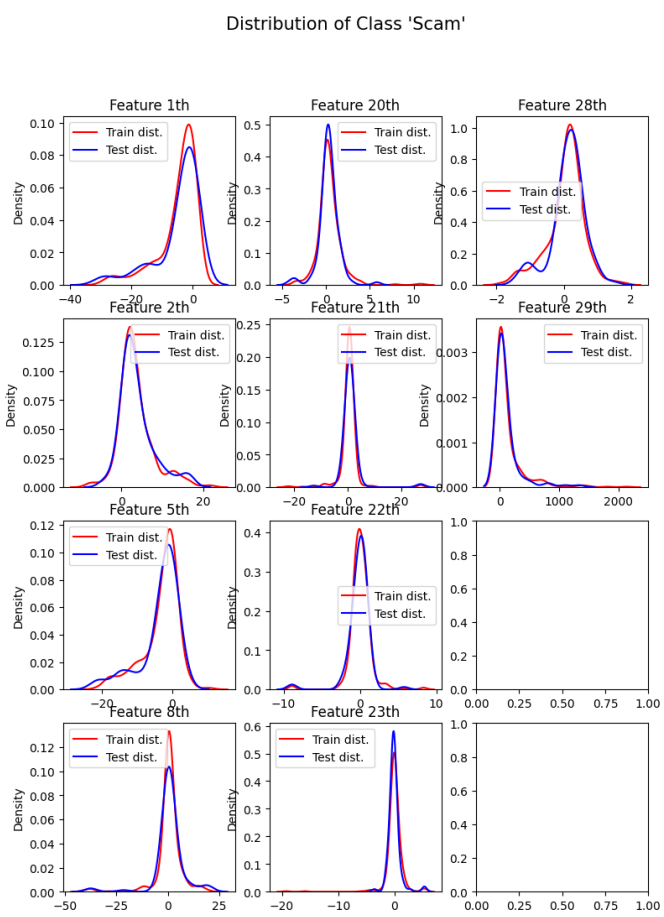
با توجه به شکل بالا، به بالانس کردن داده آموزش نیاز داریم.

توجه: مدل از این قسمت، تا به انتها، چندین بار آموزش داده شد، اما هیچ گاه روی کلاس "تقلب" داده آزمون (اعتبارسنجی) نتیجه خوبی دریافت نمی شد ولی برای کلاس "تقلب" داده آموزش، نتیجه خـوبی بدسـت مـی آمـد. بـا تفسیر این موضوع، به این نتیجه رسیدم که امکان دارد که با اعمـال SMOTE روی داده آمـوزش، پخـش داده بـرای کلاس "تقلب" تغییر کرده باشد و به همین دلیل بر روی داده آزمون، نتایج خوبی دریافـت نمـی شـود. بنـابراین ، تـابع چگالی احتمال برخی از ویژگی های داده آزمون برای کـلاس تقلـب، هـم بـرای داده آزمـون و هـم بـرای داده آمـوزش (SMOTE شده) ترسیم شد تا این فرضیه تایید شود.



همان طور که در شکل بالا ملاحظه می شود، تابع چگالی احتمال بـرای داده SMOTE شـده(داده آمـوزش) و برای داده آزمون (SMOTE نشده) یکسان نیست. (قرمز مربوط به داده آموزش و آبی مربوط به داده آزمون است)

دلیل این تفاوت از آنجا نشأت می گیرد که، پخش داده آموزش و آزمون، برای کلاس تقلب، یکسان نیست، پس یکبار دیگر، تقسیم داده را انجام می دهیم، با این تفاوت که هر کدام از کلاس ها را به صورت جداگانه split می کنیم و سپس با هم ترکیب می نماییم. با استفاده از دستور `train_test_split` مقادیر آمـوزش و آزمـون هـر کـلاس جـدا شدند ولی از آنجا که پخش داده آموزش و آزمون کلاس تقلب مـی بایسـت یـک باشـد، یـک `random_state` مناسـب انتخاب شد تا به پخش یکسانی برسیم.

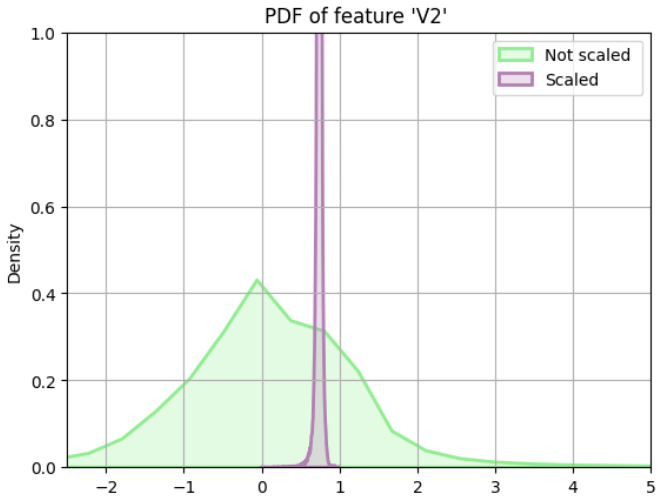


با توجه به شکل بالا، به پخش داده یکسانی برای آزمون و آموزش رسیده ایم. حال داده های آزمون و آموزش را با هم ترکیب می کنیم و محاسبات را ادامه می دهیم.

```
Shape of train input data: (227845, 29)
Shape of test input data: (56962, 29)
Data distribution of Train data for each class:
Class
0    227452
1       393
Name: count, dtype: int64
Data distribution of Test data for each class:
Class
0    56863
1       99
```

در نهایت، داده های آموزش و آزمون به صورت بالا در خواهند آمد.

در مرحله بعد، داده ها را با استفاده از `MixMaxScaler`، نرمال می کنیم.

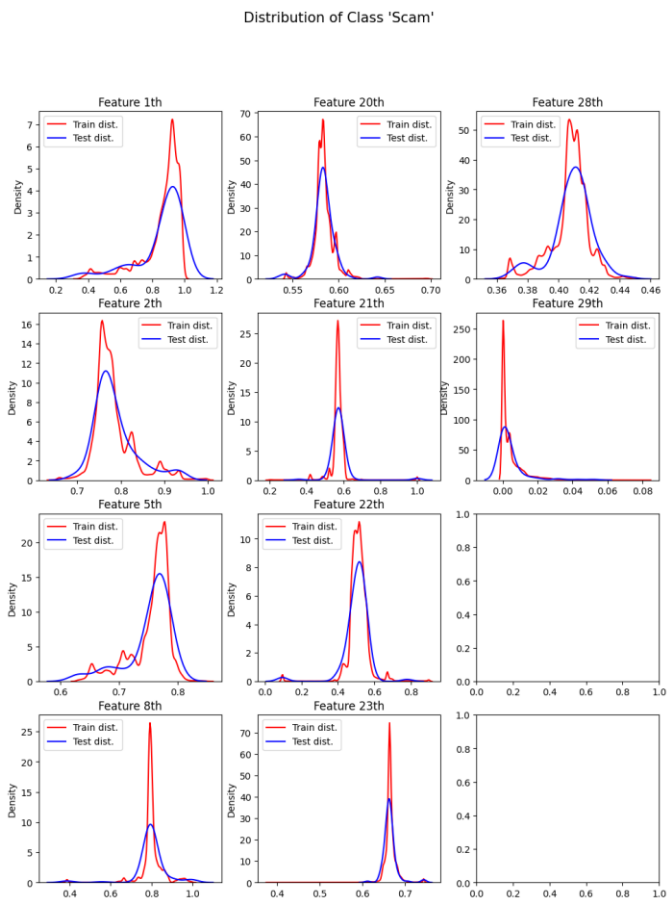


تصویر بالا، نرمال سازی داده را برای ویژگی V2 نمایش می دهد. همانطور که مشاهده مـی شـود، تمـامی داده بین ۰ و ۱ قرار گرفته اند.

مقاله برای نمایش عملکرد مدل خود، یکبار بدون Smote و AE طبقه بندی را انجام داد و باردیگر با این ابزار ها مدل خود را ایجاد کرد و سپس آن ها را با هم مقایسه نمود. در این تمرین، مدل بدون اسـتفاده از SMOTE و AE را آموزش نمی دهیم و صرفا مدل را با استفاده از این دو ابزار ایجاد می کنیم.

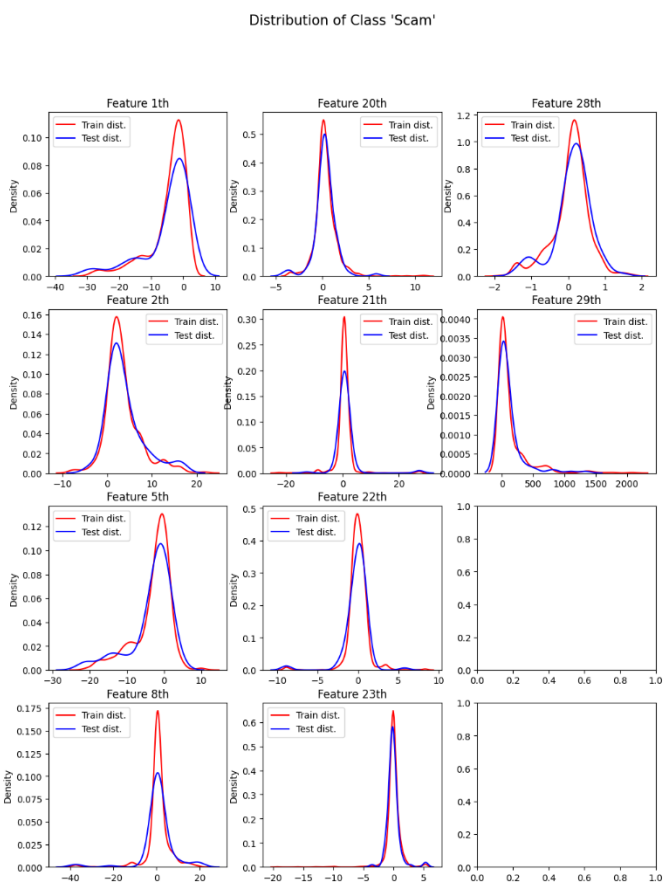
قدم بعـد، اسـتفاده از SMOTE بـرای بـالانس کـردن داده اسـت. بـرای اسـتفاده از ایـن روش، از کتابخانـه imblearn استفاده می شود.

از SMOTE برای اصلاح تعادل در داده آموزش استفاده شد، ولی باز هم مشکل به هـم خـوردن تـابع چگـالی احتمال کلاس تقلب پیش آمد.

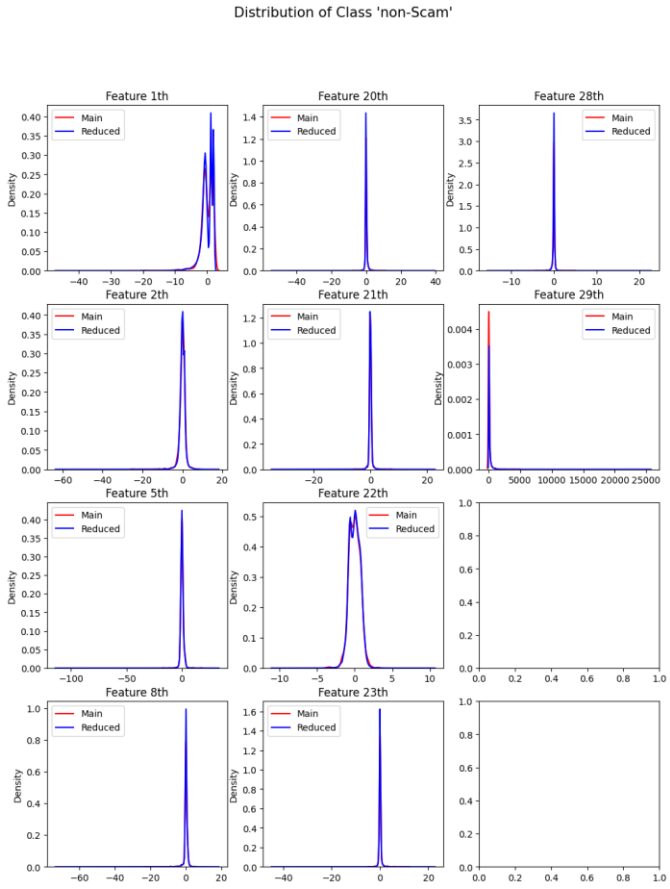


ملاحظه می شود که در شکل بالا، باز هم مشکل یکسان نبودن پخش داده های مربوط به کـلاس تقلـب پـیش آمد.

برای راهکار این مشکل، از تعداد داده های کمتری برای کلاس داده عادی استفاده شده؛ یعنی به جای اسـتفاده از تمامی ۲۰۰۰۰۰ نمونه، تنها از ۱۰۰۰ نمونه استفاده کردیم.



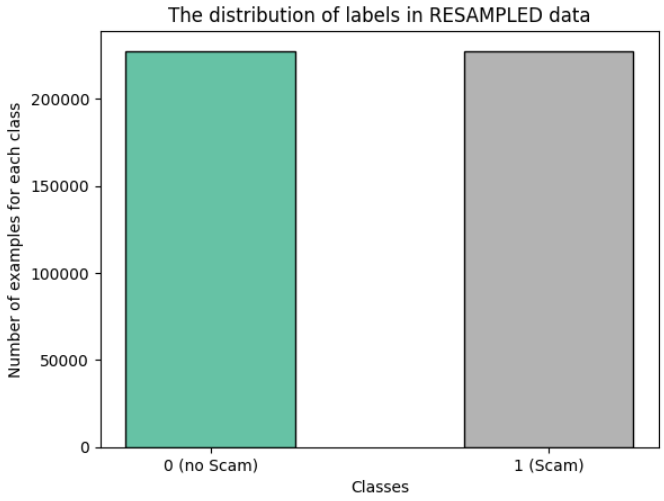
همانطور که مشاهده می شود پخش داده در این حالت، نسبت به حالت قبلی ، بسیار پیشرفت کرده است و مـی توان گفت که داده آموزش و آزمون برای کلاس تقلب، بسیار به نزدیک هستند و انتظار داریم که نتایج مشـابهی بدسـت آید.



تصویر بالا مربوط به پخش داده کلاس عادی می باشد. یعنی ۱۰۰۰ داده انتخاب شده، نماینده خوبی برای تمامی داده ها هستند.

```
Shape of RESAMPLED X: (2274, 29)
Data distribution of RESAMPLED Class:
Class
0    1137
1    1137
```

بعد از استفاده از SMOTE، تعداد داده ها به شکل بالا در خواهد آمد. درشکل زیر هم پخش داده جدید را ملاحضه می کنید.



حال از این بخش دیتا جدید برای ادامه مسیر استفاده می کنیم.

قدم بعدی، اعمال AE به داده ها می باشد. برای این کار از کتابخانه tensorflow استفاده می نماییم.

```
Model: "Autoencoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 EC_1 (Dense)                (None, 22)                660

 EC_2 (Dense)                (None, 15)                345

 Latent (Dense)              (None, 10)                160

 DC_1 (Dense)                (None, 15)                165

 DC_2 (Dense)                (None, 22)                352

 OUT (Dense)                 (None, 29)                667

=================================================================
Total params: 2349 (9.18 KB)
Trainable params: 2349 (9.18 KB)
Non-trainable params: 0 (0.00 Byte)
```

ساختار AE ایجاد شده را در تصویر بالا مشاهده می نمایید. حال باید این شبکه را بر روی داده های ورودی آموزش دهیم. از تابع ReLU به عنوان تابع فعالسازی تمامی لایه ها استفاده شد.

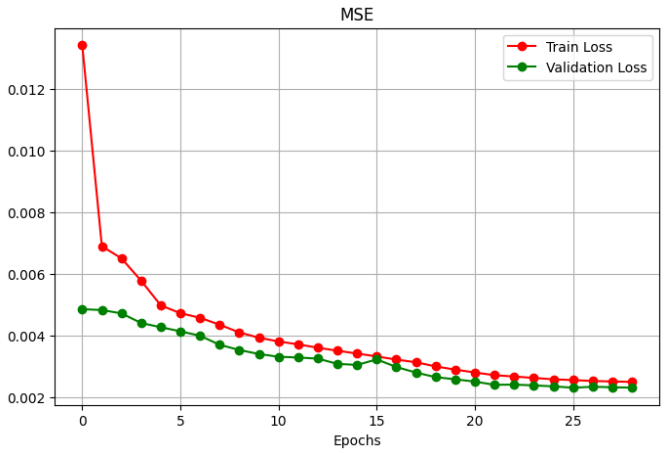
البته باید این نکته ذکر شود که، داده های آموزش را قبل از اعمال به AE، به آن ها یک نویز گوسی اضافه می کنیم. در مقاله ذکر نشده بود که مقدار این نویز چقدر می باشد، بنابراین یه نویز گوسی به shape داده آموزش ایجاد کرده و آن را به داده آموزش اضافه می کنیم. داده نویزی با استفاده از np.random.rand تولید می شود و در یک ضریب (۰.۰۰۱) ضرب می شود و به داده نهایی اضافه می شود.

برای تابع هزینه، همانند مقاله، از MSE بهره گرفته شد. برای بهینه سازی از الگوریتم Adam استفاده کردیم.

در صورت سوال خواسته شده بود که مدل با بهترین وزن ها (کمترین خطا اعتبارسنجی) ذخیره شود. برای این کار می توان از callbackی به نام earlystopping استفاده کرد. این callback را به هنگام آموزش مدل استفاده می کنیم که هم بهترین مدل را بازگرداند و هم از آموزش بیش از حد مدل خودداری کند. از طرفی با استفاده از ModelCheckpoin، بهترین وزن ها را هم ذخیره می نماییم.

```
x_train_scaled
x_train_scaled
epochs = 30
batch_size = 128
validation_data = (x_test_scaled, x_test_scaled)
callbacks = [checkpoint, early_stopping]
```

بالا، پارامترهای مربوط به آموزش AE نمایش داده شده است.



شکل بالا، نمودار خطا (MSE) مربوط به مدل، حین آموزش را نمایش می دهد.

بعد از denoise کردن داده ها، باید طبقه بندی را انجام دهیم.

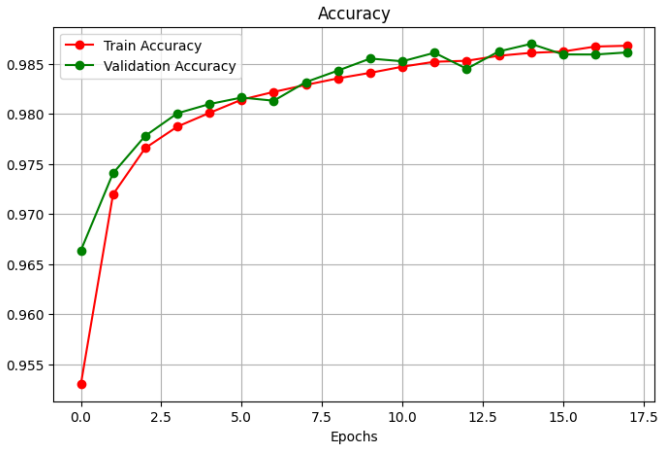
```
Model: "CLASSIFIER"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Dense_1 (Dense)             (None, 22)                660

 Dense_2 (Dense)             (None, 15)                345

 Dense_3 (Dense)             (None, 10)                160

 Dense_4 (Dense)             (None, 5)                 55

 Classifier (Dense)          (None, 2)                 12

=================================================================
Total params: 1232 (4.81 KB)
Trainable params: 1232 (4.81 KB)
Non-trainable params: 0 (0.00 Byte)
```

ساختار شبکه به صورت بالا می باشد. برای تمامی لایه ها به جز لایه آخر، از تابع فعالسازی ReLU استفاده شد. همانند مقاله، در لایه آخر از SoftMax استفاده کردیم.

برای تابع هزینه از SparseCategoricalCrossentropy استفاده کردیم (به دلیل این که از دو نورون برای لایه آخر استفاده شده بود). از تابع بهینه سازی Adam نیز به عنوان optimizer برای این MLP استفاده می کنیم.

نکته مهم این است که قبل از آموزش دادن مدل، باید داده های آموزش و اعتبارسنجی خود را به AE داده تا عمل denoise روی آن هاانجام شود و سپس مدل MLP را آموزش می دهیم.

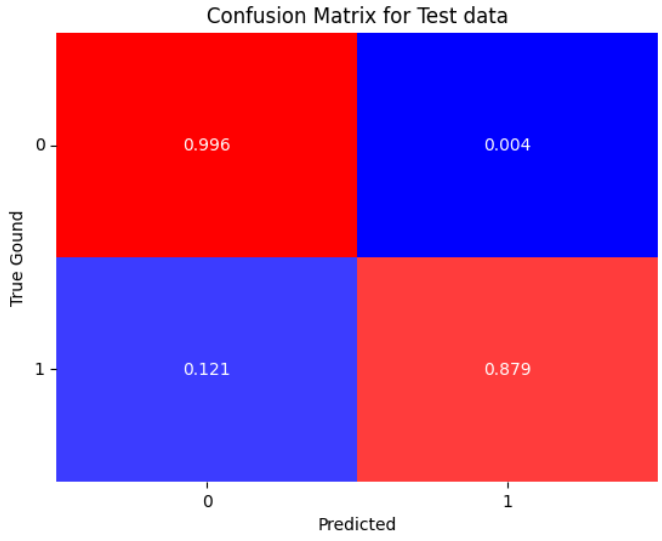
به دلیل استفاده از earlystopping، با اینکه ۳۰ epoch برای آموزش در نظر گرفته شده بود، به دلیل پیشرفت نداشت. با استفاده از این callack، بهترین مدل بر اساس دقت اعتبارسنجی، باز گردانده شد.



تصویر بالا، دقت و recall محاسبه شده روی داده اعتبارسنجی را نمایش می دهد.

```
Classification report for Train Data
              precision    recall  f1-score   support

     no Scam       0.89      0.99      0.94      1137
        Scam       0.99      0.87      0.93      1137

    accuracy                           0.93      2274
   macro avg       0.94      0.93      0.93      2274
weighted avg       0.94      0.93      0.93      2274

Classification report for Test Data
              precision    recall  f1-score   support

     no Scam       1.00      1.00      1.00     56863
        Scam       0.26      0.88      0.40        99

    accuracy                           1.00     56962
   macro avg       0.63      0.94      0.70     56962
weighted avg       1.00      1.00      1.00     56962
```

در تصویر بالا، تمامی معیار های accuracy و recall و precision وf1-score گزراش شده است. همانطور که ملاحظه می شود، عملکرد مدل برای داده آموزش مناسب است. اما در برای داده آزمون، برای کلاس عادی، عملکـرد خوبی دارد ولی برای کلاس تقلب، عملکرد آموزش خوب ولی عملکرد آزمون مناسب نیست. دلیل هم این اسـت کـه بـه دلیل SMOTE پخش داده آزمون و آموزش عوض شده است. اما با این حال، این نتایج از نتایج حالات قبلـی آزمـایش شده بهتر بود (در حالات قبلی f1-score ۰.۰۶ بود که گزارش نشده اند)

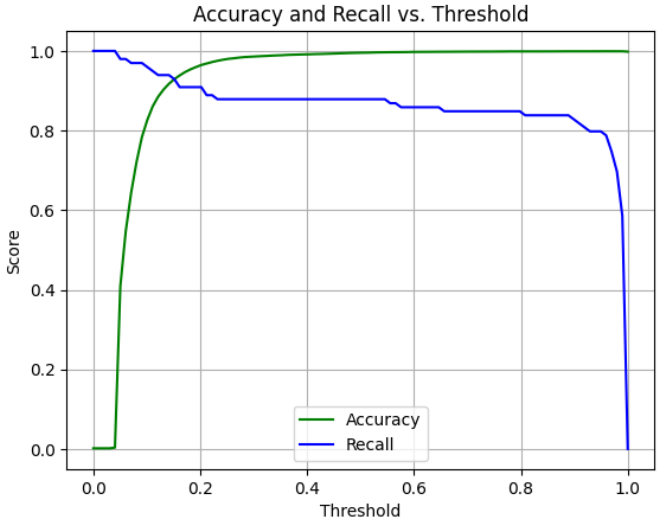


این تصویر نیز مربوط به ماتریس درهم ریختگی می باشد.

دقت به تنهایی نمی تواند معیار خوبی برای ارزیابی مدل ، به هنگام وجود عدم تعادل در داده ها باشد. به این دلیل که این معیار کلاس داده مورد بررسی را درنظر نمی گیرد و صرفا با درست یا اشتباه تخمین زدن لیبل کار می کند. مثلا از از میان ۱۰۰ داده که ۹۸ تای آنها نرمال و ۲ تای آن ها عیب باشند، مدل همه را نرمال در نظر بگیرد، دقت ۹۸ درصد خواهیم داشت و این درحالی است که هیچ یک از داده های عیب تشخیص داده نشده است.

برای حل این مشکل می توان از معیار f1-score استفاده کرد. زیرا این معیار، با ترکیب دو معیار recall و precision، برای هر کلاس، هم داده های درست تخمین زده شده را در نظر می گیرد، هم داده های اشتباه لیبل زده شده. یعنی برای هر کلاس داده های True positive و False positive و Flase negative را در محاسبات خود دخیل می کند. مثلا برای مسئله ای که بیان شد، f1-score برای کلاس نرمال، برابر ۰.۹۹ و برای کلاس عیب برابر ۰ خواهد بود که به خوبی نشان می دهد که مدل در مقابل عمل طبقه بندی چگونه عمل کرده است.

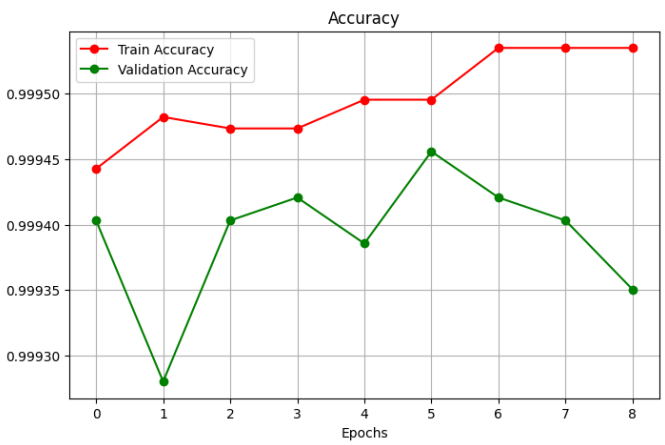
عملکرد در مقابل threshold ها مختلف

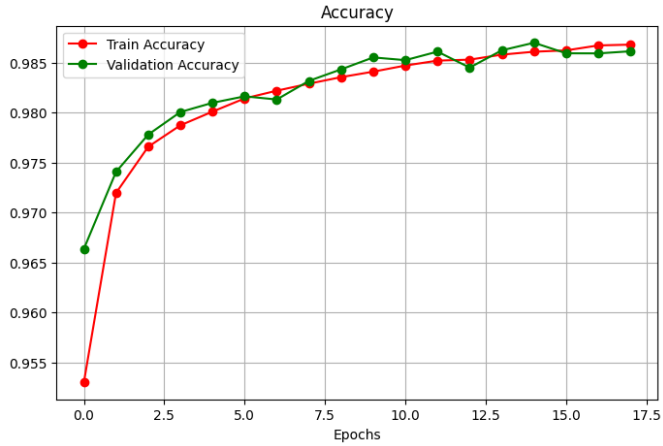


نمودار accuracy و recall براساس آستانه درنظر گرفته شده برای مدل آموزش داده شده روی داده آزمون.

عملکرد مدل بدون ازبین بردن عدم توازن در داده و نویز گیری

نکته: در تمامی تصاویر این بخش، تصاویر سمت راست مربوط به مدل متعادل و denoise **شده** (مدل قبلی) می باشد و تصویر سمت چپ مربوط به مدل متعادل و denoise **نشده** (مدل جدید) می باشند.



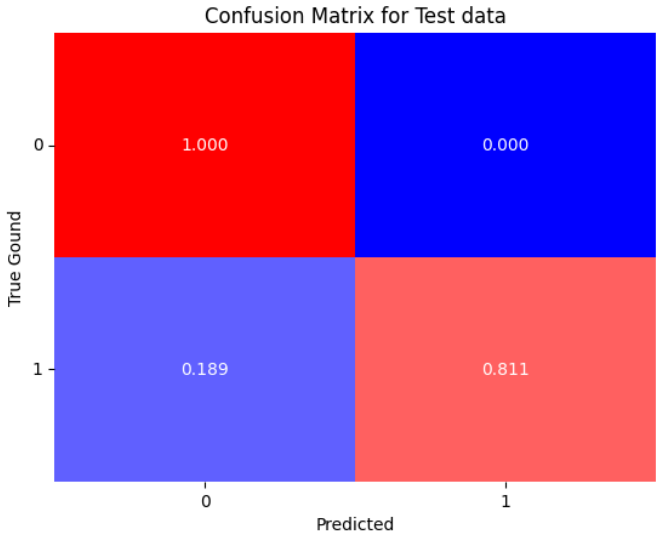


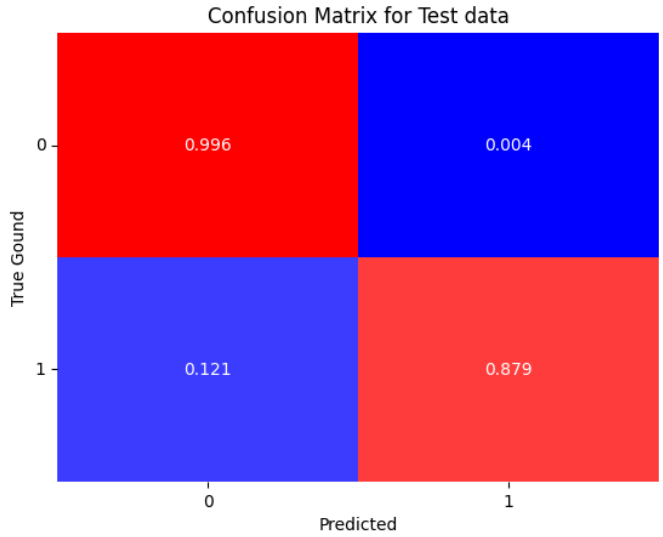
نمودار دقت، دقت بالاتری را نسب به مدل حالت قبل نشان می دهد، اما همان طور که بیان شد، دقت نمیتواند در این حالت معیار مناسبی برای سنجش عملکرد مدل باشد. برای بهتر درک کردن عملکرد مدل، به بیان موارد دیگری می پردازیم.

	precision	recall	f1-score	support
no Scam	1.00	1.00	1.00	56851
Scam	0.86	0.81	0.83	111
accuracy			1.00	56962
macro avg	0.93	0.91	0.92	56962
weighted avg	1.00	1.00	1.00	56962

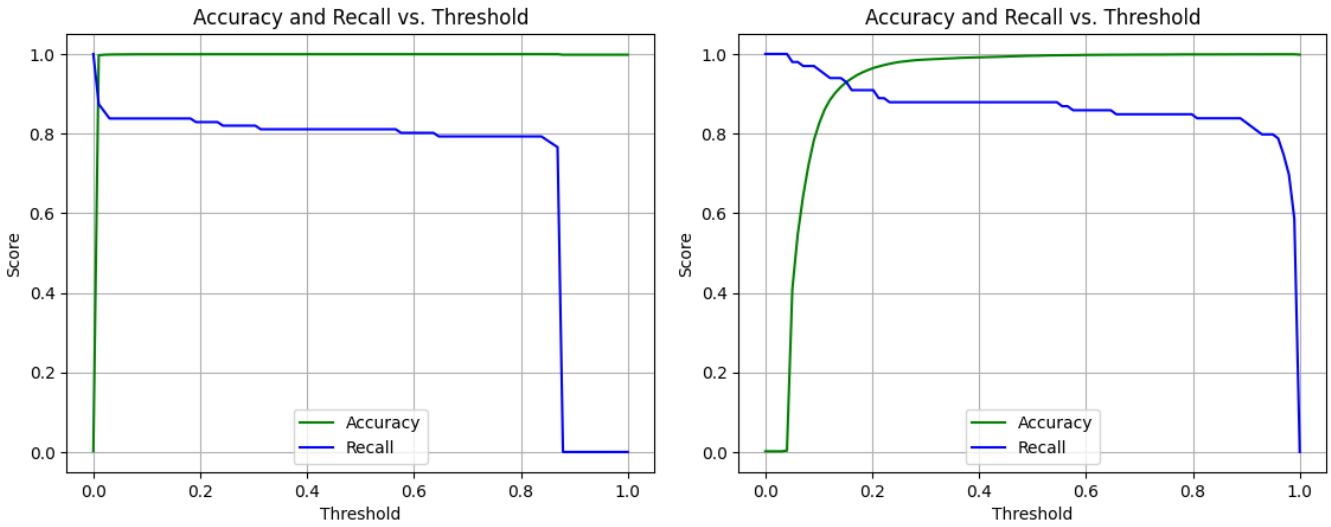
	precision	recall	f1-score	support
no Scam	1.00	1.00	1.00	56863
Scam	0.26	0.88	0.40	99
accuracy			1.00	56962
macro avg	0.63	0.94	0.70	56962
weighted avg	1.00	1.00	1.00	56962

در تشخیص کلاس no Scam، هر دو مدل عملکرد خوبی دارند اما به دلایل گفته شده، عملکرد مدل پیشنهادی در تشخیص





با درنظر گرفتن ماتری درهمریختگی، می توان گفت که عملکرد مدل پیشنهادی تاحد قابل قبولی بهتر است.



تصویر بالا ضعف مدل متوازن و denoise نشده را به خوبی نمایش می دهد.مخصوصا در recall. با این کـه از لحاظ دقت عملکرد مدل پیشنهادی ضعیف تر است، اما از نظر recall عملکرد بهتری دارد.