دانشکده مهندسي برق و کامپیوتر

نام و نام خانوادگی :

امیر اسماعیل زاده نوبری

شماره دانشجویی

40101924

درس یادگیری ماشین

مینی پروژه دوم

استاد درس:

دکتر علیاری

بهار 1403


بسم الله الرحمن الرحیم

Contents

لینک GIT

لینک colab

سوال 1

۱ پرسش یک

هدف از این سوال آزمایش الگوریتم SVM در نمونه‌های مختلف روی دیتاست معروف گل‌زنبق[1] است. مراحل زیر را یک به یک انجام دهید و موارد خواسته‌شده در گزارش خود به همراه کدها ارسال کنید.

آ.

آ. در مرحلهٔ اول دیتاست را فراخوانی کنید و اطلاعاتی نظیر ابعاد، تعداد نمونه‌ها، میانگین، واریانس و همبستگی ویژگی‌ها را به‌دست آورید و نمونه‌های دیتاست را به تصویر بکشید (مثلاً با استفاده از t-SNE). سپس، با توجه به اطلاعات عددی، آماری و بصری بدست آمده، تحلیل کنید که آیا کاهش ابعاد می‌تواند در این دیتاست قابل استفاده باشد یا خیر.

دیتاست را فراخوانی کرده و اطلاعات لازم را از آن دریافت می کنیم:

ابعاد:

```
X.shape , y.shape
((150, 4), (150,))
```

تعداد و توضیحات نمونه ها کلاس ها و ویژگی ها :

```
**Data Set Characteristics:**

:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive attributes and the class
:Attribute Information:
    - sepal length in cm
    - sepal width in cm
    - petal length in cm
    - petal width in cm
    - class:
            - Iris-Setosa
            - Iris-Versicolour
            - Iris-Virginica
```

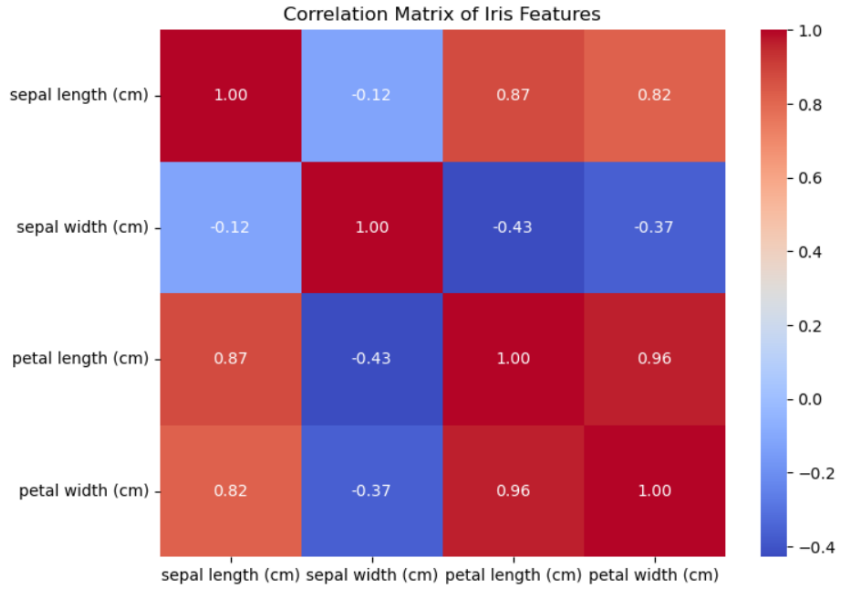
150 نمونه (50 برای هرکلاس) و 4 ویژگی و 3 کلاس داریم.

ویژگی های احتمالاتی دیتاست :

```
:Summary Statistics:

============= ==== ==== ======= ===== ====================
                Min  Max   Mean    SD   Class Correlation
============= ==== ==== ======= ===== ====================
sepal length:   4.3  7.9   5.84   0.83    0.7826
sepal width:    2.0  4.4   3.05   0.43   -0.4194
petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
============= ==== ==== ======= ===== ====================
```

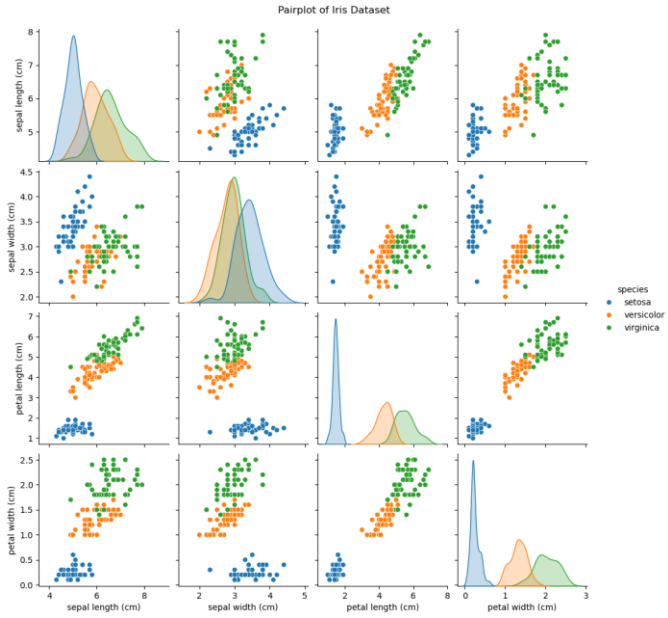
به ترتیب از چپ به راست مینیمم ، ماکسیمم ، میانگین ، انحراف از معیار و همبستگی بین کلاس ها میباشند.

همچنین همبستگی بین ویژگی ها :



مشاهده میکنیم که ویژگیpetal length و petal width بالا ترین همبستگی را با کلاس ها دارند پس از ارزش بالاتری برخوردارند اما باهم نیز همبستگی بالایی دارند که یعنی وابستگی بالایی دارند و صرفا میتوان فقط از یکی از آنها استفاده کرد و دیگری اطلاعات منحصر به فرد تری ندارد.

Pair plot :

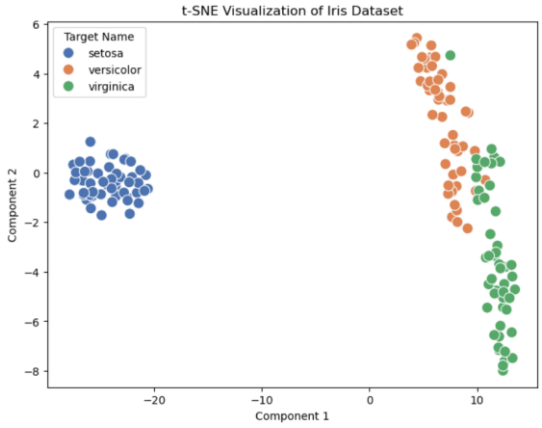


همچنین در اینجا نیز شباهت بین petal width و petal length مشاهده می شود همچنین می بینیم که در pairplot ویژگی sepal width وsepal length طبقه بندی دشوار خواهد بود

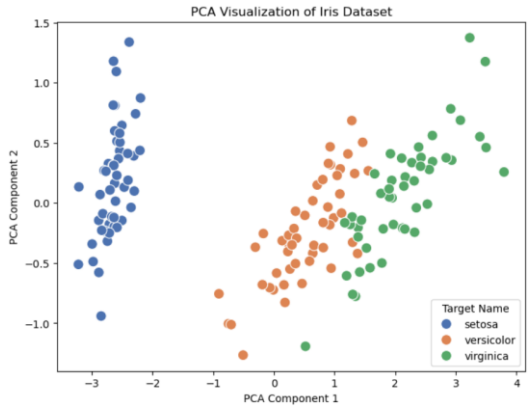
Visualization :

حال برای visualization با کاهش بعد از روش های زیر دیتا ست را به تصویر می کشیم.

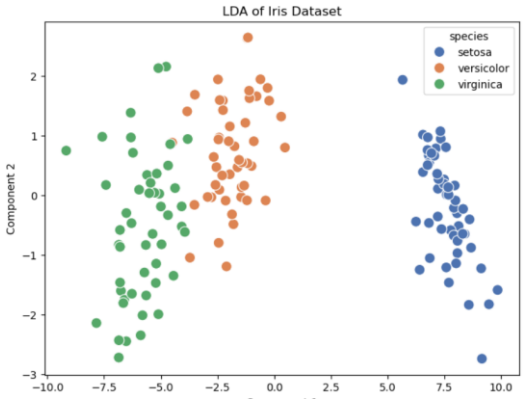
Tsne :



PCA :



: LDA



فرق بین tsne و PCA و LDA :

هدف:

pca : متد خطی بدون نظارت کاهش بعد با تمرکز بر حفظ ماکسیمم واریانس

Lda : متد خطی با نظارت کاهش بعد با تمرکز بر تفکیک پذیری کلاس ها

Tsne : متد غیرخطی بدون نظارت کاهش بعد با تمرکز بر حفظ ساختار محلی

با دستور `pca.explained_variance_ratio_` واریانس توضیح داده شده در PCA را محاسبه میکنیم:

```
[0.92461872 0.05306648]
```

این بدین معنا است که اولین مؤلفه اصلی (PC1) حدوداً 92.46٪ از واریانس را از داده‌ها توضیح می‌دهد ،دومین مؤلفه اصلی (PC2) حدوداً 5.31٪ از واریانس را از داده‌ها توضیح می‌دهد و به طور کلی 97.77٪ از کل واریانس را توضیح می‌دهند .

اگر دو مولفه اول میزان قابل توجهی از واریانس را توضیح دهند معمولاً بالای 80 درصد، کاهش ابعاد داده به دو بعد با استفاده از PCA موجه است.

همچنین برای LDA که بر روی تفکیک پذیری عمل می کند باید ببینیم که داده های داده شده تفکیک پذیری خوبی دارند یا نه که از روی شکل میبینیم که LDA نیز موجه است.

برای tsne که ساختار محلی داده‌ها را نشان می‌دهد و فاصله‌های بین نقاط را در یک فضای کم‌ابعاد حفظ می‌کند. این تجسم تایید بصری از خوشه‌ها در داده‌ها ارائه می‌دهد که برای تحلیل اکتشافی داده‌ها مفید است.

ب.

ب. با استفاده از الگوریتم SVM، با هستهٔ خطی، داده‌ها را طبقه‌بندی کنید و ماتریس درهم‌ریختگی آن را بدست آورید و مرزهای تصمیم‌گیری را در فضای دوبعدی (کاهش بُعد از طریق یکی از روش‌های آموخته‌شده با ذکر دلیل) ترسیم کنید.

حال داده ها را با standardscaler استاندارد سازی میکنیم.دلیل استفاده ازstandardscaler بجای minmax این است که svm به پراکندگی داده ها حساسیت بالایی داردو همچنین pca نیز به واریانس داده ها حساس است.

حال داده های scaled شده را به test و train با درصد تقسیم 80% تقسیم می کنیم و svm را روی داده های train آموزش می دهیم.

دقت آموزش برابر 96.9% است.

```
0.9666666666666667
```

همچنین ماتریس پراکندگی برای تست برابر:

```
  Confusion Matrix of test data:
  [[10  0  0]
   [ 0  3  1]
   [ 0  0 16]]
```

که میبینیم فقط 1 داده کلاس 2 به اشتباه در کلاس 3 طبقه بندی شده .

و ماتریس پراکندگی برای تمام داده ها:

```
Confusion Matrix of all data:
[[50  0  0]
 [ 0 45  5]
 [ 0  1 49]]
```

که 6 داده misclassified داریم .

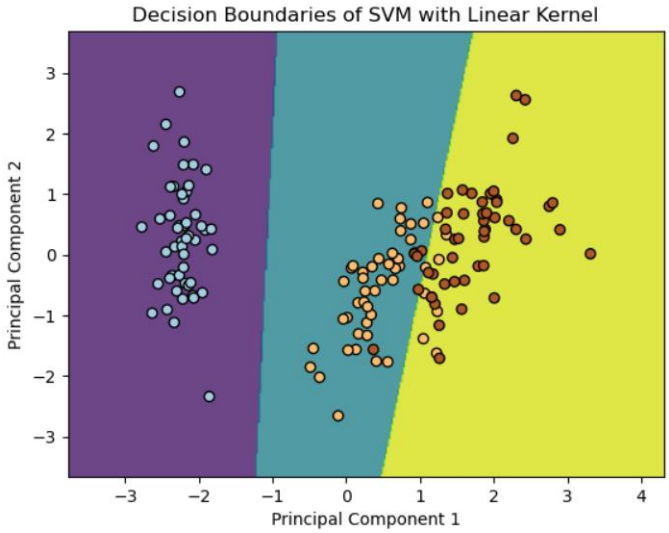
همچنین ضرایب وزن و بایاس support vector به شکل زیر است:

```
#print(svm.support_vectors_)
print('weights : ',svm.coef_[0] , 'bias : ',svm.intercept_[0])
```

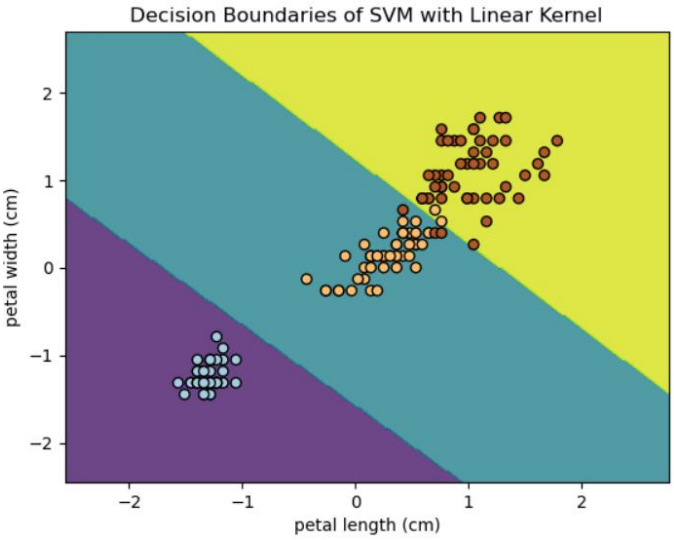
```
weights :  [-0.46020088  0.33751306 -0.8639836  -0.93632798] bias :  -1.4759502026067934
```

حال چون 4 ویژگی داریم و فضا 4 بعدی میشود(ضرایب بالا) ، برای تصویر کردن مرز های تصمیم نیاز به کاهش بعد داریم که یک بار با pca و یکبار صرفا تصویر برای فقط 2 ویژگی میکشیم.

: PCA



برای ویژگی های Petal length و Petal width داریم:



ج.

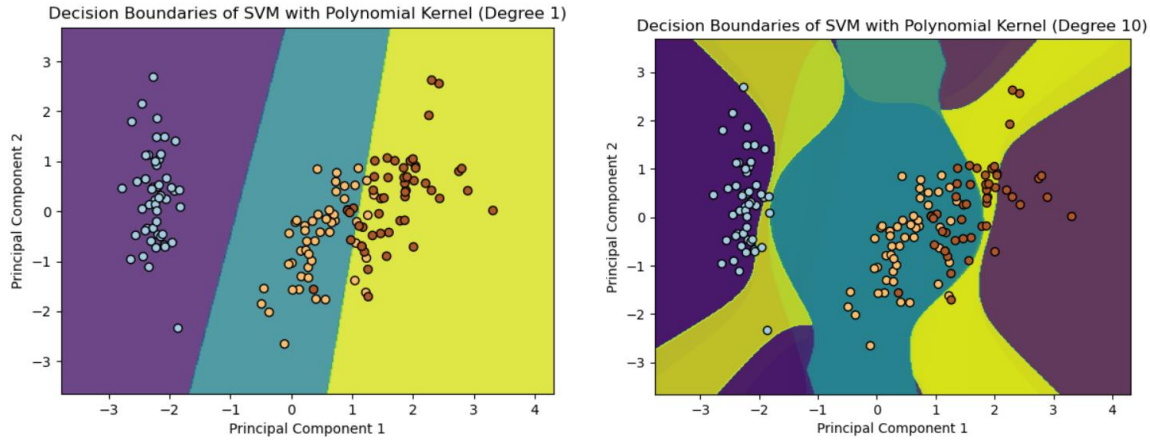
ج. بخش قبلی را با استفاده از هسته‌های چند جمله‌ای و با استفاده از کتابخانهٔ scikit-learn از درجه یک تا ۱۰ پیاده سازی کنید و نتایج را با معیارهای مناسب گزارش کرده و مقایسه و تحلیل کنید. در نهایت، با استفاده از کتابخانهٔ imageio جداسازی ویژگی‌های اصلی را (کاهش بُعد از طریق یکی از روش‌های آموخته‌شده با ذکر دلیل) برای درجات ۱ تا ۱۰ در قالب یک GIF به تصویر بکشید و لینک دسترسی مستقیم به فایل GIF را درون گزارش خود قرار دهید.

بخش قبلی را با استفاده از هسته های polynomial و با استفاده از کتابخانهٔ از درجه 1 تا ۱۰پیاده سازی می کنیم و برای نتایج از معیار accuracy استفاده کرده و مقایسه می کنیم.

```
Best degree found by GridSearchCV: 1
Accuracy with best degree: 0.9667
Degree 1: Accuracy = 0.9667
Degree 2: Accuracy = 0.8667
Degree 3: Accuracy = 0.9667
Degree 4: Accuracy = 0.8333
Degree 5: Accuracy = 0.9333
Degree 6: Accuracy = 0.8667
Degree 7: Accuracy = 0.8667
Degree 8: Accuracy = 0.8000
Degree 9: Accuracy = 0.8333
Degree 10: Accuracy = 0.8000
```

می بینیم که بهترین درصد دقت برای مرتبه 1 است.همچنین با دستور GridSearchCV ، cross validation انجام دادیم .

لینک دسترسی به gif خواسته شده .



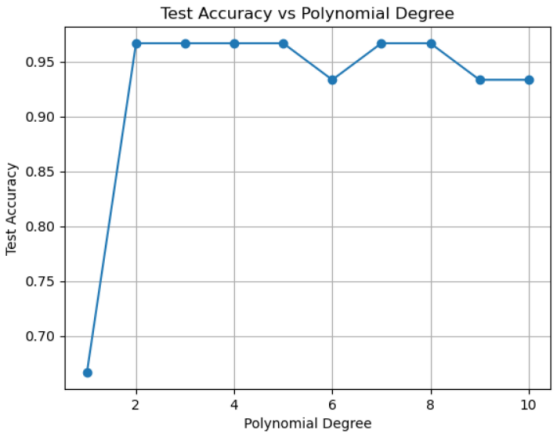
دربالا مرز های تصمیم را برای کرنل چند جمله ای با دو درجه 1 و 10 مشاهده می کنیم.

د.

د. حال الگوریتم SVM را برای مورد قبلی، بدون استفاده از کتابخانهٔ scikit-learn و به‌صورت From Scratch پیاده‌سازی کنید. در این بخش لازم است که یک کلاس SVM تعریف کنید. این کلاس می‌بایست حداقل دارای سه تابع (متد) Polynomial_kernel، Fit و Predict باشد. متد Polynomial_kernel می‌بایست با دریافت درجه‌های ۱ تا ۱۰، هسته‌های چندجمله‌ای را محاسبه کند. دقت الگوریتم را با افزایش درجه گزارش کنید و نتایج حاصل را با بخش قبلی مقایسه کنید. در این قسمت نیز جداسازی ویژگی‌های اصلی را برای درجات ۱ تا ۱۰ در قالب یک GIF به تصویر بکشید پیوند دسترسی مستقیم آن را در گزارش خود قرار دهید.

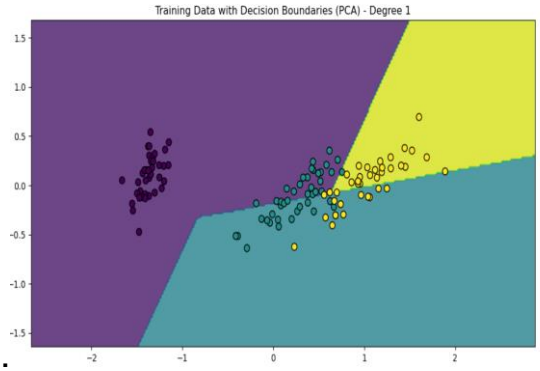
نتایج:

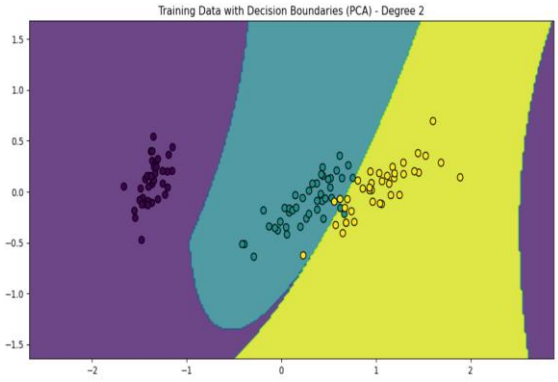
```
GIF saved as 'Q1D_polynomial_kernel_degrees.gif'
Accuracy for degrees 1 to 10:
Degree 1: 0.67
Degree 2: 0.97
Degree 3: 0.97
Degree 4: 0.97
Degree 5: 0.97
Degree 6: 0.93
Degree 7: 0.97
Degree 8: 0.97
Degree 9: 0.93
Degree 10: 0.93
```



[لینک GIF]

همچنین مرز کرنل برای درجه 1 و 2:





تفاوت بین نتایج پیاده‌سازی دستی و کتابخانه می تواند به تنظیم پارامتر ها و روش های بهینه ساز و..... ربط داشته باشد. در قسمت چپ دقت با روش دستی و در راست با کتابخانه آمده است:

```
Accuracy for degrees 1 to 10:
Degree 1: 0.67
Degree 2: 0.97
Degree 3: 0.97
Degree 4: 0.97
Degree 5: 0.97
Degree 6: 0.93
Degree 7: 0.97
Degree 8: 0.97
Degree 9: 0.93
Degree 10: 0.93
```

```
Degree 1: Accuracy = 0.9667
Degree 2: Accuracy = 0.8667
Degree 3: Accuracy = 0.9667
Degree 4: Accuracy = 0.8333
Degree 5: Accuracy = 0.9333
Degree 6: Accuracy = 0.8667
Degree 7: Accuracy = 0.8667
Degree 8: Accuracy = 0.8000
Degree 9: Accuracy = 0.8333
Degree 10: Accuracy = 0.8000
```

سوال 2

سوال 1 کامل حل شده بجای سوال 2 این قسمت صرفا برای این که حیف بود گذاشتم و نیازی به تصحیح ندارد.

تنظیم مسئله و مشکل با روش‌های موجود

مقاله با بحث در مورد چالش‌های کلی در گسترش مدل‌های ماشین بردار پشتیبان (SVM) دوکلاسه به مسائل دسته‌بندی چندکلاسه آغاز می‌شود. روش‌های سنتی یا مسئله چندکلاسه را به چند مسئله دسته‌بندی دوکلاسه تقسیم می‌کنند (مانند یکی در برابر یکی یا یکی در برابر همه) یا یک مسئله بهینه‌سازی پیچیده‌تری را که در آن تمام کلاس‌ها به طور همزمان در نظر گرفته می‌شوند، حل می‌کنند.

این روش‌های مرسوم معمولاً ناکارآمد هستند زیرا یا شامل حل تعداد زیادی از SVMهای دوکلاسه می‌شوند که باعث افزایش پیچیدگی محاسباتی می‌شود یا با یک مسئله بهینه‌سازی مقیاس بزرگ سروکار دارند که اغلب از نظر محاسباتی سنگین و پیچیده است.

معرفی GenSVM

GenSVM با ارائه راه حلی برای این چالش‌ها، SVM چندکلاسه جامعی را پیشنهاد می‌کند که مرزهای دسته‌بندی را در فضایی با بعد $K-1$ ایجاد می‌کند، جایی که K تعداد کلاس‌ها است. این کار با استفاده از استراتژی رمزگذاری سیمپلکس جدید انجام می‌شود.

مدل چندین وزن‌دهی از خطاهای طبقه‌بندی نادرست را در تابع زیان خود گنجانده است، که این امکان را می‌دهد تا از طریق یک مسئله بهینه‌سازی تکی بر روی انواع مختلف SVMهای چندکلاسه تعمیم یابد.

GenSVM از الگوریتم بزرگ‌نمایی تکراری برای حل بهینه‌سازی به طور مؤثر استفاده می‌کند بدون اینکه نیاز به فرمول‌بندی دوگانه باشد، که باعث افزایش کارایی محاسباتی می‌شود، به ویژه برای مجموعه‌های داده بزرگ.

نوآوری‌ها و مشارکت‌های نظری

رمزگذاری سیمپلکس: نوآوری اصلی GenSVM در استفاده از رمزگذاری سیمپلکس نهفته است، که با کاهش ابعاد مسئله، نمایش مرزهای دسته‌بندی را ساده می‌کند. این نمایش نه تنها در کارایی محاسباتی کمک می‌کند بلکه مناطق ابهام رایج در سایر روش‌های ابتکاری مانند یکی در مقابل یکی و یکی در مقابل همه را حذف می‌کند.

الگوریتم بزرگ‌نمایی تکراری: جنبه نوآورانه دیگر GenSVM استفاده از الگوریتم بزرگ‌نمایی تکراری است که روشی منظم برای تقریب حل به طور مؤثر فراهم می‌کند. این رویکرد امکان استفاده از شروع‌های گرم را در طول اعتبارسنجی متقابل و جستجوی شبکه‌ای به طور قابل توجهی سرعت بخشیده و فرآیند آموزش مدل را تسریع می‌بخشد.

انعطاف‌پذیری در جریمه خطاهای طبقه‌بندی نادرست: انعطاف‌پذیری در جریمه‌های خطاهای طبقه‌بندی از طریق وزن‌دهی‌ها و نُرم‌های مختلف در تابع زیان امکان می‌دهد GenSVM به طور مؤثرتری به ویژگی‌های مختلف مجموعه داده‌ها سازگار شود.

ارزیابی تجربی

مقاله به طور مفصل عملکرد GenSVM را با هفت روش دیگر SVM چندکلاسه موجود در مجموعه‌های داده مختلف مقایسه می‌کند. نشان داده می‌شود که GenSVM در دقت پیش‌بینی و زمان آموزش رقابتی است و در بسیاری از معیارها عملکرد بهتری دارد.

این مقایسه‌ها مزایای عملی GenSVM را در کاربردهای واقعی، جایی که هم سرعت و هم دقت حیاتی هستند، برجسته می‌کنند.

کدگذاری سیمپلکس Simplex encoding :

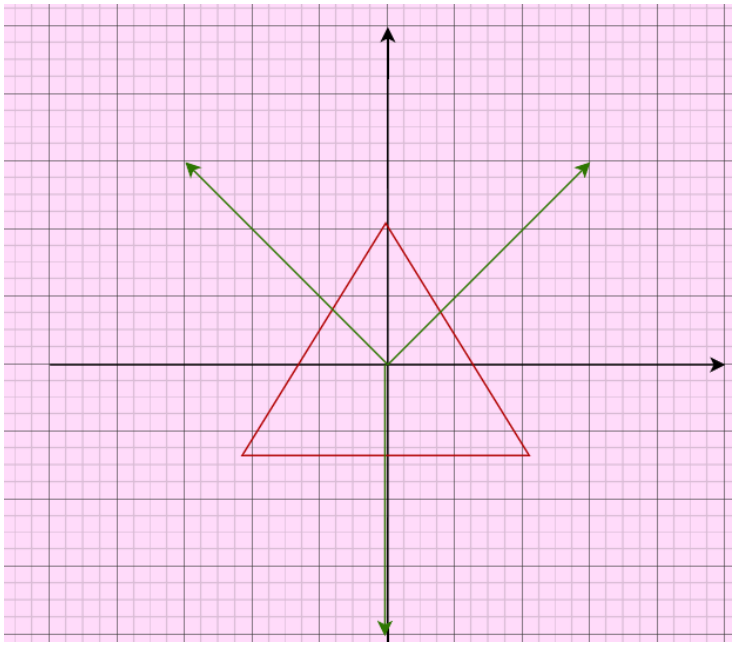
1. کاهش ابعاد:

- مسئله چندکلاسه به یک فضای (K-1) بعدی تبدیل می‌شود، جایی که (K) تعداد کلاس‌ها است. این رویکرد ابعاد مسئله را کاهش می‌دهد و فرآیند بهینه‌سازی را ساده می‌کند.

2. نمایش سیمپلکس:

- در فضای (K-1) بعدی، نقاط داده بر روی رئوس یک سیمپلکس (K) منظم نگاشت می‌شوند. هر راس سیمپلکس نمایانگر یک کلاس است.

- مرزهای تصمیم‌گیری بین کلاس‌ها به عنوان نیمسازهای عمود بر وجوه این سیمپلکس تعریف می‌شوند.



3. محاسبه خطای طبقه‌بندی:

- خطای طبقه‌بندی برای یک نقطه داده با تصویربرداری آن بر روی مرزهای تصمیم‌گیری در فضای سیمپلکس تعیین می‌شود.

- خطا متناسب با فاصله نقطه تا مرزهای تصمیم‌گیری است، که تفسیر هندسی واضحی از وظیفه طبقه‌بندی را فراهم می‌کند.

5. **بهینه‌سازی**:

- نگاشت از فضای ورودی به فضای سیمپلکس به منظور به حداقل رساندن خطاهای طبقه‌بندی بهینه می‌شود.

- GenSVM از یک تابع خطا استفاده می‌کند که در پارامترهای مورد نظر محدب است و مسئله بهینه‌سازی به صورت اولیه حل می‌شود. این امر امکان آموزش کارآمد حتی با داده‌های بزرگ را فراهم می‌کند.

6. **مثال تصویری**:

- مثالی با سه کلاس و دو ویژگی برای توضیح فرآیند ارائه شده است. پس از اعمال یک تبدیل هسته RBF، داده‌ها به فضای سیمپلکس نگاشت می‌شوند و راه‌حل بهینه با به حداقل رساندن خطاهای طبقه‌بندی تعیین می‌شود.

به طور خلاصه، کدگذاری سیمپلکس در GenSVM یک روش هندسی شهودی و محاسباتی کارآمد برای طبقه‌بندی چندکلاسه فراهم می‌کند که با نگاشت نقاط داده به فضای سیمپلکس با ابعاد پایین‌تر و تعریف مرزهای تصمیم‌گیری واضح بین کلاس‌ها انجام می‌شود.

$$\begin{cases} \boldsymbol{X_i} \in \mathbb{R}^m : \quad object\ vector\ correspondin\ to\ m\ attributes \\ \boldsymbol{y_i} \in \{1, \dots, K\}\ for\ i \in \{1, \dots, n\}: \quad class\ lables\ of\ objects \\ \boldsymbol{W} \in \mathbb{R}^{m \times (K-1)} : \quad weight\ matrix \\ \boldsymbol{t} \in \mathbb{R}^{K-1} : \quad translation\ vector \end{cases}$$

که در نهایت object یا شی i در فضای سیمپلکس k-1 بعدی برابر میشود با(ورژن خطی):

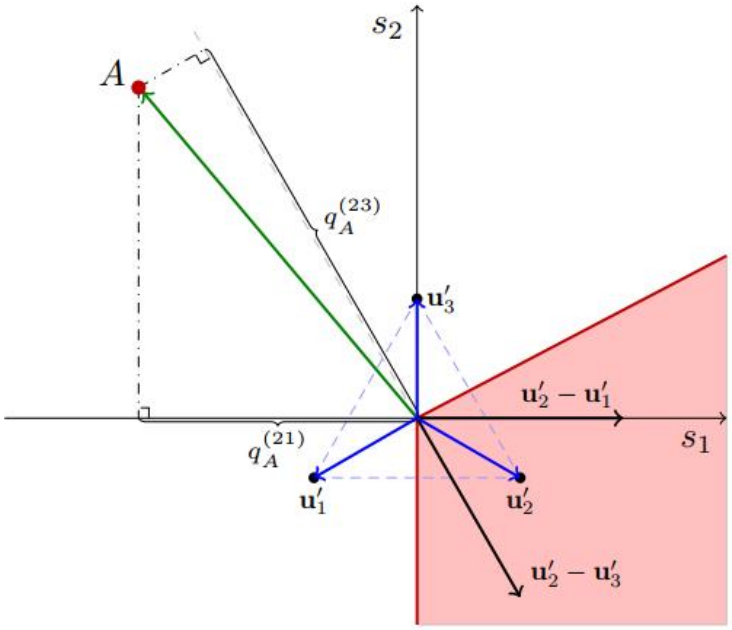
$$\boldsymbol{s}_i' = \boldsymbol{X}_i' \boldsymbol{W} + \boldsymbol{t}'$$

برای بدست آوردن خطای misclassified ها $\boldsymbol{s}_i'$ مربوط به داده روی مرز های تصمیم گیری تصویر می شوند . برای اینکه خطا ها با فاصلیشان از مرز تصمیم گیری متناسب باشند ، یک k-simplex معمولی در $\mathbb{R}^{K-1}$ با فاصله 1 بین هر جفت راس (مثلث متساوی الاضلاع با ضلع 1) استفاده شده است .

$\boldsymbol{U}_k$ را ماتریس $K \times (K-1)$ موقعیت سیمپلکس میگیریم که ردیف های آن u_k' هست که موقعیت هر راس k را در خود دارد $k \in \{1, \dots, K\}$ و $l \in \{1, \dots, K-1\}$.

المان های $\boldsymbol{U}_k$ از این قبیل اند :

$$u_{kl} = \begin{cases} -\frac{1}{\sqrt{2(l^2+l)}} & \text{if } k \leq l \\ \frac{l}{\sqrt{2(l^2+l)}} & \text{if } k = l+1 \\ 0 & \text{if } k > l+1. \end{cases}$$



اسکالر $q_i^{(kj)}$ را تعریف کرده شده تا اندازه فاصله تصویر داده یا شی i ام تا مرز بین کلاس k ام و j ام را محاسبه کند .

$$q_i^{(kj)} = (\mathbf{x}'_i\mathbf{W} + \mathbf{t}')(\mathbf{u}_k - \mathbf{u}_j).$$

و تابع هزینه Huber hinge به خاطر انعطاف پذیری در استفاده از دیتاست های مختلف و خاص استفاده شده :

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

این خطا به تمام اندیس های $q_i^{(y_i j)}$ بجز $j \neq y_i$ اعمال می شود .

حال برای هر داده خطا با احتساب بقیه کلاس ها با استفاده از نرم l_p محاسبه می شود.

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

که در خود نوعی regularization برای وزن های هوبر شامل می شود.

وزن ها به گونه زیر محاسبه می شوند.

$$\rho_i = \frac{n}{n_k K}, \quad i \in G_k,$$

که $G_k = \{i: y_i = k\}$ داده های متعلق به کلاس k و $n_k = |G_k|$ و تابع هزینه یا تابع اتلاف کلی به شکل زیر در می آید :

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

که $\lambda tr\boldsymbol{W}'\boldsymbol{W}$ ترم penalty برای overfit نشدن است. و $\lambda > 0$ پارامتر Regularization است.لازم به ذکر است که این تابع اتلاف اثبات convex بودن دارد.

تخمین (predict) لیبل کلاس ها به گونه زیر می باشد :

(W*,t*) پارامتر هایی که تابع اتلاف را کمینه می کنند هستند ، x'_{n+1} را اول با استفاده از فرمول $s'_i = X'_i W^* + t'^*$ در فضای سیمپلکس مپ میکنیم.سپسبا توجه به اینکه نزدیک ترین راس کجا هست(نرم اقلیدسی مربعی) لیبل می زنیم.

$$\hat{y}_{n+1} = \arg\min_k \|\mathbf{s}'_{n+1} - \mathbf{u}'_k\|^2, \quad \text{for } k = 1, \ldots, K.$$

همچنین برای کمینه کردن تابع اتلاف از الگوریتم IM استفاده شده.

Algorithm 1: GenSVM Algorithm

Input: $\mathbf{X}, \mathbf{y}, \boldsymbol{\rho}, p, \kappa, \lambda, \epsilon$
Output: $\mathbf{V}$

$K \leftarrow \max(\mathbf{y})$
$t \leftarrow 1$
$\mathbf{Z} \leftarrow [\mathbf{1}\ \mathbf{X}]$
Let $\overline{\mathbf{V}} \leftarrow \mathbf{V}_0$
Generate $\mathbf{J}$ and $\mathbf{U}_K$
$L_t = L_{\text{MSVM}}(\overline{\mathbf{V}})$
$L_{t-1} = (1 + 2\epsilon)L_t$
while $(L_{t-1} - L_t)/L_t > \epsilon$ **do**
  for $i \leftarrow 1$ **to** n **do**
    Compute $\overline{q}_i^{(y_i j)} = \mathbf{z}_i' \overline{\mathbf{V}} \boldsymbol{\delta}_{y_i j}$ for all $j \neq y_i$
    Compute $h\left(\overline{q}_i^{(y_i j)}\right)$ for all $j \neq y_i$ by (3)
    if $\varepsilon_i = 1$ **then**
      Compute $a_{ijy_i}^{(1)}$ and $b_{ijy_i}^{(1)}$ for all $j \neq y_i$ according to Table 4 in Appendix C
    else
      Compute ω_i following (10)
      Compute $a_{ijy_i}^{(p)}$ and $b_{ijy_i}^{(p)}$ for all $j \neq y_i$ according to Table 4 in Appendix C
    end
    Compute α_i by (12)
    Compute $\boldsymbol{\beta}_i$ by (13)
  end
  Construct $\mathbf{A}$ from α_i
  Construct $\mathbf{B}$ from $\boldsymbol{\beta}_i$
  Find $\mathbf{V}^+$ that solves (14)
  $\overline{\mathbf{V}} \leftarrow \mathbf{V}$
  $\mathbf{V} \leftarrow \mathbf{V}^+$
  $L_{t-1} \leftarrow L_t$
  $L_t \leftarrow L_{\text{MSVM}}(\mathbf{V})$
  $t \leftarrow t + 1$
end

سوال 3

۳ پرسش سه

مقالهٔ Credit Card Fraud Detection Using Autoencoder Neural Network برای پیاده‌سازی این قسمت در نظر گرفته شده است. پس از مطالعهٔ مقاله به سوالات زیر پاسخ دهید.

آ.

آ. بزرگ‌ترین چالش‌ها در توسعهٔ مدل‌های تشخیص تقلب چیست؟ این مقاله برای حل این چالش‌ها از چه روش‌هایی استفاده کرده است؟

یکی از چالش‌های بزرگ در تشخیص تقلب، نامتوازن بودن داده‌ها است. در داده‌های مربوط به تراکنش‌های کارت اعتباری، تعداد تراکنش‌های تقلبی بسیار کمتر از تراکنش‌های عادی است که این موضوع می‌تواند منجر به کاهش دقت مدل‌های تشخیص تقلب شود.

از دیگر چالش های ذکر شده در این مقاله این است که با استفاده از الگوریتم‌های oversampling برای افزایش تعداد نمونه‌های کلاس اقلیت می‌تواند نویز ایجاد کند و در نتیجه دقت مدل را کاهش دهد.

روش استفاده شده در مقاله برای حل این چالش‌ها:

1. بیش‌نمونه‌برداری با استفاده از تکنیک SMOTE: این تکنیک برای ایجاد نمونه‌های مصنوعی از کلاس اقلیت استفاده می‌شود تا تعادل بین کلاس‌ها برقرار شود.

2. استفاده از denoising autoencoder (DAE): برای حذف نویز و بهبود کیفیت داده‌های بیش‌نمونه‌برداری شده، از این نوع شبکه عصبی استفاده می‌شود. این شبکه می‌تواند داده‌های آلوده به نویز را تمیز کند و ویژگی‌های مفید را استخراج کند.

3. استفاده از deep fully connected NN: این مدل برای انجام طبقه‌بندی نهایی و تشخیص تراکنش‌های تقلبی استفاده می‌شود. مدل با استفاده از لایه‌های متصل عمیق و تابع هزینه

softmax cross entropy دقت بالایی در تشخیص تقلب دارد.

4. ارزیابی با استفاده از accuracy و recall rate: مقاله از معیارهایی مانند دقت و نرخ بازگشت برای ارزیابی عملکرد مدل استفاده کرده است تا مطمئن شود که مدل توانایی تشخیص تراکنش‌های تقلبی را دارد.

در مجموع، این مقاله با استفاده از ترکیب SMOTE و (DAE) به بهبود دقت تشخیص تقلب در مجموعه داده‌های نامتوازن پرداخته است.

ب.

ب. در مورد معماری شبکهٔ ارائه‌شده در مقاله به‌صورت مختصر توضیح دهید.

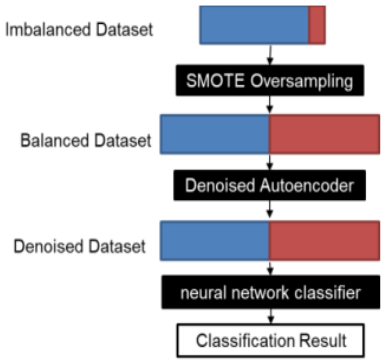


Fig. 5 Flowchart of the porcess

معماری شبکه

1. خودرمزگذار حذف نویز (Denoising Autoencoder - DAE)

هدف: حذف نویز از داده‌ها و استخراج ویژگی‌های مهم.

ساختار

ورودی: داده‌های با نویز (29 ویژگی).

لایه‌های مخفی: 3 لایه تمام متصل با(22,15,10,15,22) نورون

خروجی: بازسازی داده‌های ورودی با 29 ویژگی.

تابع هزینه: مربعات خطا (Squared Loss Function).

Table 2. Model design for denoised autoencoder

Dataset with noise (29)
Fully-Connected-Layer (22)
Fully-Connected-Layer (15)
Fully-Connected-Layer (10)
Fully-Connected-Layer (15)
Fully-Connected-Layer (22)
Fully-Connected-Layer (29)
Square Loss Function

2. شبکه عصبی عمیق برای طبقه‌بندی

هدف: طبقه‌بندی نهایی و تشخیص تراکنش‌های تقلبی.

ساختار

ورودی: داده‌های پاکسازی شده توسط DAE (29 ویژگی).

لایه‌های مخفی: لایه های تمام‌متصل با (2,5,10,15) نورون.

خروجی: دو کلاس (عادی و تقلبی).

تابع هزینه: (SoftMax Cross Entropy Loss Function).

Table 3. Model design for classifier

Denoised Dataset (29)
Fully-Connected-Layer (22)
Fully-Connected-Layer (15)
Fully-Connected-Layer (10)
Fully-Connected-Layer (5)
Fully-Connected-Layer (2)
SoftMax Cross Entropy Loss Function

روند کلی فرآیند

1. Preprocessing: حذف ویژگی "time" و نرمال‌سازی ویژگی "amount". دیگر ویژگی‌ها از طریق PCA به دست آمده‌اند و نیازی به نرمال‌سازی ندارند20 % را برای تست کنار می گذاریم.

2. oversampling: ایجاد تعادل بین کلاس‌های عادی و تقلبی با استفاده از تکنیک SMOTE.

3. Denoising Autoencoder: آموزش DAE 7لایه (مقاله) برای حذف نویز از داده‌های oversample شده و تولید داده‌های تمیز.

4. Deep NN Classifier: استفاده از داده‌های تمیز شده برای آموزش شبکه عصبی عمیق و انجام طبقه‌بندی نهایی.

این معماری ترکیبی از تکنیک‌های بیش‌نمونه‌برداری و حذف نویز با شبکه‌های عصبی عمیق است که دقت و قابلیت تشخیص تراکنش‌های تقلبی را بهبود می‌بخشد

ج.

ج. مدل ارائه‌شده را پیاده‌سازی کرده و با استفاده از این دیتاست آموزش دهید. برای جلوگیری از بیش‌برازش، آموزش مدل را طوری تنظیم کنید که در انتهای آموزش، بهترین وزن‌های مدل بر اساس خطای قسمت اعتبارسنجی بازگردانده شود.

```
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
...
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
```

در دیتاست داده null وجود ندارد ، 31 ستون داریم که بنا بر گفته های paper، Time را حذف کرده وAmount را نرمالایز می کنیم.

همچنین class را که کلاس target هست جدا می کنیم.

```
raw dataset shape: (284807, 31)
X: (284807, 29)
y:(284807,)
Class
0    284315
1       492
Name: count, dtype: int64
```

تعداد داده ها در بالا آمده و همینطور تعداد داده های نرمال و فالت را هم مشاهده می کنیم که میبینیم تعداد داده های فالت که با 1 نشان داده ایم بسیار کمتر از داده های نرمال است و دیتاست imbalance است.

برای رفع آن از دستور smote که روش oversampling است استفاده میکنیم.

تعداد داده های resample شده:

```
X_res.shape, y_res.shape
```

```
((568630, 29), (568630,),
```

سپس داده ها را به آموزش و تست با ضریب تقسیم 0.8 تقسیم می کنیم.

```
X_train.shape , X_test.shape
```

```
(454904, 29), (113726, 29))
```

اتو انکودری با ساختار و پارامتر های زیر تعریف میکنیم:

```
autoencoder = Sequential([
    Input(shape=(input_dim,),name='input'),
    Dense(22, activation='tanh'),
    Dense(15, activation='tanh'),
    Dense(10, activation='leaky_relu', name="encode"),
    Dense(15, activation='tanh', ),
    Dense(22, activation='leaky_relu',),
    Dense(input_dim, activation='linear')
])
autoencoder.compile(optimizer='Nadam', loss='mean_squared_error')
```

سپس داده ها را نویزی می کنیم :

```
noise_factor = 0.1
X_train_noisy = X_train + noise_factor * np.random.normal(loc=0.0, scale=0.5, size=X_train.shape)
X_train_noisy = np.clip(X_train_noisy, 0., 1.)
```

برای اتو انکودر نویز زدا باید داده های نویزی به داده های بدون نویز برسند. همچنین با دستور earlystoping آموزش مدل را طوری تنظیم می کنیم که در انتهای آموزش، بهترین وزن های مدل بر اساس خطای قسمت اعتبارسنجی بازگردانده شود:

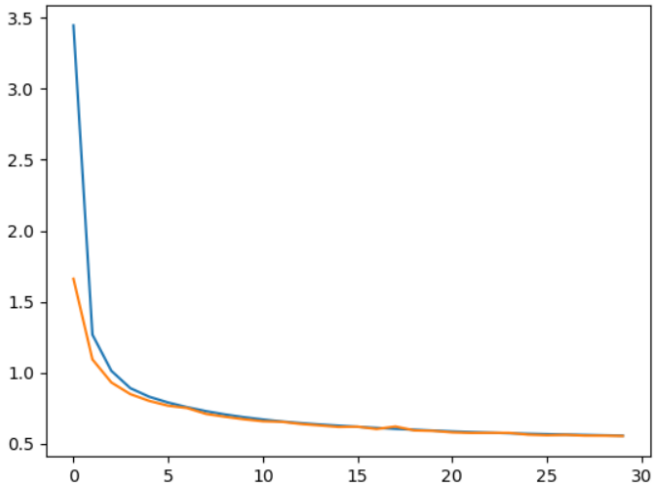
```
# Train the autoencoder
dae_history = autoencoder.fit(X_train_noisy, X_train, epochs=30, batch_size=200, shuffle=True, validation_split=0.2,
                              callbacks=[EarlyStopping(monitor='val_loss', patience=15, restore_best_weights=True)])
```

و برای جلوگیری از بیش برازش، آموزش مدل را طوری تنظیم می کنیم که در انتهای آموزش، بهترین وزن های مدل بر اساس خطای قسمت اعتبارسنجی بازگردانده شود.

```
Define the ModelCheckpoint and EarlyStopping callbacks
heckpoint = ModelCheckpoint('best_model.keras', monitor='val_loss', save_best_only=True, mode='min')
arly_stopping = EarlyStopping(monitor='val_loss', patience=5, mode='min', verbose=1, restore_best_weights=True,min_delta=0.005)
```

```
1820/1820 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.5651 - val_loss: 0.5605
Epoch 28/30
1820/1820 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.5575 - val_loss: 0.5551
Epoch 29/30
1820/1820 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.5582 - val_loss: 0.5545
Epoch 30/30
1820/1820 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 0.5560 - val_loss: 0.5524
```

نمودار loss آموزش و validation را می کشیم:



سپس encoder را به داده ها اعمال کرده تا کاهش بعد صورت گیردو خروجی را به یک deep mlp classifier میدهیم:

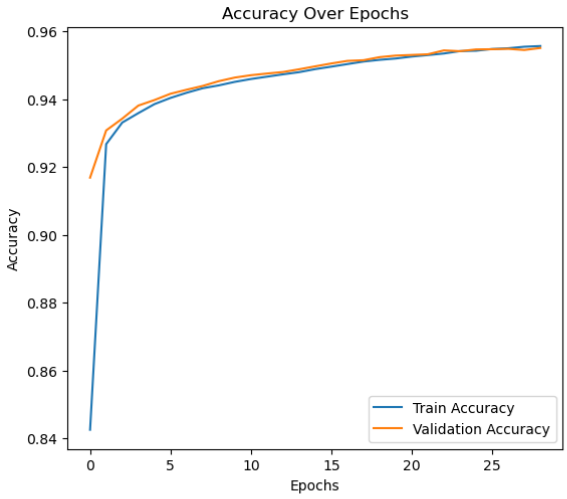
ساختار طبقه بند ما اینگونه است:

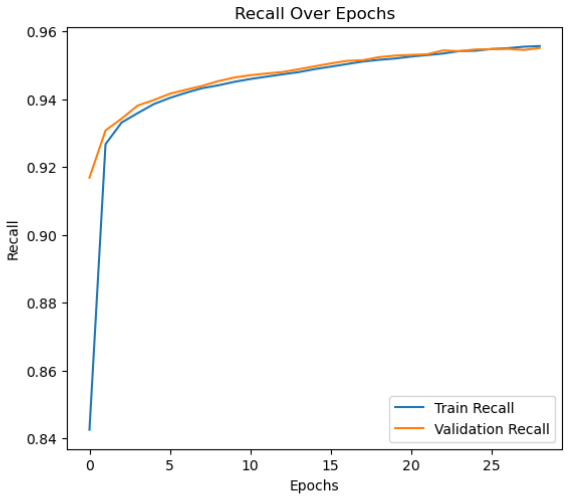
```
classifier = Sequential([
    InputLayer(shape=(10,)),
    Dense(22, activation='tanh'),
    Dense(15, activation='tanh'),
    Dense(10, activation='relu'),
    Dense(5, activation='relu'),
    Dense(2, activation='softmax')
])
classifier.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy',Recall()])
```

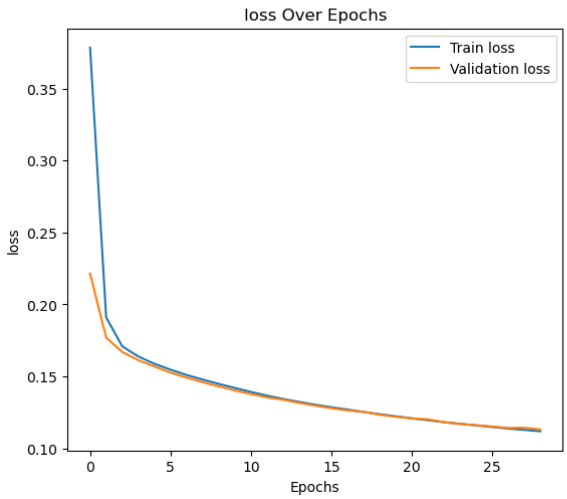
و برای جلوگیری از بیش برازش، آموزش مدل را طوری تنظیم کنید که در انتهای آموزش، بهترین وزن های مدل بر اساس خطای قسمت اعتبارسنجی بازگردانده شود .

```python
# Define the ModelCheckpoint and EarlyStopping callbacks
checkpoint = ModelCheckpoint('best_model.keras', monitor='val_loss', save_best_only=True, mode='min')
early_stopping = EarlyStopping(monitor='val_loss', patience=5, mode='min', verbose=1, restore_best_weights=True,min_delta=0.005)

# Train the classifier
history = classifier.fit(X_train_encoded, y_train_onehot, epochs=50, batch_size=200, shuffle=True, validation_split=0.2, callbacks=[checkpoint, early_stopping])

```

نمودار accuracy وrecall و lossرا رسم می کنیم







د.

د. ماتریس درهم‌ریختگی را روی قسمت ازمون داده‌ها رسم کنید و مقادیر Accuracy، Precision، Recall و f1score را گزارش کنید. فکر می‌کنید در مسائلی که توزیع برچسب‌ها نامتوازن است، استفاده از معیاری مانند Accuracy به تنهایی عمل‌کرد مدل را به‌درستی نمایش می‌دهد؟ چرا؟ اگر نه، کدام معیار می‌تواند به‌عنوان مکمل استفاده شود؟

```
Accuracy: 0.9533879675711798
Precision: 0.955891987605135
Recall: 0.9505229425643554
F1 Score: 0.9531999046517582
Confusion Matrix:
 [[54441  2491]
 [ 2810 53984]]
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.95     56932
           1       0.96      0.95      0.95     56794

    accuracy                           0.95    113726
   macro avg       0.95      0.95      0.95    113726
weighted avg       0.95      0.95      0.95    113726
```

نتایج در بالا آمده اند که میبینیم تماما بالای 95% میباشند که مقدار خوبی است.

دقت (Accuracy): نسبت نمونه‌های صحیح به کل نمونه‌ها.

بازخوانی (Recall): نسبت نمونه‌های مثبت صحیح به نمونه های مثبت صحیح+نمونه های منفی ناصحیح است. این معیار نشان‌دهنده توانایی مدل در تشخیص نمونه‌های مثبت است.

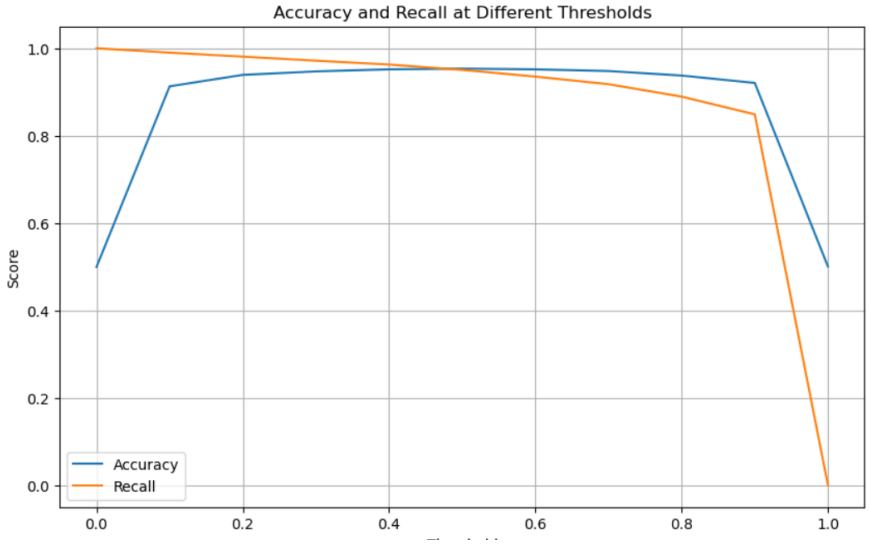
$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negetive}$$

در تشخیص تقلب، نامتوازن بودن داده‌ها یکی از چالش های بزرگ است. در داده‌های مربوط به تراکنش‌های کارت اعتباری، تعداد تراکنش‌های تقلبی بسیار کمتر از تراکنش‌های عادی است که این موضوع می‌تواند منجر به کاهش دقت مدل‌های تشخیص تقلب شود. به همین دلیل برای ما مهم تر است که داده های تقلبی بهتر شناسایی شوند و چون accuracy برای تمام داده ها است معیار مناسبی نیست.

معیار recall که به آن حساسیت یا نرخ تشخیص هم گفته می‌شود، هرچند میتواند معیار مناسبی باشد ، چون معیار آن شامل درست تشخیص دادن تقلب و به غلط نادیده گرفتن تقلب است.

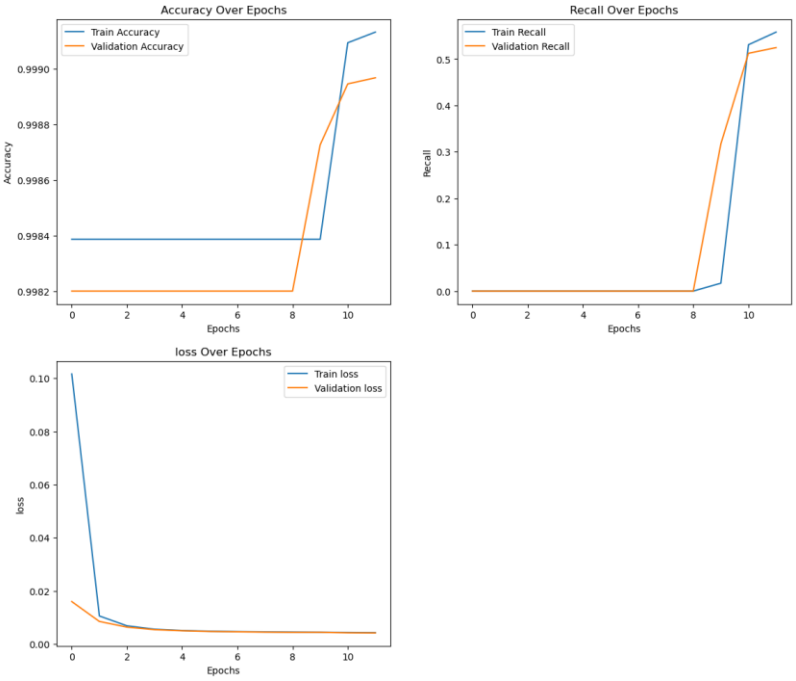
ه.

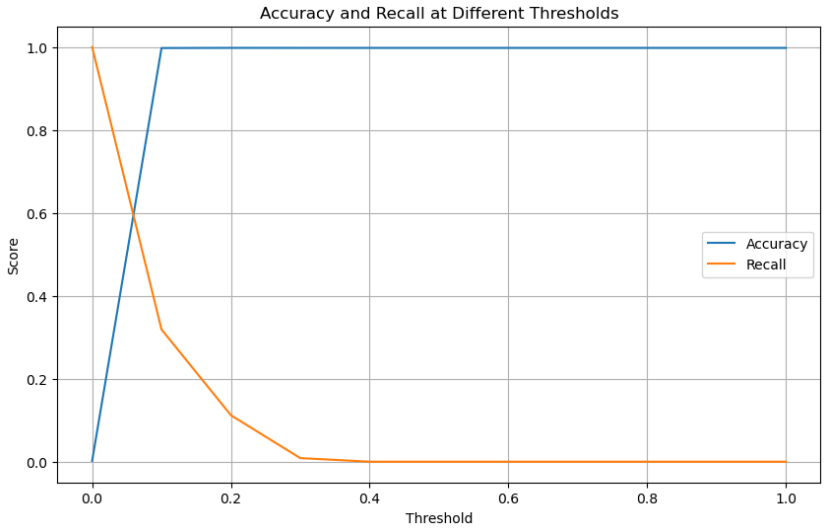
ه. با آستانه‌های مختلف برای Oversampling عمل‌کرد مدل را بررسی کرده و نمودار Recall & Accuracy را مانند شکل ۷ مقاله ترسیم کنید.



و.

و. مدل را با استفاده از داده‌های نامتوازن و بدون حذف نویز، آموزش داده و موارد بخش قبلی را گزارش کنید و نتایج دو مدل را با هم مقایسه کنید.






```
Accuracy: 0.9979635546504687
Precision: 0.0
Recall: 0.0
F1 Score: 0.0
Confusion Matrix:
 [[56846     0]
 [  116     0]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56846
           1       0.00      0.00      0.00       116

    accuracy                           1.00     56962
   macro avg       0.50      0.50      0.50     56962
weighted avg       1.00      1.00      1.00     56962
```

با توجه به حرف های زده شده در قسمت قبل مشاهده می کنیم که دیتای بدون oversampling هرچند accuracy بالا نشان میدهد که به معنی دقت بالا برای تمام داده هاست ولی چون در دیتاست imbalance تعداددادهٔ های اقلیت(تقلب) بسیار پایین است این داده های misclassified شده تاثیر چندانی در accuracy ندارند درحالی که از اهمیت بالاتری برخوردارند.

اما recall آن پایین است که معیار ارزیابی درستی هست(در قسمت قبل توضیح داده شد) .