[redacted] شماره دانشجویی 9622[redacted]

گزارش سری سوم تمرین

کلیات شبکه :

برای این پروژه از یک آتو انکودر که فقط با داده های غیر جعلی آموزش داده شده استفاده شده است

این شبکه وظیفه دارد بازسازی داده های غیر جعلی را با خطای کمی انجام دهد

در اصل ما در آموزش این شبکه خطای بازسازی را در روند آموزشش کاهش میدهیم

سپس از شبکه آموزش داده شده استفاده میکنیم تا اینبار داده های تست را پیشبینی کنیم اینکار را واضحا با خطای نسبتا بالاتری انجام خواهیم داد

با این ترتیب خواهیم فهمید اگر برای بازسازی یک ورودی با خطای بالایی مواجه شدیم این ورودی یک تراکنش جعلی بوده است

از این روند با نام Anomaly Detection نیز یاد میشود

پیش پردازش داده ها :

داده های این پروژه حجم بسیار بالایی برای پردازش های آینده داشت و نیاز بود در اولین مرحله حجم داده ها را با تغییر تایپ آن ها کم کرد تا بتوان در آینده روی آن ها پردازش انجام داد

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 590540 entries, 0 to 590539
Columns: 434 entries, TransactionID to DeviceInfo
dtypes: float64(399), int64(4), object(31)
memory usage: 1.9+ GB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 506691 entries, 0 to 506690
Columns: 433 entries, TransactionID to DeviceInfo
dtypes: float64(399), int64(3), object(31)
memory usage: 1.6+ GB
```

این عکس حجم داده های train و testرا در حالت خامشان نشان میدهد

ابتدا برای هر سیتون مقدار بیشینه را محاسبه میکنیم سپس با استفاده از آن تصمیم میگیرم که چه تایچ جدیدی را انتخاب کنیم

میبینیم که در حالت خام 31 ستون مقدار کیفی دارد که بعدا آنها را کمی میکنیم فعلا فقط بروی تغییر تایپ کمی ها تمرکز میکنیم

در زیر بازه مقدار های ممکن برای هر تایپ را میبینیم

Number of bits	Min. value	Max. value
8 bit	-128	127
16 bit	-32768	32767
32 bit	-2147483648	2147483647

Float8 وجود ندارد*

همه ستون ها بیشینه ای کمتر از 2147483647 داشتند و میتوان همه را در دسته های float16یا float32 جا سازی کرد

```python
def reduceMem(df) :

    print('\n Reducing Memory, Please Wait ... \n')

    for col in df.columns :
        if str(df[col].dtype) != 'object':
            maximum = df[col].max()

            if -32768 <= maximum <= 32767 :
                df[col] = df[col].astype(np.float16)

            elif -2147483648 <= maximum <= 2147483647 :
                df[col] = df[col].astype(np.float32)

            else :
                df[col] = df[col].astype(np.float64)

    return df

    print('\n Reducing Memory Was Done Seccessfuly, Have a Good Day Sir !\n')
```

عکس زیر حجم گرفته شده توسط داده های train و test را بعد از تغییر تایپ ها نشان میدهد

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 506691 entries, 0 to 506690
Columns: 433 entries, TransactionID to DeviceInfo
dtypes: float16(335), float32(67), object(31)
memory usage: 577.0+ MB
None
<class 'pandas.core.frame.DataFrame'>
Int64Index: 590540 entries, 0 to 590539
Columns: 434 entries, TransactionID to DeviceInfo
dtypes: float16(336), float32(67), object(31)
memory usage: 673.6+ MB
None
```

البته در مراحل بعدی کد وقتی که داده ها را در مرحله پایانی آماده فبت کردن میکنیم

یک بار داده ها را نرمال میکنیم که خود این عمل باعث میشود بتوانیم تمام ستون هارا در float16 جا کنیم که باز به کم حجم شدن آن ها کمک میکند

عکس زیر حجم اشغال شده را بعد از نرمال سازی نشان میدهد

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20663 entries, 358521 to 80060
Columns: 393 entries, TransactionID to DeviceType_mobile
dtypes: float16(393)
memory usage: 15.6 MB
<class 'pandas.core.frame.DataFrame'>
Int64Index: 569877 entries, 0 to 192487
Columns: 393 entries, TransactionID to DeviceType_mobile
dtypes: float16(393)
memory usage: 431.5 MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506691 entries, 0 to 506690
Columns: 393 entries, TransactionID to DeviceType_mobile
dtypes: float16(393)
memory usage: 379.8 MB
```

در مرحله بعدی باید مقادیر خالی را پر کنیم برای این کار میشد صرفا تمام ستون هایی که مقادیر خالی به درصد زیادی داشتند را دور ریخت

این کار هم به سرعت اجرا کد کمک میکرد هم میتوانست دیتا های اضافی را از بین ببرد

اما از آنجایی که در این داده ها مقادیر خالی نیز میتوانند معانی داشته باشند تصمیم بر این شد که بجای دور ریختن ستون ها آن ها را با مد هر ستون پر کنیم

اما دو نکته قابل ذکر است

1 – برای اینکار حتما و حتما باید داده های جعلی و غیر جعلی را جدا کنیم چرا که اگر بدون انجام این کار مد هر ستون را محاسبه کنیم در اصل مد داده های غیر جعلی را بر داده های خالی که مربوط به ردیف ها جعلی هستند نیز جایگزین کرده ایم که باعث تسخیص غلط در آینده خواهد شد

2 – تعداد ستون ها حدود 450 ستون بود و بعدا نیز نیاز خواهیم داشت تعدادی از ستون ها را برای کمی کردن گسترش دهیم که خود باعث افزایشی در تعداد ستون ها میشود و روند اجرا را بسیار کند میکند

پس با توجه به این دو نکته تصمیم بر این گرفته شد تا ابتدا ستون هایی که بیشتر از 85 درصد مقادیر خالی در خود دارند را ابتدا حذف و ستون های دیگر را با روند توضیح داده شده پر کنیم

(ایده بهتر این بود که بعد از جدا سازی داده های جعلی و غیر جعلی تعداد مقادیر خالی را به درصد در هر دو داده بدست آوریم اینگونه اگر درصد مقادیر خالی در هر ستون در داده های جعلی بیشتر از همان درصد در داده های غیر جعلی میبود تصمیم گرفته شود که آن ستون حفظ شود

چرا که اگر در داده های جعلی مثلا از ستون x به مقدار 80 درصد مقدار خالی داشتیم و همان ستون در داده های غیر جعلی 50 درصد مقدار خالی داشت میشد نتیجه گرفت که این ستون مقادیر خالی ارزشمندی دارد و باید حفظ شود

اما متاسفانه به دلیل مشکلات دیگر موفق به پیاده سازی این روش نشدم)

در سه عکس زیر سه مرحله از تعداد ستون ها را میبنییم (سمت چپ داده های train و سمت راست داده های test هستند)

1 – حالت اولیه :

```
(590540, 434) (506691, 433)
```

2 – پس از حذف ستون های با بالای 85 درصد مقدار خالی :

```
(506691, 367) (590540, 368)
```

3 – پس از اضافه کردن (کمی کردن) ستون ها بر اساس ستون های کیفی :

```
(590540, 398) (506691, 397)
```

همچنین در این عکس مشاهده میکنیم که تعداد مقادیر خالی در هر جدول چقدر بوده :

```
Amount of Null values in train Data :  115523073

Amount of Null values in test  Data :  90186908
```

قابل ذکر است تعدادی (4 تا) از ستون ها برای کمی شدن مقادیر یکتای زیادی داشتند و بهینه نبود آنها را به روش one hot vector کمی کنیم پس با label encoding کمی شدند این ستون ها را در عکس زیر مشاهده مکنیم

```
Loading...
for col in test.columns :
    if test[col].dtype == object :
        print(col)
```

```
P_emaildomain
R_emaildomain
id_31
DeviceInfo
```

داده های ما حال برای فیت کردن آماده اند نمایی از آن ها را در عکس زیر میبینیم :

	TransactionID	TransactionDT	TransactionAmt	card1	card2	card3	card5	addr1	addr2	dist1	dist2	C1
358521	0.606934	0.559570	0.012520	0.678711	0.144043	0.381592	0.919922	0.206787	0.836914	0.000097	0.0	0.000214
282997	0.479248	0.433838	0.006256	0.833008	0.779785	0.381592	0.919922	0.452393	0.836914	0.000097	0.0	0.002134
284796	0.482178	0.438721	0.006130	0.584473	0.222046	0.381592	0.905273	0.516113	0.836914	0.000097	0.0	0.000214
290902	0.492676	0.450195	0.002451	0.083679	0.441895	0.381592	0.919922	0.236328	0.836914	0.001361	0.0	0.000427
155635	0.263428	0.199219	0.021500	0.552734	0.965820	0.381592	0.919922	0.774902	0.836914	0.000097	0.0	0.000427
...	...	...	...	...	...	...	...	...	...	...	...	...
415021	0.702637	0.660156	0.004688	0.488770	0.441895	0.381592	0.919922	0.236328	0.836914	0.000097	0.0	0.000427
231999	0.392822	0.344482	0.001840	0.784180	0.895996	0.381592	0.919922	0.854492	0.836914	0.000097	0.0	0.000427
199476	0.337891	0.282227	0.083923	0.488770	0.441895	0.381592	0.919922	0.343262	0.836914	0.000097	0.0	0.000640
313705	0.531250	0.492188	0.031113	0.162109	0.022003	0.381592	0.868652	0.059082	0.836914	0.000097	0.0	0.000214
80060	0.135620	0.104248	0.002373	0.496094	0.392090	0.648926	0.277344	0.236328	0.836914	0.000097	0.0	0.001494

20663 rows × 393 columns

آموزش شبکه :

در این مرحله برای ساخت شبکه از سه قسمت استفاده میکنیم :

Encoder :

این قسمت داده ها را به فضای latent space فشرده شده کرده و چکیده ای از آن ها را به laten space میبرد

ساختار این قسمت را در عکس زیر مشاهده میکنیم :

```
Model: "Encoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_encoder (InputLayer)   [(None, 393)]             0
_________________________________________________________________
encoder_layer1 (Dense)       (None, 256)               100864
_________________________________________________________________
encoder_layer2 (BatchNormali (None, 256)               1024
_________________________________________________________________
encoder_layer3 (Dropout)     (None, 256)               0
_________________________________________________________________
encoder_layer4 (Dense)       (None, 128)               32896
_________________________________________________________________
encoder_layer5 (BatchNormali (None, 128)               512
_________________________________________________________________
encoder_layer6 (Dropout)     (None, 128)               0
_________________________________________________________________
encoder_layer7 (Dense)       (None, 64)                8256
_________________________________________________________________
output_encoder (Dense)       (None, 32)                2080
=================================================================
Total params: 145,632
Trainable params: 144,864
Non-trainable params: 768
```

Decoder :

این قسمت فضای latent space و فشرده شده هر داده را گرفته و سعی در بازسازی آن داده از طیق چکیده شده آن داده میکند

ساختار این قسمت در زیر آمده است :

```
Model: "Decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_decoder (InputLayer)   [(None, 32)]              0
_________________________________________________________________
decoder_layer1 (Dense)       (None, 32)                1056
_________________________________________________________________
decoder_layer2 (Dense)       (None, 64)                2112
_________________________________________________________________
decoder_layer3 (BatchNormali (None, 64)                256
_________________________________________________________________
decoder_layer4 (Dropout)     (None, 64)                0
_________________________________________________________________
decoder_layer5 (Dense)       (None, 128)               8320
_________________________________________________________________
decoder_layer6 (BatchNormali (None, 128)               512
_________________________________________________________________
decoder_layer7 (Dropout)     (None, 128)               0
_________________________________________________________________
decoder_layer8 (Dense)       (None, 256)               33024
_________________________________________________________________
output_layer (Dense)         (None, 393)               101001
=================================================================
Total params: 146,281
```

قسمت 3 : ساخت آتوانکودر با استفاده از دیکودر و انکودر :

در این قسمت به سادگی فقط دو قسمت قبل را با هم تلفیق میکنیم تا هر داده ورودی را بتوان با خطایی باز سازی کرد

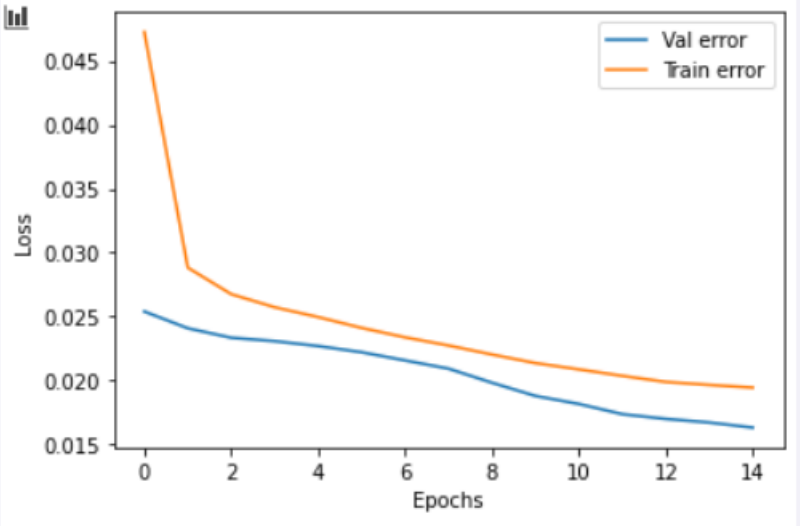
سپس این آتوانکودر را آموزش میدهیم

```
_________________________________________________________________
None
Model: "autoencoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_autoencoder (InputLaye [(None, 393)]             0
_________________________________________________________________
Encoder (Functional)         (None, 32)                145632
_________________________________________________________________
Decoder (Functional)         (None, 393)               146281
=================================================================
Total params: 291,913
Trainable params: 290,761
Non-trainable params: 1,152
_________________________________________________________________
```

در

نتایج بدست آمده از آموزش اتوانکودر را میتوان در این عکس دید

برای loss function از mae استفاده شده*



البته میتوان با تعداد اپیاک های بیشتر نتایج را بهتر کرد اما از انجایی که هدف پروژه کار با آتوانکودر ها بود سعی بر این بود که وقت و انرژی را صرف تحلیل روش های بهتر و مقایسه آن ها بود

در نهایت فقط کاری که باقی میماند این است که داده های test را با این شبکه پیشبینی و با یک ثابت به اسم threshold تصمیم بگیریم چه مقدار خطا در بازسازی به معنی جعلی بودن داده است

این مقدار با آزمون و خطای بسیار بدست آمده و توضیحی زیادی برای آن ندارم

مقدار برابر 0.035 بود که در سایت کگل به درصد دقت 78 رسید .

فایل این پیشبینی در فایل ارسالی رر قرار داده شده است