به نام خدا

دانشگاه شهید بهشتی

دانشکده علوم ریاضی

گزارش تمرین سه شبکه عصبی

زهرا دهقانی تفتی (۹۶۲۲۲۰۳۷)

آذر ۹۹

فهرست مطالب

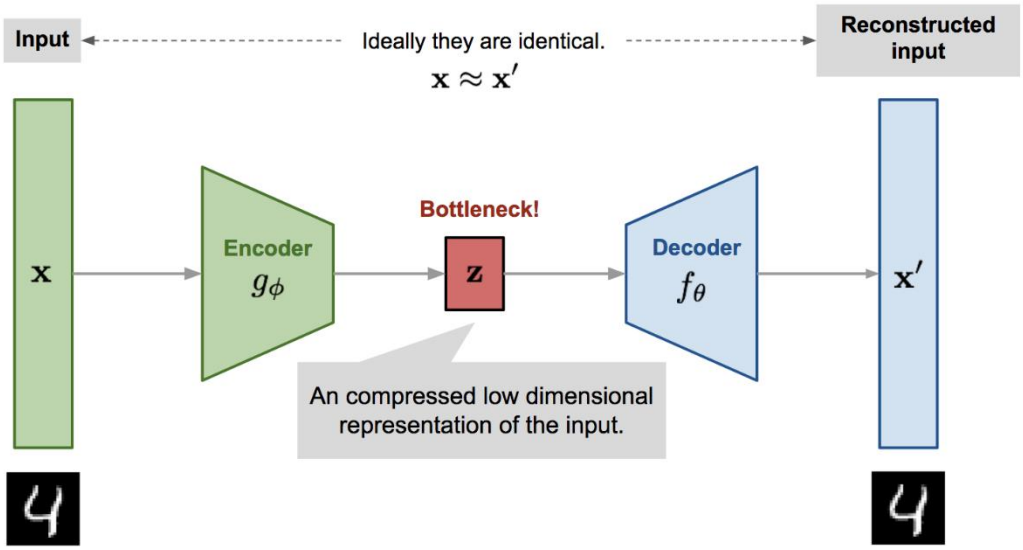
تشخیص تقلب ۲
راه حل و ایده های کلی ۲
ارزیابی نتایج ۸
آزمایش ۳: ۸
آزمایش ٤: ۹
آزمایش ٥: ۱۰
جمع بندی و نتیجه گیری ۱۲

تشخیص تقلب

راه‌حل و ایده‌های کلی

در این مسئله هدف این است که با یادگیری الگوی اطلاعاتی جزییات تراکنش‌هایی که قبلا انجام شده، بتوانیم تقلب را در تراکنش‌هایی که در داده تست به ما داده شده را تشخیص دهیم.

تشخیص تقلب در این مسئله را باید استفاده از یک مدل autoencoder انجام دهیم. ساختار autoencoder ها به این صورت است که از دو بخش encoder و decoder تشکیل شده‌اند. بخش encoder برای تبدیل داده‌ها به یک فضای فشرده‌تر و کوچک‌تر است و بخش decoder برای بازسازی داده‌ها از فضای کوچک‌تر به فضای اصلی است. در autoencoder باید بتوانیم هر آنچه را که به آن به عنوان ورودی دادیم، را برای خروجی بازسازی کنیم و هرچقدر این ورودی و خروجی شباهت بیشتری داشته باشند به آن معنی است که مدل ما الگو و توزیع آن مجموعه داده را یاد گرفته است و reconstruction error آن کم می‌شود.الگوی کلی یک autoencoder به صورت زیر است.



از مسئله‌ی تشخیص تقلب تحت عنوان anomaly detection نیز یاد می‌شود به این معنی که ما داده‌ای داریم و آن داده روند نرمالی دارد ولی یک سری داده‌هایی با روند ناهنجار در داده‌های ما ظاهر می‌شوند و تعداد داده‌هایی که باعث ناهنجاری می‌شوند خیلی کم است به همین دلیل تشخیص آن‌ها دشوار است.

در این مسئله ایده‌ی ما این است که یک Autoencoder بسازیم و آن را تنها با داده‌هایی که نرمال هستند و تقلب در آن‌ها نیست آموزش دهیم. با این کار مدل ما روند و توزیع داده‌های نرمال را یاد می‌گیرد و انتظار داریم وقتی یک داده‌ی نرمال را به آن بدهیم با خطای کمی بتواند همان داده را برای ما بازسازی کند و reconstruction error کم باشد. چون مدل روی داده های نرمال آموزش دیده و خطای بازسازی مدل برای داده‌های نرمال کم است؛ انتظار داریم اگر داده‌ای به آن ورودی داده شود

که توزیع نرمالی ندارد را نتواند به خوبی بازسازی کند و reconstruction error برای آن زیاد شود و آن داده را به عنوان fraud تشخیص دهیم.

پس کاری که می‌کنیم این است که بخش نرمال داده‌های آموزشی را جدا می‌کنیم و با استفاده از آن‌ها Autoencoder را آموزش می‌دهیم. خطای بازسازی داده‌های آموزشی و اعتبارسنجی را روی این مدل بدست می‌آوریم. انتظار داریم چون داده‌های اعتبارسنجی در آموزش مدل به کار نرفته خطای بازسازی بیشتری داشته باشد. بنابراین ماکسیمم خطای بازسازی داده‌های اعتبارسنجی روی این مدل را به دست می‌آوریم و آن را به عنوان یک threshold در نظر می‌گیریم. بعد از اتمام آموزش مدل، داده‌های تست را به مدل می‌دهیم، اگر خطای بازسازی داده‌ای از threshold بیشتر شد به این معنی است که مدل توزیع آن را بلد نبوده که بتواند آن را بازسازی کند و آن داده را به عنوان fraud در نظر می‌گیریم. اما اگر خطای بازسازی از threshold کمتر شد به این معنی است که داده از نوع نرمال بوده و مدل با توزیع آن آشنایی داشته و توانسته آن را باکمترین خطا بازسازی کند.

فرمول خطای بازسازی طبق شکل به صورت زیر است:

Reconstruction error : $x - x'$

من در کد از mean squared error به عنوان خطای بازسازی استفاده کردم.

فایل داده‌ها شامل ٤ فایل به نام‌های train_transaction، train_identity، test_transaction و test_identity است که فایل‌های transaction مربوط به جدول تراکنش‌ها است و فایل‌های identity مربوط به اطلاعات شناسه‌ای تراکنش‌ها است. برای استفاده از اطلاعات باید فایل‌های transaction و identity برای داده‌های آموزشی را با هم concat کنیم که برای هر تراکنش همه‌ی اطلاعات لازم را داشته باشیم. همین کار را برای داده‌های تست هم انجام می‌دهیم.

داده‌های این پروژه بسیار سنگین هستند و در ابتدا که از google drive خوانده می‌شوند حدود ٥ گیگ از حافظه را اشغال می‌کنند. برای اینکه حافظه کمتری اشغال شود و حافظه crash نکند باید تایپ عناصر موجود در داده‌ها را تغییر دهیم.

pandas به صورت خودکار، داده‌هایی که از نوع دسته بندی هستند را object در نظر می‌گیرد و با این کار حافظه زیادی اشغال می‌شود و دلیل آن این است که لیستی از pointer ها به آدرس حافظه را ذخیره می‌کنند. برای بهبود حافظه این نوع داده‌ها را به نوع category تغییر می‌دهیم.

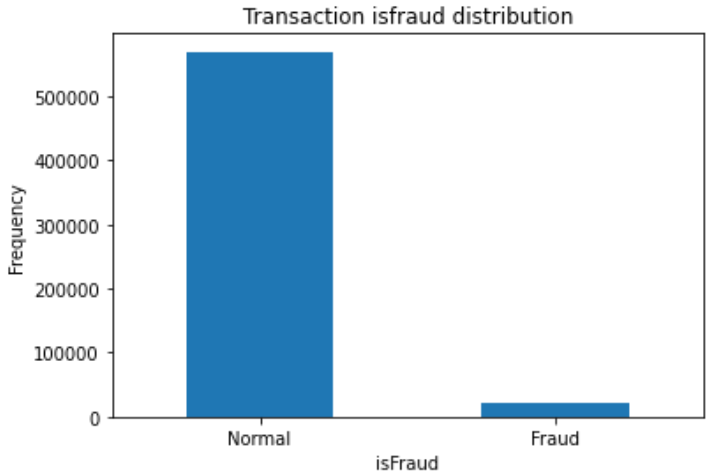
تعداد زیادی از داده‌ها به صورت اعشاری هستند. داده‌های که از نوع عدد اعشاری هستند به صورت float64 ذخیره شده‌اند که ٨ بایت از حافظه را اشغال می‌کنند. آن‌ها را به float تبدیل می‌کنیم با اینکار ٤ بایت از حافظه اشغال می‌شود.

می‌توان داده‌هایی که از نوع اعداد صحیح هستند و به صورت int64 ذخیره شده‌اند و ٨ بایت از حافظه را اشغال می‌کنند به int تبدیل کرد و حجم حافظه اشغالی را نصف کرد و در صورت اطمینان از مثبت بودن داده‌ها، آن‌ها را به صورت بی علامت تبدیل کرد تا حافظه‌ی کمتری مصرف کنند. ولی چون تعداد این نوع داده‌ها زیاد نیستند از اینکار صرف نظر می‌کنیم.

با این تغییر تایپ‌ها مموری به شدت آزادتر می‌شود!

حال باید به سراغ مرتب سازی داده‌ها برویم. باید جاهای خالی را پر کنیم و وِیژگی‌های نامربوط و آن‌هایی را که مقادیر زیادی null دارند را حذف کنیم و ارتباط بین داده‌ها را دربیاوریم تا بتوانیم ویژگی‌های تکراری را حذف کنیم و ویژگی‌های مفیدتری استخراج کنیم و با اینکار نتیجه بهتری بگیریم.

در ابتدا نگاهی به توزیع داده‌های نرمال و تقلب می‌اندازیم.



طبق شکل بالا می‌بینیم که تعداد داده‌های تقلب در داده‌های آموزشی کم است بنابراین جز روند نرمال حساب نمی‌شوند و تشخیص آن‌ها سخت است. از بین ۵۹۰۵۴۰ داده آموزشی که داریم، ۲۰۶۶۳ تقلب وجود دارد.

در داده‌های آموزشی که ما داریم ۱۱۵۵۲۳۰۷۳ خانه خالی وجود دارد به عبارتی ٤٥ درصد از جدول خالی است که این‌ها باید کنترل شوند. به این صورت که قرارداد می‌کنیم ستون‌هایی که بیشتر از ۵۰ درصد مقدار null دارند را از جدول حذف می‌کنیم زیرا اطلاعات خاصی به ما نمی‌دهند. تعداد چنین ستون‌هایی ۲۱٤ است و این ۲۱٤ ستون را از جدول حذف می‌کنیم.

بین ستون‌های باقیمانده، تعداد null های هر ستون را بدست می‌آوریم. احتمال اینکه بین ستون‌هایی که تعداد مقادیر خالی برابری دارند ارتباط وجود داشته باشد هست. این کار را برای تعدادی از ستون‌هایی که با حرف V شروع می‌شوند انجام می‌دهیم. برای مثال از V12 تا V74 را مورد بررسی قرار می‌دهیم. به این صورت که ستون‌هایی که تعداد null برابری دارند را با هم بررسی می‌کنیم و با رسم ماتریس confusion برای آن ویژگی‌ها وابستگی بین ویژگی‌ها را مشخص می‌کنیم. قرار داد می‌کنیم ویژگی‌هایی را که بیشتر از ۷۵ درصد باهم شباهت دارند را در یک دسته قرار دهیم. حالا چند دسته به وجود میاید که در هر دسته یک ویژگی را انتخاب می کنیم و بقیه را به دلیل تکراری بودن حذف می‌کنیم. با اینکار از بین V12 تا V74 تنها ۲۳ ویژگی مورد استفاده قرار می‌گیرد و ٤۰ ویژگی دیگر حذف می‌شود.

همچنین ستون‌هایی که تنها در آن‌ها یک مقدار متمایز (V107) وجود دارد را نیز حذف می‌کنیم.

ستون هایی را که بیشتر از ۹۰ درصد داده‌ی تکراری دارد را نیز پاک می‌کنیم زیرا اکثر مقادیر این ستون‌ها یکسان هستند و اطلاعات خاصی به ما نمی‌دهند . که ٥٤ ستون با این ویژگی داریم. ستون مربوط به تشخیص تقلب نیز در این لیست هست که چون خروجی است باید ازاین لیست پاک شود و بقیه ی ستون‌های این لیست باید هم از داده‌های آموزشی و هم از داده‌های تست پاک شوند.

بررسی می‌کنیم اگر سطری داریم که همه‌ی ستون‌های آن خالی است را پاک کنیم که طبق کد چنین ستونی نداریم.

بعد از اینکار باید جاهای خالی باقیمانده هم در داده‌های آموزشی و هم در داده‌های تست پر شوند.

ما در داده‌هایمان دو نوع داده‌های دسته بندی(به صورت متنی) و عددی داریم که برای پر کردن هر کدام روش جدگانه‌ای باید به کار ببریم. برای پر کردن ستون‌های مربوط به دسته بندی که در اینجا ۹ ستون دسته بندی داریم، باید هر خانه توسط عنصری که بیشترین تکرار را در آن ستون داشته پر شود یعنی چنین خانه‌هایی در داده‌های آموزشی با mode آن ستون در داده آموزشی پر می‌شوند و چنین خانه‌هایی در داده‌های تست با mode آن ستون در داده‌های آموزشی پر می‌شوند زیرا فرض می‌شود ما هیچ اطلاعاتی در مورد داده‌های تست نداریم.

برای پر کردن ستون‌های مربوط به داده‌های عددی آن خانه‌های خالی چه درد داده‌های تست و چه آموزشی باید توسط میانگین آن ستون در داده‌ی آموزشی پر شود. با این کار تمام جاهای خالی پر می‌شوند. با اجرای دو قطعه کد زیر مطمئن می‌شویم هیچ جای خالی‌ای در داده‌های ما وجود ندارد.

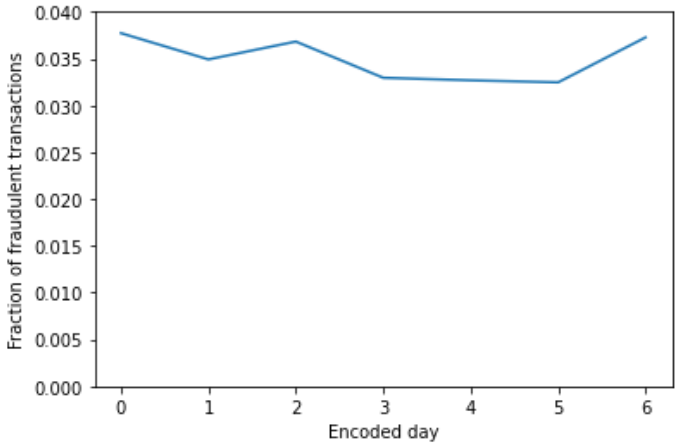
```
[ ]  train.isnull().sum().sum()

     0

[ ]  test.isnull().sum().sum()

     0
```

بعد از اینکار وارد آماده کردن داده‌ها برای دادن به شبکه عصبی می‌شویم. برای اینکه داده‌ها توسط شبکه عصبی قابل فهم باشد باید همگی به صورت عدد درآیند. بعنی داده‌های متنی را باید با استفاده از label encoding به صورت عدد درآورد. برای مثال اگر ستونی ۵ مقدار متمایز متنی دارد با label encoding مقادیر ستون‌های آن به جای داده‌های متنی، عدد از ۰ تا ٤ می‌شود. بعد از اینکار برای بهبود یادگیری شبکه و جلوگیری از نسبت دادن وزن برای مثال به داده‌ای که مقدار ٤ گرفته در مقایسه با آن که مقدار ۰ گرفته از روش one hot encoder استفاده می‌شود؛ به اینصورت که به تعداد مقادیر متمایز در هر ستون، به جدول ستون اضافه می‌شود برای مثال اگر ۵ مقدار متمایز در ستونی وجود داشت به آن ۵ ستون نسبت می‌دهد ولی در ایجا چون تعداد ستون‌ها زیاد است و داده ها نیز زیاد هستند از one hot encoder صرف نظر می‌کنیم.

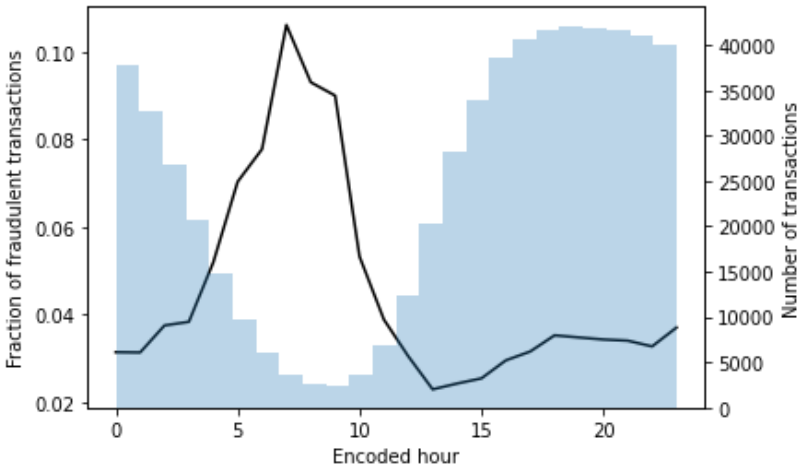
بعد از همه این کارها جدول را چاپ می‌کنیم تا مقادیر آن را مشاهده کنیم. ستون TransactionDT که در صورت پروژه در مورد آن توضیح داده شده بود و گفته شد بود اختلاف زمانی از یک مرجع است را مشاهده می‌کنیم. اما گفته نشده که این زمان در چه واحدی است. از آن جا که مقادیر این ستون زیادند به سال، ماه، ساعت و دقیقه نمی‌تواند باشد زیرا این اطلاعات از شروع سال ۲۰۱۷ به ما داده شده‌اند و اگر در هر یک از واحدهای گفته شده باشند از سال ۲۰۲۰ که الان در آن هستیم فراتر می‌رود!

پس نتیجه می‌گیریم مقادیر این ستون به ثانیه هستند. این ستون را به دو ستون روزهای هفته و ساعت‌های روز تبدیل می‌کنیم تا بتوانیم اطلاعات بیشتری را ازآن استخراج کنیم و ستون TransactionDT را حذف می‌کنیم.

نمودار توزیع تقلب بر حسب روزهای هفته را رسم می‌کنیم تا ارتباط آن‌ها را تشخیص دهیم ولی طبق نمودار زیر ارتباط چندانی بین روزهای هفته و تقلب صورت گرفته وجود ندارد پس این ستون اطلاعات چندانی به ما نمی‌دهد بنابراین آن را حذف می‌کنیم.



حال نمودار توزیع تقلب بر حسب ساعت‌های روز را رسم می‌کنیم. در نمودار زیر، سایه‌های آبی رنگ در نمودار نشان دهنده‌ی توزیع تراکنش صورت گرفته در ساعات روز است و خط مشکی نشان دهنده توزیع تراکنش‌های تقلب است که می‌بینیم در ساعاتی که تراکنش کمتر انجام می‌شود و به خصوص شب‌ها تراکنش‌های تقلب انجام می‌شود. این اطلاعات می‌تواند به ما کمک کند.



بعد از اینکار باید فقط قسمت نرمال داده‌های آموزشی را به دلایلی که در ابتدای گزارش گفته شده، برای آموزش به autoencoder دهیم. اینکار را انجام می‌دهیم سپس برچسب‌های isFraud را از داده‌ها حذف می‌کنیم زیرا نوع یادگیری autoencoder به صورت unsupervised است و بدون برچسب انجام می‌شود و فقط توزیع داده‌ها را یاد می‌گیرد.

یک ستون دیگر که باید از داده‌های تست و اموزشی حذف شود، TransactionID است و دلیل آن این است که هر تراکنش شماره مخصوص به خود را دارد و هیچ تاثیری در تعیین تقلب بودن یا نبودن آن تراکنش ندارد پس اطلاعاتی به ما نمی‌دهد و باید حذف شود.

یک راه دیگر که برای کاهش تعداد ستون‌ها (ویژگی‌ها) و فشرده‌تر کردن داده‌ها می‌توان از آن استفاده کرد، روش PCA است. کارکرد این روش به این صورت است که مولفه‌های اصلی را شناسایی می‌کند به این صورت که بعدها و ویژگی‌ها را در جهاتی که بیشترین پراکندگی (واریانس) را داشته باشند در نظر می‌گیرد. این بعدها باید بر هم عمود باشند که همین شرط تضمین می‌کند که ویژگی‌ها نسبت به هم کواریانس (همبستگی) ندارند.

برای استفاده از این روش باید از کتابخانه sklearn.decomposition، PCA را فراخوانی کنیم. سپس تعداد بعدهایی که می‌خواهیم داشته باشیم (تعداد بعدهای فشرده شده در فضای جدید) را تعیین می‌کنیم و روی مجموعه داده‌های آموزشی fit می کنیم. (در کد این پروژه از روش PCA استفاده نشده و از روش‌هایی که قبل‌تر توضیح داده شد استفاده شده.)

حال باید داده‌هایی را که داریم نرمال کنیم تا با هم قابل مقایسه گردند زیرا هر ویژگی بازه‌های مختلفی از مقادیر را می‌تواند داشته باشد و این کار باعث می‌شود مقایسه پذیر نباشند.

برای نرمال سازی من از روش MinMaxScaler استفاده کردم؛ زیرا با اینکار تمام داده‌ها بین ۰ تا ۱ در می‌آیند و در بدترین حالت که تعداد اعداد منفی داده‌ها زیاد باشد به رنج ۱- تا ۱ درمی‌آیند. در خروجی Decoder در Autoencoder نیز از تابع فعالیت sigmoid استفاده می‌کنیم که اعداد خروجی در بازه‌ی ۰ تا ۱ باشند و به آسانی قابل مقایسه شوند و تابع loss دقیق‌تر محاسبه گردد.

بعد از نرمال سازی 0.2 از داده‌ها را به عنوان داده‌های اعتبارسنجی جدا می‌کنیم.

بعد از کار روی داده‌ها وارد آموزش autoencoder ها می‌شویم.

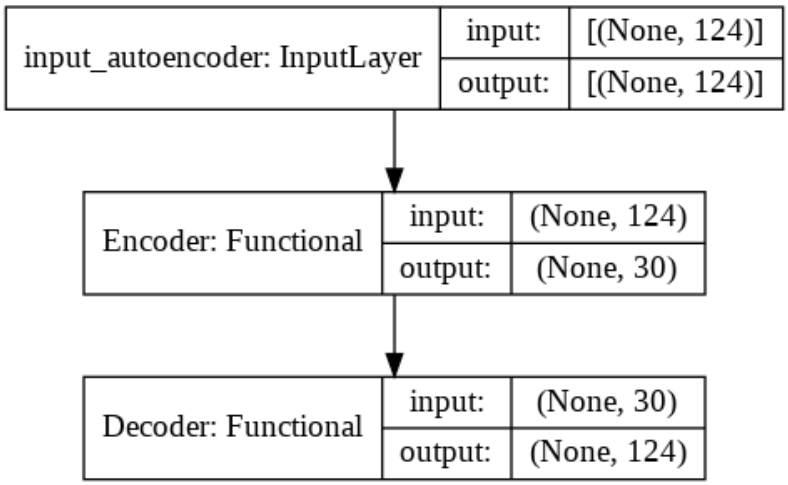
ارزیابی نتایج

در این قسمت می‌خواهیم بهترین ساختار Autoencoder را برای داده‌های خود بدست آوریم که کم‌ترین reconstruction error را داشته باشد. مدل‌های ما به صورت multi-layer perceptron هستند.

! توجه : autoencoder1 و autoencoder2 بر روی داده‌هایی آموزش داده شده که به روش standard نرمال شده‌اند و تابع فعالیت لایه‌ی خروجی decoder آن‌ها linear در نظر گرفته شده بخاطر همین نتیجه خوبی ندارند و اعداد ورودی و خروجی آنها در بازه‌ی مختلفی هستند. تنها تفاوت این دو مدل این است که در autoencoder2 سایز دسته بیشتر گرفته شده‌است. بنابراین این دو مدل خوب نیستند و مدل‌های بعدی روی داده‌هایی که با روش minmax نرمال شده‌اند آموزش داده شدند و تابع فعالیت خروجی برای decoder آن‌ها را sigmoid گذاشتیم که بازه‌ی اعداد ورودی و خروجی یکی باشد و نتیجه بهتری بگیریم.

آزمایش ۳:

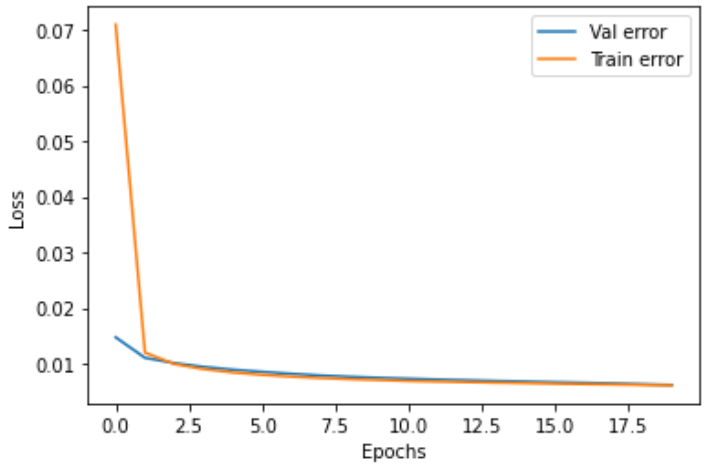
در این مدل ما ساختاری شبیه شکل زیر داریم. در این آزمایش ما از کل داده‌های آموزشی برای آموزش مدل استفاده نمی‌کنیم چون تعداد داده‌ها زیاد هست و سعی می‌کنیم با همان تعداد داده کم مدل خوبی بسازیم. در این مدل ورودی ما ۱۲٤ ویژگی دارد که تعداد ستون‌های باقیمانده از جدول‌های ما است که داده‌های مهمی داشتند. فضای میانی ما ۳۰ نورون دارد و در خروجی نیز انتظار داریم همان ورودی را بتوانیم بازسازی بکنیم.



با ساختن و آموزش این مدل در ۲۰ ایپاک به نتیجه خوبی می‌رسیم و خطا روی داده‌های اموزشی و اعتبارسنجی به صورت زیر است.

```
Epoch 20/20
 391/391 [==============================] - 1s 3ms/step - loss: 0.0061 - val_loss: 0.0062
```

طبق شکل زیر در این مدل خطا روی داده‌های آموزشی و اعتبارسنجی کاهش پیدا می‌کند و مدل خوبی است.



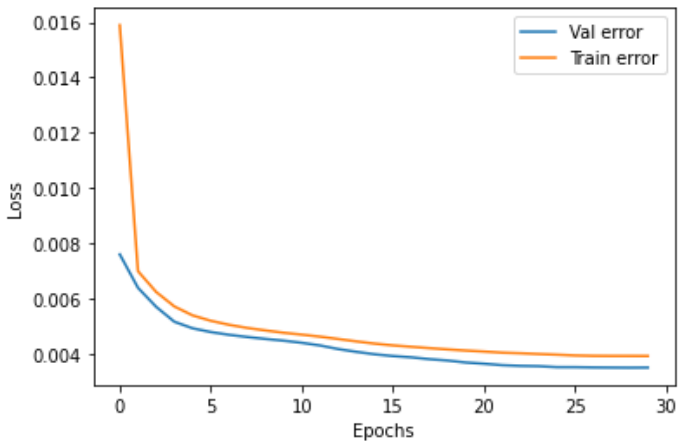
آزمایش ٤:

در این مدل همان ساختار مدل قبل را داریم ولی می‌خواهیم مدل را با کل داده‌های آموزشی آموزش دهیم و تعداد ایپاک ها را نیز به ۳۰ افزایش داده‌ایم. انتظار می‌رود در این صورت چون روی تعداد بیشتری نمونه یادگیری انجام می‌شود عملکرد بهتری داشته باشد. طبق خروجی مدل در ایپاک ۳۰ ام این مدل بهتر از مدل قبل عمل می‌کند و خطا روی داده‌های اعتبارسنجی و آموزشی نسبت به مدل قبل کم‌تر شده است.

```
Epoch 30/30
3562/3562 [==========================] - 12s 3ms/step - loss: 0.0039 - val_loss: 0.0035
```

دلیل این بهتر شدن این است که روی کل داده‌های آموزشی لرن شده و نمونه‌های بیشتری را دیده و ساختار و توزیع داده‌های نرمال را بهتر یاد گرفته است. و این افزایش تعداد ایپاک هم باعث شده تعداد مرتبه بیشتری داده‌ها را از اول ببینید و به خوبی داده‌ها را یاد بگیرد.

طبق شکل زیر خطای بازسازی مدل آتوانکدر ما به خوبی و با سرعت مناسبی روی داده‌های اعتبارسنجی و آموزشی کم شده‌است و این نشان‌دهنده‌ی این است که مدل ما ساختار و توزیع داده‌های آموزشی را به خوبی یادگرفته است.



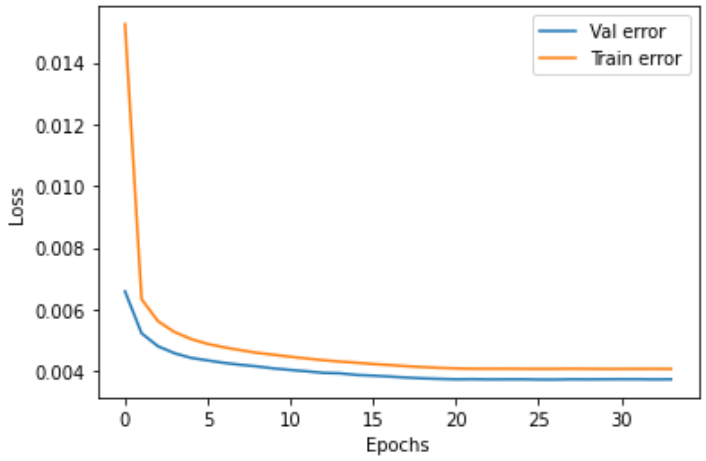
در این شکل اورفیت مشاهده نمی‌شود و این نشان از این دارد که مدل قابلیت تعمیم دارد به این معنی که؛ آن چیزی که از داده‌های آموزشی یادگرفته را توانسته به داده‌هایی که در آموزش آن استفاده نشده‌اند یعنی داده‌های اعتبارسنجی نیز تعمیم دهد و مدل ما این الگوها را حفظ نکرده است. تا اینجای کار مدل ٤ به عنوان بهترین مدل انتخاب می‌شود زیرا در مقایسه با مدل قبلی خطای کمتری روی داده اعتبارسنجی و آموزشی دارد.

آزمایش ٥:

ساختار و معماری این مدل نیز مشابه مدل ٣ است ولی در اینجا تعداد ایپاک‌ها را به ٦٠ افزایش دادیم تا خطای مدل کمتر شود ولی میبینیم که مدل در ایپاک ٣٤ام متوقف شده و این به معنی آن است که مدل در آستانه overfit شدن بوده و در حال حفظ ساختار داده‌ی آموزشی بوده و نه در یادگیری آن، به خاطر همین از این نقطه به بعد قرار بوده خطا روی داده‌های اعتبار سنجی افزایش پیدا کند، که متوقف شده است. طبق نتایجی که در ایپاک ٣٤ام از این مدل می‌گیریم در مقایسه با مدل ٤ خطای داده‌ی آموزشی و تست آن بیشتر است و همچنان مدل ٤ بهترین مدل ما است.

```
Epoch 34/60
3562/3562 [======================] - 12s 3ms/step - loss: 0.0041 - val_loss: 0.0037
Epoch 00034: early stopping
```

در شکل زیر روند کاهش خطا روی داده‌های تست آموزشی را تا ایپاک ٣٤ام مشاهده می‌کنیم.



تا این جای کار ما بهترین مدل را بدست آوردیم که مدل ٤ است.حال باید یک threshold بدست آوریم و این را برای پیش‌بینی تقلب بودن یا نبودن داده مورد استفاده قرار دهیم. به این صورت که با استفاده از مدل ٤ نتایج را برای داده تست پیش بینی می‌کنیم. خطای بازسازی را برای هر داده تست بدست می‌آوریم. سپس این خطا را با threshold ای که بدست آوردیم مقایسه می‌کنیم. اگر این مقدار خطا از threshold بیشتر شد به این معنی است که مدل ساختار این داده را بلد نبوده که خطای بازسازی آن از یک حدی بیشتر شده‌است و آن را به عنوان Fraud تشخیص می‌دهیم. ولی اگر خطای بازسازی برای داده تست

از آن threshold کمتر بود به این معنی است که مدل ساختار آن داده را به خوبی بلد بوده و توانسته با کمترین خطا آن را بسازد پس از نوع داده نرمال تشخیص داده می‌شود. که کد مربوط به این قسمت به صورت زیر است:

```
[ ]  for i in range(0,len(mse)):
       if mse[i] <= threshold1 :
         mse[i] = 0
       else:
         mse[i] = 1
```

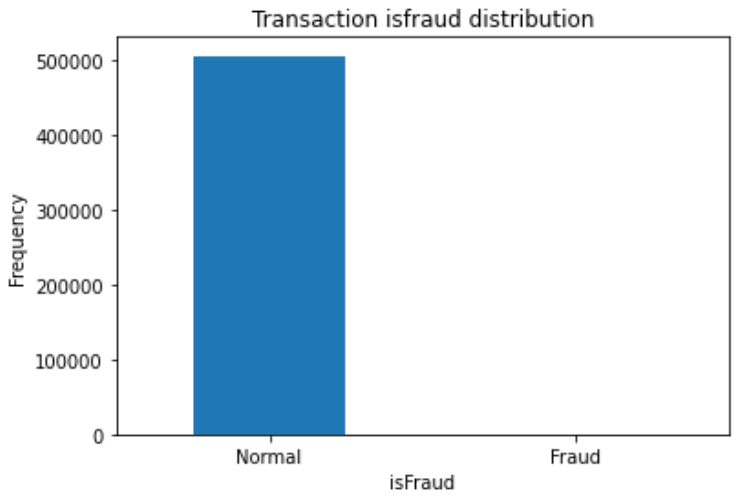
مقادیر ستون مربوط به بودن یا نبودن تقلب را با mse جدید که صفر و یک است جاگذاری می‌کنیم.

حال برای بدست آورن threshold کاری که می‌کنیم این است که بیشترین خطای بازسازی برای داده‌های اعتبارسنجی را بدست آوریم زیرا منطقی است که معمولا خطای داده‌های اعتبارسنجی بیشتر از خطای داده‌های آموزشی باشد زیرا مدل روی داده‌های آموزشی لرن شده پس آن‌ها را یادگرفته ولی از داده‌های اعتبار سنجی برای آموزش مدل استفاده نشده و انتظار خطای بیشتری روی آن‌ها داریم.

Threshold = max reconstruction error on validation dataset

بعد از بدست آوردن threshold کار پیش بینی برای داده‌های تست را انجام می‌دهیم و جواب برای داده‌های تست را در یک فایل به اسم submission.csv ذخیره می‌کنیم.

همانطور که می‌بینیم طبق مدلی که داشتیم و threshold ای که تنظیم کردیم از بین ۵۰۶۶۹۱ داده‌ی تستی که داریم ۱۴۹ تقلب کشف شده‌است.



جمع‌بندی و نتیجه‌گیری

با توجه به اینکه ورودی وخروجی آتوانکدر ما باید یکسان باشد نتیجه می‌گیریم برای استفاده از آن در تشخیص تقلب یا وقوع ناهنجاری، آتوانکدر ابتدا باید روی داده‌های نرمال (داده هایی که احتمال وقوع آن‌ها بیشتر است) آموزش داده شود تا توزیع ساختار آن‌ها را یاد بگیرد و با استفاده از معماری‌های مختلف و تکنیک‌های دیگر آنقدر باید خوب آموزش ببیند تا اختلاف بین داده‌ی ورودی و خروجی بازسازی شده برای داده‌های نرمال کم شود. از این پس در مرحله تست هر داده‌ای که به آن داده شود باید بتواند آن را با کمترین خطا بسازد ولی اگر داده‌ای به آن داده شد که اختلاف ورودی و خروجی (خطای بازسازی) آن از یک حدی بیشتر شد نتیجه می‌گیریم داده‌ی ما غیرنرمال بوده که مدل نتوانسته آن را به خوبی بازسازی کند.

از دیگر نتیجه‌ای که از آزمایش‌های انجام شده می‌گیریم این است که بازه‌ی اعداد ورودی و خروجی باید یکسان باشد برای مثال اگر داده‌های ورودی ما بین صفر و یک هستند خروجی نیز باید در همین بازه باشد و برای این کار باید در لایه‌ی آخر دیکدر از تابع فعالیت sigmoid استفاده شود و یا اگر اعداد ورودی ما بین ۱- تا ۱ هستند برای تابع فعالیت لایه‌ی آخر دیکدر باید از tanh استفاده شود در غیر اینصورت نتیجه‌ای که می‌گیریم درست نیست و خطای زیادی خواهد داشت.

همچنین استفاده از لایه‌های dropout در بین لایه‌های آتوانکدر باعث کم شدن وقوع overfit می‌شود