گزارش پروژه ۳

مهرانه مقتدائی‌فر ۹۷۲۲۲۰۸۶ و محمدرضا صیدگر ۹۷۲۲۲۰۵۵

در ابتدا با توجه به بررسی داده های موجود، دو سری جدول داریم، تراکنش ها و شناسه ها از آنجایی که داده های جدول شناسه ها از جدول تراکنش ها کمتر بود و جدول تراکنش ها بخش زیادی از جدول شناسه ها را پوشش میداد در ابتدا کل جدول شناسه ها را کنار گذاشتیم و با تراکنش ها کار کردیم.

البته اگر از دو جدول باهم ددیگه استفاده کنیم قطعا نتیجه خیلی بهتری خواهیم گرفت، از جدول شناسه ها میتوان در انتها برای درست کردن مدل استفاده کرد و داده های موجود در آن را نیز مورد بازرسی و استفاده قرار داد.

اما کاری که میکنیم در ابتدا با جدول identity کاری نداریم و به مرتب کردن داده های جدول تراکنش ها میپردازیم.

مرتب کردن داده ها:

همانطور که مشاهده میکنید این جدول شامل ۳۹۴ ستون از ویژگی های مختلف میباشد. مقدار زیادی از این ستون ها دارای مقادیر NAN یا همون پوچ هستند.

این داده ها باید مرتب شوند به طوری که در کل جدول دیگر هیچ داده خالی نداشته باشیم.

برای پر کردن داده های خالی از چندین روش استفاده کردیم.

ابتدا دیدیم که در هر ستون و از هر ویژگی چه مقداری از آن خالی است. درواقع در تک تک ستون ها گشتیم و تعداد پوچ ها را خارج کردیم و در num_of_nulls قرار دادیم. سپس هر ستونی را که تعداد عناصر پوچ آن بیشتر از نصف تعداد عناصر هر ستون بود(یعنی نصف تعداد سطر ها 295270 = 590540/2) در یک لیستی به نام null_columns قرار دادیم. در واقع این لیست، لیست ستون هایی است که میخواهیم از جدولمان حذف کنیم.

سپس در لیست train_null_list همان ستون هایی که قرار از از جدول داده های ما حذف شود را قرار دادیم و اسم آنها را نمایش دادیم.

در مرحله بعدی این ستون ها را از جدولمان drop میکنیم. جدول ما به ۲۲۰ ستون تبدیل میشود. در این مرحله ستون TransactionID را نیز حذف کردیم چون به نظر تاثیری در نتیجه نهایی و کاری که میخوایم انجام بدهیم ندارد.

حال تعدادی از ستون های جدول هستند که نوع آنها object است ، اما تعداد بیشتری عددی هستند. این داده های عددی را همه را با میانگین آنها پر میکنیم.

پس از انجام اینکار تمامی ستون های عددی که NAN هستند پر میشوند. حال تنها ستون های Object ما هستند که باید مقادیر خالی آن را نیز به نوعی پر کنیم. در اینجا از دو روش ستون ها را پر میکنیم.

ابتدا ستون هایی که نوع آنها object هست و دارای داده ی پوچ هستند را نمایش دادیم، ۸ ستون داریم، تمامی آنها را با نمودار نشان دادیم تا بفهمیم که ۱.انواع داده های موجود در این ستون ها چی هستند و ۲. فراوانی کدام یک از آنها بیشتر است.

همانطور که میبینید، در ۶ ستون یکی از داده ها میزان قابل توجی از بقیه بیشتر است، به همین دلیل مقادیر پوچ این ستون ها (P_emaildomain, M1, M2, M3, card4, card6) را با آن داده ای که فراوانی بیشتری داشته (مد آنها) پر میکنیم.

دو ستون M4,M6 باقی میماند. که همانطور که در نمودار میبینیم، مقدار F های ستون M6 خیلی تفاوتی با مقدار T های آن ندارد، به همین دلیل این ستون را با F پر نکردیم بلکه با روش دیگری به اسم forward fill پر میکنیم، به این شکل که مقدار ستون توجه

میکند و به اولین مقدار پوچی که برسد آن ها با مقدار قبلی پر میکند و همینطور تا آخر ادامه دارد. برای ستون M4 نیز به همین روند عمل میکنیم.

سپس تمامی مقادیر پوچ جدول به همین شکل پر میشوند و دیگر هیچ مقدار پوچی نداریم.

مرجله بعد این است که داده ها را برای دادن به شبکه آماده کنیم.

در ابتدا ستون isFraud را از داده ها جدا میکنیم و به عنوان label ذخیره میکنیم. زیرا در انتها برای تشخیص جعلی بودن یا نبودن به آن نیازی داریم.

مثل همیشه داده های object را باید encode کنیم. برای داده های categorical از one hot encoding استفاده کردیم و سپس تمامی داده ها را scale کردیم.

در اینجا کار پردازش داده ها و مرتب کردن آنها به اتمام میرسد. نوبت آن است که مدل را بسازیم.

ساختن مدل و توضیح عملکرد شبکه:

میخواهیم یک مدل autoencoder بسازیم. در واقع مدل های اتوانکدر به این گونه عمل میکنند که بردار ورودی داده ها را دریافت میکنند سپس این بردار را به فضای ویژگی ها (latent vector) میبرد. سپس این بردار را باید decode کنیم و دوباره این مدل سعی میکند تا بردار ورودیه اصلی را بسازد

اتوانکدری که ما در اینجا قرار دادیم دقیقا همانند یک شبکه معمولی دارای لایه های مختلف با تعداد نرون های متفاوت و شامل لایه های dropout نیز میباشد. از آنجایی که بردار scale شده که توسط ستون های جدول به دست آمده بود، ۲۹۲ تا بود پس درواقع بردارد ورودی ما نیز باید ۲۹۲ نرون داشته باشد تا همه داده ها را دریافت کند.

حال لایه های encoder را با تعداد نرون های مختلف قرار دادیم، دو لایه dropout نیز گذاشتیم، همانطور که مشخص است در لایه آخر یعنی خروجی انکدر ما ۳۰ تا نورون وجود دارد. در واقع خروجی انکودر همان بردار latent ما هست. که در مرحله بعد ورودی دیکدر ما خواهد بود.

لایه های دیکدر ما برعکس لایه های انکدر ما هست طوری که خروجی آن دقیقا باید ۲۹۲ تا(به اندازه ی داده هایی که به مدل داده ایم) باید باشد. لایه ها بهتر است طوری ساخته شوند که تعداد پارامتر های انکدر و دیکدر تقریبا یکسان باشند تا مدل ما عملکرد بهتری داشته باشد.

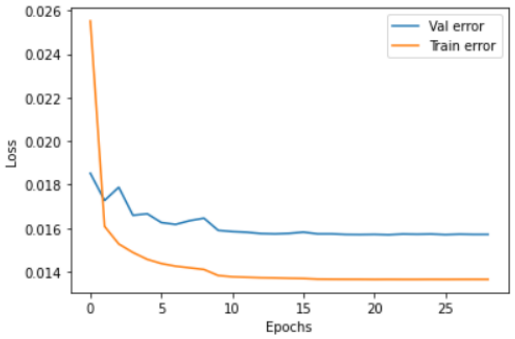
حال پس از آنکه ساختار encoder و decoder را مشخص کردیم نوبت آن است که autoencoder را بسازیم.

همانطور که گفته شد مدل اتوانکدر در ابتدا بردار ورودی را میگیرد سپس به فضای latent انکد میکند و در آخر دوباره بردار latent را به بردار خروجی decode میکند.

در تمامی مراحل از تابع فعالسازی relu استفاده کردیم و فقط برای خروجی از sigmoid

تابع loss که از آن استفاده کردیم mean squared error است.

در این مرحله اندازه بردار latent ما برابر با ۳۰ بود و با استفاده از بهینه گرadam و validation split = 0.2 و batch_size = 128 نتیجه زیر حاصل شد

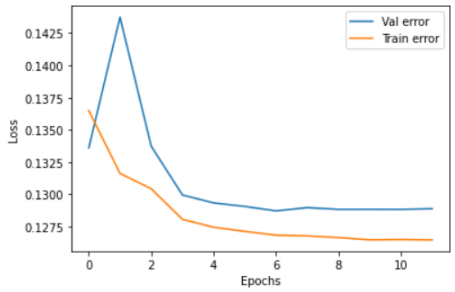


مقدارloss آن تقریبا برابر با 0.1 شد که مقدار خوبی است

حال پس از آنکه اتوانکدر مورد نظرمان را آموزش دادیم نوبت آن است که در یک classifier از آن استفاده کنیم. اینگونه که آن ورودی شبکه مورد نظری که میخواهیم برای طبقه بندی از آن استفاده کنیم، درواقع خروجی است که مدل autoencoder ما ساخته است.

آن را ورودی میدهیم و توسط لایه های متفاوت، شبکه را آموزش میدهیم. همانطور که میبینید دوباره از dropout برای جلوگیری از overfitting استفاده کردیم. در اینجا تابع loss که استفاده میکنیم با تابع autoencoder ما متفاوت است و در اینجا از تابع binary_crossentropy استفاده کردیم. بهینه گر ما همان adam است و validation split = 0.3 در نظر گرفته شده.

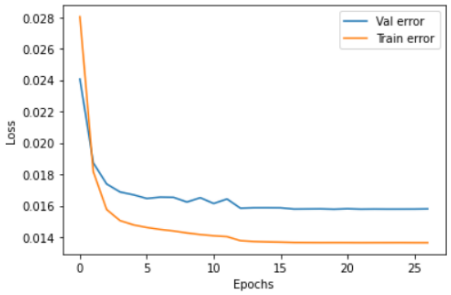
با لرن کردن شبکه توجه میشویم که دقت ما بر روی داده های آموزشی چیزی در حدود ۹۶٪ است، و مقدار loss که داریم نیز خوب است.



برای مدل دوم کاری که انجام میدهیم آن است که مقدار بردار خروجی encoder مان را کم میکنیم، یعنی از ۳۰ به ۱۰ کاهش میدهیم پس از آنکه این کار را انجام دادیم دوباره مدل اتوانکدر خود را آموزش میدهیم،

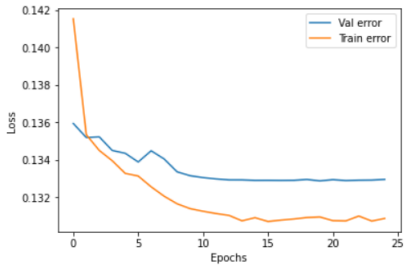
مقدار validation split را نیز به 0.3 تغییر دادیم

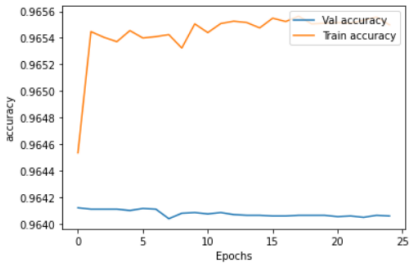
مشاهده میکنیم که مقدار loss ما نسبت به حالت قبلی کمتر شده و به شکل نمودار زیر در می آید



همانطور که گفتیم باید این مقدار ورودی را نیز به شبکه classifier نیز بدهیم، در اینجا تغییراتی نیز در ساختار شبکه ایجاد کردیم و درواقع تعداد پارامتر های آن را کم میکنیم و تعداد لایه ها نیز کمتر از حالت قبل است

در این حالت شبکه ما یادگیری بهتری داشت و مقدار دقت آن خیلی تغییری نکرد اما میزان loss آن خیلی بهبود یافت. در شکل زیر نمودار دقت و loss را باهم مشاهده میکنیم.





مدل آخر بهترین مدل ما بود، وزن و خود مدل در درایو ذخیره شده و لینک آنها ارسال میشود.

به دقت خوبی بر روی داده های آموزشی رسیدیم. حال داده های تست را نیز باهم بررسی میکنیم.

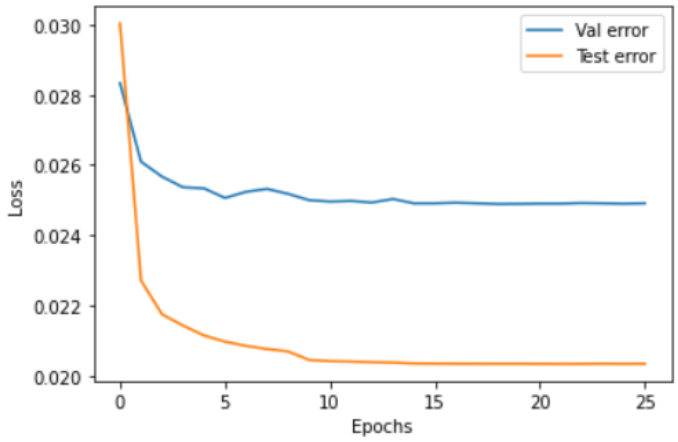
همانطور که میبینید در ابتدا مانند داده های آموزشی باید داده های تست را نیز مرتب کنیم چون در آنها نیز تعداد زیادی NAN وجود دارد. به همین دلیل دقیقا مشابه مرحله اول کار بر روی داده های تست نیز همان کار ها را انجام دادیم.

برای حذف ستون های پوچ، دقیقا گشتیم و ستون هایی که در داده های آموزشی به طور کل حذف کرده بودیم را در داده های تست پیدا کردیم و دقیقا همان ها را نیز در اینجا از بین بردیم. دقیقا ۱۷۴ تا ستون مشابه را حذف میکنیم.

در داده های تست ستونی به نام isFraud نداریم، و درواقع پس از آنکه عملیات حذف داده های پوچ را انجام میدهیم به جای ۲۱۹ تا ستون که در داده های آموزشی داشتیم در اینجا ۲۱۸ تا داریم. بقیه کار ها را نیز مشابه داده های آموزشی انجام میدهیم و به شبکه اتوانکدر میدیم.

در انتها دقیقا مشابه داده های آموزشی، ۲۹۲ تا ستون ویژگی داریم که به اتوانکدر مورد نظر میدیم و این مدل دقیقا مشابه مدلی است که برای داده های آموزشی استفاده شد، میخواهیم تفاوت loss آنها را مقایسه کنیم.

همانطور که در نمودار زیر مشاهده میکنید مقدار loss که داریم تا حد خوبی پایین است، پس به نظر میرسد که مدل ما برروی داده های تست نیز خوب کار میکند.



نتیجه گیری:

در انتها نیز باید گفت که به دلیل زیاد بودن حجم داده ها، میتوانیم این داده ها را با جدول identity ترکیب(merge) کنیم که در اون صورت داده های جدول کمتر میشوند (زیرا تعداد سطر ها از ۵۹۰ هزار تا به تعداد سطر های جدول identity که حدود ۱۴۰ هزار تا هست کاهش می یابد.)

اگر داده ها را ترکیب کنیم مقدار NANها بیشتر خواهد شد. همچنین ستون های زیادی در این جدول (idهای مختلف) کاربردی ندارند و می توانیم آنها را حذف کنیم، اما در کل تعداد داده های کمتری نسبت به حالت قبلی خواهیم داشت و شبکه ما راحت تر آموزش خواهد دید. همچنین تعدادی از ستون ها اطلاعات اضافه تری نیز به ما میدهند. اما در کل خیلی متفاوت نخواهد بود.

ما دیگر بر روی داده های identity کاری نکردیم اما در این مدلی که ما داریم نیز در نهایت به دقت خوبی هم بر روی داده های تست و هم بر روی داده های train رسیدیم و این نشان میدهد که شبکه ما به درستی تشخیص میدهد و دقت ۹۶٪ دارد که بسیار خوب است.