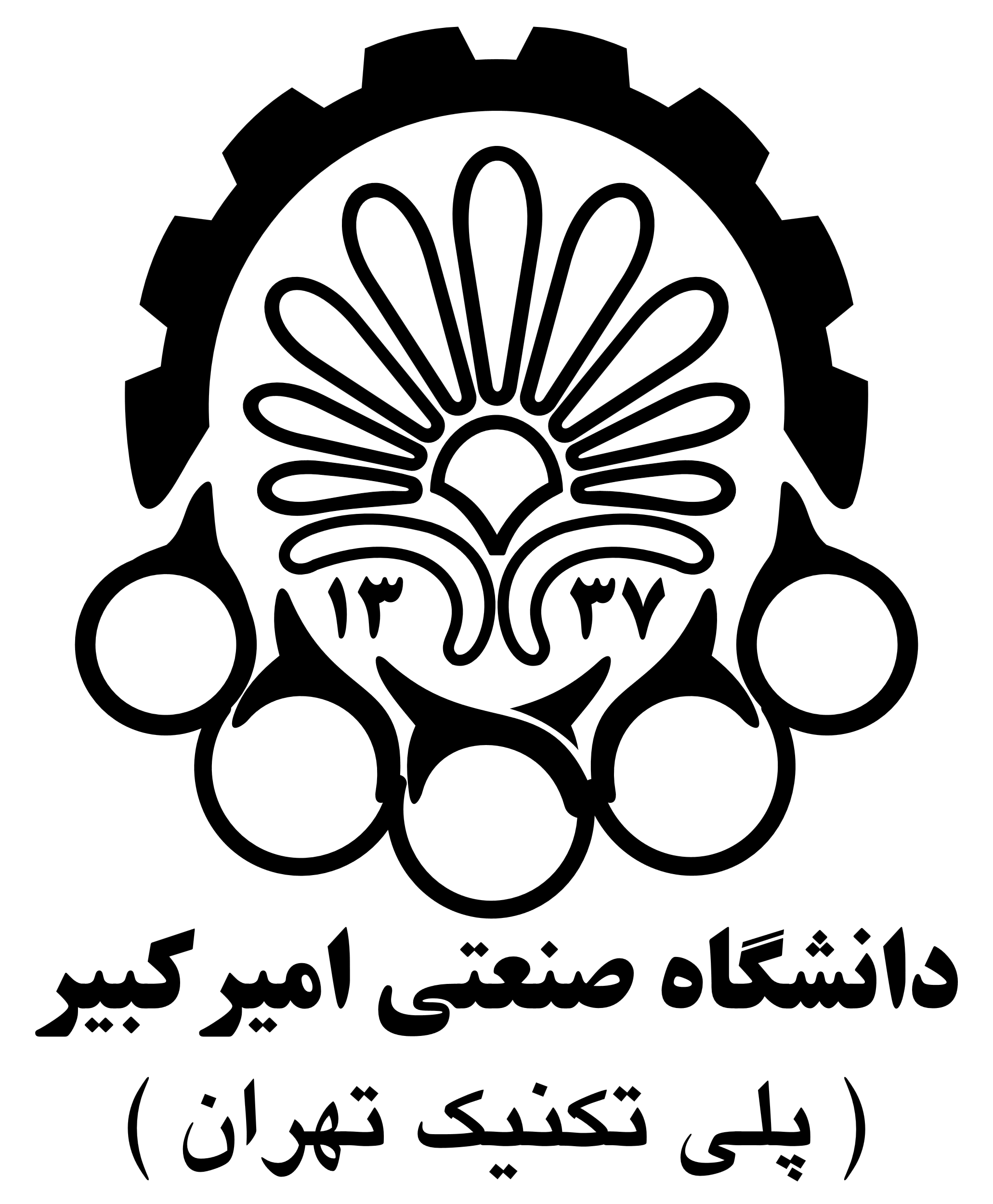
****

**دانشکده مهندسی کامپیوتر**

**تمر ین دوم درس شبکه های عصبی**

**دکتر صفابخش**

**غلامرضا دار 400131018**

**زمستان 1400**

فهرست مطالب

[سوال 1) 3](#_Toc98108227)

[1) 4](#_Toc98108228)

[2) 11](#_Toc98108229)

[3) 13](#_Toc98108230)

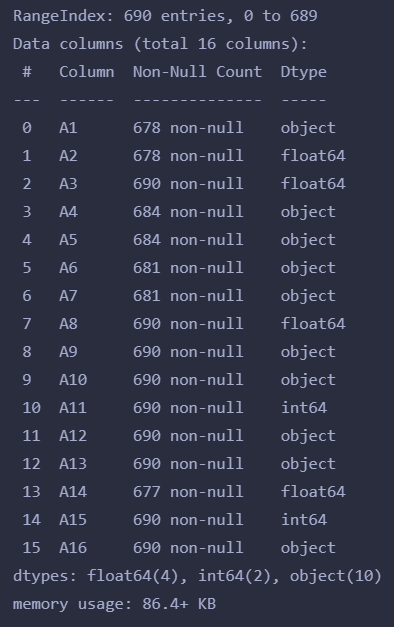
[4) 16](#_Toc98108231)

[5) 18](#_Toc98108232)

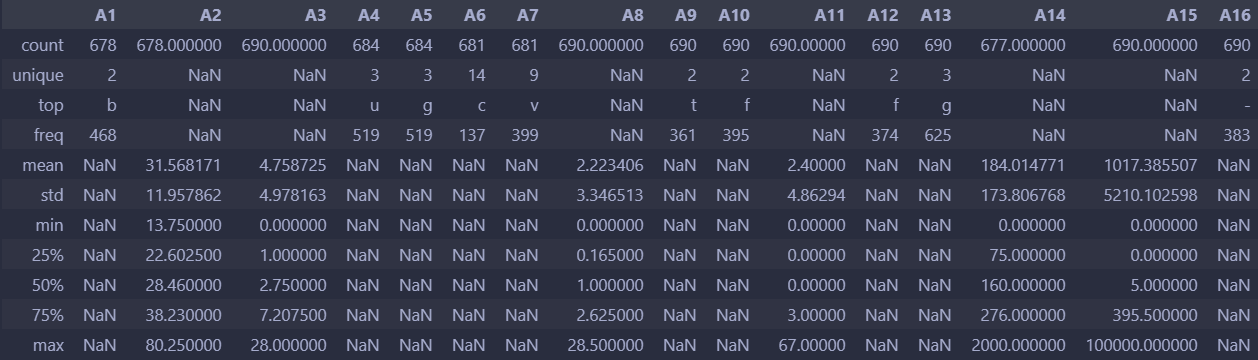
[6) 20](#_Toc98108233)

# سوال 1)

قبل از شروع، لازم است دیتاست را فراخوانی کنیم و کمی با آن آشنا شویم. همانطور که در طرح سوال گفته شده، این دیتاست شامل 690 داده با 15 ویژگی مختلف است.



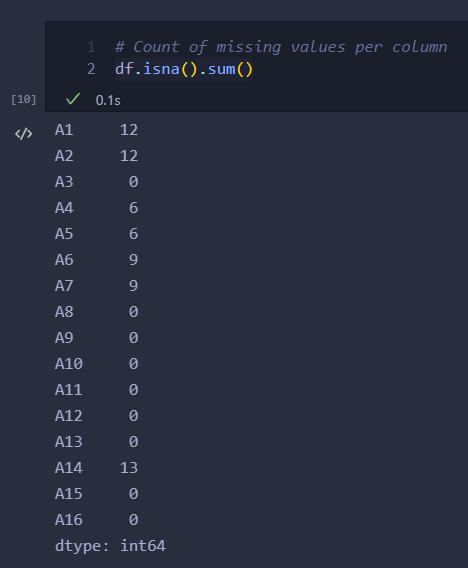
تعداد 6 ویژگی از نوع عددی هستند و 9 ویژگی دیگر از نوع Categorical هستند. همچنین ستون مربوط به Label داده ها نیز به صورت Categorical با دو مقدار ‘+’ , ‘-‘ است. همچنین همانطور که در تصویر زیر میبینید، مقادیر ستون های عددی در بازه های مختلف و با خواص آماری مختلفی هستند.



## 1)

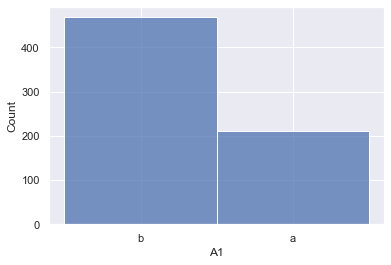
در این بخش با کمک اطلاعاتی که نسبت به ویژگی های مختلف کسب کردیم میخواهیم به پیش پردازش داده ها بپردازیم. برای اینکار 3 مرحله "**تصحیح مقادیر گم شده**"، "**نرمال سازی**" و "**کدگذاری (Encoding)**" را بر روی تک تک ویژگی ها اعمال میکنیم.

بنابراین در مرحله اول میخواهیم بدانیم چه ویژگی هایی مقادیر گم شده دارند.



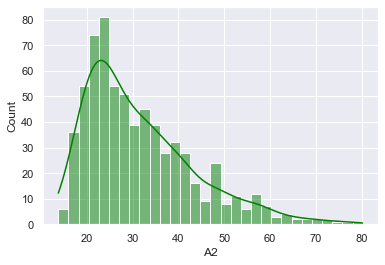
همانطور که در تصویر بالا مشاهده میکنید، ویژگی های A1, A2, A4, A5, A6, A7, A14 دارای مقادیر گم شده هستند که در این بین ویژگی های A2 و A14 عددی هستند.

با ویژگی **A1** شروع میکنیم :



مشاهده میشود که ویژگی A1 دارای دو مقدار ‘a’, ’b’ است. تعداد 468 داده مقدار b و تعداد 210 داده مقدار a را برای این ویژگی دارند. از آنجایی که مقدار b فراوانی تقریبا دوبرابری نسبت به a دارد احتمالا اگر در آن 12 داده ای که مقداری برای ستون A1 وجود ندارد b را قرار دهیم مشکلی پیش نیاید.

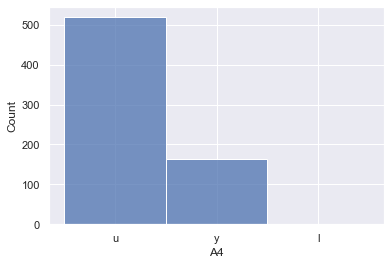
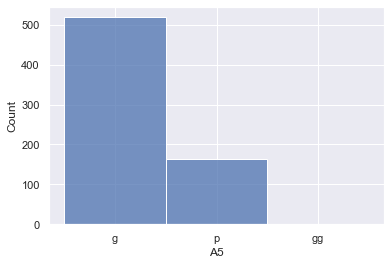
ویژگی **A2 :**



ویژگی A2 یک ویژگی عددی است. بنابراین بهتر است برخلاف ویژگی A1 که از Mode برای جایگزینی مقادیر گم شده استفاده کردیم، اینجا از Mean استفاده کنیم. مقدار Mean برای این ویژگی عدد 31.56 است.

ویژگی های  **A4, A5:**

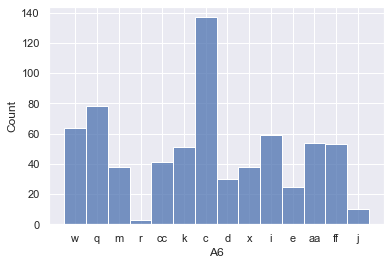
این دو ویژگی مانند ویژگی A1 کتگوریکال هستند و با توجه به توزیعی که مقادیر مختلف آنها دارد میتوانیم همانند A1 عمل کنیم و بجای مقادیر گم‎شده از Mode داده ها استفاده کنیم.

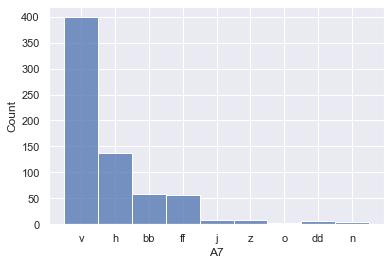


ویژگی های  **A6, A7:**

تفاوتی که این دو ویژگی با ویژگی های گسسته قبلی دارند تعداد مقادیر یکتای آنهاست. همانطور که مشخص است این دو ویژگی تعداد بسیار بیشتری مقدار مختلف به عنوان ارزش خود قبول میکنند که این باعث میشود درستی انتخاب Mode برای تمامی خانه های خالی با شک همراه باشد. متاسفانه به دلیل حفظ امنیت کاربران، ستون ها و مقادیر این دیتاست با اسامی تصادفی و بی معنی جایگزین شدند. این باعث میشود نتوانیم برای این مرحله و مراحل بعد از معنای ویژگی ها استفاده کنیم و از تکنیک های Feature Engineering استفاده کنیم. به عنوان مثال ویژگی A7 **میتواند** میزان حقوق دریافتی ماهانه اشخاص(در بسته های گسسته) باشد با توجه به اینکه حقوق های بالاتر را افراد کمتری دارند که در این صورت استفاده از Mode برای این ویژگی گزینه مناسبی خواهد بود. اما همانطور که ذکر شد برای این دیتاست نمیتوانیم از معانی ویژگی‎ها استفاده کنیم.

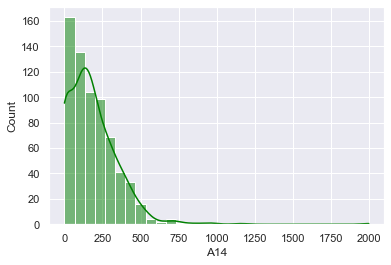
اما با توجه به اینکه تعداد داده های گم شده برای این دو ویژگی خیلی کم است (6 داده از 690 داده) احتمالا جایگزینی با Mode برای این دو ویژگی حتی اگر اشتباه هم باشد تاثیر زیادی نگذارد.



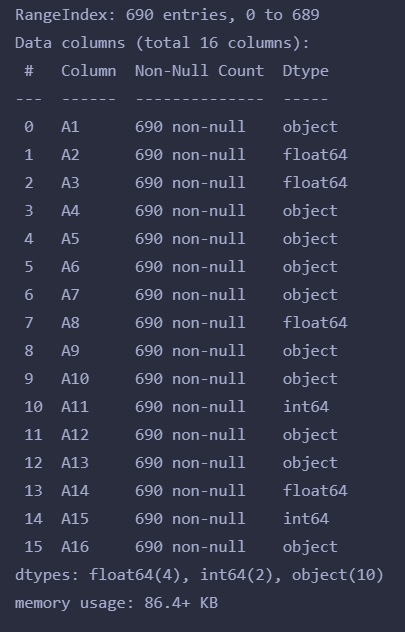


ویژگی **A14** :

در نهایت به ویژگی A14 میرسیم که همانند ویژگی A2 یک ویژگی عددی است. جایگزینی مقادیر گم شده این ویژگی با میانگین نیز انتخاب خوبی است.



همانطور که در تصویر زیر میبینید پس از انجام این مرحله همه ویژگی ها تعداد 690 داده دارند یعنی داده های گم شده به کلی از بین رفتند.

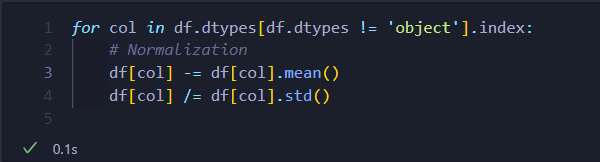


مرحله بعدی پیش‎پردازش میتواند **حذف ویژگی های اضافی** یا حتی **ترکیب ویژگی های مختلف** با هم باشد(مثلا روز و سال و ماه را به یک ویژگی عددی تبدیل کنیم). اما همانطور که قبلا هم گفتیم معنی اصلی این ویژگی ها به دلیل مسائل امنیتی پوشانده شده اند بنابراین برای این دیتاست این مرحله را انجام نمیدهیم.

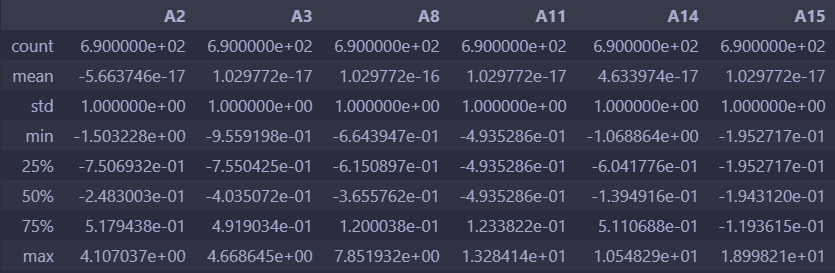
**نرمال سازی مقادیر عددی:**

همانطور که میدانید، خیلی از الگوریتم های یادگیری ماشین با اعداد کوچک و حول صفر رفتار بهتری میکنند و این امر باعث همگرایی سریعتر بسیاری از الگوریتم ها میشود. بنابراین در این مرحله تمام ویژگی های عددی دیتاست را نرمال سازی میکنیم.

برای نرمال سازی ویژگی ها کافیست میانگین مقادیر یک ویژگی را از همه مقادیر آن ویژگی کم کنیم، سپس مقادیر حاصل را بر انحراف از معیار مقادیر تقسیم کنیم.



خواص ویژگی های عددی پس از نرمال سازی را در تصویر زیر مشاهده میکنید.

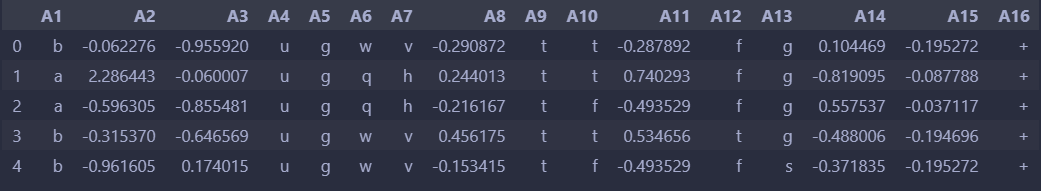


همانطور که میبینید پس از نرمال سازی ویژگی ها، میانگین همه ویژگی ها تقریبا **صفر** و انحراف از معیار ویژگی ها حدودا **یک** میباشد.

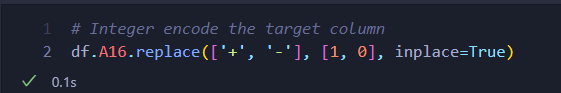
**Encoding**

همانطور که میدانید اکثر الگوریتم های یادگیری ماشین نمیتوانند با داده های Categorical کار کنند. به همین دلیل نیاز است تمامی ویژگی هایی که مقادیر غیر عددی دارند را طوری تغییر دهیم که تمامی داده های موجود در دیتاست ما عدد باشند.

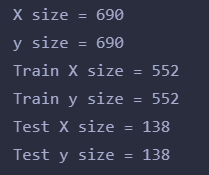
برای انکد کردن ویژگی های Categorical به دو روش عمل میکنیم. برای ویژگی هایی که ترتیب دارند(مانند مقطع تحصیلی : دیپلم/کارشناسی/کارشناسی ارشد/...) کافیست هر رشته را با یک عدد به ترتیب جایگزین کنیم. اما همانطور که بارها گفته شد ما نمیدانیم در این دیتاست چه ویژگی هایی مقادیر ترتیبی دارند و چه ویژگی هایی خیر. به عنوان مثال مقطع تحصیلی میتواند یکی از همین 15 ویژگی دیتاست ما باشد. به همین علت از راه دیگری به اسم One-hot Encoding برای همه ستون ها استفاده میکنیم.



و برای ستون آخر که Label ما است از یک Integer Encoding ساده استفاده میکنیم به این شکل که به جای علامت – عدد صفر و به جای علامت + عدد 1 را جایگزین میکنیم.

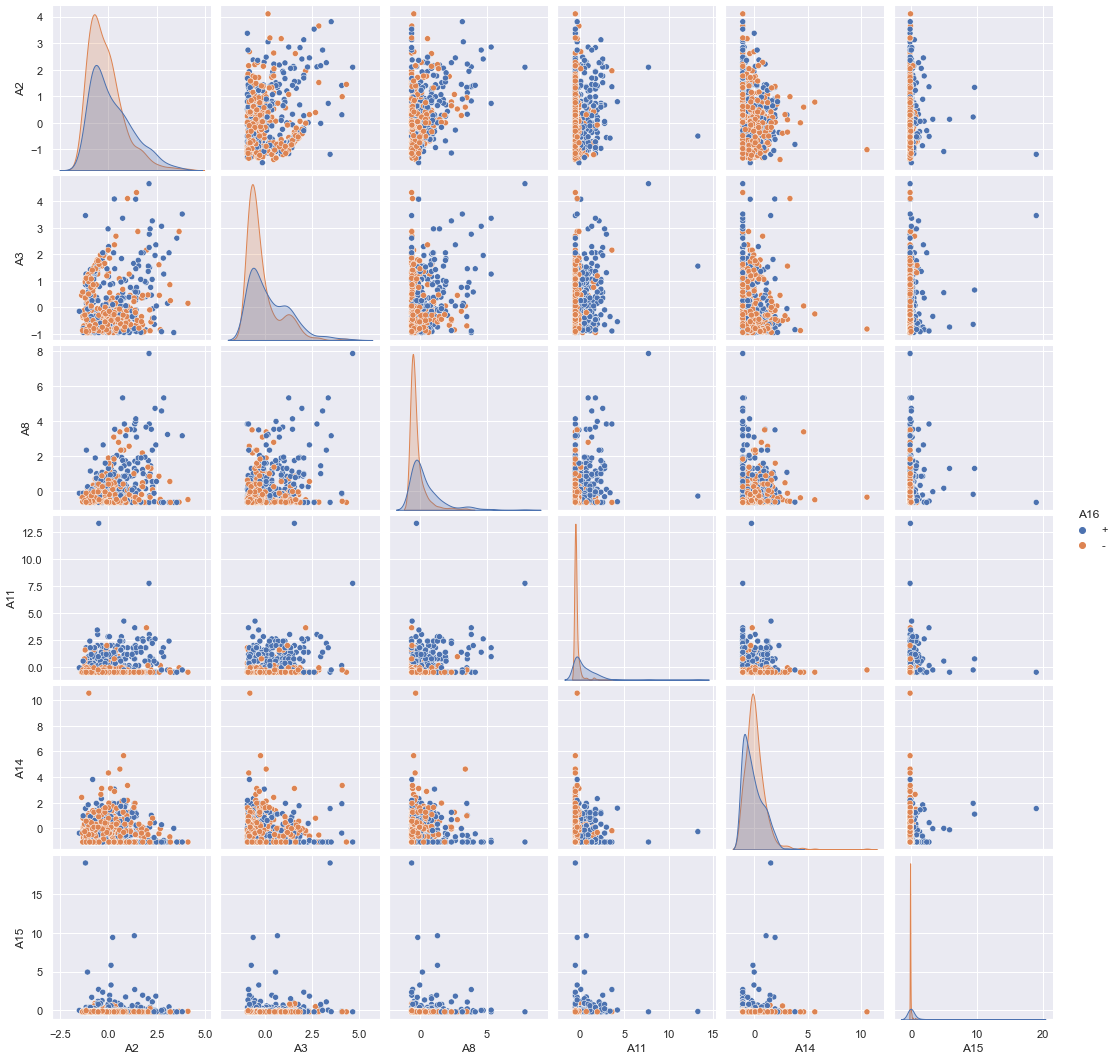


## 2)

در ادامه دیتاست را به دو دسته آموزش و آزمون تقسیم میکنیم (80% آموزش - 20% آزمون). در تصویر زیر تعداد داده های موجود در هر دسته را مشاهده میکنید.

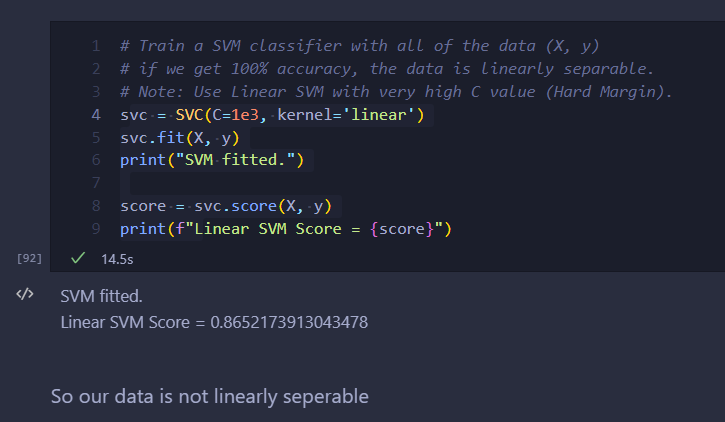
*نکته : دلیل اینکه دسته Validation را در این مرحله تولید نکردیم این است که API تنسورفلو پارامتری به اسم Validation\_split دارد که به طور خودکار بخشی از دسته آموزش را برای ارزیابی استفاده میکند.*

*پس از انجام جداسازی میتوانیم ویژگی های مختلف را دو به دو در ارتباط با هم رسم کنیم و مشاهده کنیم که آیا دو ویژگی وجود دارند که بتوانند به صورت خطی تمام داده ها را جدا کنند یا خیر؟*

**

*این نمودار اطلاعات زیادی به ما نمیدهد چون تا جایی که دیده میشود هیچ دو ویژگی ای وجود ندارند که به طور خطی داده هایی با برچسب های مختلف را از هم جدا کنند.*

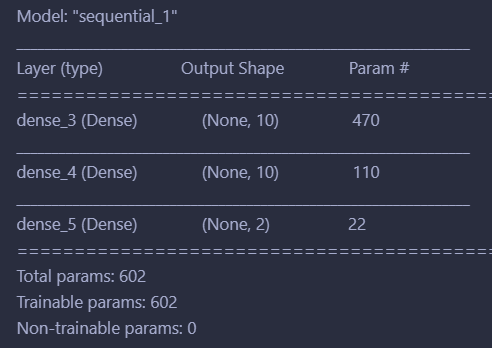
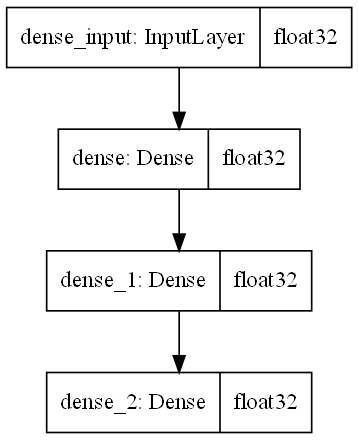
*در ادامه برای بررسی جداپذیری خطی دیتاست از Hard Margin Linear SVM استفاده میکنیم. به این شکل که تمام داده را به SVM میدهیم و انتظار داریم اگر داده ها جداپذیر خطی باشند، SVM با دقت 100 درصد آنها را دسته بندی کند. پس از انجام آزمایش متوجه میشویم که SVM به دقت حدود 80 درصد میرسد که به این معنی است که داده ها جداپذیر خطی نیستند.*

**

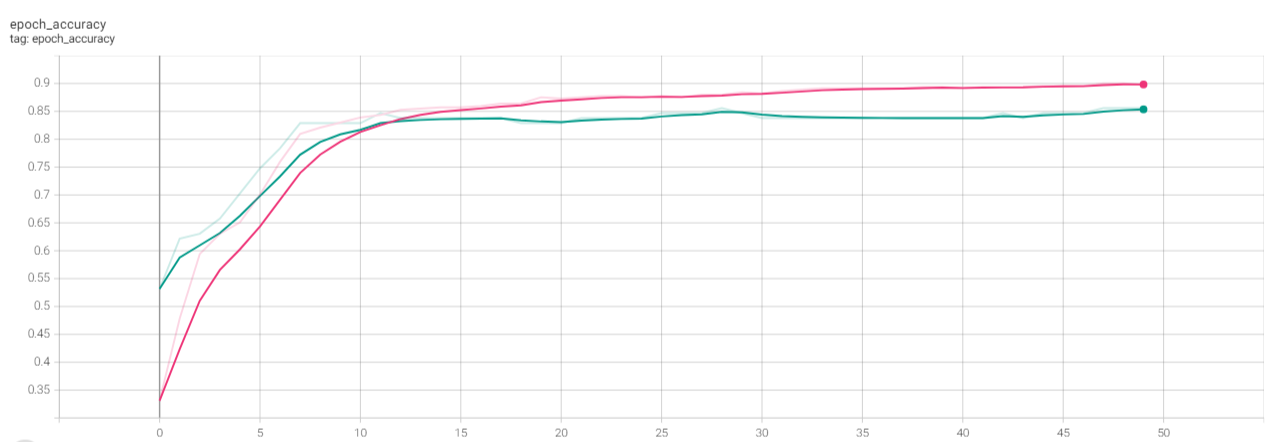
## 3)

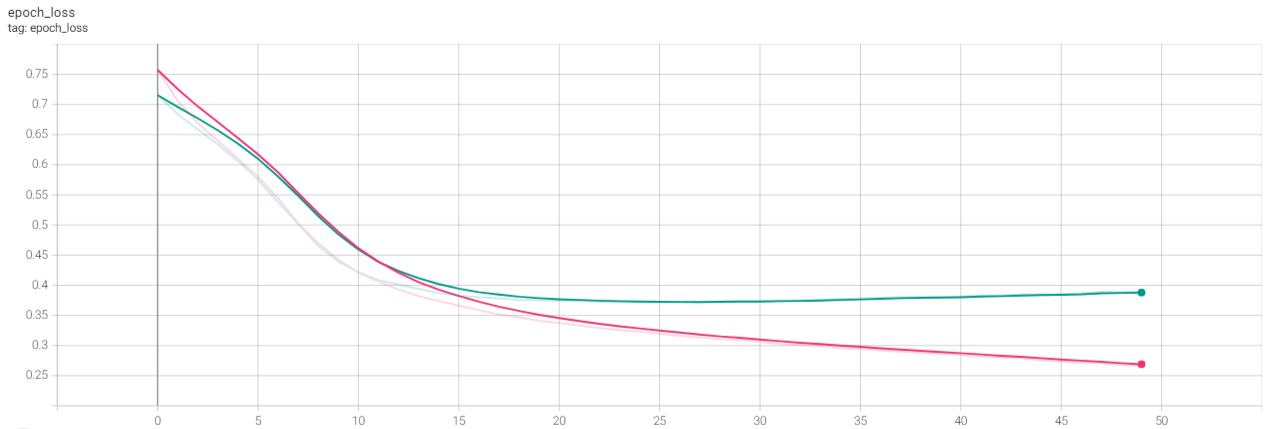
*در این بخش یک شبکه پرسپترونی چند لایه برای دسته بندی داده ها میسازیم. از آنجایی که متوجه شدیم داده ها به صورت خطی جدایی پذیر نیستند لازم است تعداد لایه های مخفی این شبکه را حداقل 2 در نظر بگیریم تا ارتباطات غیرخطی بین داده ها هم پیدا شوند.*

*در تصویر زیر میتوانید ساختار شبکه ذکر شده را مشاهده کنید. همانطور که مشخص است برای شروع، دو لایه مخفی از نوع Dense یا تماماً متصل داریم. همچنین به عنوان ورودی مدل به ازای هر کدام از 47 ویژگی یک نورون ورودی داریم. تعداد نورون های لایه مخفی در این مدل به خصوص برابر با 10 در نظر گرفته شده اند. تابع فعال سازی لایه های مخفی Relu و تابع فعال سازی لایه آخر Sigmoid است. برای آموزش از Adam به عنوان الگوریتم بهینه سازی و از تابع sparse categorical crossentropy به عنوان loss function استفاده شده است.*

**

*پس از اتمام آموزش این مدل، نمودار های صحت بر اساس epoch و loss بر اساس epoch را رسم کردیم. (نمودار قرمز آموزش و سبز اعتبارسنجی است)*

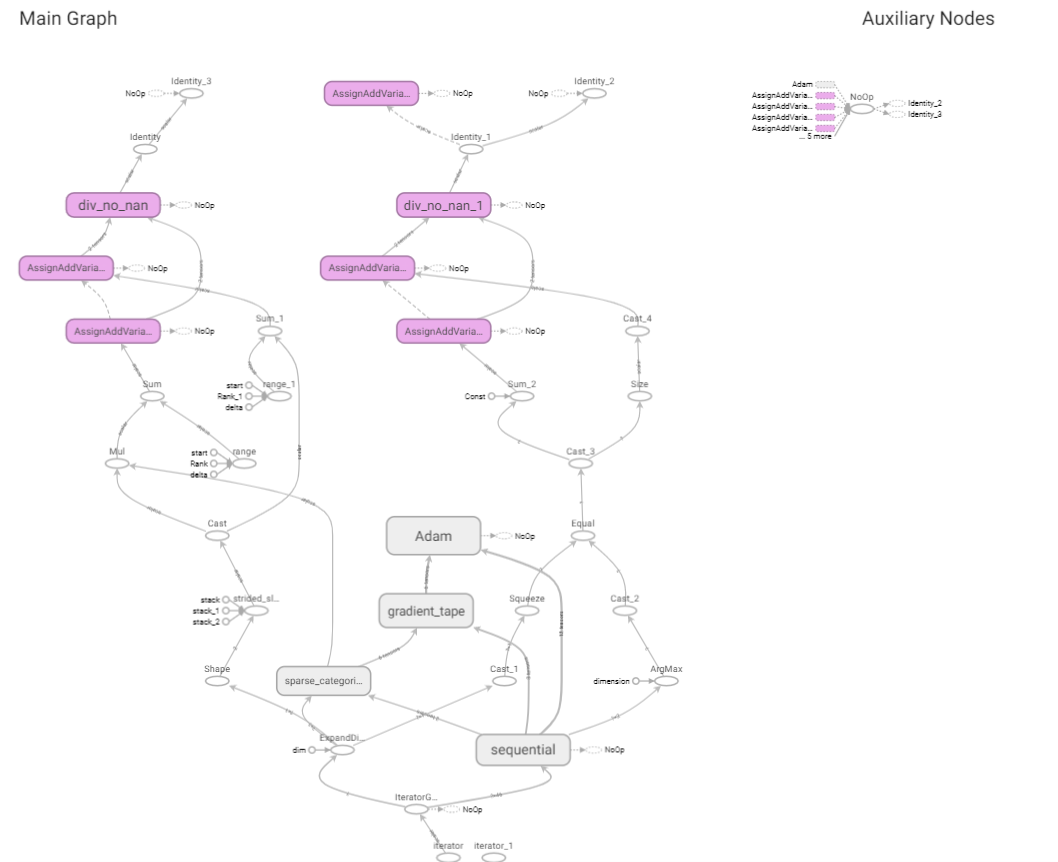
**

**

*همانطور که مشاهده میکنید نمودار epoch-accuracy صحت حدود 85 درصد پس از 50 epoch را نمایش میدهد اما اگر به نمودار epoch-loss توجه کنیم میبینیم که منحنی validation\_loss (سبز رنگ) پس از حدود 20 epoch دیگر کاهش نمی یابد در صورتی که training\_loss همچنان درحال کاهش است. این به این معنی است که مدل از اینجا به بعد در حال overfit شدن است.*

*در بخش های بعدی با استفاده از Hyper parameter Tuning و Regularization سعی میکنیم جواب بهتری از یک شبکه پرسپترونی چند لایه بگیریم.*

*گراف حاصل از این مدل را در تصویر زیر مشاهده میکنید. ( خروجی تب graph از Tensorboard)*

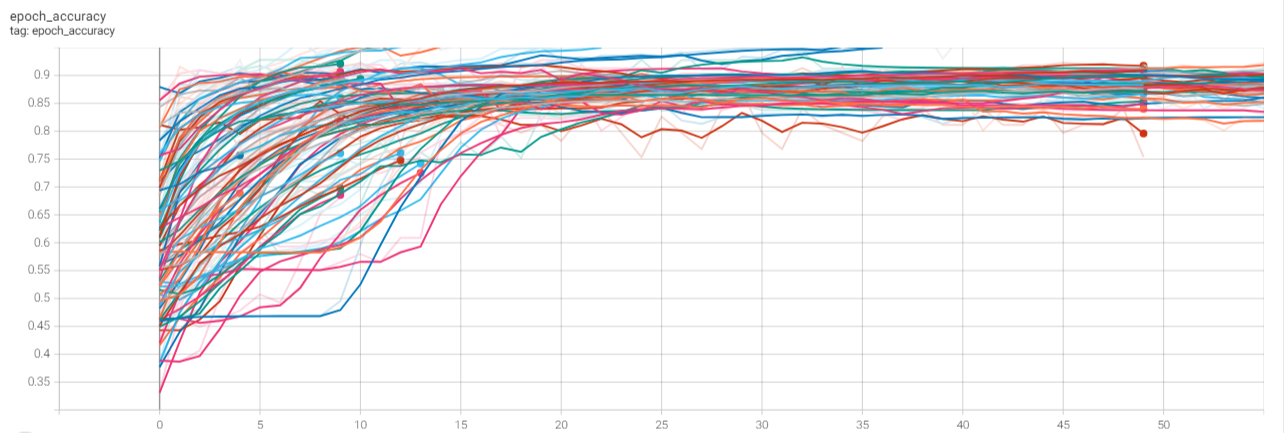
**

## 4)

*در این بخش از سوال میخواهیم سعی کنیم بهترین ابرپارامترها را برای مدل خود پیدا کنیم. برای این منظور تعدادی متغیر داریم:*

* *تعداد لایه های مخفی*
* *تعداد نورون های هر لایه مخفی*
* *تابع فعال سازی هر لایه*
* *و ...*

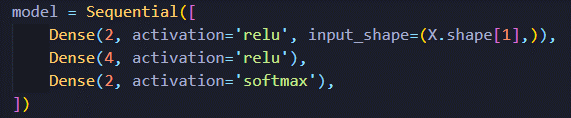
*پس از اجرای تعداد زیادی آزمایش با مدل های مختلف به یک سری نتایج رسیدیم.*

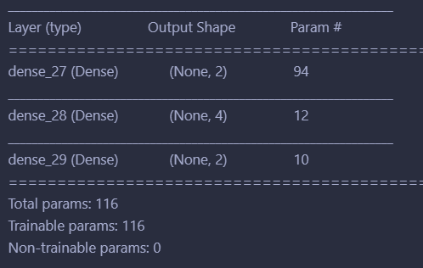
**

*اول اینکه با توجه به تعداد فیچری که داریم (47) و تعداد داده آموزش موجود (483)، مدل های بسیار عمیق و یا بسیار پهن خیلی زود overfit میشوند. به همین دلیل در انتخاب های نهایی بر روی مدل هایی با تعداد لایه کمتر و تعداد نورون های کمتر تمرکز کردیم. همچنین تغییر دادن تابع فعال سازی لایه ها تاثیر زیادی نداشت به همین دلیل برای سادگی مقایسه کم کردن تعداد حالت های ممکن برای همه لایه ها از Relu استفاده کردیم.*

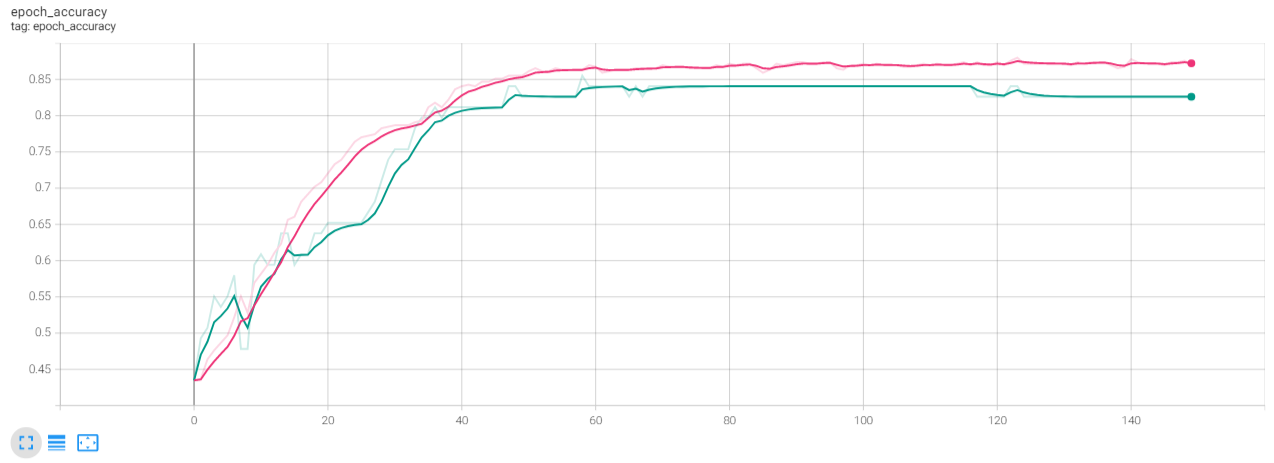
*همچنین در زمینه پیدا کردن بهترین ابرپارامترها از Keras Tuner نیز استفاده شد. (برای دیدن تمام آزمایش ها و نتایج مدل های مختلف به نوت بوک مراجعه کنید.*

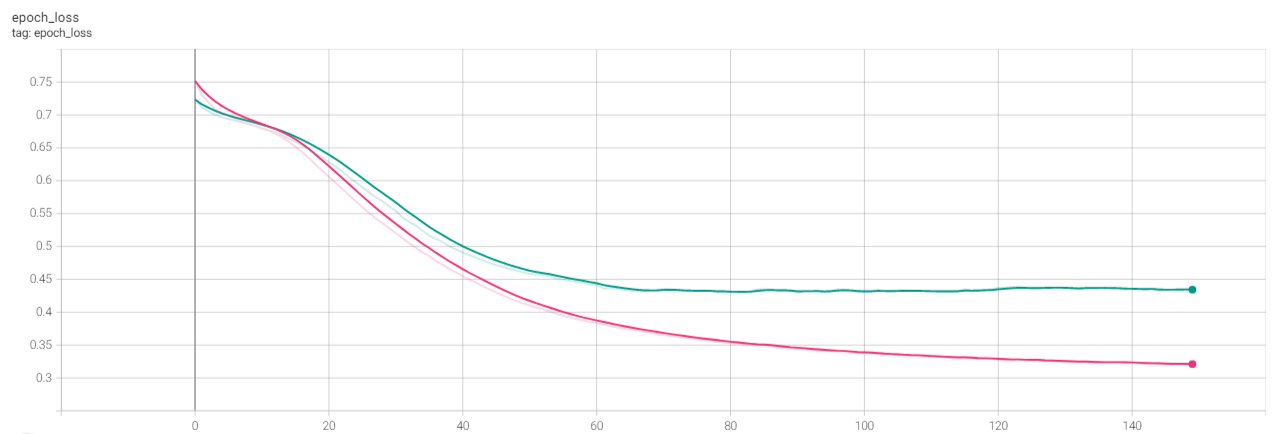
*در نهایت مدلی که در تصویر زیر میبینید به عنوان بهترین مدل انتخاب شد.*

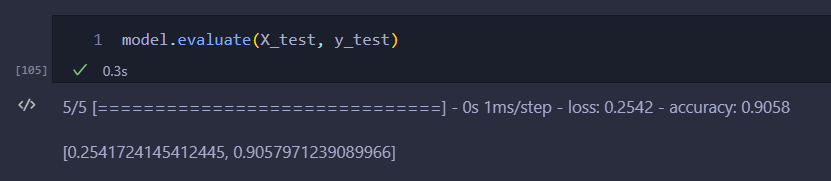
**

**

*نمودار های epoch-accuracy و epoch-loss را برای بهترین مدل میتوانید در تصاویر زیر مشاهده کنید. (نمودار قرمز آموزش و سبز اعتبارسنجی است)*

**

**

**

*صحت تست بر روی این مدل برابر با* ***90******درصد*** *بود.*

*برای مقایسه از XGBOOST برای دسته بندی داده ها استفاده کردیم و صحت 84 درصد را بدست آوردیم.*

*از جمله مشکلاتی که در این مسئله وجود داشت تعداد کم داده اعتبارسنجی بود. با توجه به اینکه 690 داده داشتیم، تعداد داده ها در دسته اعتبارسنجی برابر با 69 عدد بود که این باعث می‎شد بین ران های مختلف مقادیر مختلفی بدست بیاید و همچنین نتیجه validation\_acc, validation\_loss خیلی معتبر نباشند. این که در این مسئله از one hot encoding هم استفاده کردیم و تعداد ویژگی ها به 47 عدد رسید هم در این مشکل بی تاثیر نبود.*

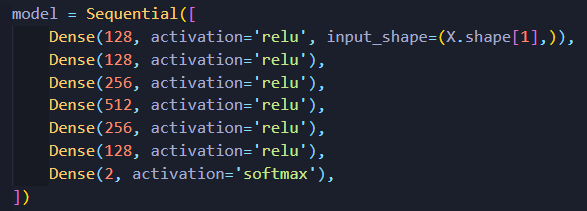
*اما با این وجود با مقایسه نمودار ها و نتیجه صحت تست بین این مدل و مدل بخش قبل میتوانیم ببینیم که مدل های کوچک تر برای این مسئله مناسب تر هستند.*

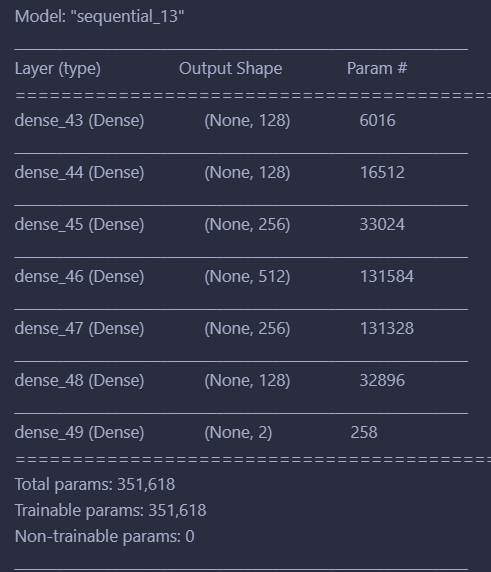
## 5)

*در این بخش سعی داریم مدل را به طور مصنوعی طوری تغییر دهیم که مدل Overfit شود. لازم است قبل از انجام این کار کمی به دلایل عمده Overfitting آشنایی داشته باشیم. یکی از دلایل عمده بیش برازش کمبود داده آموزش است. هر چه داده آموزش کمتر باشد مدل به راحتی بیشتر میتواند داده های آموزش را حفظ کند. در این سوال تصمیم گرفتیم به تعداد داده های آموزش دست نزنیم. یکی دیگر از دلایل Overfitting پیچیدگی زیاد مدل است. به گونه ای که مدل توانایی حفظ کردن تمام داده آموزش را داشته باشد. با افزایش تعداد لایه های مخفی و زیاد کردن تعداد نورون های هر لایه میتوانیم Capacity مدل را افزایش دهیم و عمل حفظ کردن داده آموزش را برای مدل راحت تر کنیم. انتظار داریم در این حالت به عدد های بسیار بالا در train\_acc برسیم و valid\_acc را تا حد زیادی کم کنیم. از آنجایی که تعداد داده اعتبارسنجی در این مسئله کم است ممکن است نتوانیم به خوبی این حفظ شدن داده آموزش را مشاهده کنیم و به طور شانسی دقت های بالایی بگیریم ولی با داشتن داده اعتبار و تست بیشتر باید کاهش چشم گیری در صحت اعتبارسنجی و تست مشاهده کنیم.*

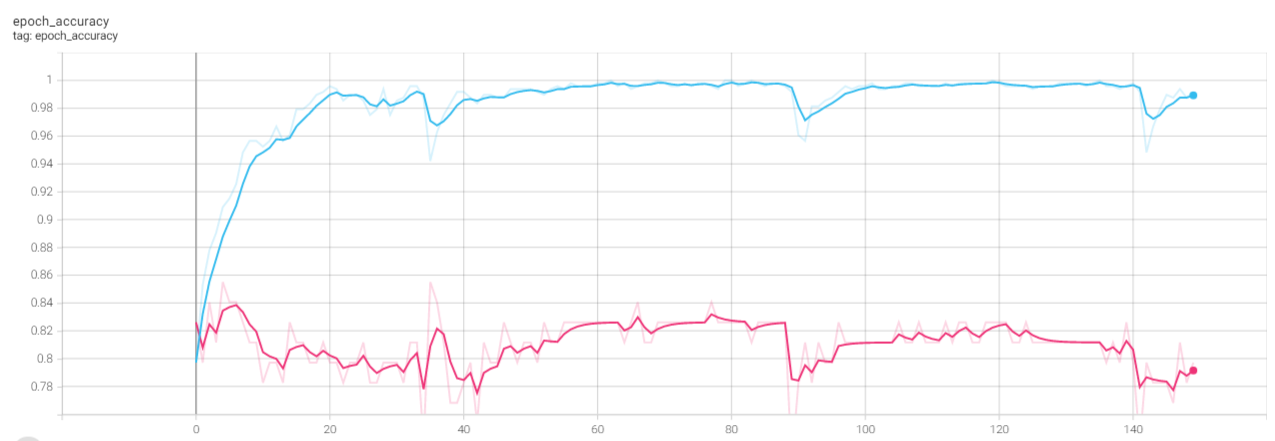
*ساختار مدل را به شکل زیر تغییر میدهیم.*

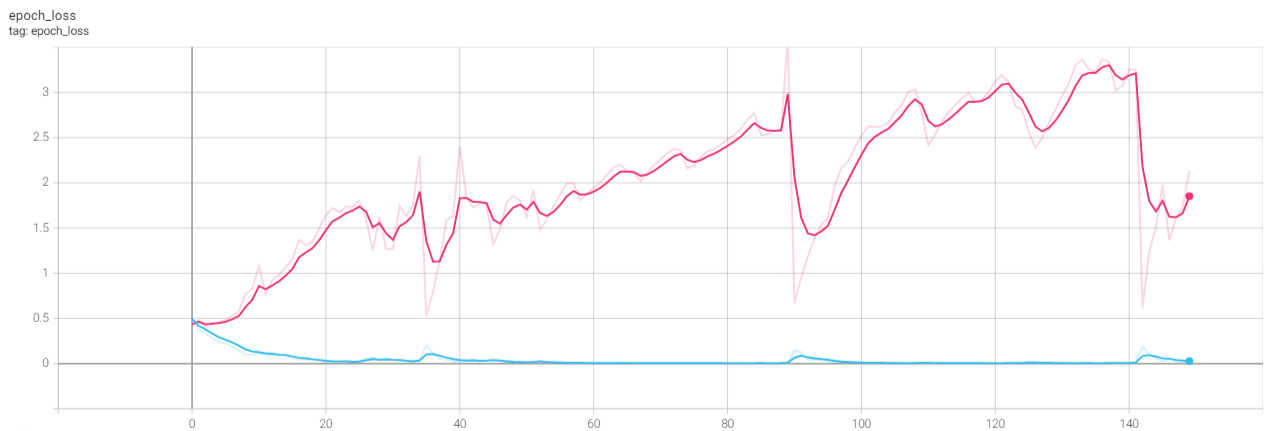
*همانطور که میبینید هم تعداد لایه ها و هم تعداد نورون های هر لایه را به شکل چشم گیری افزایش دادیم.*

**

**

*در نمودار های مربوطه هم به راحتی میتوان Overfitting را مشاهده کرد. صحت آموزش خیلی زود به اعدادی نزدیک به 100 درصد رسیده و لاس مربوط به آموزش هم بسیار به صفر نزدیک شده است. (نمودار آبی آموزش و قرمز اعتبارسنجی است)*

**

**

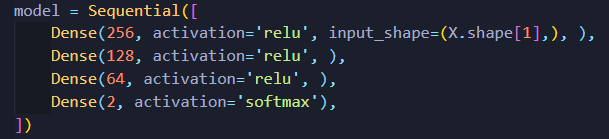
*تعداد وزن های قابل تنظیم این مدل برابر حدود 350 هزار عدد است.*

## 6)

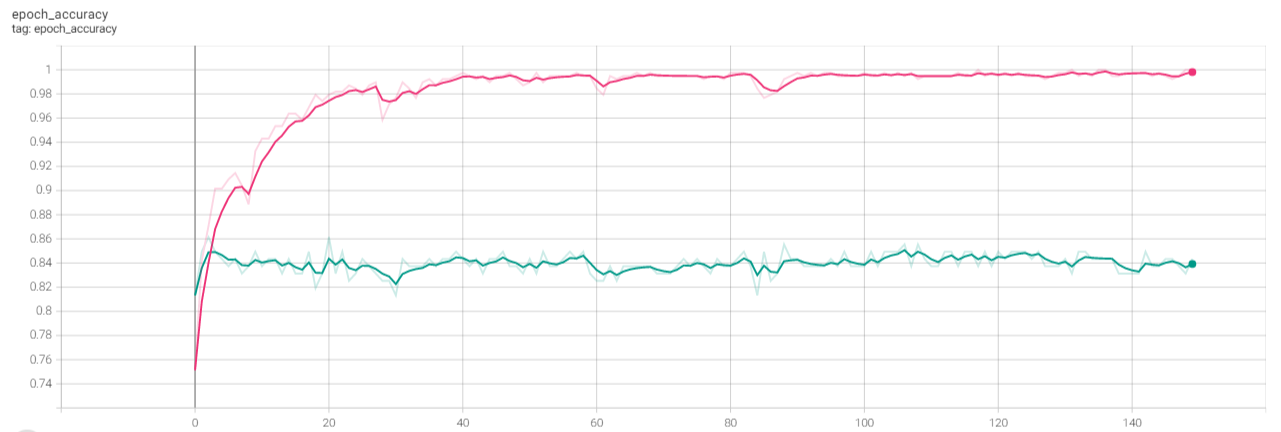
*در این بخش سعی داریم با استفاده از روش های مختلفی مانند استفاده از L1/L2 Regularization و Dropout مدل بخش قبل را بهبود ببخشیم و از حالت Overfit در بیاوریم.*

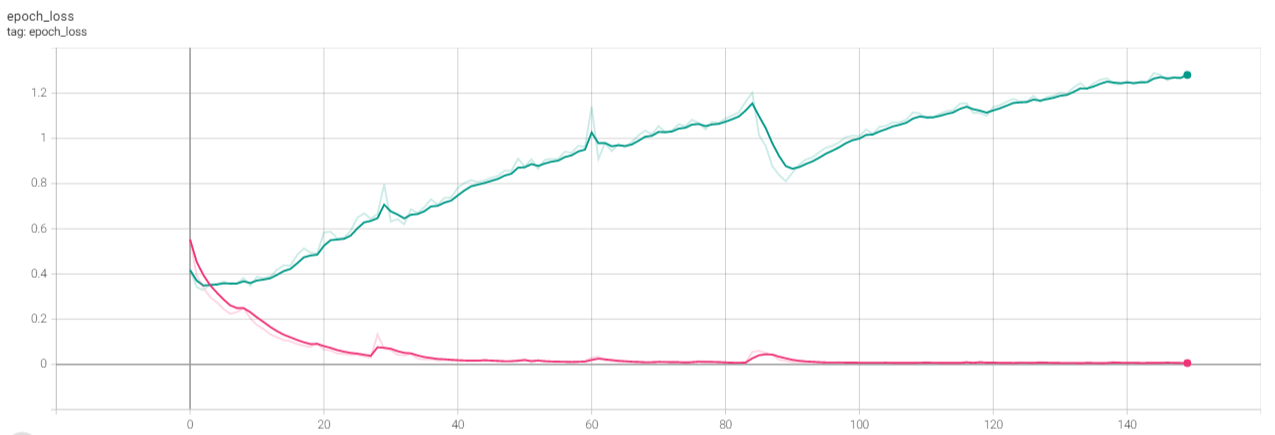
*از بین این دو روش Regularization کمک میکند عدد متناظر با وزن ها خیلی زیاد نشوند. و در واقع بزرگی وزن ها را به عنوان یک جریمه به loss function اضافه میکند و باعث میشود مدل وزن های بسیار زیادی نداشته باشد. این کار باعث میشود با وجود اینکه تعداد نورون های زیادی داریم اما تاثیر هر نورون کم تر باشد. اگر از L1 استفاده کنیم حتی باعث میشود بعضی وزن ها صفر شوند که باعث میشود پیچیدگی مدل کاهش پیدا کند.*

*و Dropout باعث میشود در حین آموزش یک سری از نورون ها خاموش شوند و عملا پیچیدگی مدل را کاهش میدهد. دلیل این که این کار مفید است این است که وابستگی نورون ها به یکدیگر را از بین میبرد و باعث میشود هر نورون اطلاعات مفید و یکتایی را مستقل از اطلاعات سایر نورون ها (ی مجاورش) بیاموزد.*

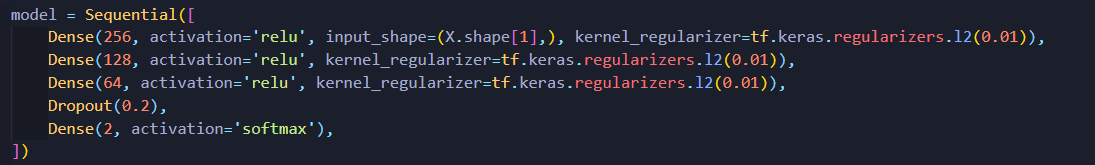
**

*(****پیش از تعمیم پذیری****: نمودار قرمز آموزش و سبز اعتبارسنجی است)*

**

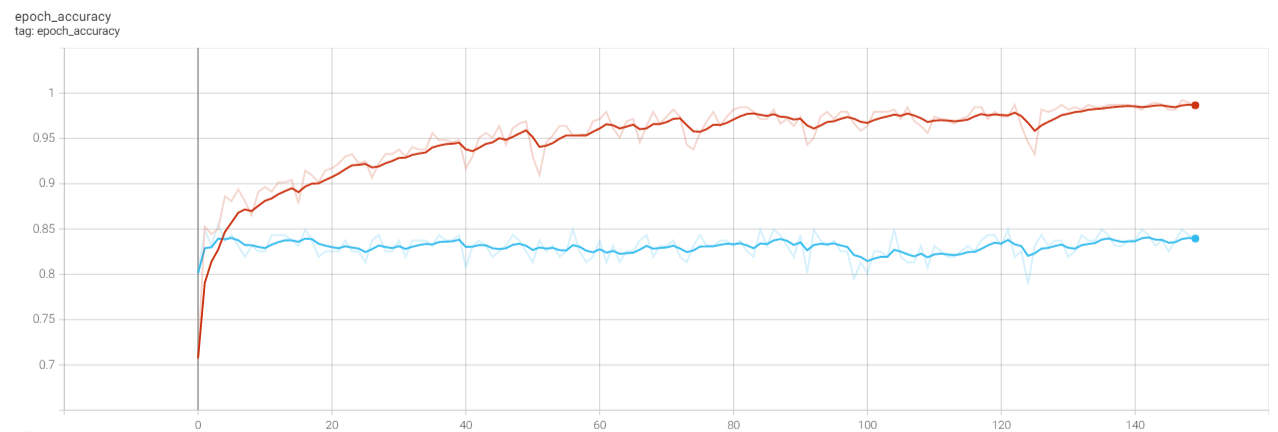
**

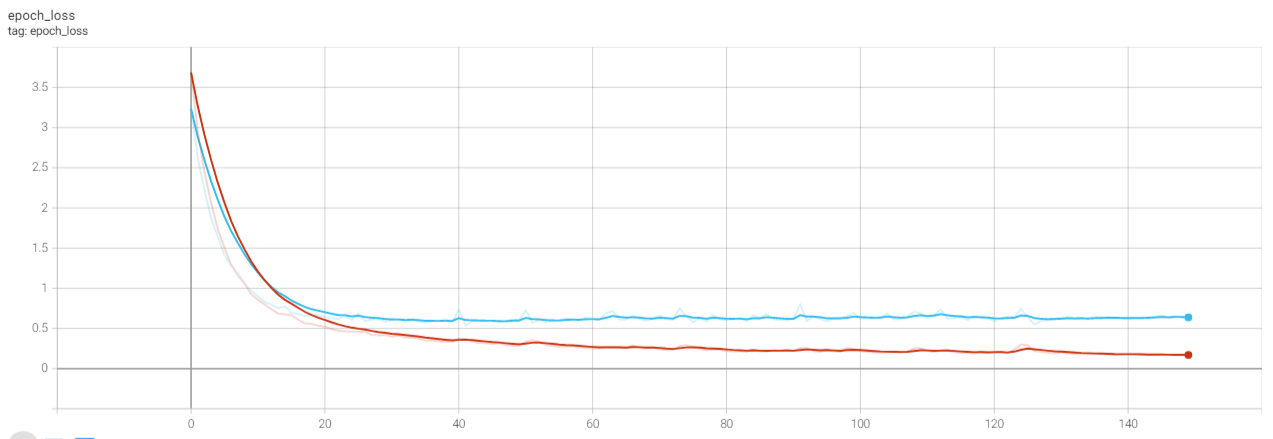
*ساختار مدل پس از اعمال Regularization و Dropout را مشاهده میکنید.*

**

*در تصاویر زیر میتوانید نمودار های epoch-loss و epoch-accuracy را بعد از اعمال Regularization , Dropout مشاهده کنید.*

*(****پس از تعمیم پذیری****: نمودار نارنجی آموزش و آبی اعتبارسنجی است)*

**

**

***پایان***