|  |  |
| --- | --- |
| نام و نام خانوادگی | **پویا شیخ الاسلامی** |
| شماره دانشجویی | **810100394** |
| تاریخ ارسال گزارش | **۱۴۰2.۰3.۰3** |

|  |  |  |
| --- | --- | --- |
|  | **به نام خدا**  **دانشگاه تهران**  **دانشکده‌ مهندسی برق و کامپیوتر** |  |
| **درس شبکه‌های عصبی و یادگیری عمیق**  **تمرین چهارم** | | |

**فهرست**

[**پاسخ 1**. **عنوان پرسش اول به فارسی** 1](#_Toc115453335)

[۱-۱. عنوان بخش اول 1](#_Toc115453336)

[**پاسخ ۲** **- عنوان پرسش دوم به فارسی** 2](#_Toc115453337)

[۱-۲. **عنوان بخش اول** 2](#_Toc115453338)

[**پاسخ ۳** **– عنوان پرسش سوم به فارسی** 3](#_Toc115453339)

[۱-۳. **عنوان بخش اول** 3](#_Toc115453340)

[**پاسخ ۴** **– عنوان پرسش چهارم به فارسی** 4](#_Toc115453341)

[۱-۴. **عنوان بخش اول** 4](#_Toc115453342)

**شکل‌ها**

شکل 1: تصویر پوشه دادگان 1

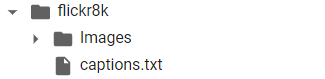
**جدول‌ها**

[جدول 1. عنوان جدول نمونه 1](#_Toc115453324)

# **پاسخ 1**. **توصیف عکس**

۱-۱. مجموعه دادگان و پیش پردازش آن ها

با توجه به محدودیت های سخت افزاری و امکانات موجود از محیط google colab برای آموزش و اجرای کد ها استفاده شده است. برای دانلود کردن دیتاست از روی سایت Kaggle به جای دانلود کردن دیتاست و آپلود آن به صورت دستی، این کار به صورت اتوماتیک در کد برنامه قرار داده شد. بدین صورت که یک فایل json از مشخصات حساب kaggle به گوگل داده شده و با توجه به آن و استفاده از API سایت، دیتاست دانلود و در پوشه مناسب خود قرار داده می‌شود. در نهایت فایل های دانلود شده به شکل زیر خواهد بود:



شکل 1: تصویر پوشه دادگان

در این پوشه یک فایل به نام captions.txt موجود است که در آن 5 جمله به ازای هر یک از عکس های موجود در دادگان موجود است. در پوشه Images نیز تمامی عکس ها موجود هستند.

در کلاس Vocabulary که برای تبدیل جملات به توکن تولید شده روند طی شده به این صورت است:

برای تبدیل جملات به توکن های مختلف از کتابخانه SPACY استفاده شده است و جمله در آن به توکن های مختلف تبدیل شده و در نهایت به حروف کوچک تبدیل شده و پس از آن دیکشنری های تبدیل کلمه به ایندکس و ایندکس به کلمه ساخته می‌شوند. در نهایت تابع numericalize برای تبدیل یک جمله به عدد ها، در آن تعریف شده است.

در کلاس FlickrDataset که برای خواندن و پردازش دادگان تعریف شده است، ابتدا داده ها خوانده، سپس دیتافریم آن ساخته شده و برای هرکدام از عکس ها، جمله متناظر آن ذخیره می‌شود.

در تابع get\_loader داده های مورد نیاز برای آموزش دادن شبکه عصبی ساخته و اماده می‌شوند.

2-۱. مدل شبکه

مدل شبکه عصبی همانطور که در داکیومنت توضیح داده شد از دو قسمت شبکه ResNet18 و LSTM تشکیل شده است که بخش اول برای استخراج ویژگی از تصویر استفاده می‌شود و بخش دوم برای ترکیب ویژگی های تصویر با ویژگی های متن توصیف تصویر و یادگیری آن استفاده می‌شود.

پس از آموزش دادن شبکه با استفاده این روش دقت کسب شده برای تصویر به اندازه کافی مناسب نبود و پس از 30 ایپاک آموزش دادن شبکه، متن تولید شده همچنان دارای دقت بالایی نبود. مقدار loss شبکه در این حالت در 10 ایپاک آخر به شکل زیر کاهش پیدا کرد(شروع خطای شبکه از 3.14 بود):

شکل 2: میزان خطای شبکه در 10 ایپاک اخر

و در نهایت نمونه خروجی آن به صورت زیر بود:



شکل 3: نمونه خروجی مدل اول

برای بهبود دقت شبکه طبق مطالعات انجام شده از شبکه های attention در بین لایه های مدل استفاده شد تا دقت را بهبود دهد.

در کد پیاده سازی شده در قسمت EncoderCNN یک مدل ResNet18 بدون در نظر گرفتن لایه های تماما متصل انتهایی آن برای انکود کردن ویژگی های تصویر استفاده شده است و لایه های آن در دو حالت freeze و unfreeze شده اند.

در قسمت دوم پیاده سازی مدل، بخش شبکه attention اضافه شده است. این قسمت به ترتیب شامل لایه های زیر است:

* انکودر
* دیکدر
* خطی ساز
* ReLU
* Softmax

و در نهایت قسمت نهایی LSTM که با دریافت پارامتر های ورودی توسط کاربر به شکل زیر ساخته شده است:

attention\_dim = 700

embed\_dim = 700

decoder\_dim = 700

dropout = 0.5

vocab\_size = len(dataset.vocab)

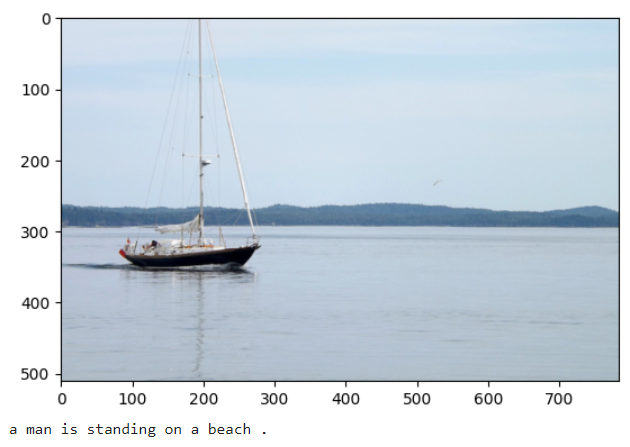
learning\_rate = 1e-03

num\_epochs = 2

3-۱. آموزش شبکه

برای آموزش شبکه در google colab به دلیل محدودیت زمانی امکان آموزش دادن شبکه بیش از 30 ایپاک موجود نبود. آموزش به صورت دوره هایی با 2 ایپاک انجام می‌گرفت و هربار خروجی شبکه بررسی می‌شد. ولی شبکه تا 30 ایپاک خروجی هایی به شکل زیر داشت:

شکل 4: خطای شبکه



شکل 5: نمونه تصاویر خروجی شبکه و توضیحات آن ها

خروجی های بالا در صورتی بودند که وزن های شبکه ResNet18 ثابت باشند. در صورت اضافه شدن پارامتر های این شبکه به پارامتر های قابل آموزش، میزان خطای شبکه و خروجی به شکل زیر خواهد بود:

.

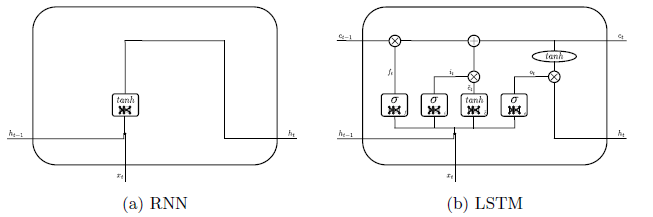
# **پاسخ ۲** **– تشخیص اندیشه**

## ۱-۲. **معماری** LSTM **و** embedding

شبکه های عصبی قدیمی (ff) نمیتوانند ترتیب زمانی که ورودی بر اساس آن اتفاق افتاده است را برون یابی کند. ایده در نظر گرفتن بلوک های کلمات ورودی به عنوان مستقل از یکدیگر در چارچوب NLP در واقع، بسیار محدود است. درست مانند فرآیند خواندن برای یک انسان منجر به حفظ تمام کلمات متن نمی شود، بلکه منجر به استخراج از میان مفاهیم بنیادی بیان شده، می شود. پس ما به یک «حافظه» نیاز داریم که فقط به در نظر گرفتن صریح ورودی های قبلی، محدود نباشد.

Translation is too long to be saved

این به طور طبیعی منجر به پیدایش شبکه های عصبی بازگشتی (RNN) شده است که، با معرفی مفهوم "وضعیت داخلی" (به دست آمده بر اساس قبلی ورودی‌ها)، در حال حاضر نوید های به خصوص در فیلد NLP نشان داده‌اند. RNN ها حاوی حلقه هایی در داخل آنها هستند که به اطلاعات قبلی اجازه عبور از مراحل مختلف تحلیل می دهد. نمای صریح از سیستم بازگشتی می تواند با باز کردن فعال‌سازی‌های شبکه با در نظر گرفتن زیرشبکه‌های مجزا به عنوان کپی های(در هر مرحله به روز می شوند) همان شبکه، به ‌دست آید.



1-شمای کلی شبکه RNN و LSTM

Translation is too long to be saved

در تئوری، RNN ها باید بتوانند اطلاعاتی را که الگوریتم یادگیری در یک دنباله آموزشی پیدا می کند در حالت های خود حفظ کنند. متأسفانه در انتشار گرادیان به سمت عقب در طول زمان، اثر مقادیر خروجی مورد نظر که به کمک تابع هزینه می آید می تواند آنقدر کوچک شود که تنها پس از چند مرحله، دیگر سهم کافی در یادگیری پارامتر نداشته باشد(ناپدید شدن شیب). این مشکل، به این معنی است که RNNs فقط یک حافظه کوتاه مدت را حفظ می کند ، زیرا قسمت‌هایی که از نظر زمانی دورترند ،کم اهمیت ترند. این باعث می شود RNN ها فقط برای دنباله های بسیار کوتاه مفید باشند.

Translation is too long to be saved

برای غلبه بر (حداقل تا حدی) مشکل حافظه کوتاه مدت، معماری های حافظه (LSTM) معرفی شدند. برخلاف یک شبکه کلاسیک RNN ، LSTM ساختار تک سلولی بسیار پیچیده تری دارد، که در آن یک شبکه عصبی با چهار شبکه جایگزین می شود که با یکدیگر تعامل دارند. با این حال، عنصر متمایز LSTM حالت سلول c است که به اطلاعات اجازه می دهد برای حرکت در طول زنجیره از طریق عملیات خطی ساده، منتقل شود. اضافه یا حذف اطلاعات توسط سه ساختار به نام "دروازه" تنظیم می شود.

Translation is too long to be saved

بدیهی است که صرف نظر از نوع شبکه عصبی مورد استفاده، به دلیل سروکار داشتن با متن، در زبان طبیعی، تعریف نوع embedding مورد استفاده ضروری است. در واقع بر خلاف زبان‌های رسمی که کاملاً مشخص هستند، زبان طبیعی از نیاز ساده برای ارتباط سرچشمه می‌گیرد ، و بنابراین حامل تعداد زیادی از ابهامات. برای اینکه بتوانیم حداقل سعی کنیم آن را درک کنیم، لازم است که نوعی "نزدیک معنایی" بین اصطلاحات مختلف، که تک کلمات را به بردارهایی با مقادیر واقعی در یک فضای مناسب تبدیل میکند، مشخص شود.

Save translation

بنابراین embedding حاصل می تواند تک کلمات را در یک نمایش عددی ترسیم کند که "معنای خود را حفظ می کند" و آن را به معنای خاص "قابل درکی"، برای کامپیوتر تبدیل می کند.

Translation is too long to be saved

امروزه راه های مختلفی برای به دست آوردن این فضای معنایی وجود دارد که عموماً شناخته شده است به عنوان تکنیک های Word Embeddings که هر کدام ویژگی های خاص خود را دارند.

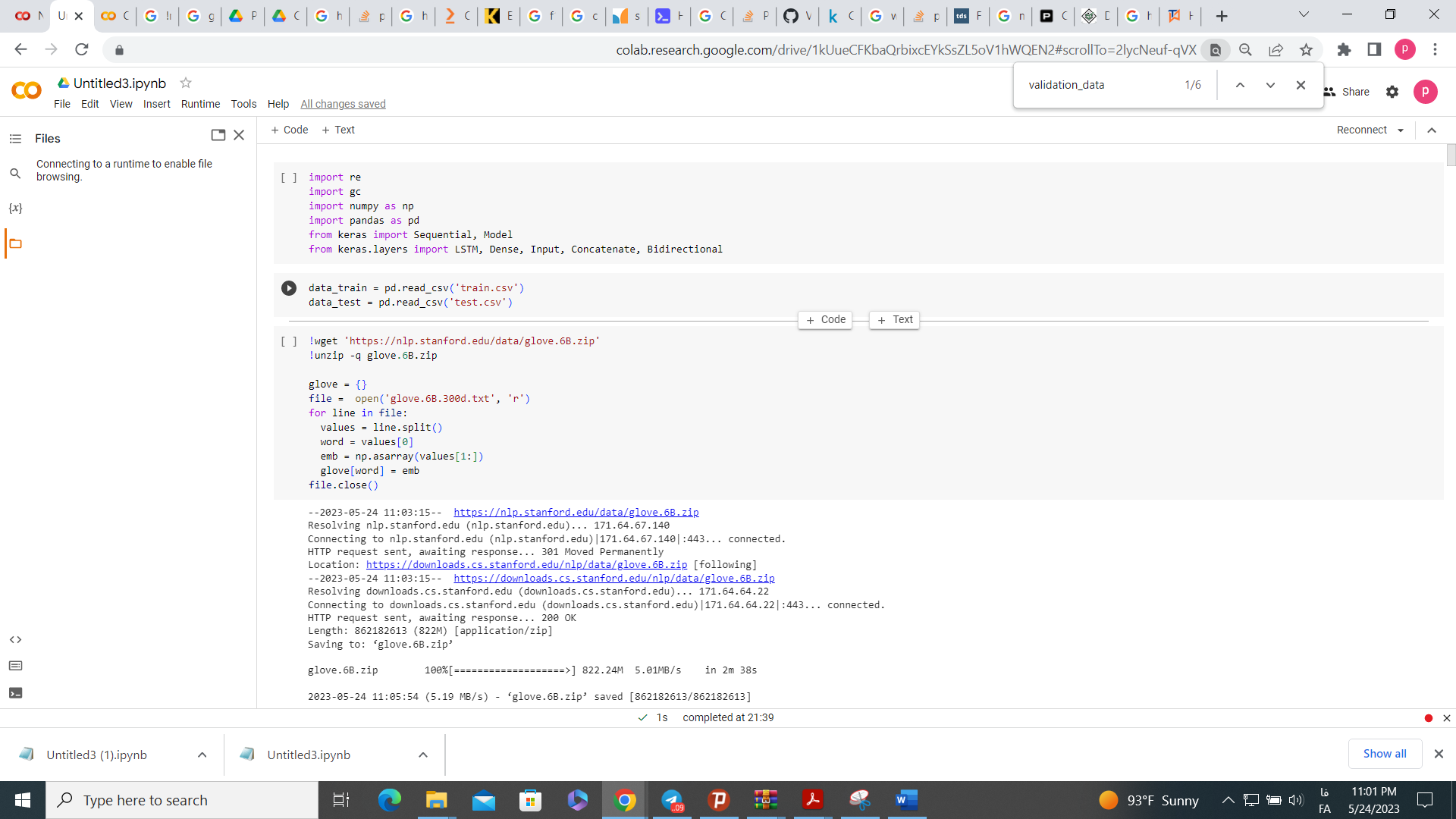
Translation is too long to be saved

یکی از محدودیت های اصلی Word Embeddings این است که (مدل های فضای برداری کلمه به طور کلی) که کلمات با معانی متعدد در یک نمایش واحد (بردار واحد در فضای معنایی) ترکیب می شوند. به عبارت دیگر، چند معنایی و همنامی به درستی مورد بررسی قرار نمی گیرند. به عنوان مثال، در جمله "باشگاهی که دیروز امتحان کردم عالی بود!"، مشخص نیست که آیا اصطلاح باشگاه به معنای کلمه ساندویچ باشگاهی، باشگاه بیسبال، باشگاه، باشگاه گلف یا هر معنای دیگری مربوط به آن باشگاه است. ممکن است داشته باشد.

در روش GloVe embedding از ان جایی که علاوه بر آمارهای محلی (کانتکست اطلاعات محلی کلمات) از آمارهای گلوبال (همزمانی کلمات-کلمات با چند معنی) برای به دست آوردن بردارهای کلمه استفاده میکند میتواند در این زمینه به ما کمک کند.

## 2-2. **پیش پردازش**

همان طور که میدانیم برای حل مشکل کلمات با معانی متعدد از آمارهای گلوبال در پردازش زبان طبیعی استفاده میشود که به این معناست که از معانی مختلف یک کلمه چندین بردار عددی میسازیم یکی از متدهایی که در روش glove نیز از ان استفاده شده است محول کردن توکن به این واژه ها برای یافتن طبقه بندی های متفاوت(معانی مختلف) است. که مطابق با چیزی که خواسته شده است به پیاده سازی پرداختیم.

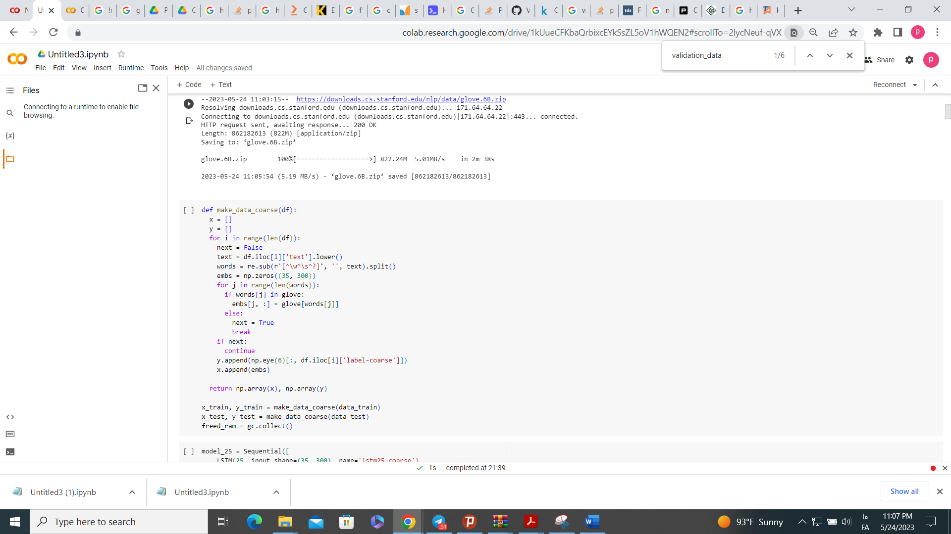


2-پیش پردازش

## 3-2.پیاده سازی طبقه بندی نیت

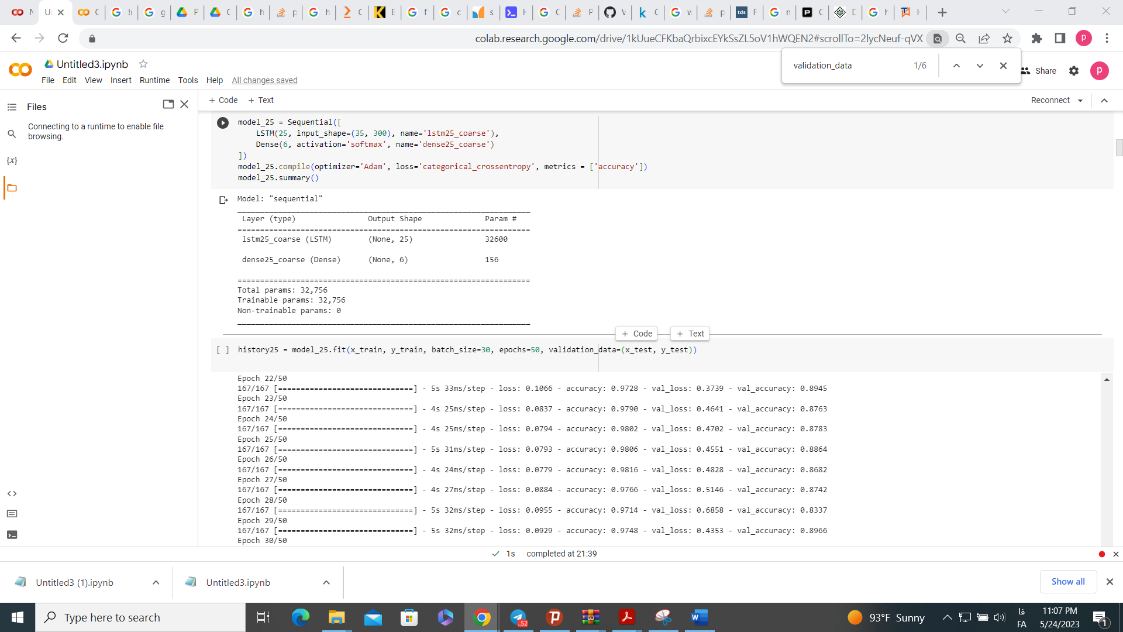
مطابق آنچه گفته شد دو مدل معرفی شده وجود دارد که اولی فقط یک سطح(اصلی-coarse) و دومی دو سطح(فرعی-fine) را بررسی میکند در ادامه به بررس گذرای که و نتایج خواسته شده صورت سوال پرداخته میشود.

در ابتدا مدل تک سطحی که فقط سطح اصلی در ان بررسی میشود پیاده سازی شده است



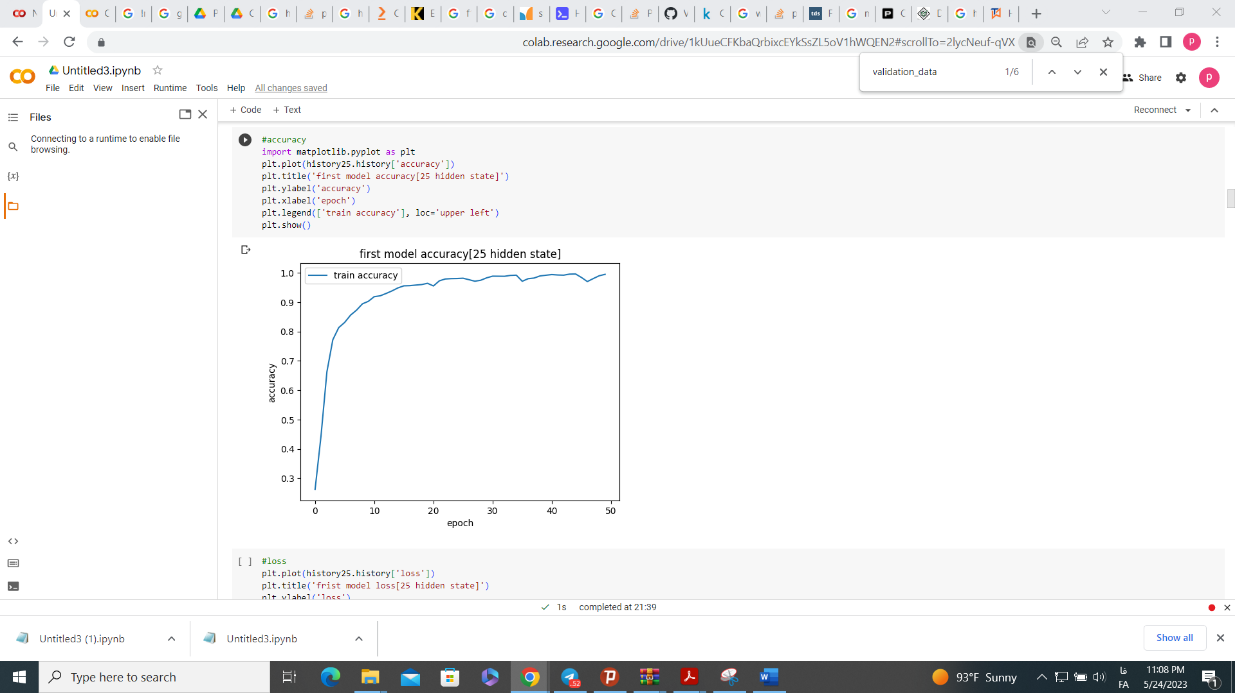
3-پیاده سازی تک سطح- اصلی

سپس در ابتدا مدل با 25 لایه پنهان پیاده سازی شده است

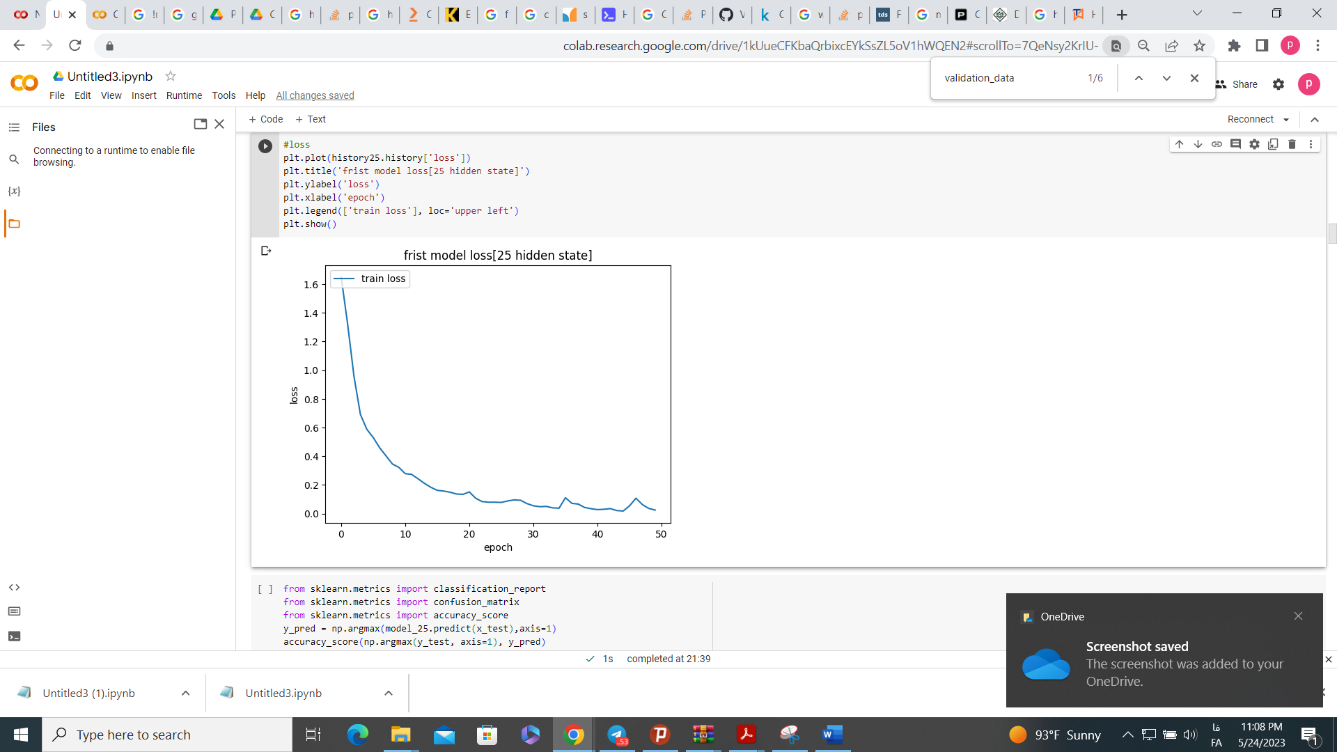


4-مدل اول-25لایه

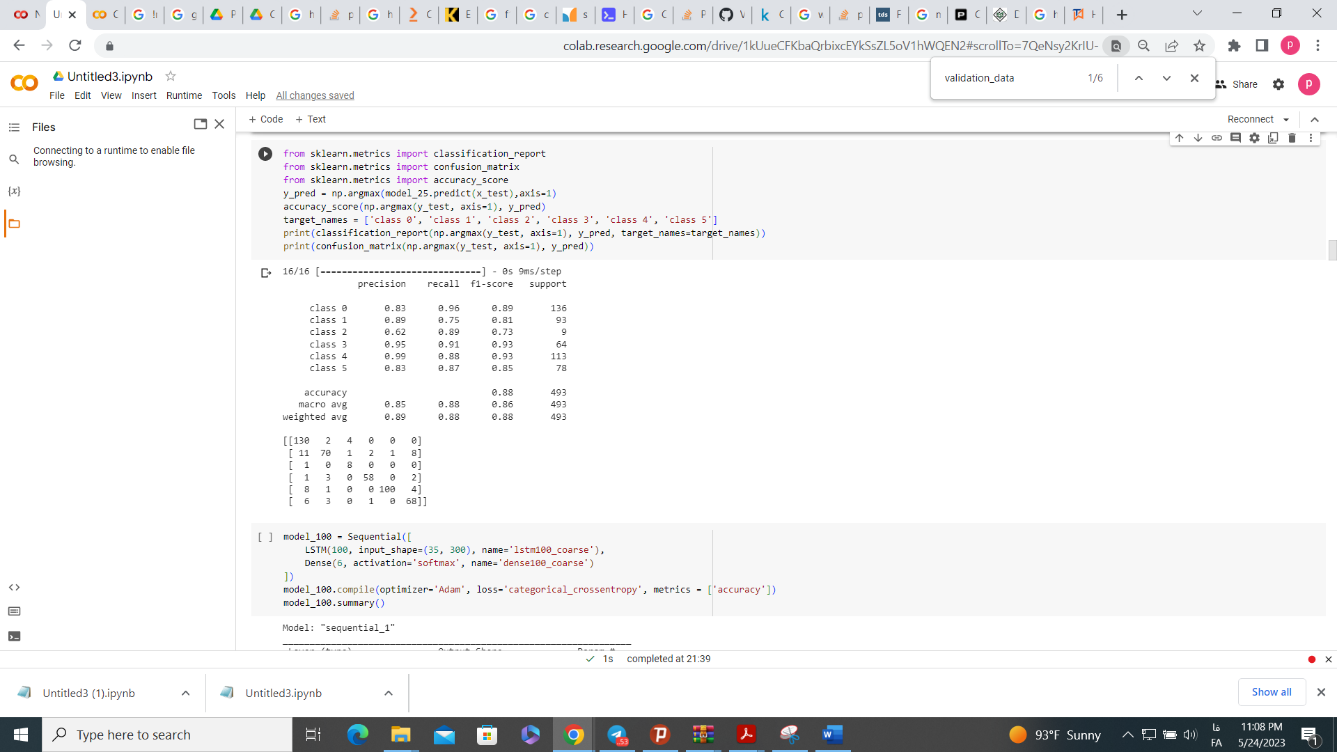
در ادامه گراف های accuracy و loss و در پایان classification-repot به همراه ماتریس آشفتگی آورده شده است.



5تابع accuracy مدل اول- 25 لایه

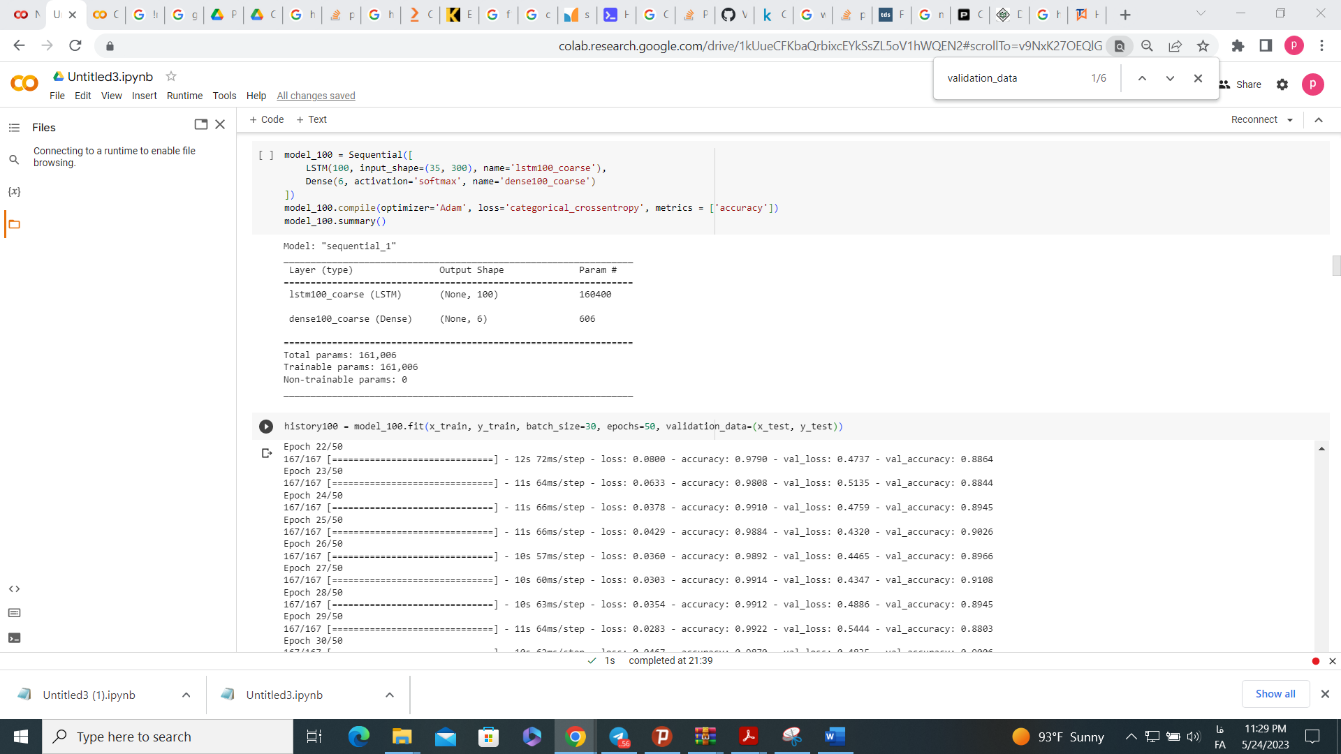


6-تابع loss مدل اولی-25 لایه



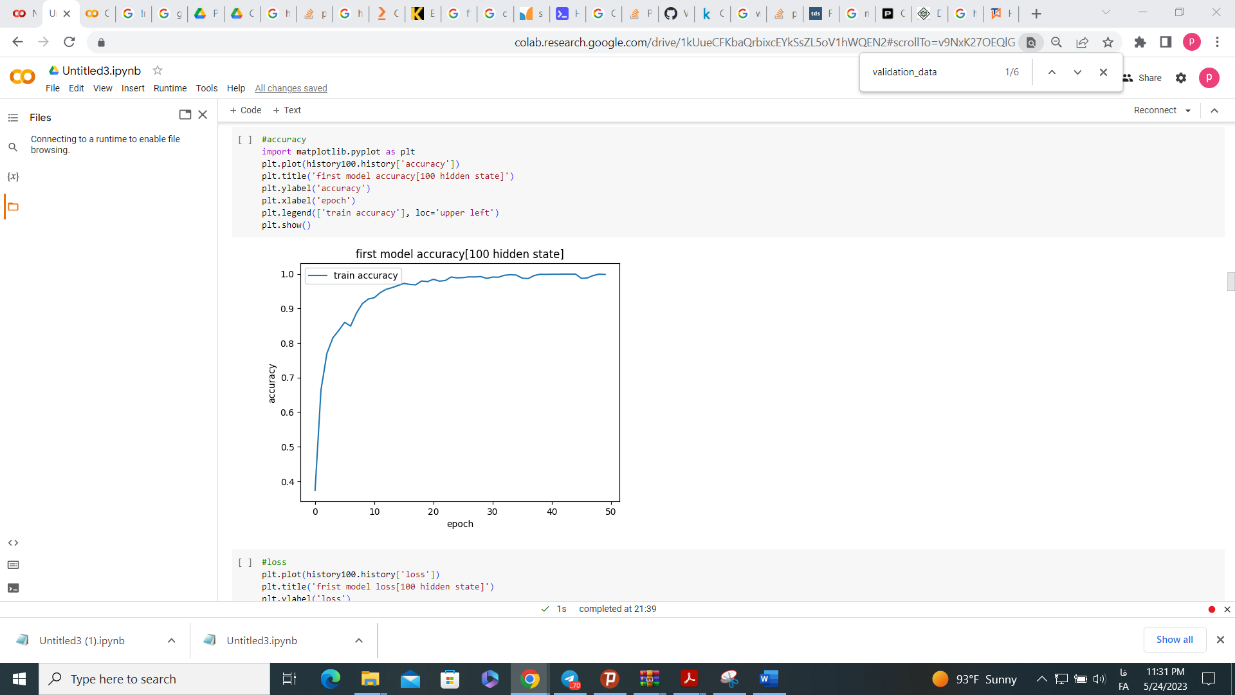
7-گزارش طبقه بند و ماتریس آشفتگی مدل اول-25لایه

سپس به پیاده سازی 100 لایه پرداخته شده است.

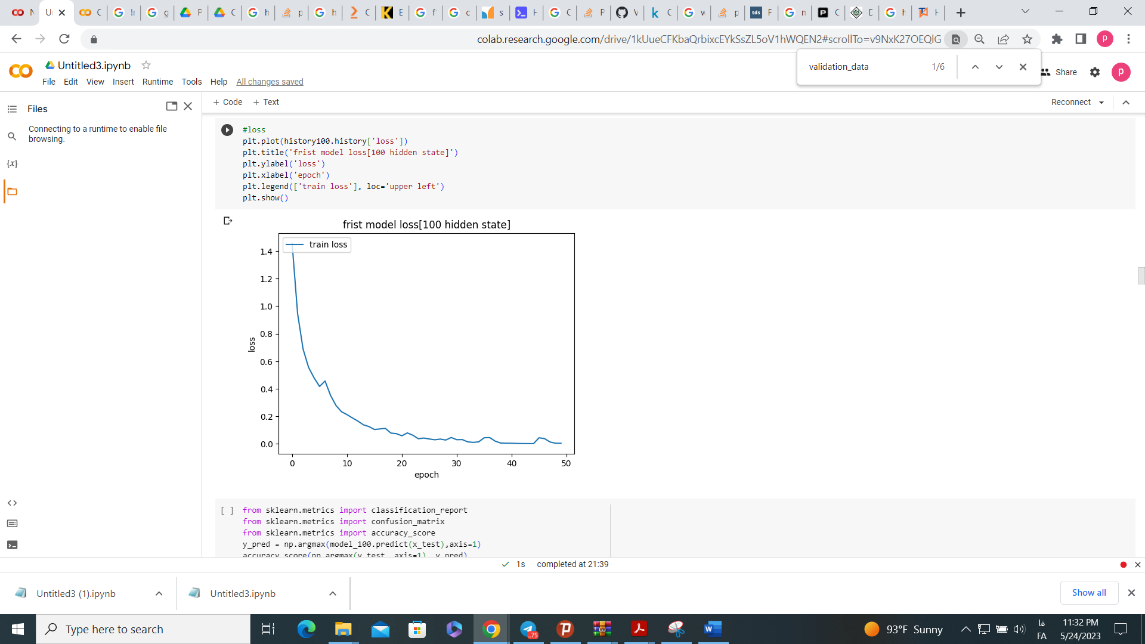


8-پیاده سازی مدل اول-100 لایه

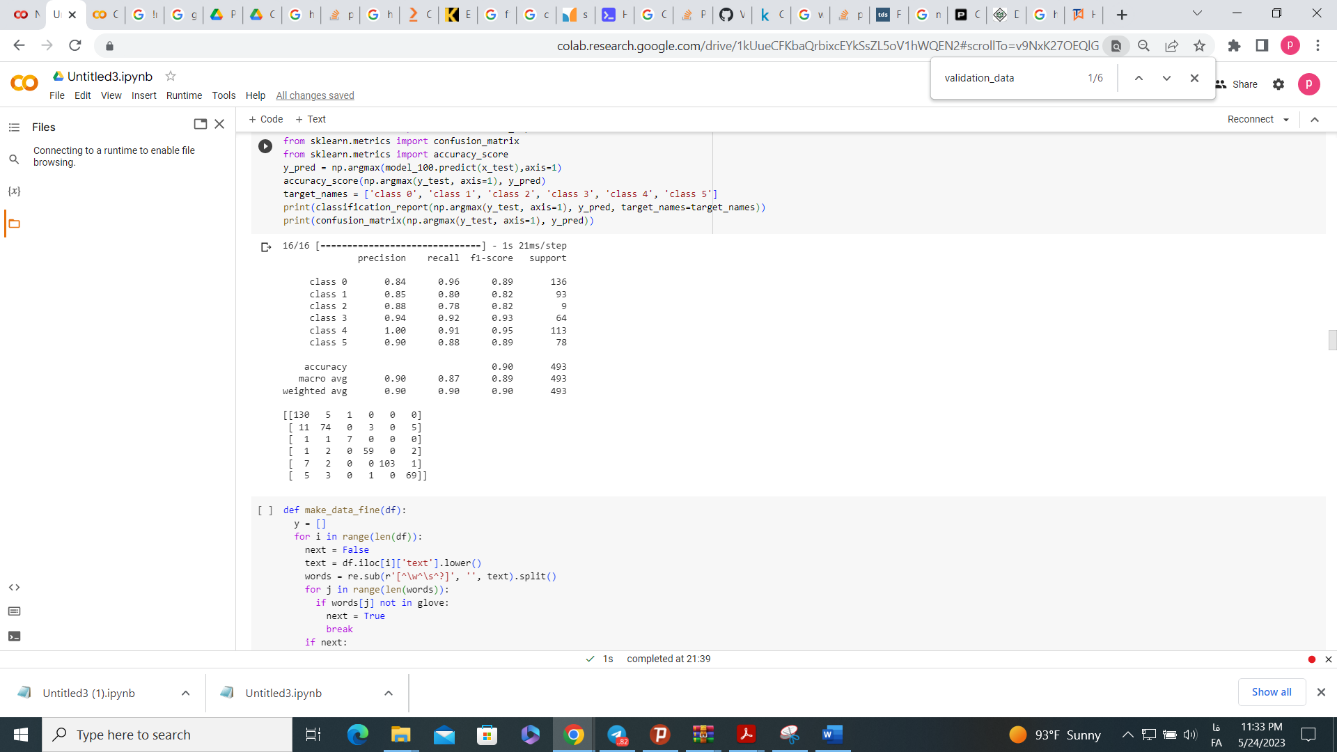
در ادامه نیز مانند قبل گراف های مربوط و دیگر اطلاعات خواسته شده اورده شده است.



9- تابع accuracy مدل اول با 100 لایه

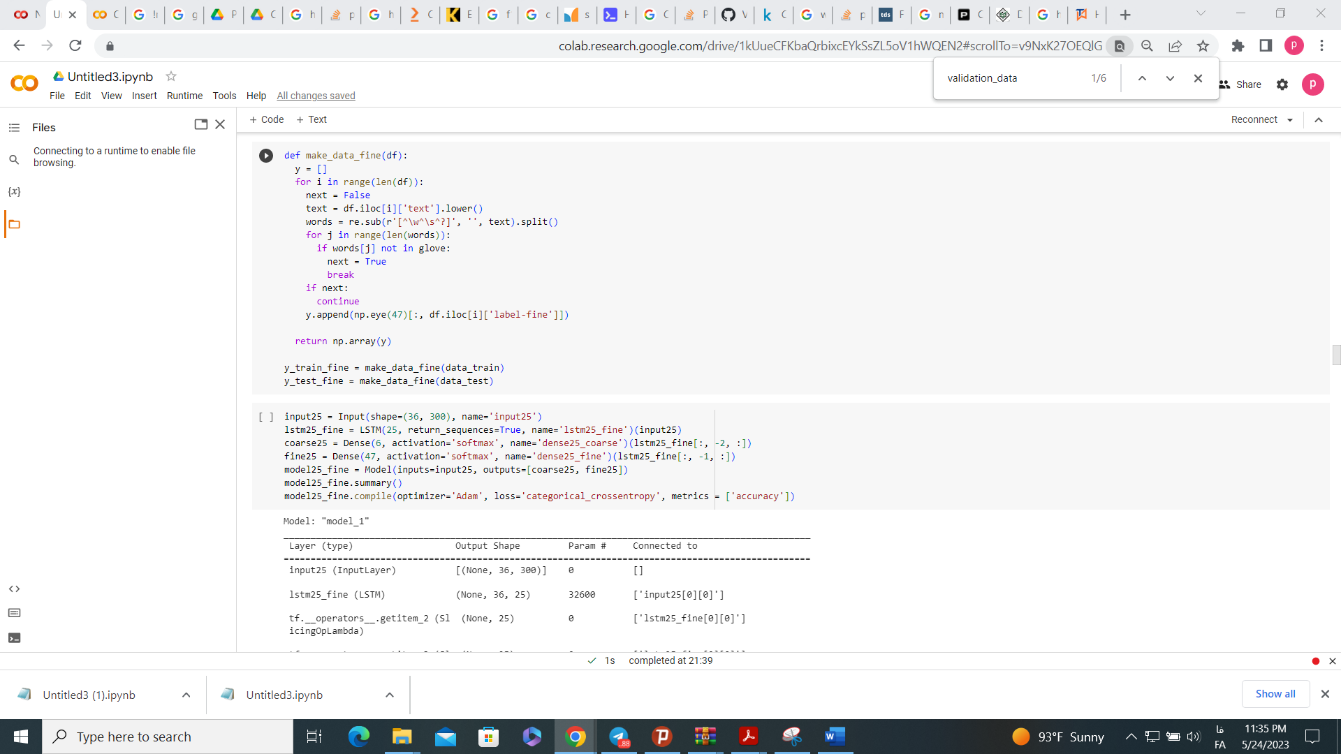


10-تابع loss با 100 لایه



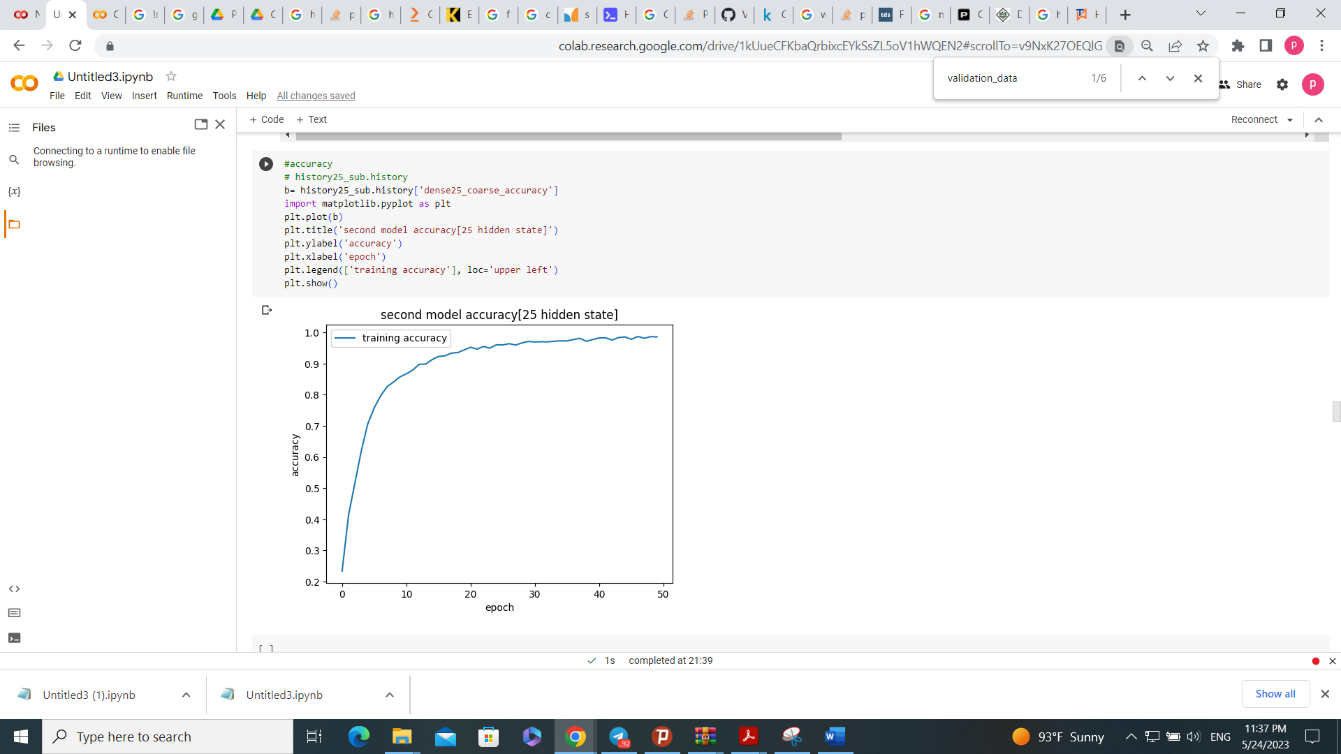
11-گزارش طبقه بند و ماتریس آشفتگی مدل اول100لایه

در اینجا اول به پیاده سازی سطح فرعی میپردازیم .

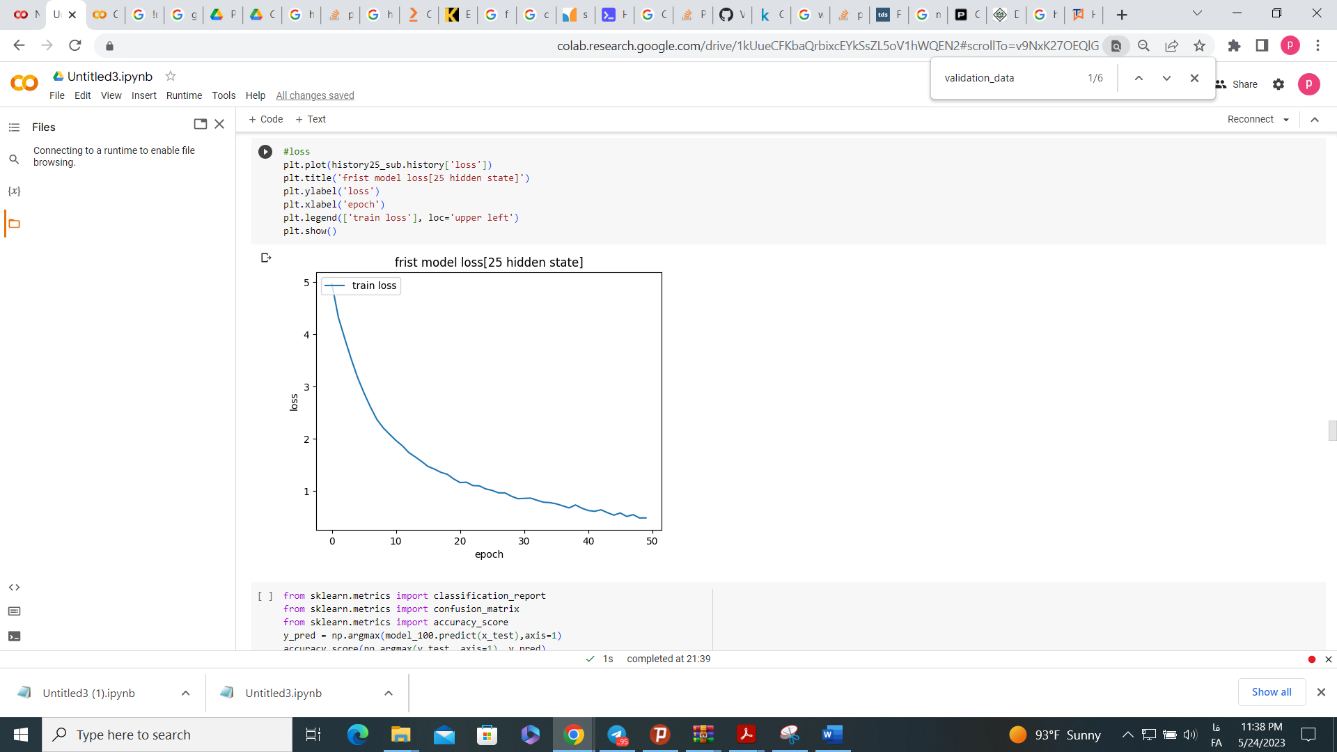


12-پیاده سازی سطح فرعی-fine

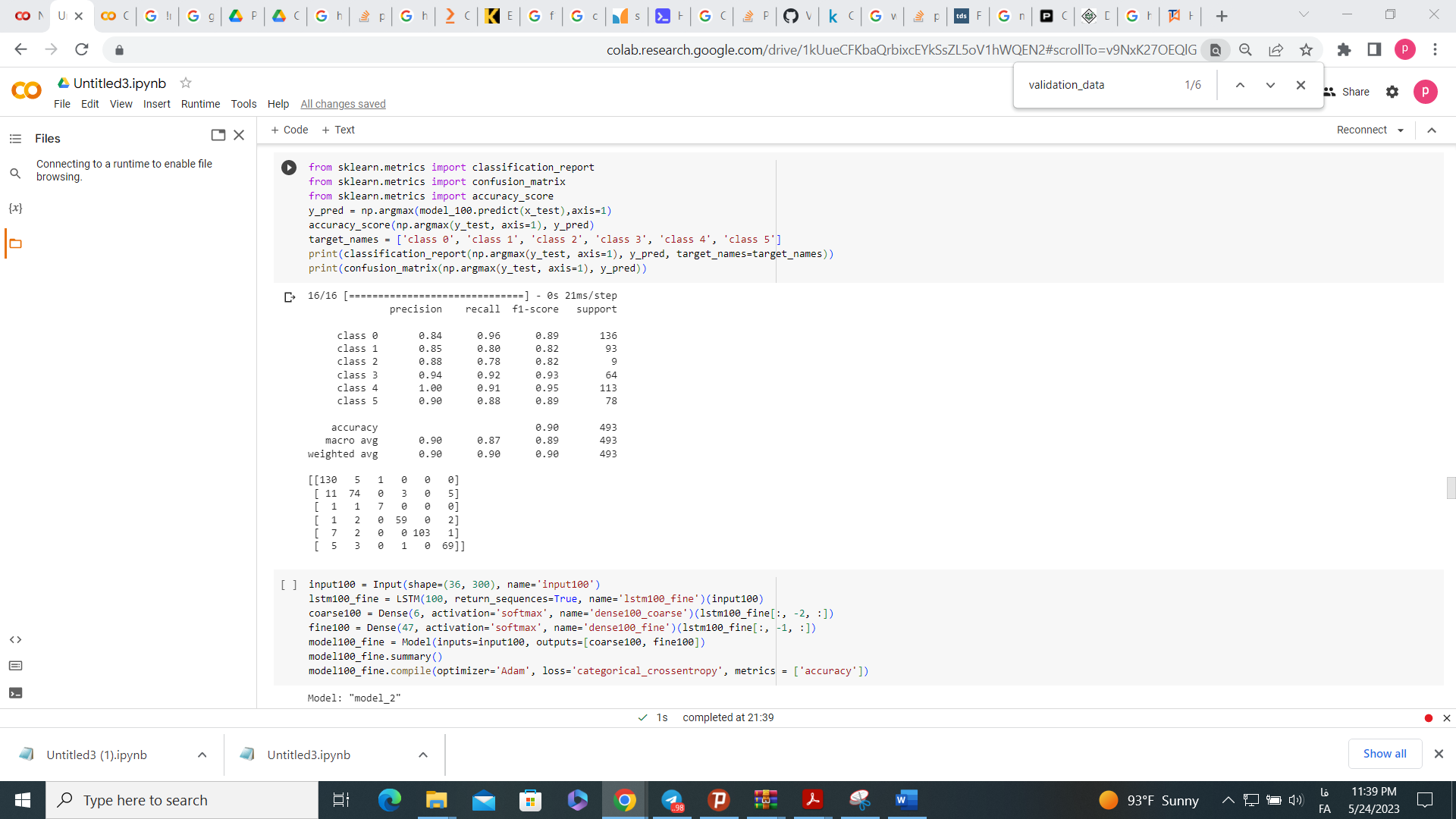
پیاده سازی 25 لایه را انجام میدهیم و در ادامه آن loss، accuracy و ماتریس آشفتگی اورده شده است.



13-تابع accuracy مدل دوم-25لایه

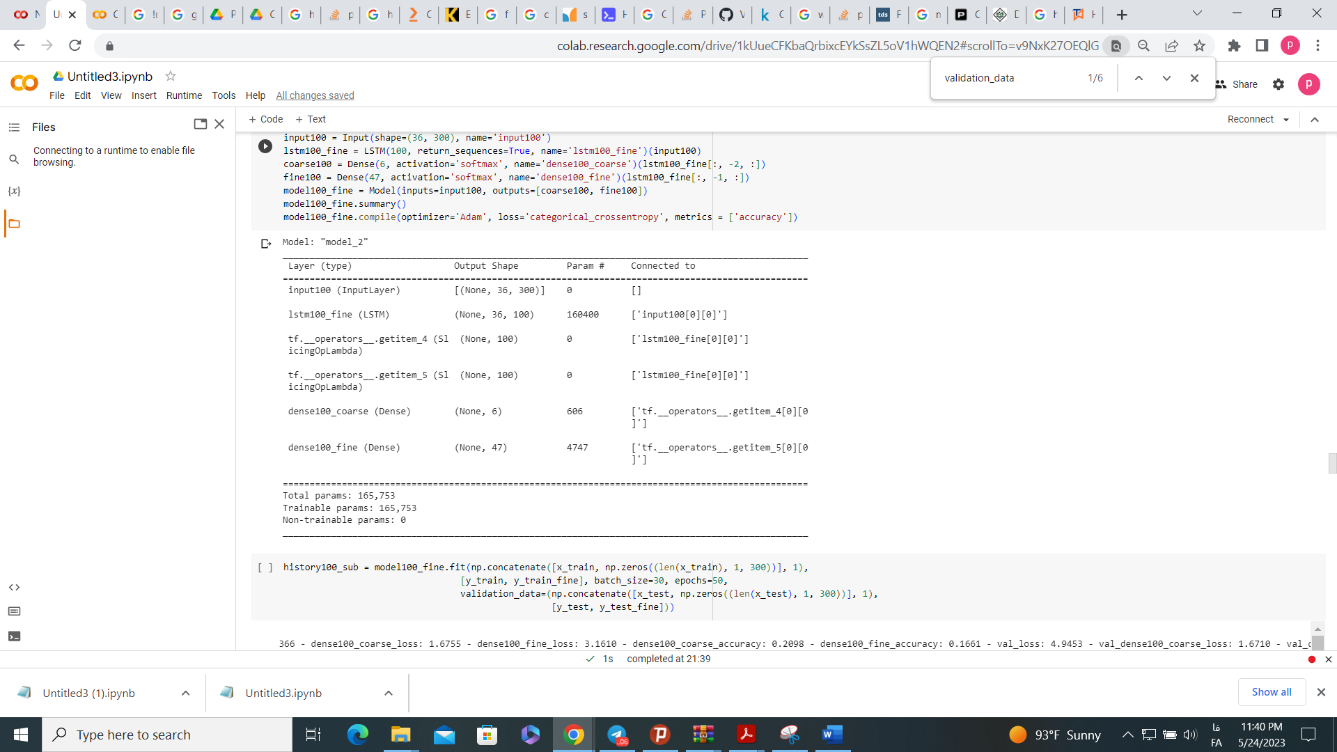


14-تابع loss مدل دوم-25لایه

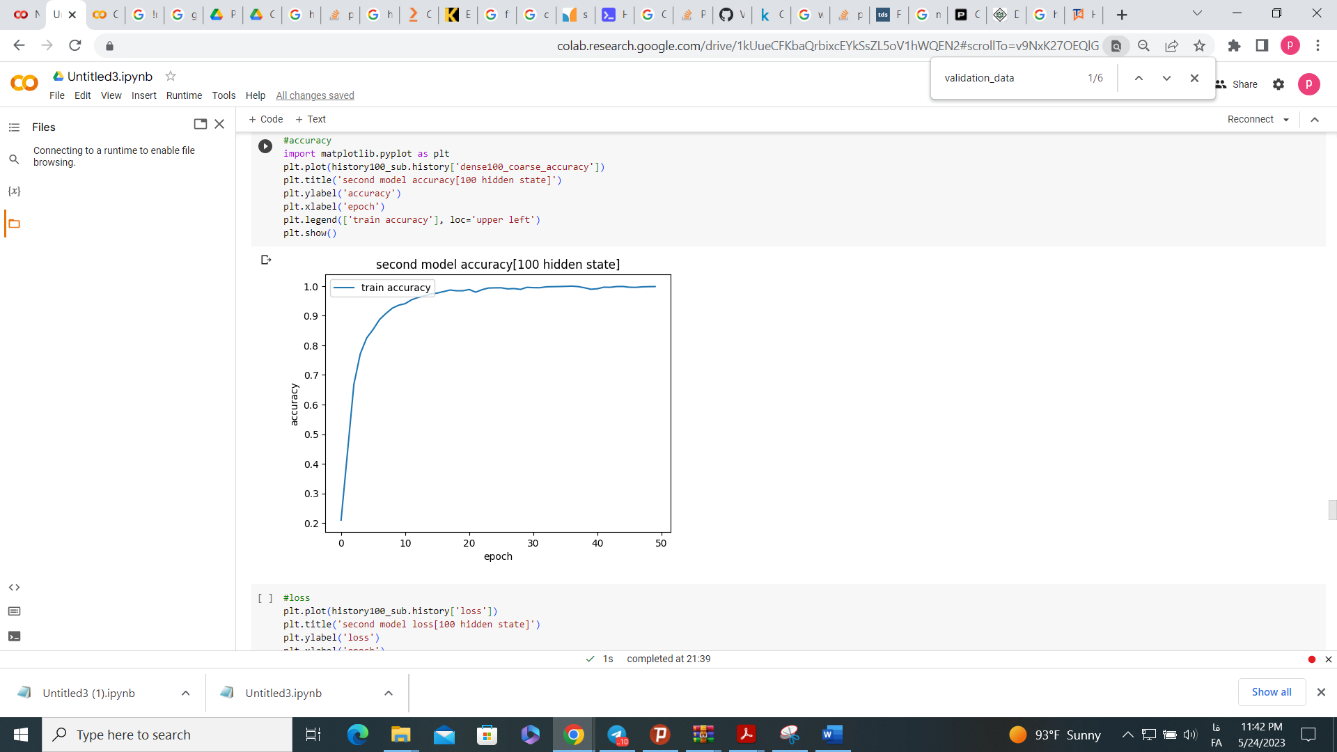


15-گزارش طبقه بند و ماتریس آشفتگی مدل دوم 25لایه

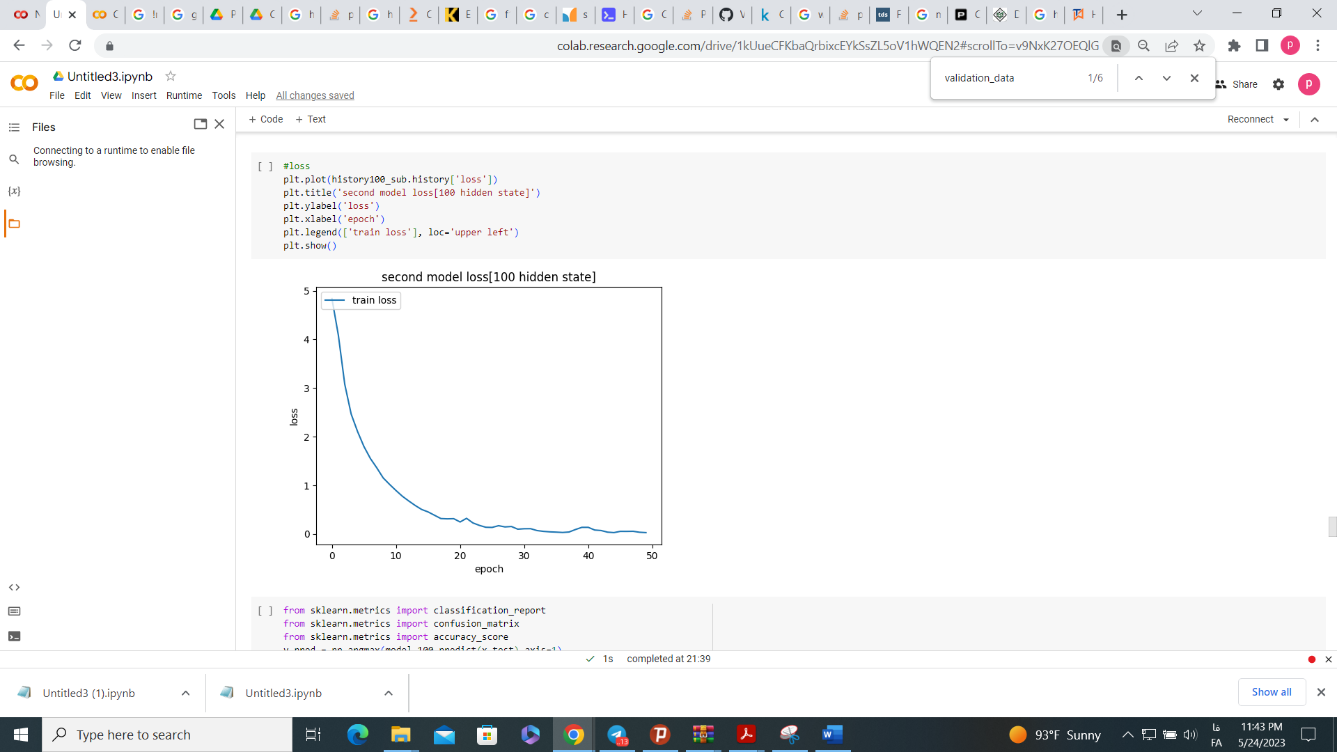
در پایان نیز همه موارد را برای 100 لایه انجام دادیم.



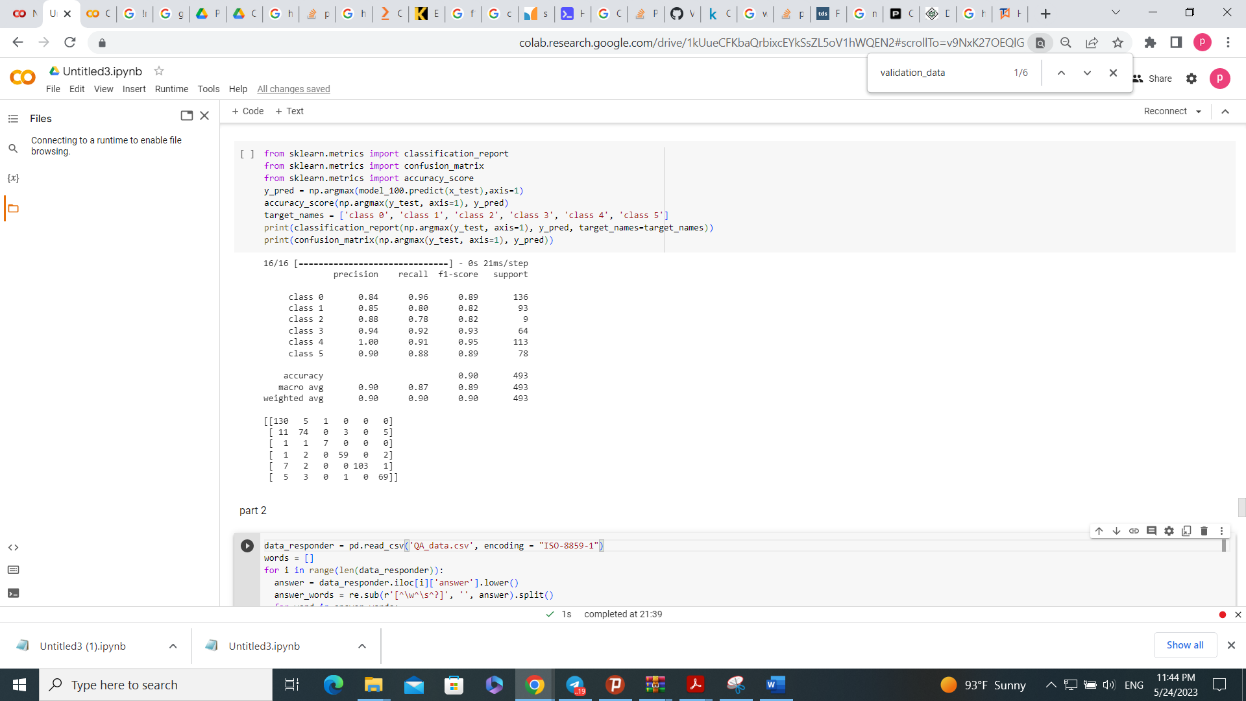
16-پیاده سازی 100 لایه مدل دوم



17-تابع accuracy مدل دوم-100 لایه



18-تابع loss مدل دوم-100 لایه

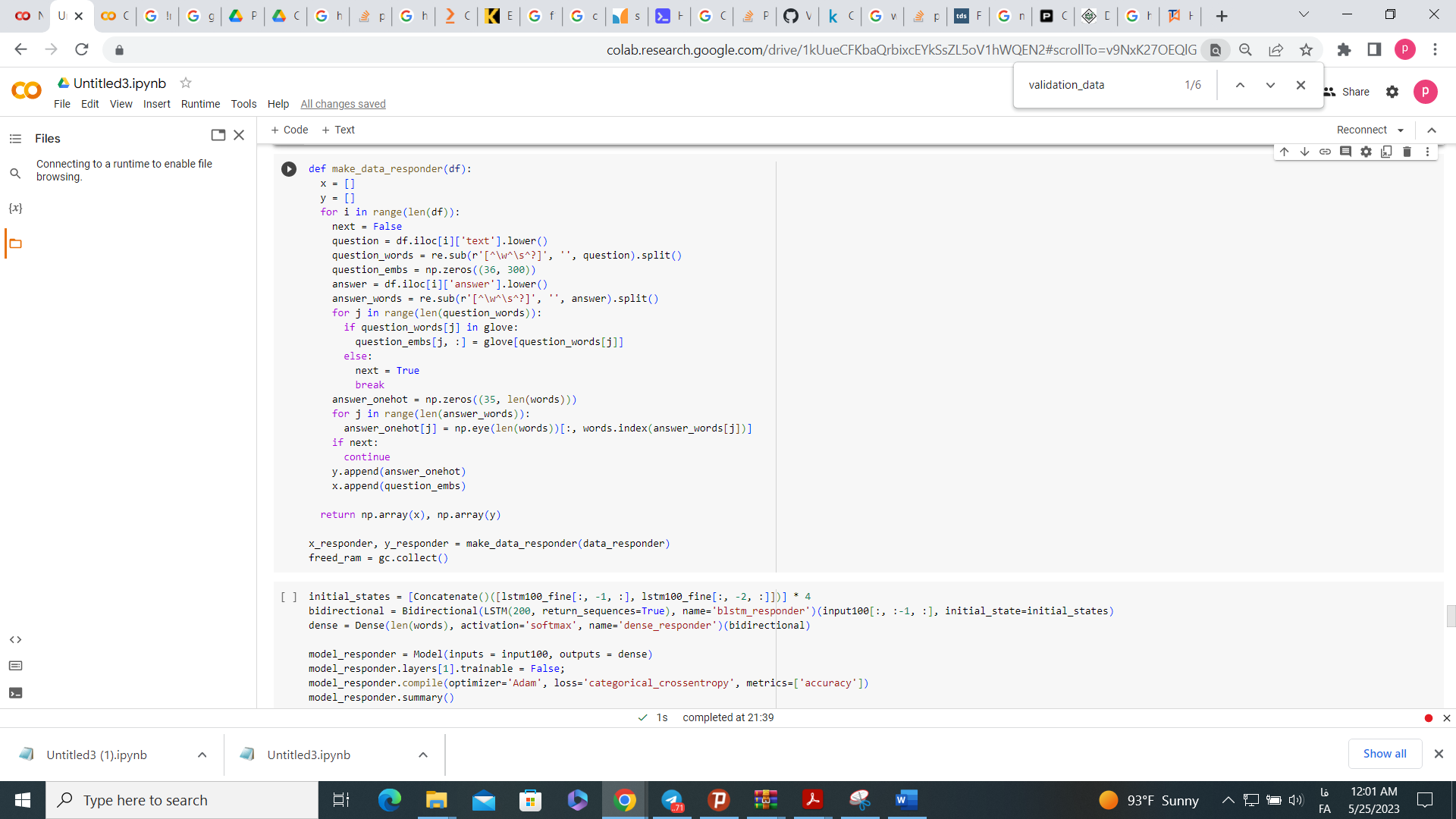


19-گزارش طبقه بند و ماتریس آشفتگی مدل دوم 100لایه

آنطور که برمی اید اولا میزان loss در تمامی مدل ها با هر تعداد لایه پنهان نزولی بوده و میزان دقت نیز صعودی است لازم به ذکر است برای این بخش تعداد ایپاک ها در تمامی مراحل روی 50 قرار دارد. همینطور مشخصاً میزان percision برای مدل دوم به طور کلی از مدل اول بیشتر است و همچنین با افزایش لایه پنهان نیز افزایش نسبی می یابد. در مورد accuracy نیز به طور مشابه همین برقرار است با این تفاوت که بین تعداد متفاوت لایه در مدل دوم تفاوت چندانی ایجاد نمیشود. در مورد f1-score و همینطور recall نیز به طور مشابه افزایش نسبی در هر مدل با افزایش تعداد لایه به ویژه در مدل اول در مدل دوم تقریبا افزایش لایه پنهان کمکی به بهبود نکرده است و گویی با همان تعداد 25 لایه هم میتوان با سخت افزار کمتر به نتایج مشابه رسید. همینطور بین دو مدل، مدل دومی بسیار بهتر عمل کرده است و ان هم به خاطر این است که در دو سطح اصلی و فرعی بررسی میکند.

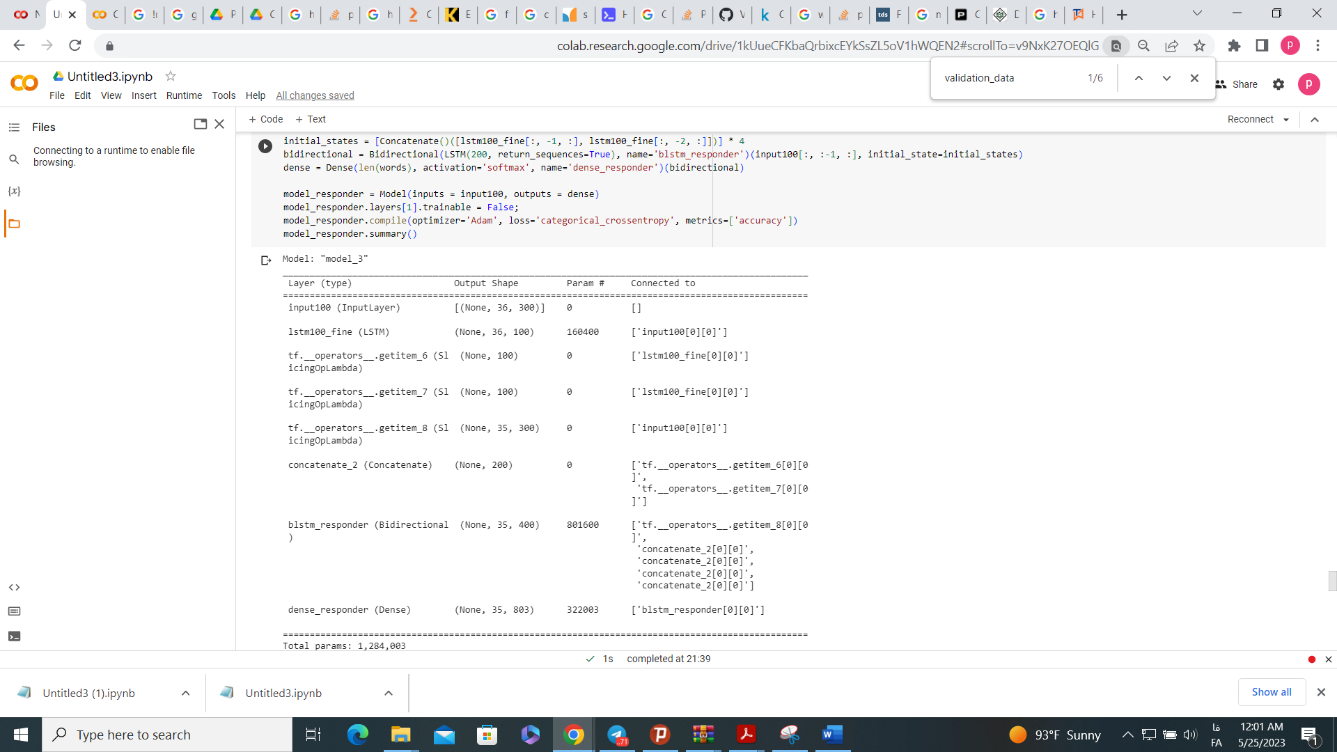
## 4-2.پیاده سازی مدل **Responder**

**در این قسمت نیز یک مدل جدید برای پرسش به پاسخ بر اساس پردازش زبان طبیعی معرفی شده است. در ادامه نیز پیاده سازی مدل اورده شده است.**

****

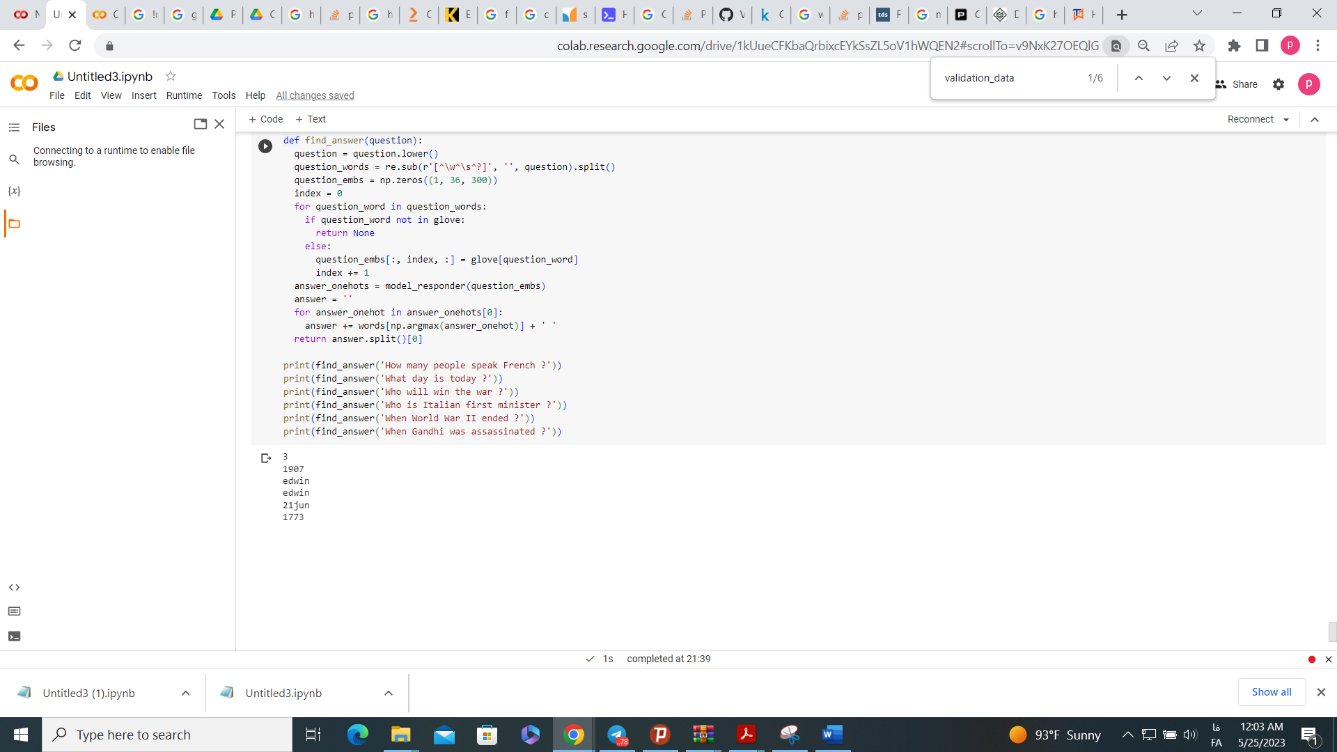
20-پیاده سازی مدل responder

**در این قسمت به اموزش مدل پرداختیم، دقت شود به علت محدودیت های سخت افزاری تا تعداد epoch مورد نظر مقاله نتوانستیم پیش برویم.**

****

21-فیت کردن مدل

**در اینجا نیز مطابق خواسته سوال پاسخ های تولید شده به پرسش ها را نشان میدهیم.**

****

**مشابه شکل توانستیم نتایجی که لزوما نیز درست(ایپاک کم- دقت ....) نیست را گزارش کنیم با توجه به صورت سوالات پاسخ های گزارش شده به ترتیب 3، 1907، Edwin،**

**Edwin، 21Juneسس، 1773 میباشد.**

# **پاسخ ۳** **– عنوان پرسش سوم به فارسی**

## ۱-۳. **عنوان بخش اول**

متن نمونه

.

# **پاسخ ۴** **– عنوان پرسش چهارم به فارسی**

## ۱-۴. **پیش پردازش**

متن نمونه

## . **۲**-۴. **MLP**

لش