**کلاس بندی با خودرمزگذار[[1]](#footnote-2)**

در این گزارش طبقه بندی یک مجموعه داده شامل حالت های مختلف دست[[2]](#footnote-3) با استفاده از AutoEncoder به عنوانclassifier استفاده شده است.

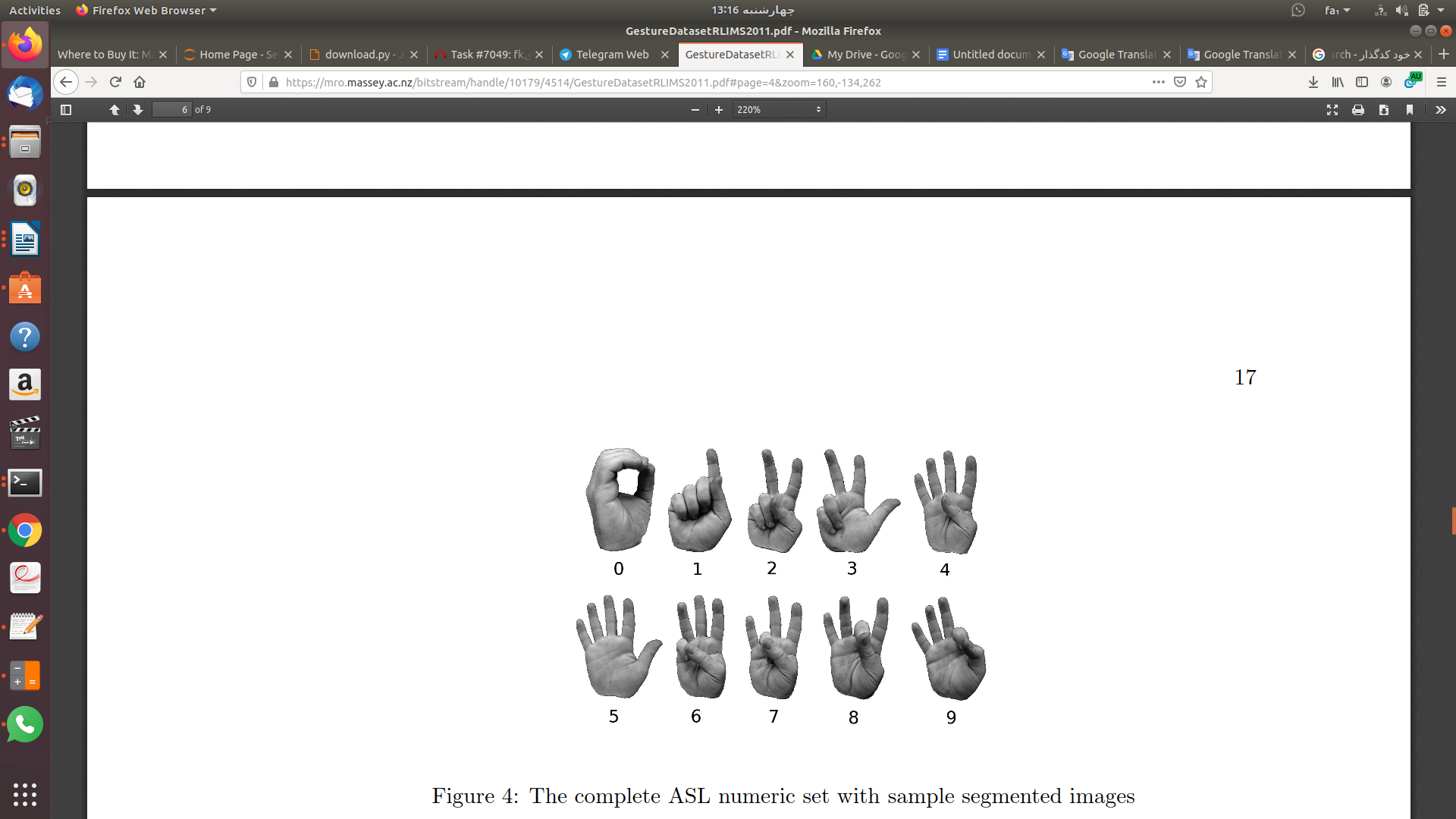
**مجموعه داده**

مجموعه داده های حرکات استاندارد زبان آمریکایی (ASL) که شامل تصاویر 2425 از 5 نفر است [1].در اصل ، 36 کلاس در مجموعه داده وجود دارد. با این حال ، بسته به نوع ویژگی استخراج استفاده شده ، حرکاتی وجود دارد که به دلیل شباهت آنها ، طبقه بندی بسیار دشوار است.نام فایل ها از یک قرار داد ساده پیروی می کند:

handX\_G\_ILL\_seg\_crop\_R.png

* x: تعداد داوطلبانی است که تصاویر آنها در آنجا ضبط شده است
* G: بیابنگر ژست است که به صورت a to z, 0 to 9 می باشد.
* ILL: بیانگر شرایط روشنایی می باشد. که به صورت بالا(top), پایین(bot) و چپ(left) و راست(right) و پخش(diff) می باشد.
* R: بیانگر تکرار ، معمولاً از 1 تا 5 است.





**خودرمزگذار**

Autoencoder یک شبکه عصبی مصنوعی بدون نظارت است که می آموزد چگونه به طور مؤثری داده ها را فشرده سازی و رمزگذاری کند و سپس می آموزد که چگونه داده ها را از حالت فشرده و رمزگذاری شده به حالت اصلی بازگرداند به طوریکه تا حد ممکن نزدیک به ورودی اصلی باشند. Autoencoder با یادگیری نحوه نادیده گرفتن نویز موجود در داده‌ها، ابعاد داده را کاهش می‌دهد.

**اجزای تشکیل دهنده ی خودرمزگذار**

خودرمزگذار از چهار جزء اصلی تشکیل شده است:

۱- کدگذار[[3]](#footnote-4): در آن مدل یاد می‌گیرد چطور ابعاد ورودی را کاهش داده و داده‌های ورودی را به یک نمایش کد گذاری شده فشرده‌سازی کند.

۲- تنگراه[[4]](#footnote-5): لایه ای که شامل نمایش فشرده‌شده داده‌های ورودی است. این لایه پایین‌ترین ابعاد ممکن از داده‌های ورودی است.

۳- کدگشا[[5]](#footnote-6): در آن مدل یاد می‌گیرد چگونه داده‌ها را از نمایش کد شده بازسازی کند تا به ورودی اصلی نزدیک شود.

4- خطای بازسازی[[6]](#footnote-7): این روشی است که عملکرد رمزگشایی و نزدیک بودن خروجی به ورودی اصلی را اندازه گیری می کند.

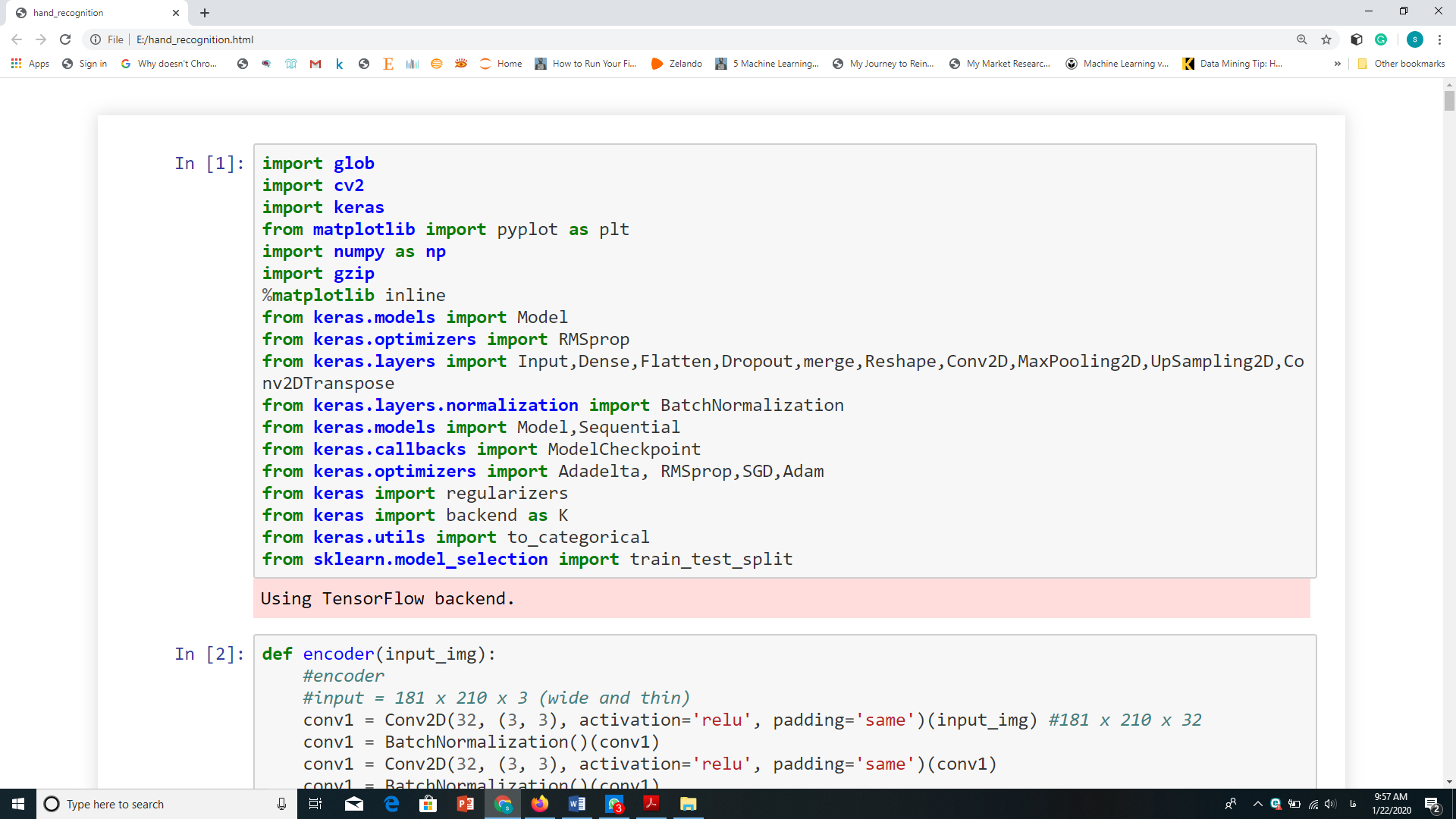
**معماری خودرمزگذار**

معماری شبکه برای خودکارسازها بسته به مورد استفاده ، می تواند بین یک شبکه ساده پیش رو[[7]](#footnote-8)، شبکه بازگشتی[[8]](#footnote-9) یا شبکه عصبی کانولوشن[[9]](#footnote-10) متفاوت باشد. در این گزارش برای ساخت لایه های رمزگذار و رمزگشا از لایه های کانولوشنی استفاده شده است.

**انجام کلاس بندی[[10]](#footnote-11) با اسفاده از خودرمزگذار**

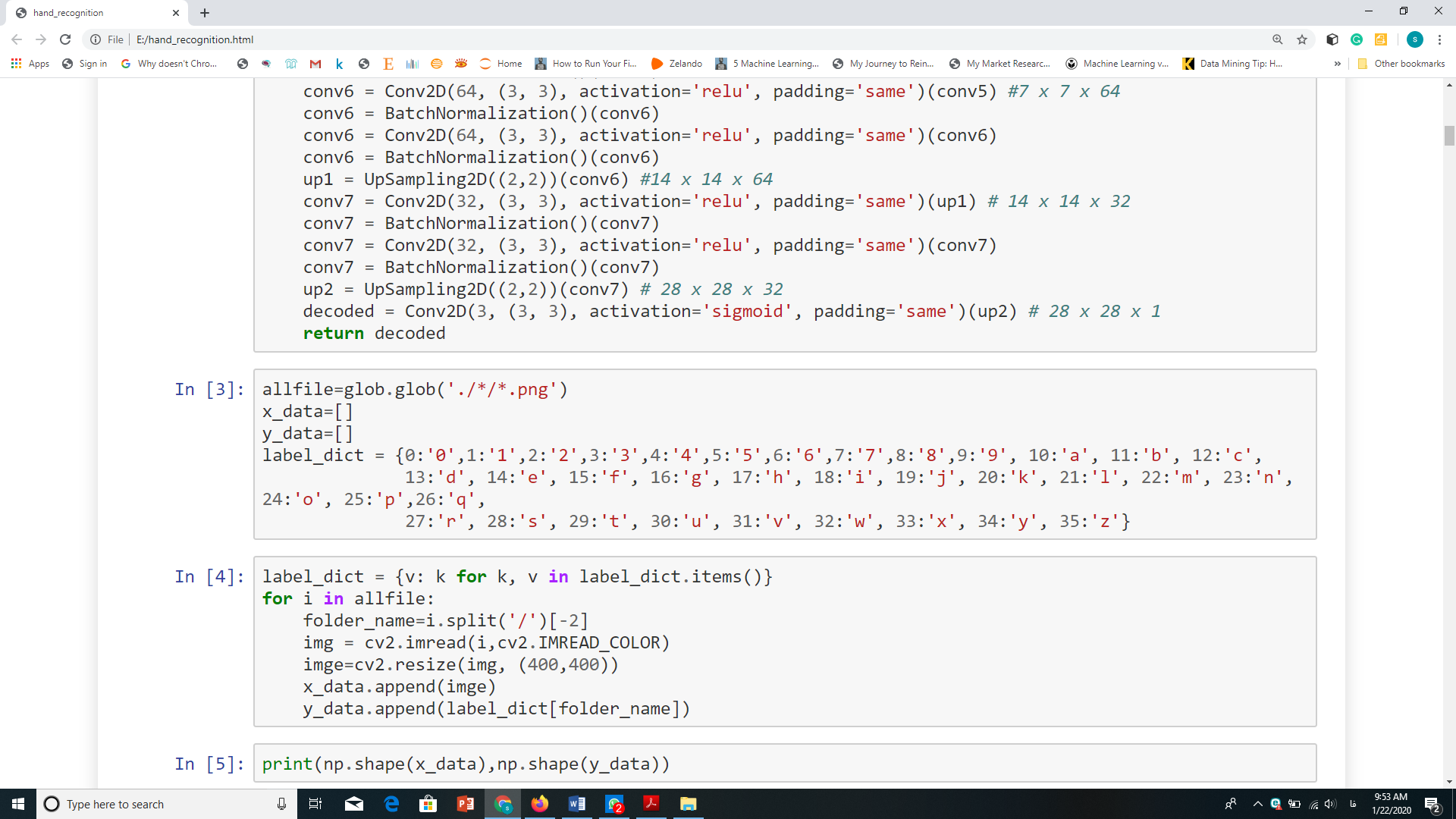
برای کلاس بندی با خودرمزگذار نیاز است که ابتدا مدلی شامل هردو رمزگذار و رمزگشا با مجموعه داده ها آموزش داده شود. در این مرحله نیازی به برچسب داده ها نیست چون قرار است مدل داده های ورودی را کاهش بعد داده و فشرده کند و بتواند دوباره آنهارا بازسازی کند. بنابراین داده ها به عنوان ورودی و خروجی به مدل داده می شوند. پس از ساخت مدل اول و ذخیره ی وزن ها، مدل دوم باید ساخته شود که اینبار فقط شامل رمزگذار(با همان معماری رمزگذار مدل اول) است و یک لایه ی کاملا متصل با تابع فعالیت softmax که برای کلاس بندی استفاده می شود. از وزن های مدل اول برای وزن دهی مدل دوم استفاده می شود. جزییات در مراحل زیر توضیح داده می شود.

**مراحل انجام کلاس بندی:**

**1- import کردن پکیج های لازم**

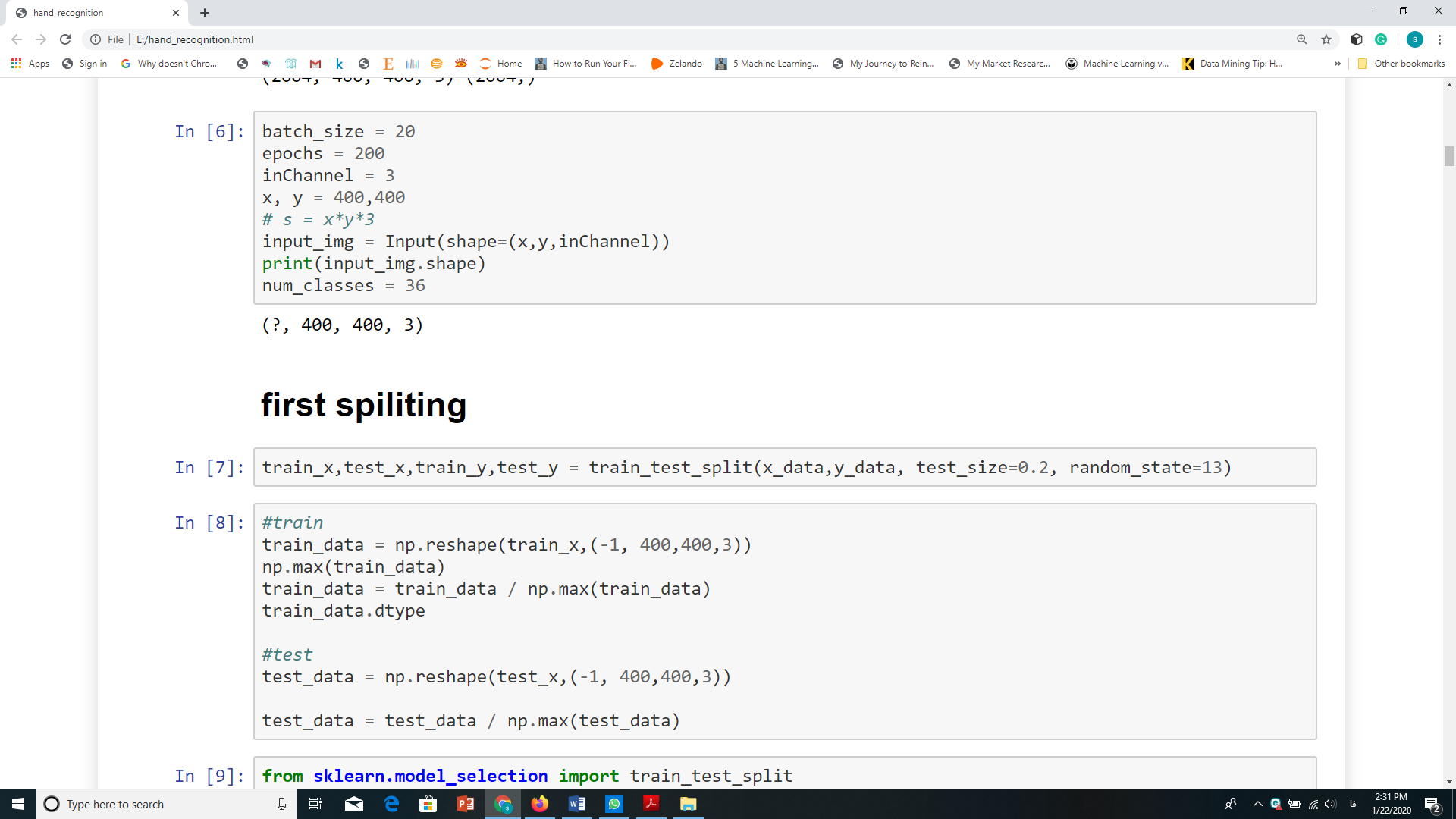
**2- بارگذاری داده ها**

با استفاده از کتابخانه ی glob داده هارا از مسیر میخوانیم. در مجموعه داده 36 کلاس داریم و برچسب ها نام پوشه ها هستند، یک دیکشنری از برچسب ها میسازیم. به ازای تمام داده ها، تصاویر در x\_data و برچسب متناظر را در y\_data قرار میدهیم.

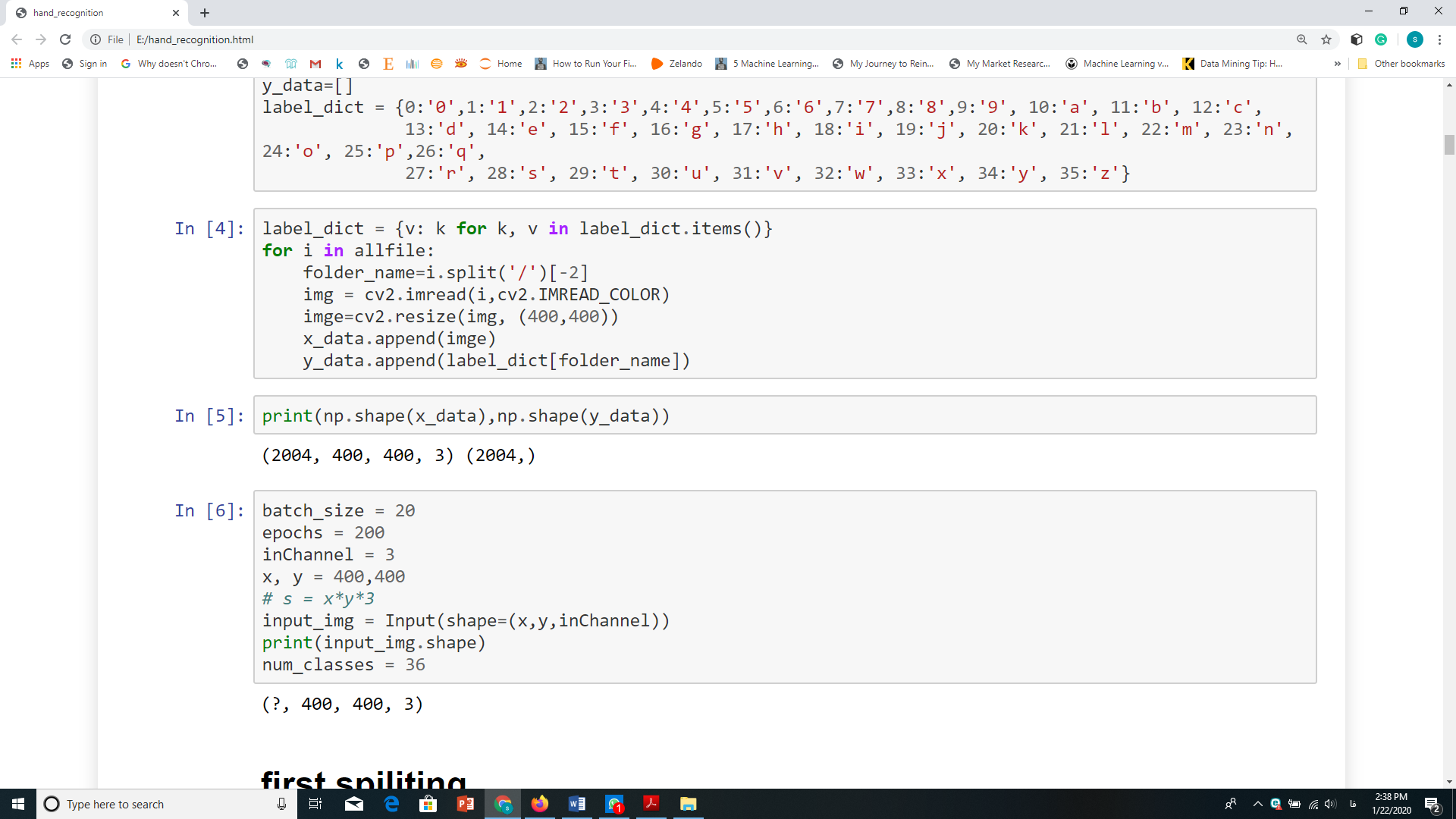


**3- پیش پردازش داده ها**

تصاویر دیتابیس در واقع تصاویر مقیاس رنگی با مقادیر پیکسلی از 0 تا 255 با ابعاد متفاوت هستند که برای ساخت مدل باید یک سایز واحد داشته باشند به همین دلیل آنها را resize میکنیم. همچنین داده ها باید به دو قسمت آموزش و تست تقسیم شوند. برای اینکار از تابع train\_test\_split() از کتابخانه ی sklearn استفاده شده است. گام بعدی reshape کردن داده ها به فرم 400\*400\*3 است که 3 برای سه کانال قرمز، سبز و آبی است. نرمال سازی داده ها با تقسیم مقادیر پیکسل ها بر بیشترین مقدار موجود، انجام شده است. مراحل روی هر دو مجموعه ی تست و آموزش انجام شده است.



**4- تعیین لایه های خودرمزگذار**



گام اول تعیین سایز batch، تعداد ایپاک، تعداد کلاس و تعریف لایه ی ورودی شبکه است.

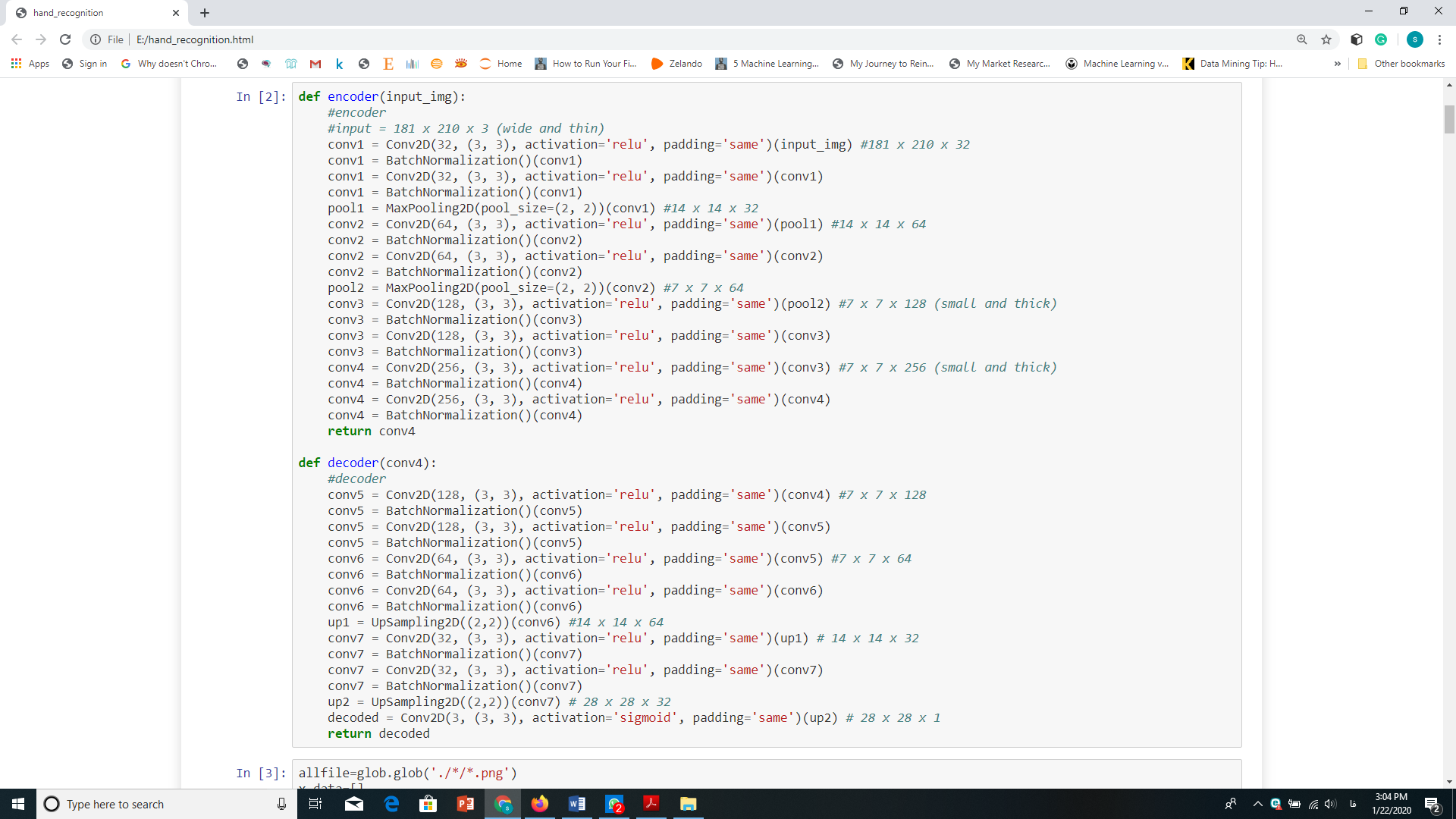
همان طور که گفته شد خودرمزگذار از دو قسمت رمزگذار و رمزگشا تشکیل شده است.

**رمزگذار**: دارای 4 بلوک کانولوشنی است، هر بلوک دارای یک لایه conv است که به دنبال آن یک لایه ی batch normalization قرار دارد. لایه max-pooling پس از بلوک های مرحله اول و دوم استفاده می شود.

* بلوک کانولوشنی اول دارای 32 فیلتر به ابعاد 3\*3 خواهد بود ، به دنبال آن یک لایه max-pooling(downsampling)،
* بلوک دوم دارای 64 فیلتر به ابعاد 3\*3 خواهد بود ، به دنبال آن یک لایه max-pooling دیگر ،
* بلوک سوم رمزگذار 128 فیلتر اندازه 3 \*3 خواهد داشت ،
* بلوک چهارم رمزگذار 256 فیلتر اندازه 3 \*3 خواهد داشت.

**رمزگشا**: دارای 3 بلوک کانولوشنی است ، هر بلوک دارای یک لایه conv است که به دنبال آن یک لایه batch normalization قرار دارد. لایه ی Upsampling پس از بلوک های کانولوشنی دوم و سوم استفاده می شود.

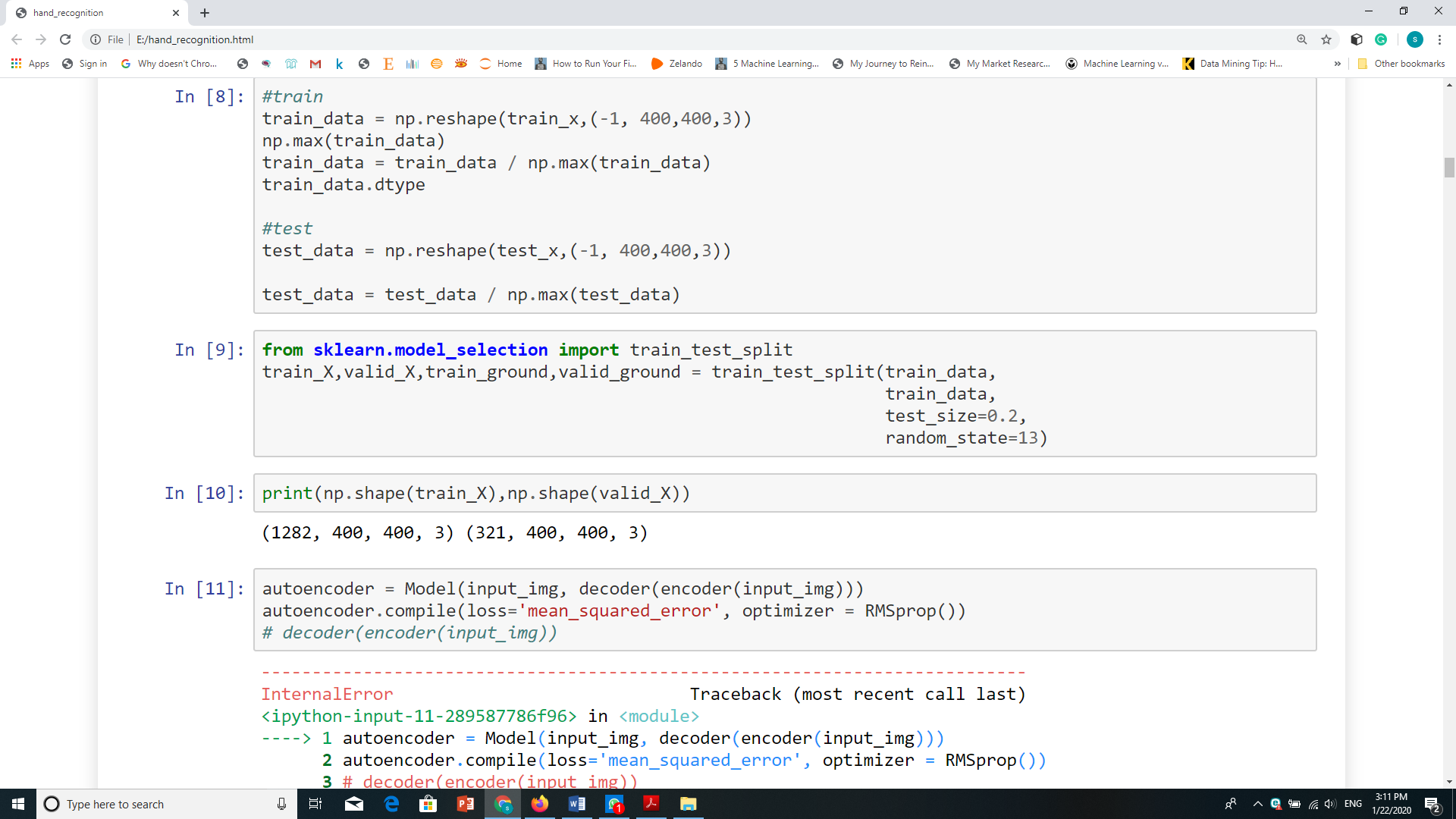
* بلوک اول دارای 128 فیلتر اندازه 3\*3 است ،
* بلوک دوم دارای 64 فیلتر به ابعاد 3\*3 خواهد بود که به دنبال آن یک لایه Upsampling دیگر ،
* بلوک سوم دارای 32 فیلتر به ابعاد 3\*3 خواهد بود که به دنبال آن یک لایه Upsampling دیگر ،
* لایه نهایی رمزگذار 3 فیلتر به اندازه 3\*3 خواهد داشت که ورودی را با داشتن سه کانال رنگی بازسازی می کند.



لایه max-pooling، هر بار که از آن استفاده میشود، ورودی را با ضریب دو downsample می‌کند، در حالی که لایه upsampling هر بار که از آن استفاده می‌شود، ورودی را با ضریب دو upsample تقسیم می‌کند.

**5- ساخت مدل**

نکته ای که در کلاس بندی با خودرمزگذار وجود دارد این هست که تصاویر ورودی به شبکه هم به عنوان ورودی و هم خروجی رمزگشا به شبکه داده می شوند و از برچسب ها در این قسمت استفاده نمی شود. بنابراین مجموعه ی train\_data به دو قسمت train و validation تقسیم میشود تا برای آموزش شبکه استفاده شوند. 1282 تصویر در مجموعه ی train و 321 تصویر در مجموعه ی validation قرار میگیرد. در گام بعد مدل autoencoder با لایه Input و خروجی decoder(encoder(input\_img)) ساخته میشود و معیار mean\_squared\_error به عنوان تابع هزینه تعیین می شود.



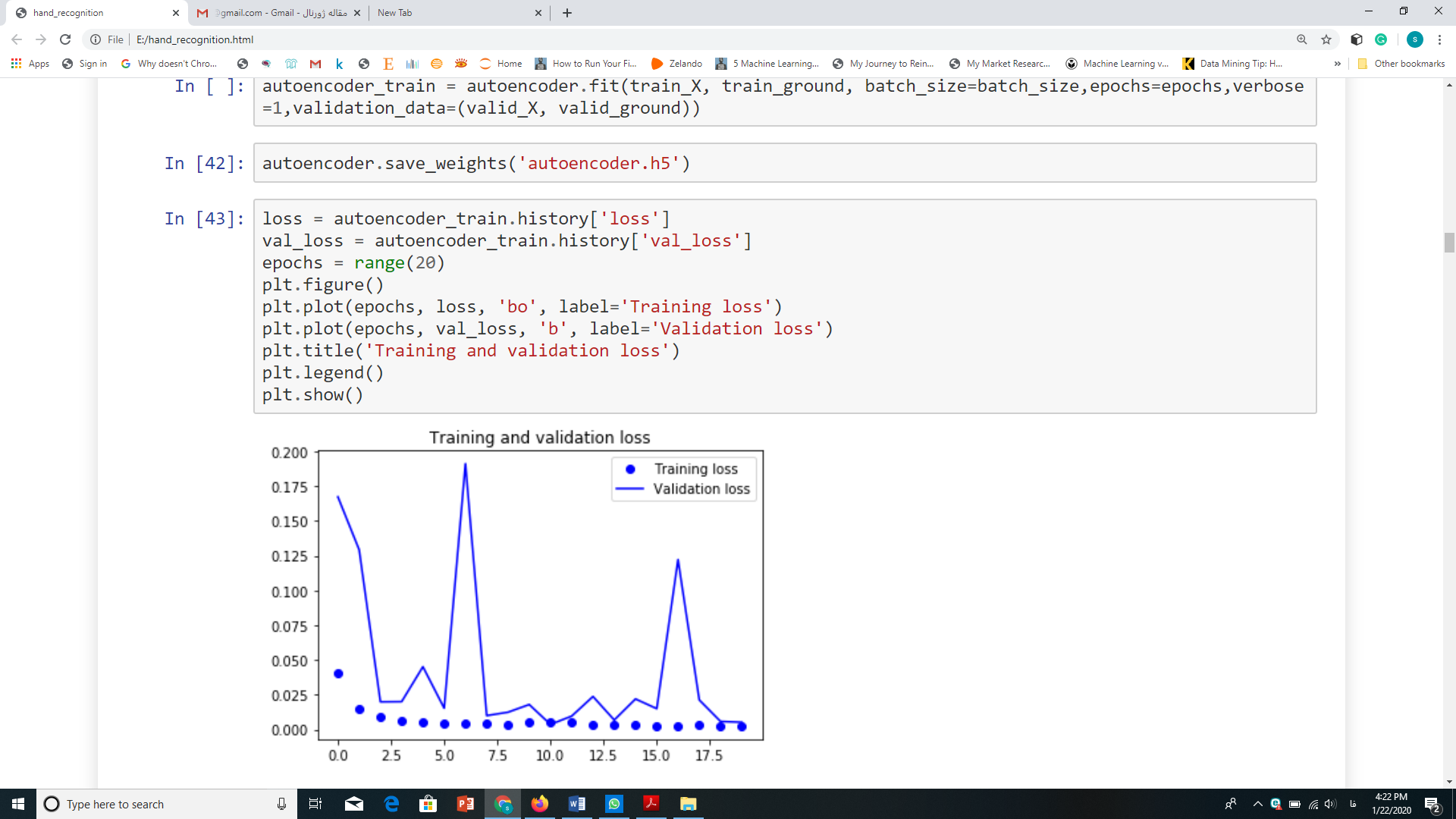
**6- آموزش مدل و ذخیره ی وزن ها**

در این قسمت با استفاده از تابع fit از کتابخانه keras داده های train و validation را به مدل میدهیم. بعد از طی شدن ایپاک های تعیین شده آموزش تمام می شود و وزن ها را ذخیره میکنیم.



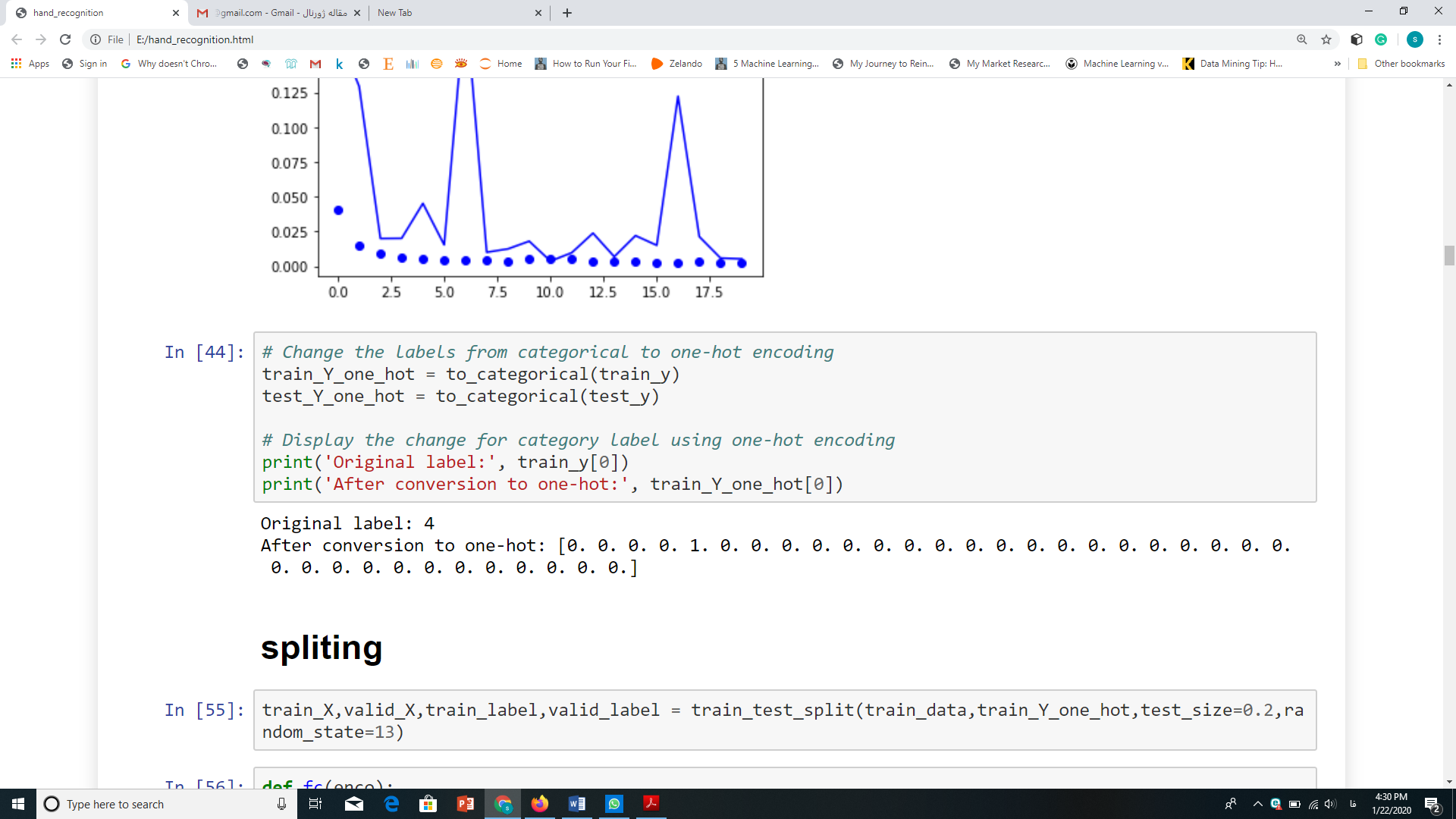
**7- رسم نمودار خطا**

در نمودار زیر خطای مدل روی مجموعه داده ی train و valididation رسم شده است.



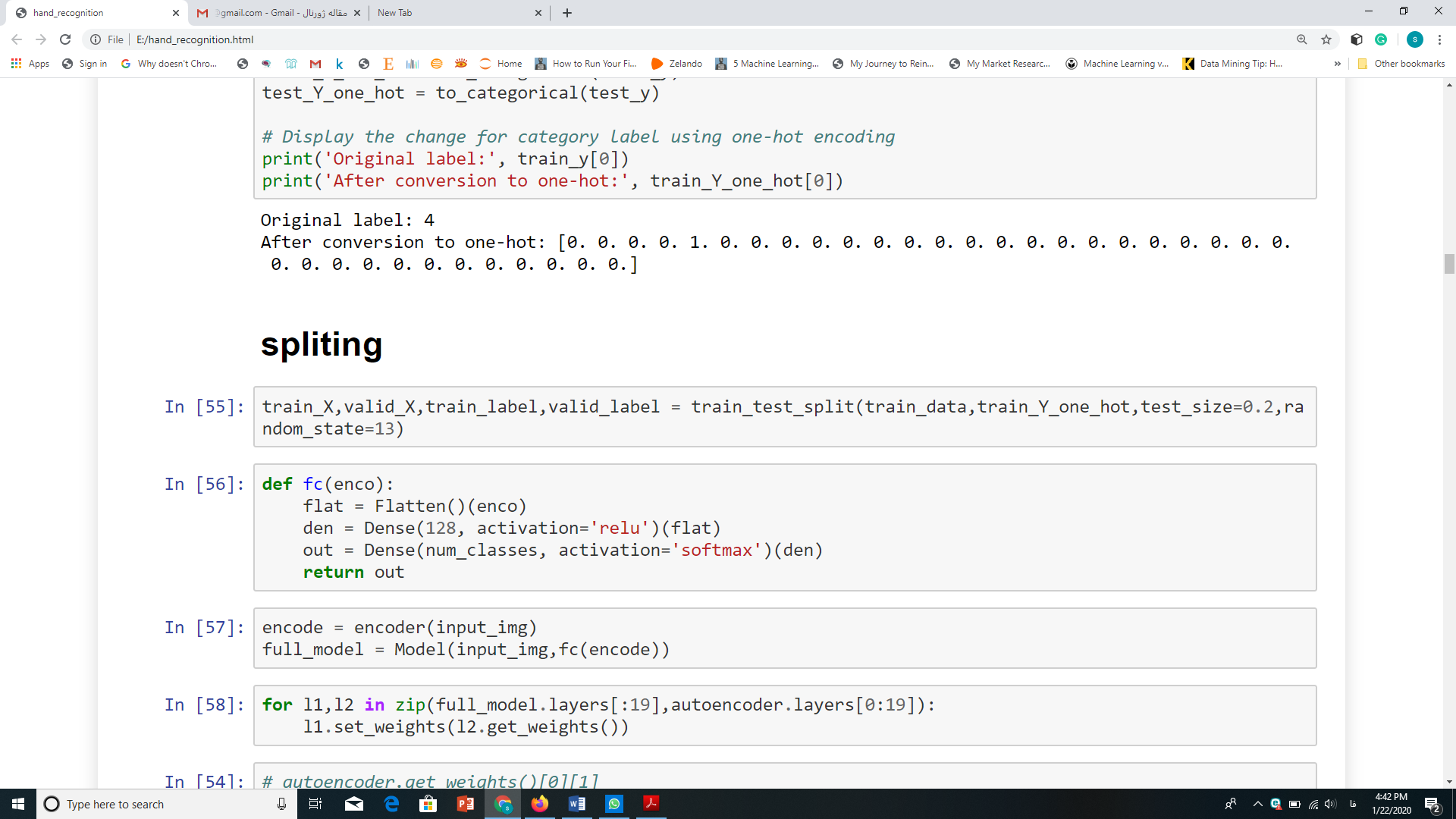
**8- one-hot کردن برچسب کلاس ها**

هر برچسب تبدیل به یک آرایه 36 خانه ای(به تعداد کلاس ها) می شود که فقط یکی از خانه های آن یک و بقیه صفر هستند. خانه ای که یک شده اندیس آن برچسب داده است.



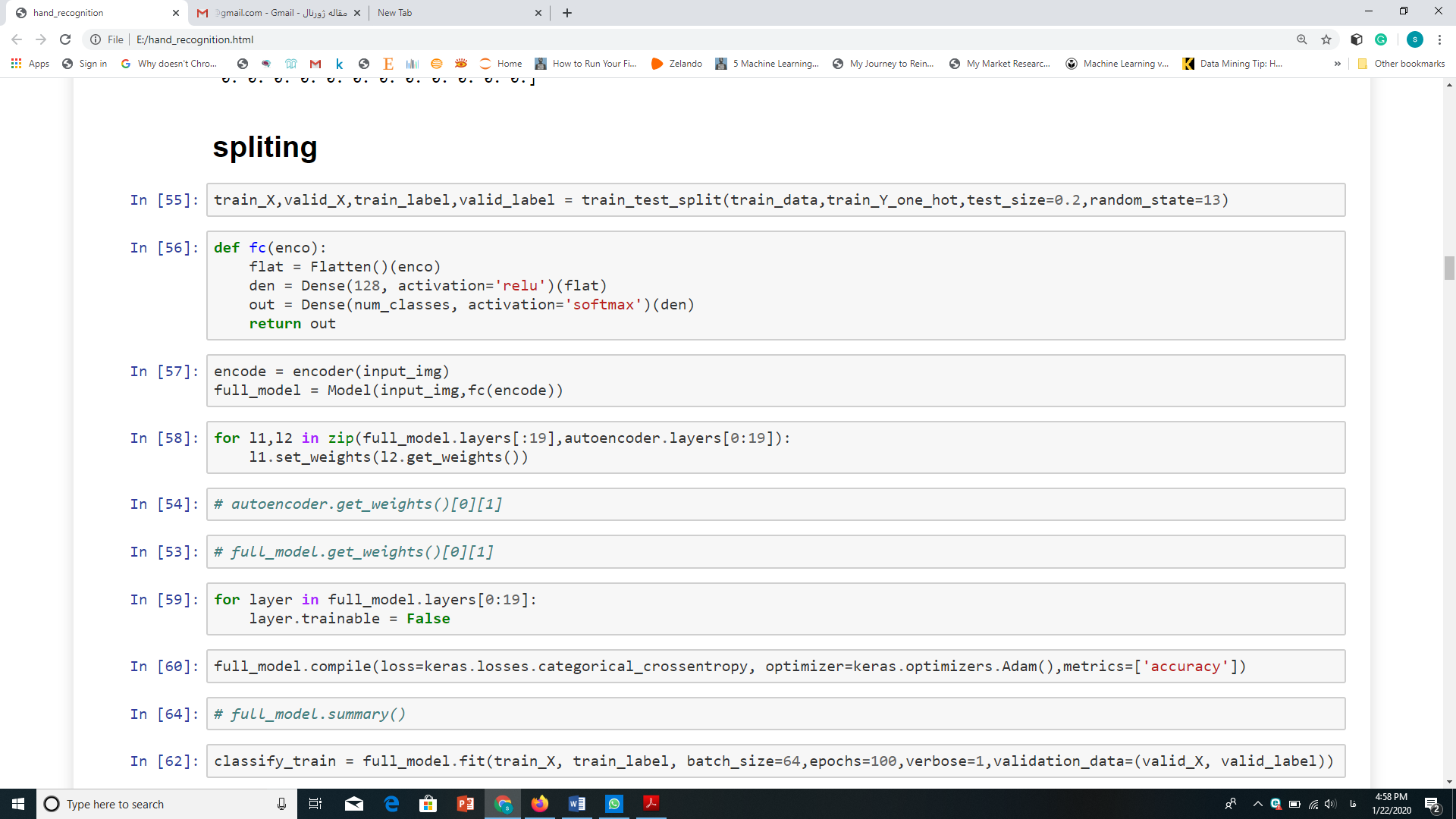
**9- تقسیم کردن داده ها برای تست و آموزش**

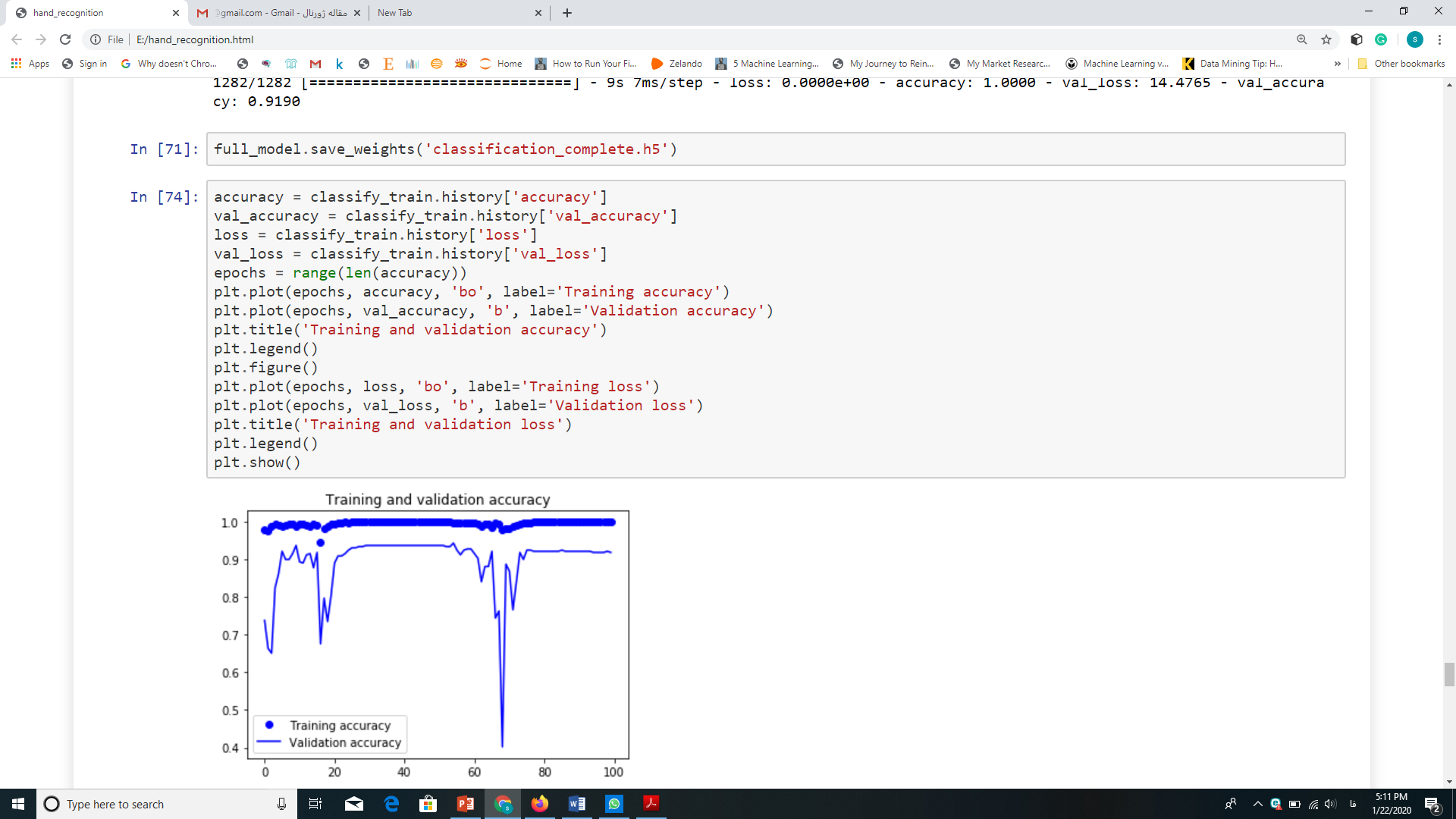
در این مرحله مانند مرحله ی 2 داده ها را به دو قسمت آموزش و تست تقسیم میکنیم اما اینبار چون هدف کلاس بندی است از برچسب های one-hot شده هم استفاده میکنیم.



**10- ساخت مدل طبقه بند**

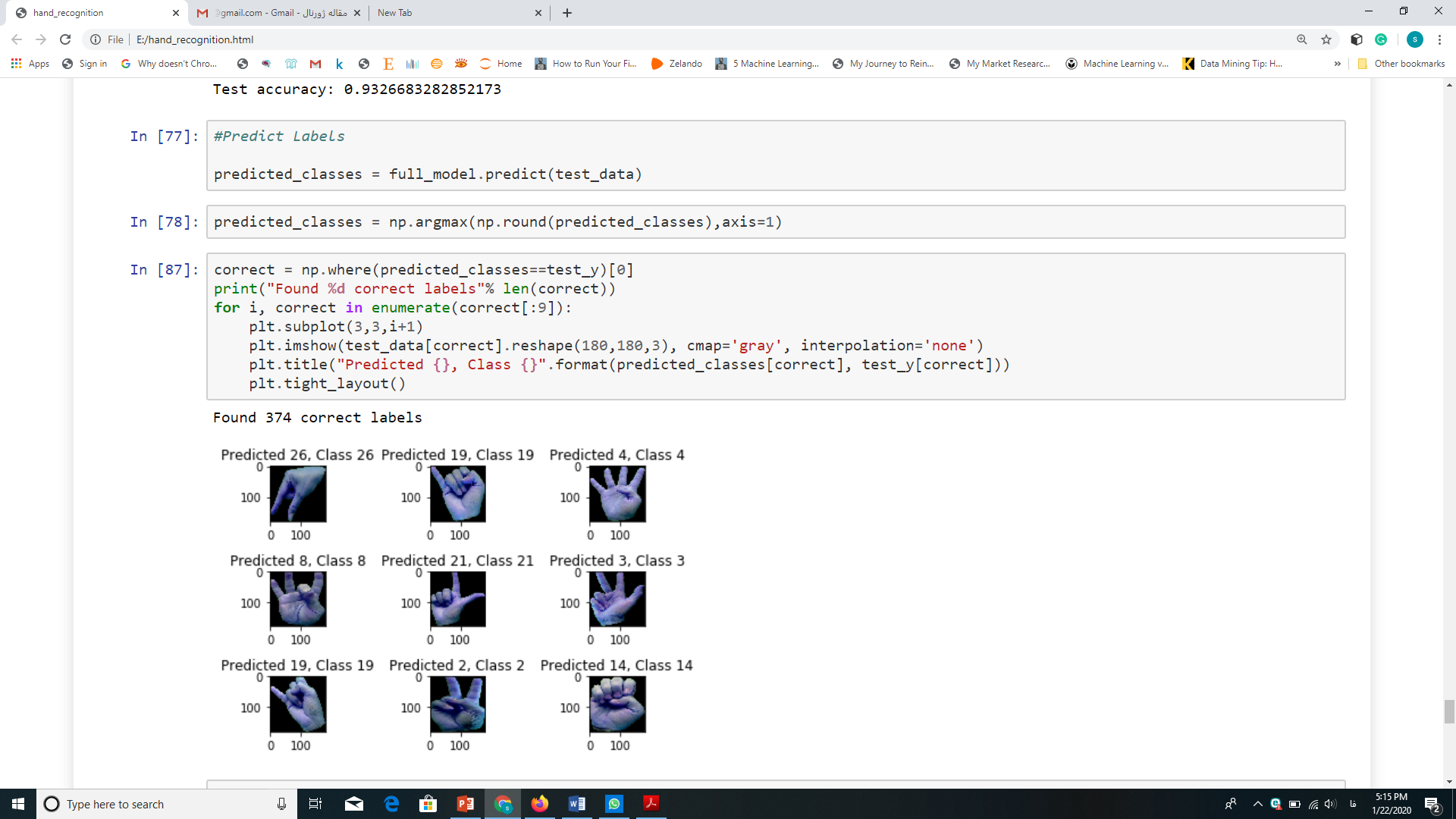
در این مرحله باید یک شبکه جدید بسازیم که معماری مشابه رمزگذار داشته باشد و وزن های خودرمزگذاری که ذخیره کرده بودیم را روی آن بارگذاری کنیم و مجددا آن را آموزش دهیم. این بار با هدف کلاس بندی آموزش انجام می شود بنابراین باید یک لایه ی کاملا متصل[[11]](#footnote-12) به انتهای مدل اضافه کنیم. علاوه بر این چون قسمت رمز گذار یکبار آموزش داده شده است دیگر نیازی به آموزش آن نیست، بنابراین layer.trainable را برای این لایه ها false میکنیم. در آخر مدل را روی داده های آموزش fit میکنیم.

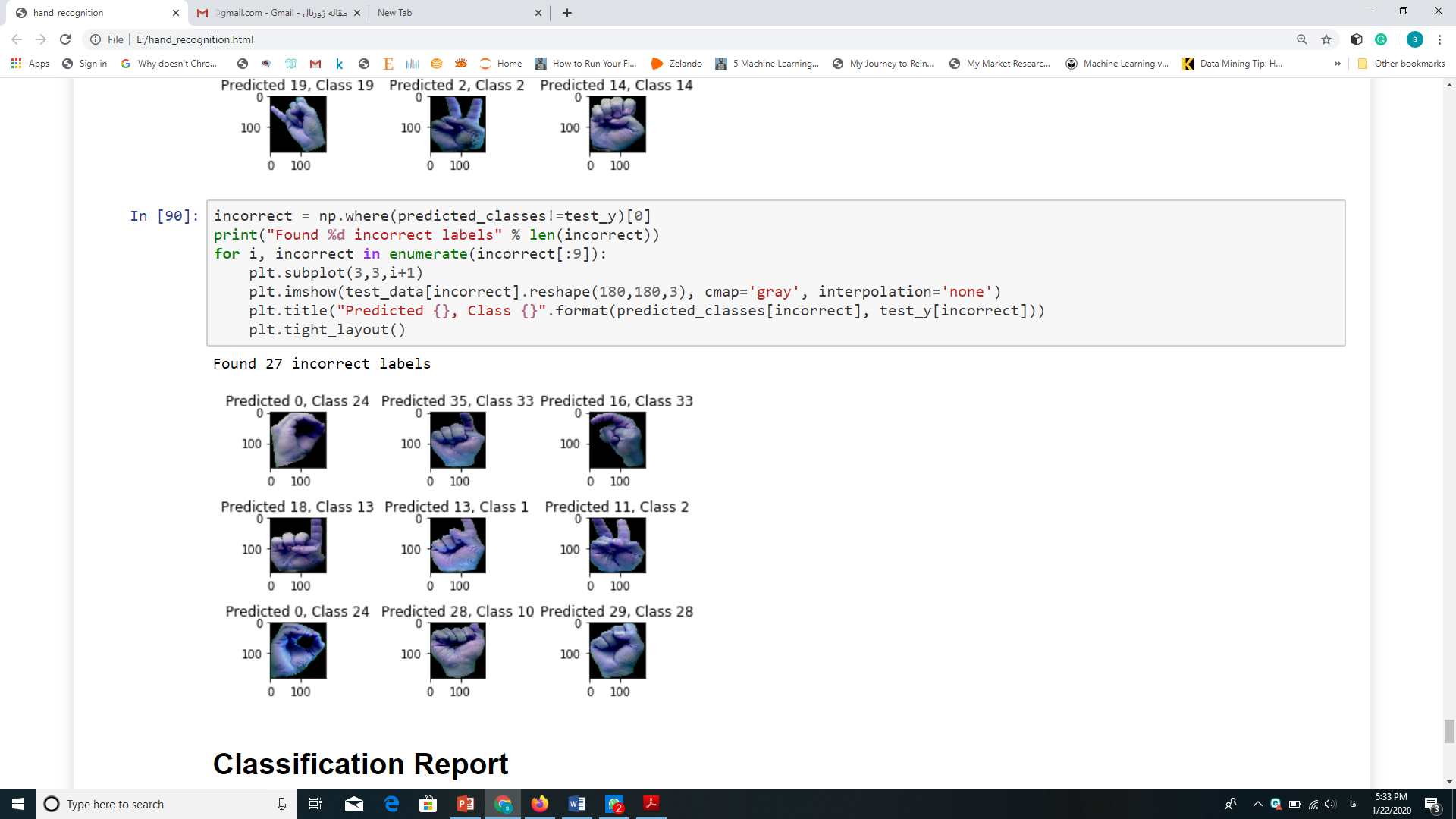




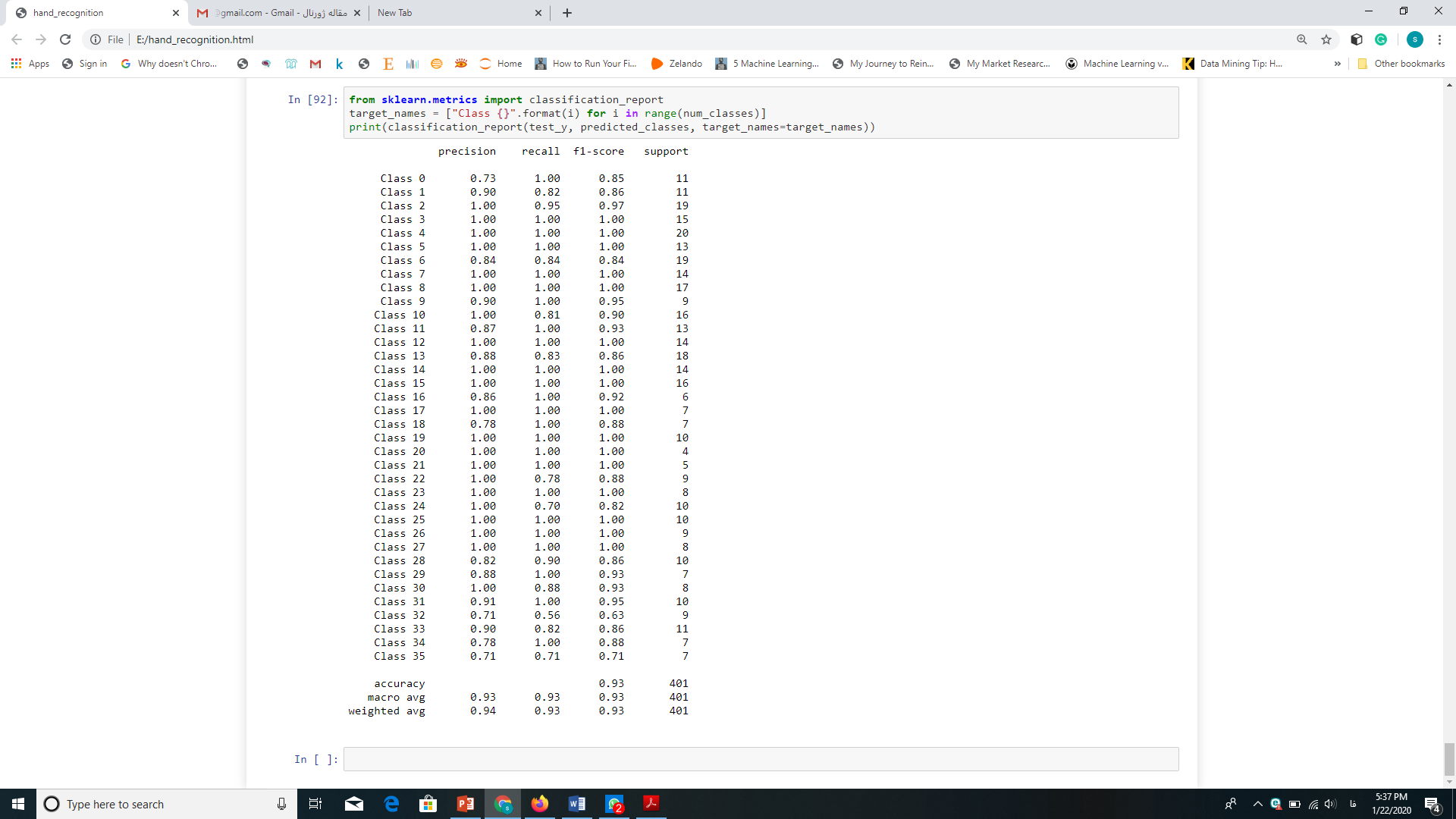
**11 – استفاده از مدل برای پیش بینی**

با استفاده از تابع predict از کتابخانه ی keras، برای مجموعه داده ی تست برچسب ها را پیش بینی میکنیم و در predicted\_class قرار می دهیم و سپس accuracy را حساب میکنیم و برای 18 تصویر نتیجه را رسم میکنیم. در 9 تصویر اول برچسب تصاویر درست تشخیص داده شدند و در 9 تصویر بعدی تشخیص اشتباه بوده است.





**12- گزارش نهایی طبقه بندی**



مراجع:

1. Barczak, A., et al., *A new 2D static hand gesture colour image dataset for ASL gestures.* 2011.

1. AutoEncoder [↑](#footnote-ref-2)
2. Hand gestures [↑](#footnote-ref-3)
3. Encoder [↑](#footnote-ref-4)
4. Bottleneck [↑](#footnote-ref-5)
5. Decoder [↑](#footnote-ref-6)
6. Reconstruction Loss [↑](#footnote-ref-7)
7. Feedforward [↑](#footnote-ref-8)
8. LSTM [↑](#footnote-ref-9)
9. Convolutional [↑](#footnote-ref-10)
10. Classification [↑](#footnote-ref-11)
11. Fully connected [↑](#footnote-ref-12)