***بسم الله الرحمن الرحیم***

پروژه ی Bitcoin Price Prediction

درس هوش محاسباتی

نام و نام خانوادگی:هیلا حیدری

شماره دانشجویی:950122680081

**گزارش فعالیت ها:**

ابتدا با زبان پایتون و سپس با jupyter notebook آشنا شدیم.

و در مرحله بعد به جستجو در اینتنت و جمع آوری اطالعات پرداختیم.

Deep learning یکی از مباحث در هوش مصنوعی است در تعریف کلی یادگیری عمیق همان یاد گیری ماشین است بطوری که در سطوح مختلف نمایش یا انتزاع (abstraction) یادگیری را برای ماشین انجام می دهد که با این کار ماشین میتواند درک بهتری از واقعیت های موجود پیدا کرده و میتواند الگو ها ی مختلف را شناسایی کند.

برای شناسای یادگیری عمیق، ابتدا نیاز به دانستن شبکه های عصبی دارید. بر اساس تعریف

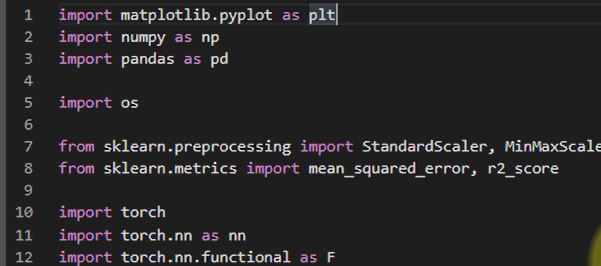
مشهور، یادگیری عمیق در واقع همان یادگیری به وسیله شبکه های عصبی ای هستند که دارای لایه های پنهانی (Hidden Layers) زیادی میباشند. هر چقدر در لایه های یک شبکه های عصبی عمیق جلو تر میرویم به مدل های پیچیده تر و کامل تری میرسیم.

یادگیری عمیق در حوزه های مختلفی مورد استفاده قرار میگیرد از جمله پیش بینی قیمت بورس و بیت کوین و...

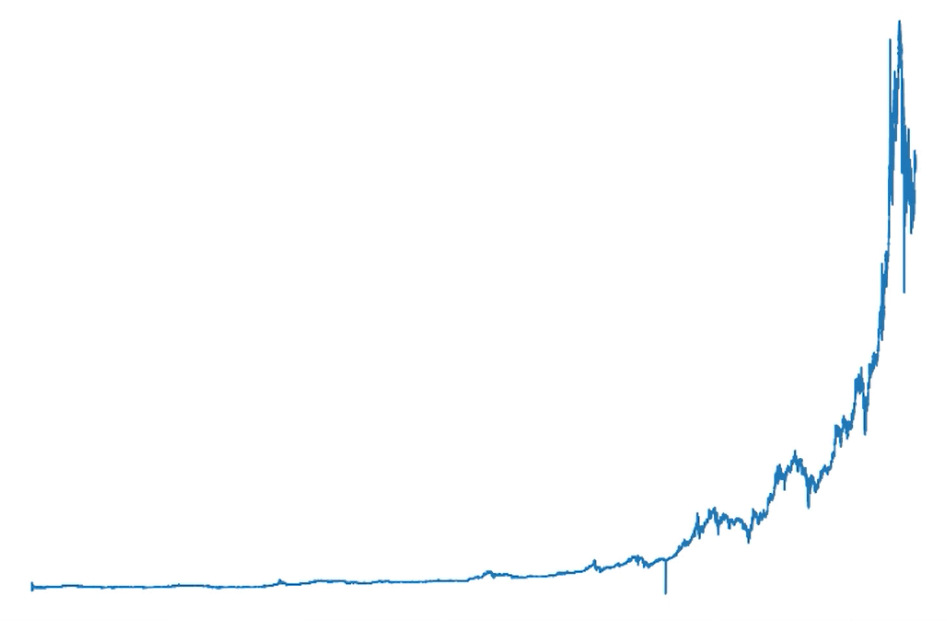
توضیحات پروژه

قرار است data را خودمان پیدا کنیم . از اکانت gitup پیدا کردم که مال خود لینک دانلودش هست و این خوش از یک وبسایتی میگیره که فایل ها را توی 2 تا 3 سال آپدیت میکنه تقریبا نزدیک به یک میلیون و دویست هزار بار قیمت هارو ذخیره کرده با ویژگی های مختلفشون پس data را اینجوری دانلود کردم

بحث libraryها:



Matplotlib.pyplot:اولی فقط برای کشیدن نقشه ها و نمودار بکار میره چون قراره یک نمودار بکشیم مثل این نمودار که کل قیمت هاست و ما قراره یک قسمتی از اون را پیش بینی کنیم

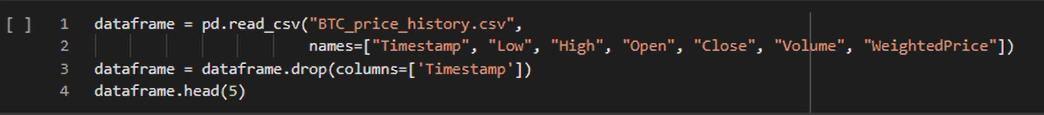


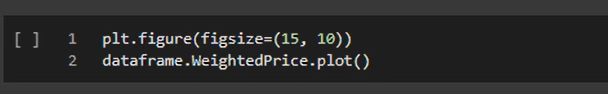
Numpy:برای محاسبات عددی پایتون است مثلا زمانی که data را میخوانیم بصورت text هستند باید آن را تبدبل کنیم به عدد و عدد ها را scale کنیم ببریم توی رنج 0و 1 همچنین امکان ایجاد آرایه و ماتریس و... را به ما میدهد

Pandas:یک کتابخونه پایتون است که برای خواندن فایل های csv است خیلی متد ها دارد که کمک کننده هستند و به صورت یک جدول تمییز برای ما درست میکنه کل عدد هارو

برای انجام تجذیه تحلیل داده ها بکار گرفته میشود و این کتابخانه برای نگه داشتن فایل ها و داده ها بصورت سطر و ستون هستند کاربرد دارد

این ها مربوط به pandas هستند





Os:برای خواندن فایل در فولدر استفاده میشود

Sklearn.preprocessing:برای scale کردن استفاده میشود در اینجا از standarscaler استفاده نکردیم

Sklearn.metrics:در اینجا از r2 استفاده نکردیم چون جواب نمیداد

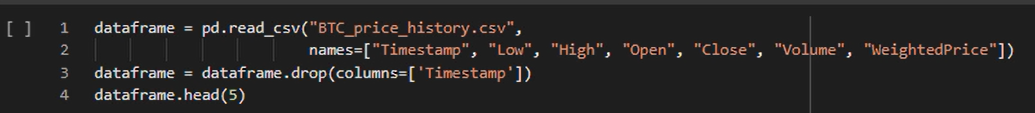
Torch:کل کاری که ما انجام میدهیم چونکه با deep learning بوده library های زیادی داریم ولی ما از torch استفاده میکنیم مثلا حتی توی average گرفتن هم از این استفاده میکنیم

Torch.nn:کل نیورال نتورک ها توی این مثلا linear که رابطه ی خطی میسازد از توی این ساخته میشود

Torch.nn.functional:این هم یکسری متد ها دارد .متد هایی هستند که محاسبات عادی دارند مثل max گرفتن ولی مخصوص شبکه های عصبی ساخته شده که سریع تر کار کنند

Generate\_dataset:

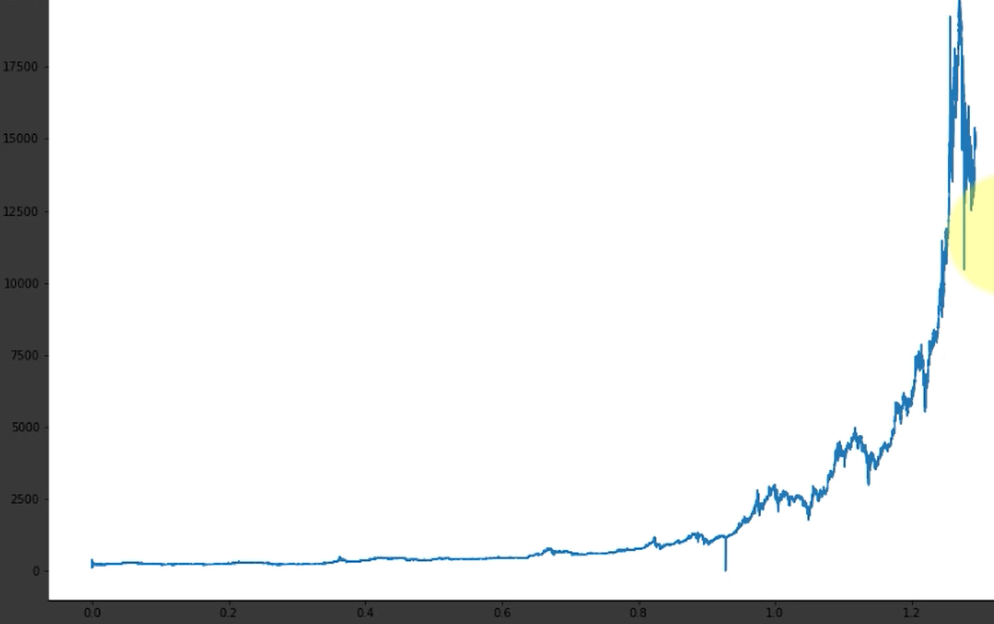
چون فایل های ما به شکل یک فایل csv است کاری که اول از همه میکنیم اینه که data مان را با pd میخوانیم



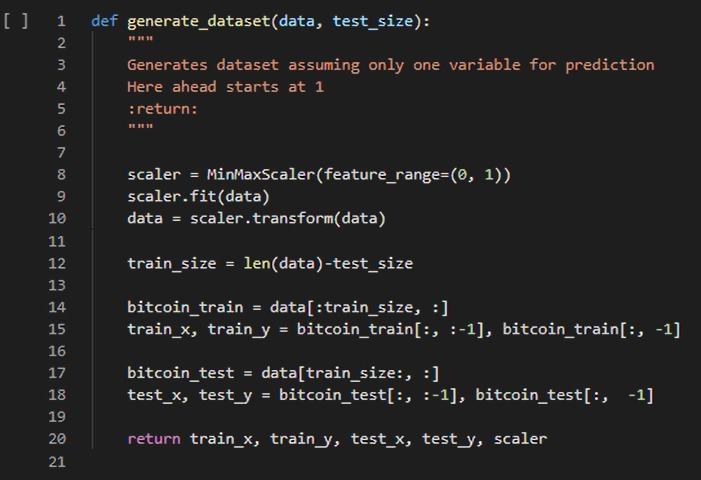
این دقیقا شکل data ما است



برای هر قیمتی که وجود دارد(weightedprice) اطلاعات مربوطه نوشته شده.برای هر کدام از این ها یک تایم زمانی داریم که من حذف کردم چون بدرد ما نمیخورد ما نیازی نداریم بدانیم مثلا توی 2007 چه قیمتی بوده فقط ترتیبشان برای ما مهم است . توی این جدول یکسری متد داریم که برامان مهم نیست و تغییری درشان ایحاد نمیکنیم چون هدف اصلی شبکه عصبی این است که متخصص توی کار نباشیم.



در حقیقت اینجا 1.2 معادل 1200000 یعنی یک و دو دهم میلیون data است.ما فرض میکنیم اینجا را که تا سال 2019 است را میدانیم بعد میخواهیم بدانیم توی 2020 چقدر میشه هدفمان این است. هر کدام از این اعداد داخل جول بالا یک نقطه در داخل نمودار هستند که مثلا اوایل که بیت کوین مشهور نبود قیمتش پایین بود ولی بعد که مشهور شد قیمتش بالاتر رفت.



اتفاقی که می افتد اول از همه minmaxscaler استفاده میکنیم که این می آید min و max data(تمام data ها) را میگیرد مثلا این جا min data 300 و max آن 20000 است پس می آید همه ی data هارا طوری میشکند که 300 بشود 03/0و 20000 بشود 99/. یعنی همه را میریزد بین 0 و 1 به صورتی که همون قدری که اول بودن همان وزن را حفظ کنند به این صورت عمل میکند مثلا فرض کنید ما 1 و 9 داریم اتفاقی که می افتد می شود 1/0 و 9/0 اگر این را scale 2برابر کنیم می شود 2 و 18 که این همان 1/0 و 9/0 است. پس کل هدفمان این است که همه را برگردانیم توی بازه 0و1

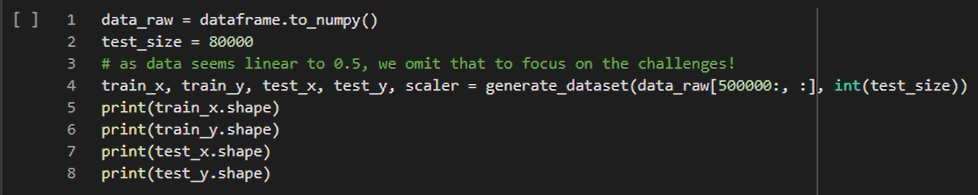
الان رنج volume همیشه جند صدم است در حالی که open یا price توی رنج چند صد هستند بعد به چند 10000 هم میرسند مشکل اینجاست که اگر ما scale نکنیم چون شبکه های عصبی فقط با عدد کار میکنند در واقع ما با عدد باید بهش بگوییم که روی price تمرکز کن به این صورت که بهش میگوییم 2 از 1 بیشتر است پس روی 2 تمرکز کن و 2 برات مهم تر است پس تمرکز ما اینجا روی open و close است و به volume به خاطر رنج کمش کاری نداریم.

برای scale کردن از minmaxscaler استفاده میکنیم.

ین کار هایی را که گفتیم را متد fit انجام میدهد.

Transform داده اولیه مان را میدهیم می آید data مان را می برد توی ویژگی که fit دارد که در اینجا منظور همان 0و1 کردن است.

این Generate\_dataset یک testsize دارد برای این است که چقدر از این 2/1 میلیون data را ببریم به عنوان فثسف استفاده کنیم. اگر testsize را بدهیم 50 درصد 2/1 میشود 6/0 که قطعا به ما جواب غلط میدهد چون اون قسمت پیک کردن های اصلی را اصلا ندیده ما testsize دستی دادیم 80000 دادیم که درواقع آخر های نمودارمان میشود که پیک کردن های اصلی را شامل شود.



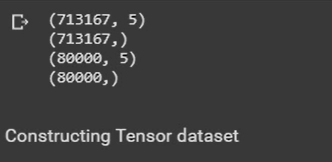
ما از data 50000 به بعد را دیدیم یعنی تا 50000 را ریختیم دور و از آن به بعد را دیدیم چون تا 50000 نوسانات خطی بود و فقط الگوریتم را کند میکرد و سود دیگری نداشت چالش اصلی از 6/0 به بعد است ولی اگر بخواهیم میتوانیم بیشتر ران کنیم.

Bitcoin\_train این کل data منهای 80000 تا (چون مقدار train\_size را 80000 دادیم) را میکند train و بقیه را میکند test

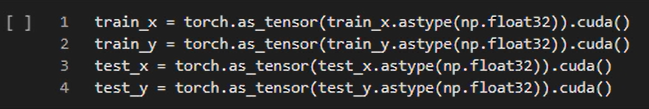
توی نمواری که داشتیم همه ی ستون ها به غیر از ستون آخر(که میشود weighted price) همه میشوند ویژگی های ورودی ما و ستون آخر چیزی است که ما میخواهیم پیش بینی کنیم پس ما 5 تا وردی داریم.

این هم دقیقا همین کار را میکند و اون 1- ها منظورش همان ستون آخر است که همه جا به عنوان test هستند

پ.ن: 1-: یعنی از ستون 0 تا ستون یکی مانده به آخر ولی 1- یعنی ستون آخر



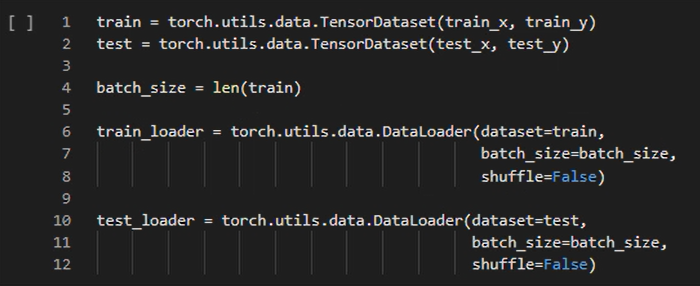
با اون شکانذنی که داشتیم 713167 تا برای train هستند و 80000 تا برای test ورودی یا همان trainsize مان 5 تا ورودی دارد که منظورمان همان 5 ستون است.



کاری که این جا میکنیم کار کردن با torch است. Torch همان np است که برای gpu ساخته شده چون اگر بخواهیم روی cpu ران کنیم خیلی طول میکشد روی gpu ران میکنیم.برای این کار از متدی به نام as\_tensor که توی torch است کمک میگیریم که همان np array را میگیرد و میکند tensor.tensor همان vector بیشتر از 2 بعد است که تا n هم میتواند برود(به vector 2بعد ماتریس میگویند).

چون ما اینجا محدودیت رم داشتیم float32 گذاشتم.

Cuda هم متدی است که باعث میشود داده ها بروند توی gpu به جای cpu که همین باعث سرعت بیشتر میشود.



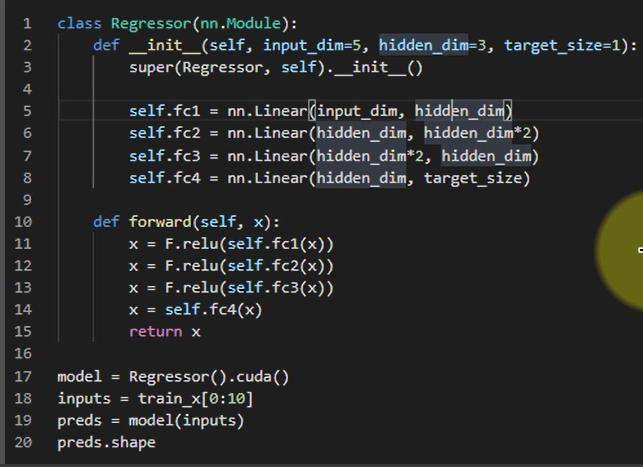
Tensordataset یک کلاس است.

ما یکسری از کار ها مثل بچ کردن data در همین کلاس است.

بچ کردن یعنی وقتی می خواهیم data را train کنیم (در اینجا نیازی نیست) مخصوصا برای تصویر که data خیلی بزرگ است نمیتوان همه را هم زمان داد چون خیلی رم میبرد همه data ها را همزمان نمیشه داد تکی هم نمیشه داد بخش بندی میکنیم به اصطلاح مثلا هر دفعه میگوییم برو 200 تاش را بگیر . Tensordataset و dataloader این کار ها را برای ما میکنند یک کلاس خاص هستند.

میتوان batch\_size داد ولی چون dataset ما کوچک است من batch\_size را کل data دادم.

batch\_size را دو تا درست میکنیم یکی برای train کردن یکی برای test کردن



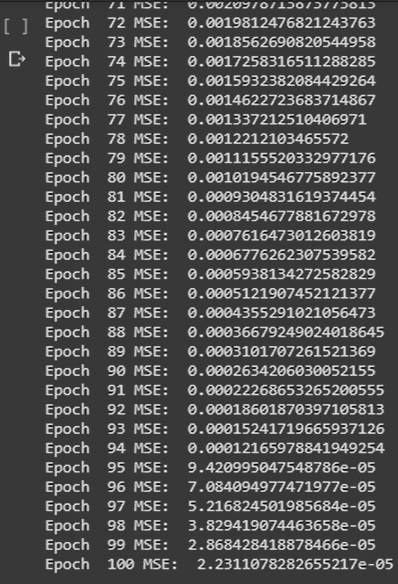
Class regressor این کل اون کلاسی است که داریم کل کار اون شبکه عصبی را برای ما انجام میدهد.

5تا ستون داریم پس ما شبکه هایی که داریم ورودیش مجبور است که 5تایی باشد فقط هم یک لایه ش (لایه ی اول) کافی است بقه را به دلخواه گذاشتم 3 چون شبکه کوچکتر بهتر است سرعتش هم بیشتر است هرجه عدد بزرگتری بزاریم data بیشتر و زمان بیشتر میشود.

Target\_size این است که نهایتا اون شبکه ای که داریم باید چه چیزی را پیش بینی کند که واسه ی ما ستون آخر یا weightedprice است پس برای Target\_size یک value می خواهیم برای هر inputی که میدهیم.

از خط 5 تا 8 در این کلاس معماری شبکه است و خط های 11تا 15 هم همون کار را انجام میدهند برای flow کردن data توی شبکه.

چون همه به هم وصل هستند اسمش را fc گذاشتیم. این شبکه عصبی ماست.ماتریس های ما 1\*1 1\*3 3\*6 6\*3 3\*5 هستند.میتوان data ها را سطر سطر داد و میتوان کل data را با هم داد ما کل data را با هم دادیم.یکبار کل data را میدهیم همه یerror هاش را حساب میکند بعد متغییر ها را برای بهبود error ها آپدیت میکند ولی با یکبار انجام این کار یک عدد را خوب کرد و یک عدد دیگر را خراب کرد پس یکبار کافی نیست انقدر این کار را تکرار میکنیم تا یاد بگیرد مثلا ما 100 بار این کار را تکرار کردیم که اگر error بار اول 15/0 بود توی بار آخر شد 2 دههزارم که این یعنی یادگرفت.



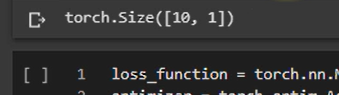
Relu تابعی است که خطی بودن را میشکند.

برای اینکه بتواند هر چیز غیر قطعی را پیش بینی کند ما به یک تابع نیاز داریم به نام activationfunction. این ها قطعا باید غیر خطی باشند.

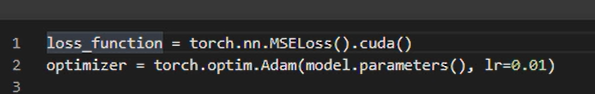
Relu تابعی است که اگر بزرگتر از صفر بود خود data و اگر کوچکتر از صفر بود صفر میشود.به همین خاطر scale را بین 0و1 قرار دادیم میتوانستیم بین 1-و1 هم قرار دهیم ولی دیگر نمیتوانستیم از تابع relu استفاده کنیم چون هیچ data را منفی نمیکرد هر چیز منفی را صفر میکرد و باز همیشه بین 0و1 نشان میداد. این تابع میتواند خطی بودن را بشکند.

پس بعد از خروجی هر کدام از اون fc ها یک تابع relu قرار میدهیم ولی برای آخری نمیگذاریم چون ما به خود چیزی که میدهد نیاز داریم .

وقتی model را میسازیم بهش یک input میدهیم این input همان tensor است و به ما price میدهد.



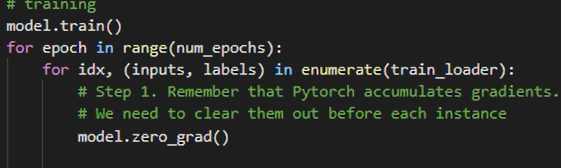
اگر به torch 10 تا ورودی بدهیم این 10 تا value برای ما پیش بینی میکند.



Lost\_function همان error ی است دیدیم.

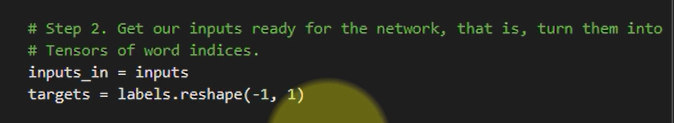
Optimizer را کپی کردم.

Lr یک ضریبی است که هر بار برای تغییر متغییر ها استفاده میشود تا error ها بهبود یابند چون تکرار را زیاد گذاشتیم(100 بار میتوان بیشتر هم گذاشت فقط زمان بیشتر میشود) پرش زیاد لازم نیست .

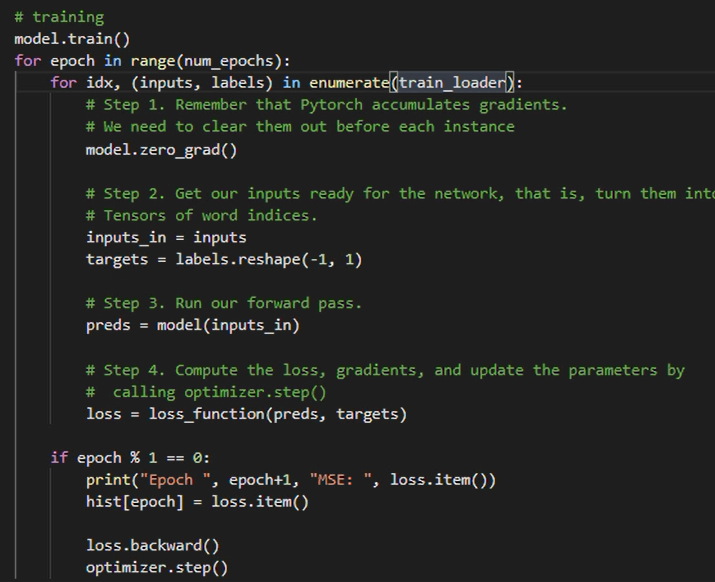


این for یک لیست به ما میدهد .epoch همون کار را چند بار تکرار میکند. این ها کد های paytoch هستند که بهش zero\_grad میگویند.

هر بار که data را میبریم تا انتها برمیگردیم و گرادیان را صفر میکنیم دوباره data را از اول میخوانیم که اگر error قبلی مان اشتباه بود دیگر error برنگردد.



Target همان ستون price است این recheck فقط برای این است که error ندهد.

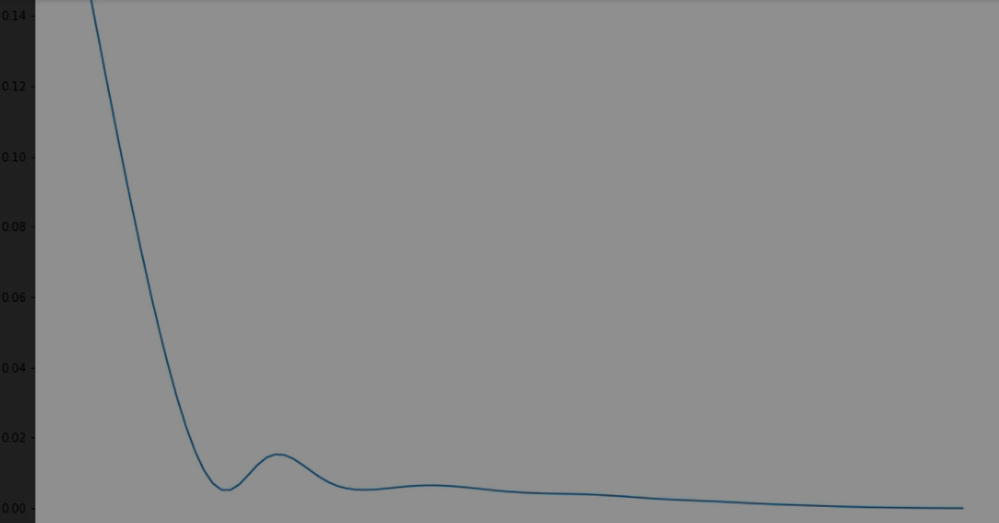


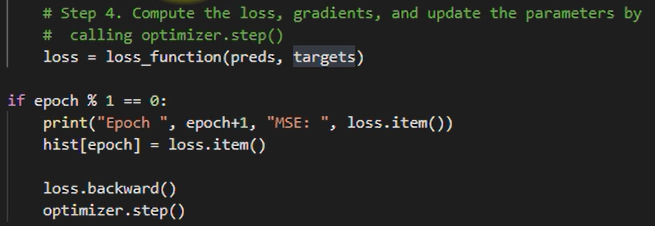
Pred همان model است که input را میدهیم و به ما price را میدهد .

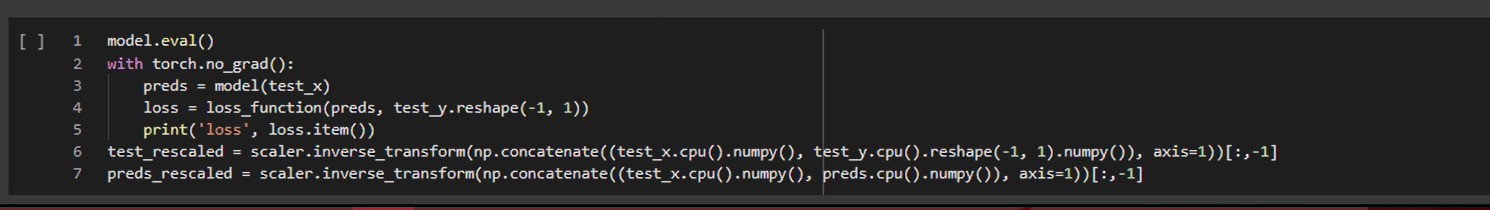
Loss را حساب میکند. در این لحظه pred و target را داریم بعد error را برمیگردانیم در حقیقت ما بهش میگوییم که چه جوری جلو میرویم.hist عمل ذخیره کردن را انجام میدهد

اوایل چون خیلی نیاموخته شیبش زیاد است ولی بعدش منطقی تر میشود.

در این نمودار اگر ما bachsize را که کل data در نظر گرفته بودیم را بگیریم مثلا یک صدم data نمودارمان زیک زاکی میشد.



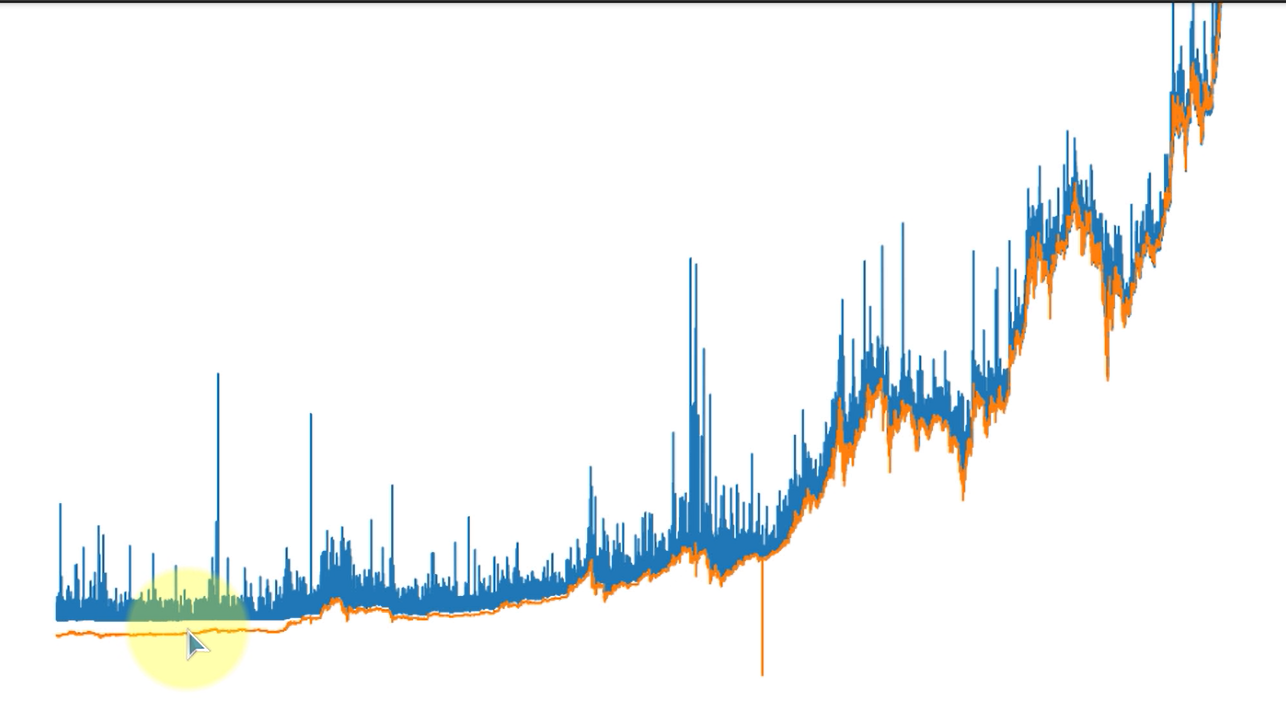




موقعی که میخواهیم model را تست کنیم برای اینکه ram و gpu کمتر مصرف شود میگوییم توی حالت eval هستیم و دیگر برای ما گرادیان حساب نکن فقط برو جلو بعد چون dataمان روی gpu بود میبریم توی cpu بعد numpy میکنیم که بتوان نمودار رسم کرد.

Scaler.inverse اعداد را بین min و max اصلی که در اینجا 300 و 20000 بود قرار میدهد.

Pred یعنی همان چیز هایی که شبکه پیش بینی کرد .



میبینیم که train را دقیقا یاد گرفته چون نمودار پیش بینی شده دقیقا روی هم قرار گرفته اند.