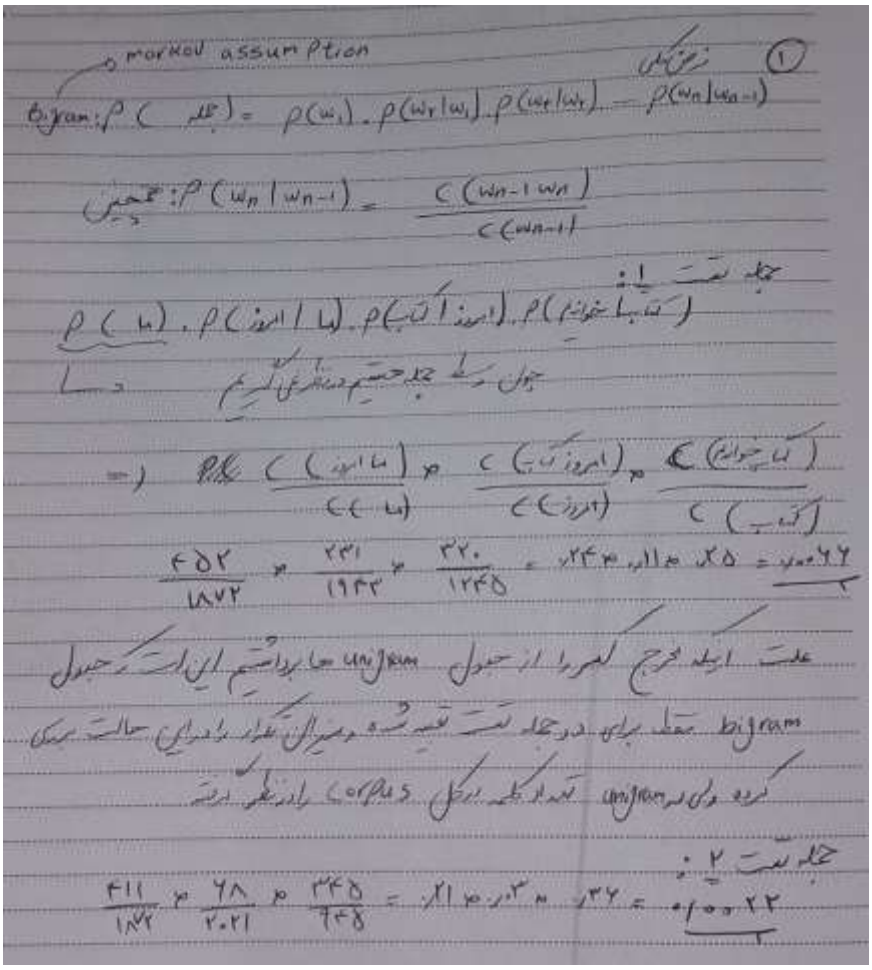
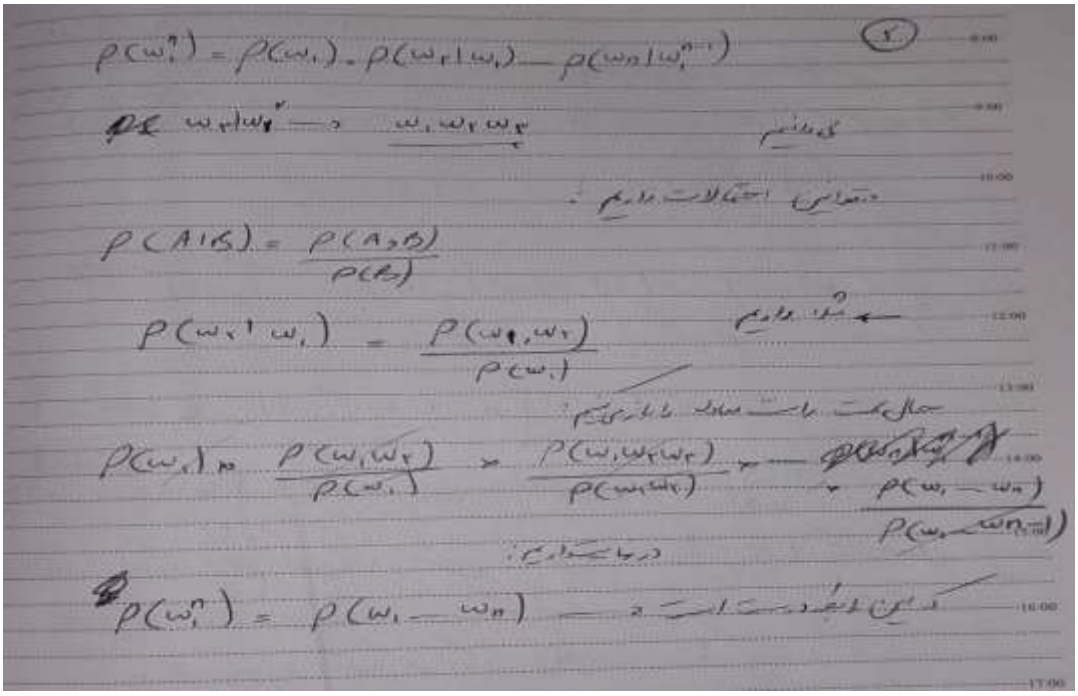
به نام خدا

امیرمحمد کمیجانی ۹۹۵۲۲۰۳۲

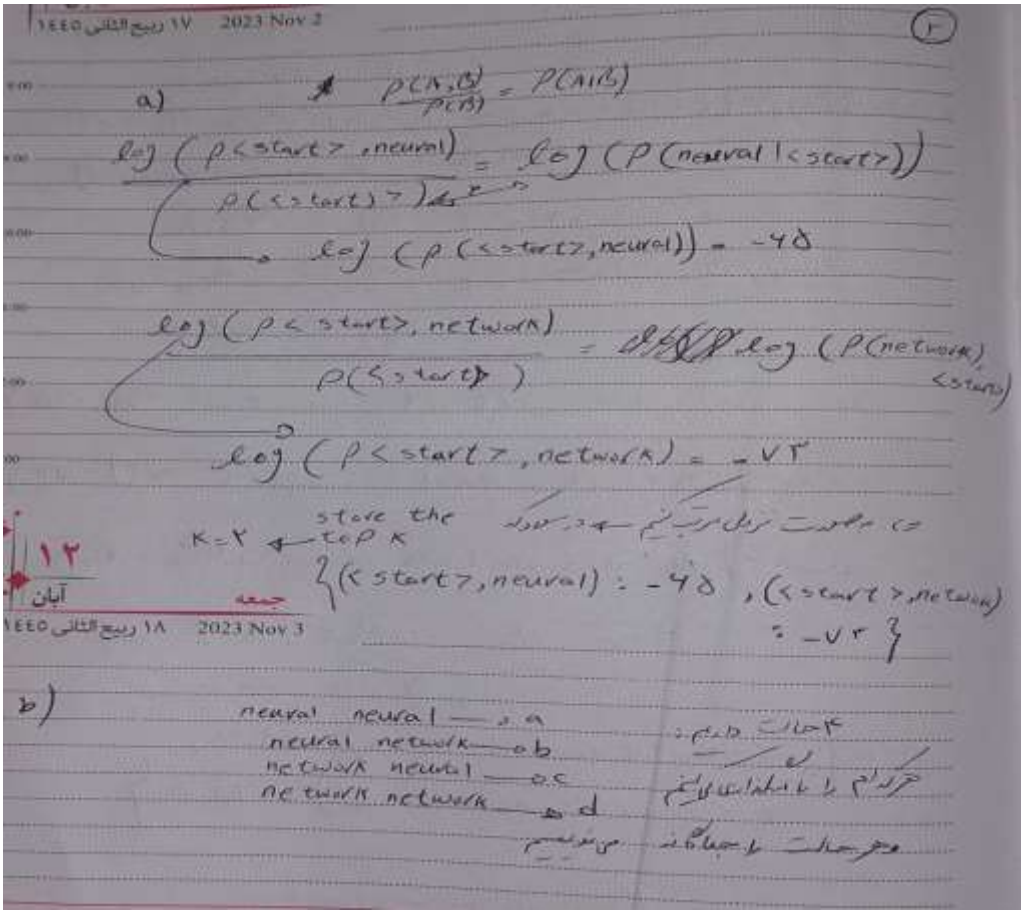
**(۱**



**(۲**



## ۳) قسمت a,b,c



2023 Nov 4

به طور کلی معادلات ما به این صورت می‌شود

a حالت: $\log(P(<start>, a)) = \log(P(neural | <start>, neural)$
$neural, neural \quad * P(<start>, neural)$

$= -۷۸ + (-۴۵) = -۴۵,۸$

برای بقیه حالات طبق الگوریتمی که در جلسه قبل گفتیم داریم و معادلات به شرح زیر به دست می‌آیند:

b حالت) neural, network → $-۷ + (-۴۵) = -۴۵,۷$

c حالت) network, neural → $-۶۱ + (-۴۳) = -۴۴,۶$

d حالت) network, network → $-۷۸ - ۷۳ = -۷۳,۸$

store the top-K, K=2

یعنی این حالت حاضر یکشون کلی:

{(<start>, neural, network): -۴۵,۷ ,
(<start>, neural, neural): -۴۵,۸}

c) ۴ حالت داریم مقادیر قبلی را داشتیم حالا می‌خوایم

neural, neural, neural: $-۴۵ - ۷۸ - ۱۰۱ = -۴۵,۸۱$

neural, neural, network: $-۴۵ - ۷۸ - ۴,۷ = -۷۰,۴$

neural, network, neural: $-۴۵ - ۷۷ - ۷۸ = -۴۴,۴$

neural, network, network: $-۴۵ - ۷۴ - ۷۴ = -۴۴,۲$

store top-K, K=2

{(<start>, neural, neural, neural): -۴۵,۸۱ ,
(<start>, neural, network, network): -۴۴,۲}

d) الگوریتم beam search یک الگوریتم حریصانه میباشد و بهینه حالات در هر مرحله را پیدا کند و بر اساس بهینه ترین حالت در هر مرحله به مرحله بعد میرود.ممکن است در این مثال ما بالاترین احتمال را پیدا نکرده باشیم چون برروی کل نود ها این الگوریتم اجرا نشده است.ممکن است در مرحله ۲ یک قسمت که بالاترین احتمال نباشد انتخاب نشود ولی در حالت کلی و به صورت global به کل مسئله نگاه کنیم راه بهینه ترین و بالاترین احتمال از آن نود بگذرد.پس به طور کلی نمیتوان به قطعیت گفت که این الگوریتم بیشترین احتمال را برای ما میابد.

e) برای دنباله به طول T اوردر زمانی برابر با o(T) میشود برای کلمات با توجه به طول M که در vocab داریم اوردر زمانی o(M) را داریم و برای k که در هر مرحله طبق سودوکد داریم store the top k و باید بالاترین ها را حفظ کنیم اوردر زمانی o(k * log(M)) را داریم.

در نهایت : O(T*M*k*log(M))

منبع کمکی برای این قسمت :

(e) **What is the runtime complexity of generating a length-$T$ sequence with beam size $k$ with an RNN?** Answer in terms of $T$ and $k$ and $M$. (Note: an earlier version of this question said to write it in terms of just $T$ and $k$. This answer is also acceptable.) **Solution:**

- Step RNN forward one step for one hypothesis = $O(M)$ (since we compute one logit for each vocab item, and none of the other RNN operations rely on $M$, $T$, or $K$).
- Do the above, and select the top $k$ tokens for one hypothesis. We do this by sorting the logits: $O(M \log M)$. (Note: there are more efficient ways to select the top $K$, for instance using a min heap. We just use this way since the code implementation is simple.) Combined with the previous step, this is $O(M \log M + M) = O(M \log M)$.
- Do the above for all $k$ current hypotheses $O(KM \log M)$.
- Do all above + choose the top $K$ of the $K^2$ hypotheses currently stored: we do this by sorting the array of $K^2$ items: $O(K^2 \log(K^2)) = O(K^2 \log(K))$ (since $\log(K^2) = 2\log(K)$). (Note: there are also more efficient ways to do this). Combining this with the previous steps, we get $O(KM \log M + K^2 \log(K))$. When one term is strictly larger than another we can take the max of the two. Since $M \geq K$, we could also write this as $O(KM \log M)$.
- Repeat this for $T$ timesteps: $O(TKM \log M)$.

**۴)**

**a)** ابتدا یکبار تعریف میکنیم که هر gate دقیقا چه نقشی را ایفا میکند.

- **Forget gate** : وظیفه این گیت مدیریت کردن دیتای گذشته میباشد و به طور کلی مدیریت میکند چه مقدار از دیتای گذشته را وارد استیت فعلی بکنیم.
- **Input gate** : این گیت وظیفه کنترل دیتای جدید را دارد و مدیریت میکند چه مقدار از دیتای جدید وارد cell state شود.
- **Output gate** : به طور کلی مثل یک فیلتر برای خروجی نهایی عمل میکند و خروجی های ما را در LSTM unit کنترل میکند.

حال با توجه به این موارد که بیان شد؛ مشخص است حذف هر کدوم چه تاثیری میگذارد. اگر گیت های input , output را حذف کنیم باعث میشود که LSTM unit ما نتواند خود را با دیتای جدید تغذیه کند و بر طبق آن عملیات ها را انجام دهد و نتواند خروجی های مناسبی را فیلتر و خروجی دهد. به طور کلی با نبود این دو گیت ما دوباره یک RNN ساده داریم با این تفاوت که به دلیل وجود گیت forget حافظه آن را میتوانیم کنترل کنیم و ظرفیت محدودتری اختصاص میدهیم.

هر دو گیت input , output برای حل مشکل اصلی RNN یعنی vanishing gradient ضروری هستند.

**b)** میدانیم هر یک از گیت ها میتوانند مقادیر از 0 تا 1 به خود بگیرند. هر چقدر به ۱ نزدیکتر باشیم هر گیت بیشتر تاثیر گذاری دارد.

- مثلا اگر مقدار گیت ورودی به ۱ نزدیک باشد اجازه میدهیم دیتای جدید تاثیر بیشتری داشته باشند و بالعکس اگر به صفر نزدیک باشند این تاثیر را عملا از بین برده ایم.
- در رابطه به گیت forget نیز به همین صورت است ؛ هر چقدر این مقدار به ۱ نزدیک باشد LSTM مقادیر بیشتری از گذشته را حفظ میکند و اگر این مقدار به صفر نزدیکتر شود ما بیشتر اطلاعات گذشته را forget میکنیم.

همانطور که در صورت سوال ذکر شده است با قرار دادن مقدار صفر برای گیت forget عملا تمام دیتای گذشته را نادیده میگیریم و فراموش میکنیم.

در اثر فراموشی کامل ما دوباره دچار مشکلات RNN میشویم:

- Long term dependency : با حذف کردن گیت forget دوباره کنترلی بر روی اینکه چه مقدار از دیتای گذشته باید استفاده شود نداریم
- Vanishing gradient : مشکل اصلی در RNN

و به طور کلی هر ورودی جدیدی که داریم از دیتای گذشته هیچ استفاده ای نداریم و ورودی جدید شبیه به ورودی ای که هیچ گاه آنرا ندیده ایم و محاسباتی بر روی آن نداشتیم میشود و محاسبات دوباره تکرار میشود.

همچنین در فرآیند آپدیت کردن cell state گیت forget نقش دارد که با اعمال مقدار 0 به آن فرآیند آپدیت نیز دچار مشکل میشود.

برای پیش بینی نیز این مشکل به وجود می آید؛ که اگر ما به متن یا دیتای گذشته قبل از دیتای فعلی نیاز داشته باشیم و بر اساس آن بخواهیم پیش بینی انجام دهیم یا پیش بینی خود را بهتر کنیم، این امکان را نداریم و پیش بینی های ما بسیار نا دقیق میشوند.

**c)**

**مزایا :**

- اضافه کردن لایه ها میتواند باعث این شود که مدل ظرفیت بیشتری برای یادگیری complex representation ها داشته باشد
- از دیگر مزایای آن feature extraction میباشد به طوری که هر لایه میتواند فیچر های خاصی را بدست آورد و به طور مثال لایه های ابتدایی فیچر های راحت تر و لایه های بعدی و عمیق تر فیچر های سخت تر و پیچیده تری را کشف کند. مثلا اگر یک تسک در همین حوزه NLP را بخواهیم بررسی کنیم لایه های ابتدایی میتواند ساختاری های اساسی و پایه ای زبان را یاد بگیرند که جز مراحل ابتدایی میباشند ولی لایه های عمیق تر میتوانند روابط معنایی بین کلمات را بیابند.
- استفاده از لایه های بیشتر به این صورت است که ما یک ساختار hierarchial داریم که اجازه میدهد هم یادگیری short term و هم long term را داشته باشیم.

**معایب :**

- **هزینه محاسباتی بیشتر :** مشخص است افزایش لایه ها محاسبات بیشتری را برای ما به همراه دارد.
- **Overfitting :** شبکه های عمیق احتمال رخداد این اتفاق در آنها بیشتر است.
- **Vanishing and exploding gradients :** که باعث دشواری در آموزش و یادگیری مدل میشوند و باید پارامتر هایی را تعریف کنیم تا بتوانیم این مشکلات را رفع کنیم.

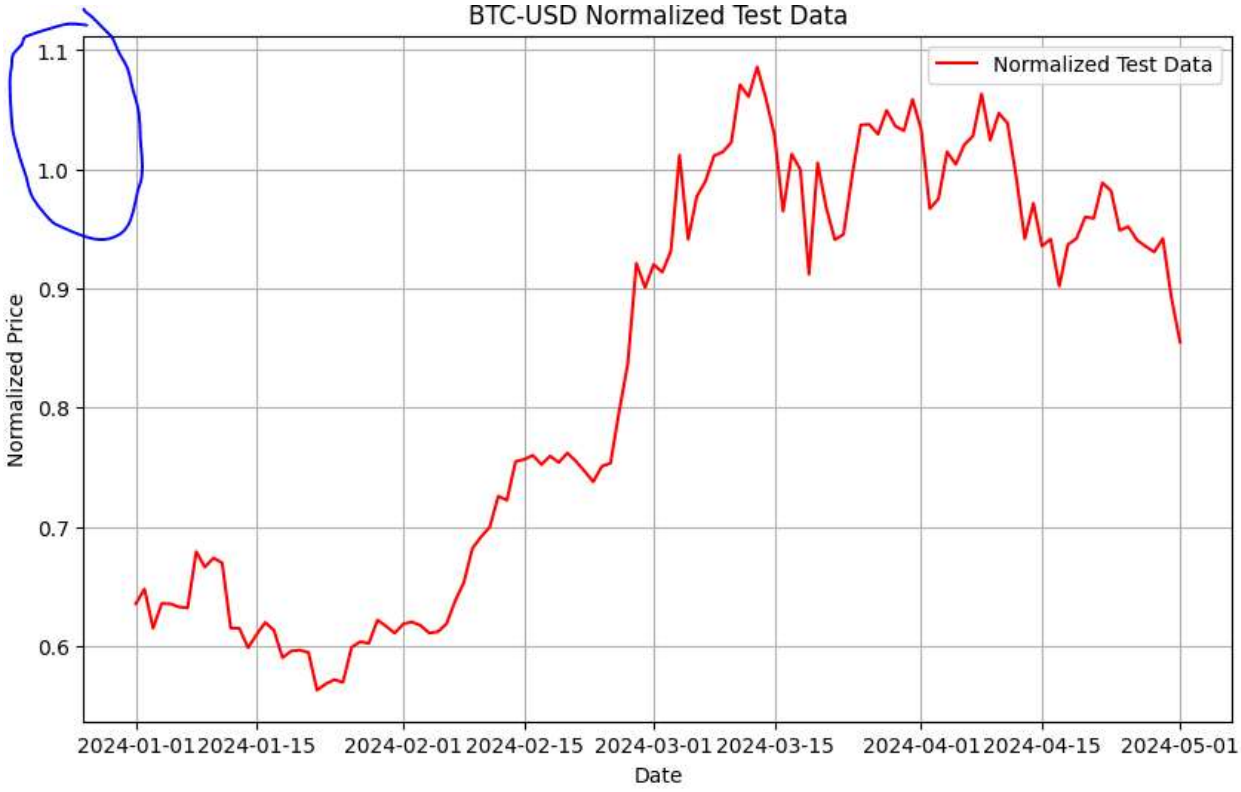
## قسمت عملی :

**۵)**

**توضیح کد ( قسمت a تا h)**

ابتدا کتابخانه yahoo finance را نصب کردیم و تمام کتابخانه های مورد استفاده را ایمپورت کردیم. در قسمت بعدی دیتای train , test را به صورت جداگانه دانلود کردیم و نمایش دادیم. و در مرحله بعد plot مربوط به هر دو را نمایش دادیم.

در مرحله بعد هر دو داده را نورمالایز کردیم.به این ترتیب که ابتدا آبجکتی از MinMaxScaler ساختیم و با استفاده از آن آبجکت یا scaler ابتدا با تابع fit_transform دیتای train را نورمالایز کردیم و بعد بر اساس آن و با استفاده از تابع transform دیتای تست را نورمالایز کردیم. و پلات هر دو را نمایش دادیم.




با توجه به اینکه مقدار ماکسیموم در نورمالایز کردن ۱ میباشد علت اینکه در اینجا 1.1 داریم این است که ما بر اساس دیتای train عملیات scale کردن را انجام دادیم و معیار ما نورمالایز شده دیتای train بوده است که در برخی مقادیر تست از حداکثر آن عبور کرده ایم.

در مرحله بعد باید دیتای خود برای test, train را آماده کنیم. این کار را با استفاده از تابع create dataset انجام دادیم. اساس کار تهیه دیتا به این صورت است که ابتدا دیتای نورمالایز شده و سپس مقدار look back را به آن پاس میدهیم.منظور از look back پنجره زمانی است که ما در آن گذشته را در نظر میگیریم.
مقدار خروجی نیز همانند خروجی ای است که در داک سوالات آورده شده است. به این صورت است که پنجره را به صورت مثلا 1 تا ۶۰ بعد ۲ تا ۶۱ و به همین صورت تا ۱۷۶۶ تا ۱۸۲۶ در نظر میگیریم.(تعداد دیتایی که برای train داشتیم 1826 بوده).

پس از تهیه دیتا نوبت به این مدل مشابه با شکل داده شده را طراحی کنیم. ابتدا به صورت sequential مدل را طراحی میکنیم سپس lstm unit ها و مقادیرdropout را در هر مرحله به آن اضافه میکنیم.همچنین مدل را با Adam optimizer و با تابع خطای ذکر شده در سوال کامپایل میکنیم.و در نهایت برای fit کردن مدل تعداد epoch , batch را نیز برای آموزش در نظر میگیریم. در این مرحله تعداد لایه ها را ۴ در نظر میگیریم.ابتدا لایه اول را با مقادیر ورودی میسازیم و بعد دو لایه میانی یعنی لایه دوم و سوم را طراحی میکنیم و طبق شکل پیش میرویم و در نهایت لایه نهایی را میسازیم.

در قسمت بعدی دیتاست مربوط به دیتای تست را میسازیم سپس عملیات predict را انجام میدهیم سپس برای اینکه به طور دقیق ببینیم چه مقادیری از قیمت واقعی را پیش بینی کرده ایم مقادیر scale شده را برعکس میکنیم تا به مقدار واقعی قیمت برسیم و سپس در انتها پلات مربوط به پیش بینی های خود و مقادیر واقعی قیمت را نمایش میدهیم.

(i

افزایش تعداد روز ها باعث میشود که تعداد بازه هایی که داریم کمتر شود ولی طول بازه ها طولانی شود به این ترتیب در مرحله آموزش مدل ما تعداد بازه های کمتری دارد که طبق آن اموزش ببیند و خود را بهبود بخشد اما اگر کاهش تعداد روزها را داشته باشیم باعث میشود طول

بازه ها کوتاه تر اما تعداد بازه ها افزایش یابد و مدل نمونه های بیشتری برای یادگیری داشته باشد.

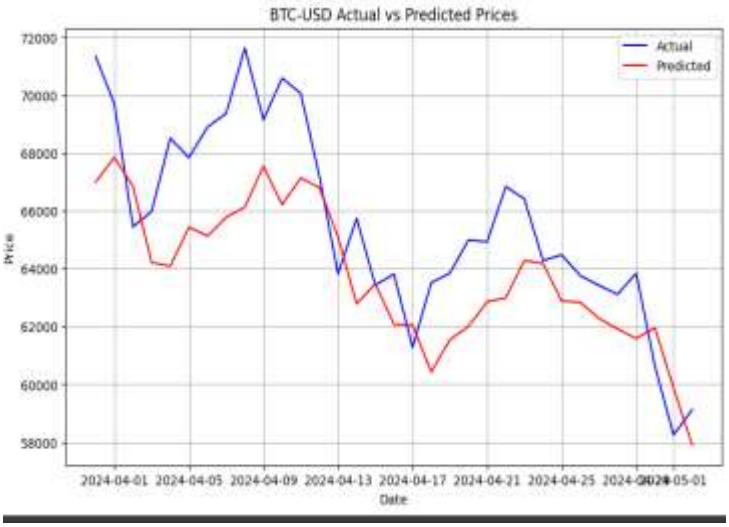
به صورت کلی کاهش و کمبود دیتا میتواند مزایا و معایب زیر را داشته باشد:

- آموزش سریعتر ، کاهش محاسبات (مزایا)
- ریسک overfitting بالاتر میرود ، مدل در برابر دیتای پرت حساس تر است ، تعمیم پذیری کاهش میابد.(معایب)

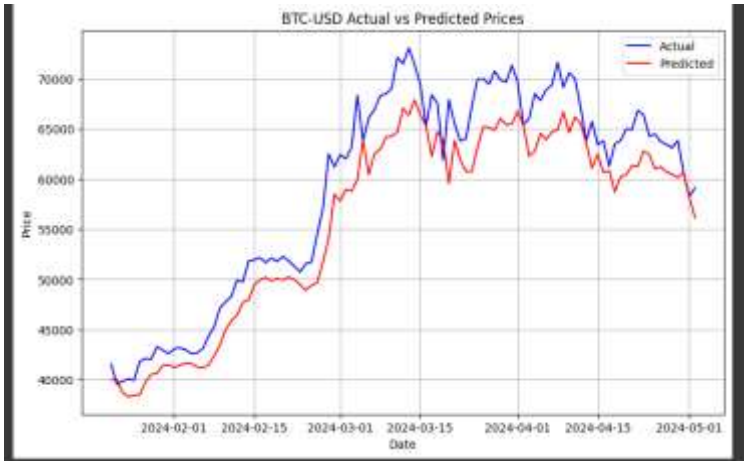
همچنین افزایش دیتا میتواند مزایا و معایب زیر را به همراه داشته باشد :

- بهبود تعمیم پذیری مدل ، stability بیشتر ، پیش بینی های دقیق تر به دلیل وجود دیتای بیشتر برای یادگیری ( مزایا)
- هزینه محاسبات بالاتر میرود ، جمع آوری دیتا سخت تر میشود(در این مثال به دلیل وجود دیتا ممکن است مشکل نخوریم) ، مدل به حدی میرسد که در برابر دیتای جدید خیلی sensitive نباشد و اصلا به دنبال تغییر نباشد که اصلا مناسب نیست.(معایب)

تصاویر زیر نشان دهنده عملکرد بهتر با طول روز کمتر هستند :




به ازای look back =90




به ازای look back = 20

که نشان میدهد مقدار پیش بینی ما به مقدار واقعی نزدیک تر است و فاصله کمتر میباشد.