دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی مهندسی کامپیوتر

گرایش فناوری اطلاعات

پیاده سازی مدل پیشبینی قیمت سهام با استفاده از یادگیری عمیق

نگارش:

مصطفی کریمی

استاد راهنما:

دکتر محمد طهماسبی

شهریور ۱۴۰۰


بسم الله الرحمن الرحیم

تقدیم

تقدیم به هر آنکه در جستجوی دانش و کسب راهی برای ارتقا خویشتن و دیگران میباشد.

تشکر و قدردانی

از خداوند متعال که همواره مرا مورد لطف و عنایت قرار داده یادش باعث آرامش و تمکین خاطر بوده و صلاح و مصلحتش همواره راهگشای زندگی بندگانش می باشد.
از استاد گرامی جناب آقای دکتر طهماسبی بسیار سپاسگزارم که این فرصت را در اختیار من گذاشتند و همواره بنده را از حمایت‌های خویش بهره‌مند نمودند. .

چکیده

حجم تبادلات مالی که هر روزه در بازارهای مالی اتفاق میافتد رقمی بسیار چشمگیر میباشد که انسان های زیادی را ترغیب به فعالیت در این بازار میکند همانطور که شاهد بودیم با شیوع گسترش ویروس کرونا و به هم خوردن بسیاری از کسب و کارهای فعال و لزوم اعمال سیاست هایی از جمله قرنطینه سراسری در تمام دنیا و از جمله کشورمان تمایل مردم به سرمایه گذاری در این بازار های مالی زیاد شد و حجم و تراکنش های قابل تاملی روزانه در این بازار ها انجام میشود لذا استفاده از هوش مصنوعی و بکارگیری توانایی محاسباتی کامپیوتر ها در جهت کسب بهترین بهره در این بازارها مورد توجه خاصی قرار گرفت در این پروژه برآنیم تا با استفاده از پیاده سازی مدلی برای پیشبینی روند قیمتی بر مبنای یادگیری عمیق می باشد، که به وسیله زبان برنامه نویسی پایتون و استفاده از کتابخوانه Keras و ساخت یک شبکه عصبی بازگشتی مبتنی بر حافظه طولانی کوتاه-مدت LSTM پیاده سازی شده است .
لذا دیتاهای فراهم شده در این برنامه مربوط به بازار سهام ایران بوده و سعی شده تا دقت و صحت مدل پیاده سازی شده را بسنجیم و درنهایت شکل نموداری آن را در هر مرحله نشان دهیم.

کلیدواژه: یادگیری عمیق، شبکه عصبی، بازارهای مالی، پایتون

فهرست مطالب

عنوان **صفحه**

فهرست جدول‌ها

عنوان **صفحه**

فهرست شکل‌ها

عنوان **صفحه**

مقدمه

۱-۱- پیشگفتار

حجم تبادلات مالی که هر روزه در بازارهای مالی اتفاق میافتد رقمی بسیار چشمگیر میباشد که انسان های زیادی را ترغیب به فعالیت در این بازار میکند همانطور که شاهد بودیم با شیوع گسترش ویروس کرونا و به هم خوردن بسیاری از کسب و کارهای فعال و لزوم اعمال سیاست هایی از جمله قرنطینه سراسری در تمام دنیا و از جمله کشورمان تمایل مردم به سرمایه گذاری در این بازار های مالی زیاد شد و حجم و تراکنش های قابل تاملی روزانه در این بازار ها انجام میشود لذا استفاده از هوش مصنوعی و بکارگیری توانایی محاسباتی کامپیوتر ها در جهت کسب بهترین بهره در این بازارها مورد توجه خاصی قرار گرفت.

امروزه مسائلی که بر محور سری های زمانی میباشد مورد توجه بوده از جمله مسائلی مانند پیشبینی قیمت سهام به طور خلاصه هدف اصلی در تحلیل سری های زمانی، ایجاد یک مدل آماری برای داده های وابسته به زمان بر اساس اطلاعات گذشته آن پدیده است به بیان دیگر در سری های زمانی برآنیم تا با ایجاد مدلی گذشته نگر امکان تصمیمات آینده را میسر سازیم.

در این پروژه در ابتدا سعی برآن شد تا با فراهم کردن دیتا های مربوط به بازار سهام ایران و تهیه و پردازش آنها برای استفاده از مدل های یادگیری عمیق سعی بر گرفتن خروجی متناسب و معقولی بود که پس از بررسی مدل بسیاری از جمله شبکه‌های عصبی پیچشی (CNN)[۱] شبکه عصبی بازگشتی(RNN) [۲]، میانگین متحرک همبسته یکپارچه (ARIMA)[۳] بر آن شدیم تا بهترین مدل برای پیشبینی را شبکه عصبی بازگشتی مبتنی بر حافظه طولانی کوتاه-مدت (LSTM)[۴] قرار داده و به بررسی و صحت نتایج بپردازیم

[۱] convolutional neural network

[۲] recurrent neural network

[۳] AutoRegressive Integrated Moving Average

[۴] Long short-term memory

۱–۲– شیوه‌های نوین به‌کاررفته در این پروژه

در این پروژه سعی بر آن شده است که کمترین میزان کد نویسی مدلی به وجود آورده و در هر قسمت از فرایند های یادگیری دقت و عملکرد مدل را به صورت نموداری نشان داده و همه‌ی فرایند های مربوطه را در محیط گوگل کولب[۱] انجام و خروجی را نمایش دهیم لازم به ذکر است مدل پیاده سازی شده بهترین مدل با کمترین میزان خطا از مدل هایی که در طول فرآیند پیاده سازی این پروژه تست شده میباشد.

۱–۳– هدف از انجام این پروژه

همان‌طور که در قسمت پیشگفتار بیان شد، وجود گردش مالی فراوان و امکان کسب و بهره‌ی مالی در بازار های مالی افراد گوناگونی را ترغیب به فعالیت در این بازار ها میکند. همچنین با وجود گسترش روز افزون تکنولوژی و وابستگی بیش از اندازه جامعه بشری به آن موضوع فراگیری هوش مصنوعی و مشتقات آن امری غیر قابل چشم پوشی میباشد لذا سعی شده در این پروژه با رویکرد پژوهشی-عملی اقدام به فعالیت نموده تا علاوه بر ارتقاء و یادگیری در این حوزه جذاب و فراگیر نتیجه خروجی آن را بتوان به صورت عملی استفاده کرد .

[۱] Google Colab

فصل ۲– مطالعات صورت گرفته و جمع‌آوری داده‌های موردنیاز

۲–۱– مقدمه

برای انجام این پروژه دانستن برخی از مسائل و موضوعات در یادگیری ماشین و شبکه‌های عصبی و مدل های مختلف آن نیاز بود و همچنین یکی از مشکلات پروژه در پیاده سازی این بود که پیشبینی قیمت در بازار های مالی به عوامل متعددی بستگی دارد و همین عوامل متعدد بسیار کار را برای ساخت مدلی که بتوان آن را مطمعن و سودآور دانست بسیار سخت کرده است. در کل بیشترین زمان اختصاص داده شده برای پیاده سازی پروژه مربوط به آموزش پیشنیاز های آن بوده که لزوم آن فراگیری مطالب مربوط به یادگیری ماشین ، یادگیری عمیق ، داده کاوی بوده است

۲–۲– مطالعات صورت گرفته

همان‌طور که بیان شد پیش‌نیاز انجام این پروژه دانستن برخی مفاهیم در یادگیری ماشین و همچنین شبکه‌های عصبی بود. با استفاده از فیلم‌های متعدد در یوتیوب همچنین مطالعات پراکنده از مقالات متفاوت در مورد شبکه‌های عصبی و نحوه کارکرد آن این قسمت از کار کامل شد و دانش موردنیاز برای پروژه کسب شد در انتها به فهرست مرجع برای یادگیری اشاره میشود.

۲–۳– جمع‌آوری داده‌های موردنیاز

یکی از مهمترین بخش تهیه داده مناسب و پیش پردازش[۱] آن میباشد داده های مربوط به بازار سهام ایران از برنامه TSE Client گرفته شده و پس متناسب سازی بعضی از ستون های مربوطه پردازش آن ها فایل نهایی که با پسوند csv ذخیره گردیده و به عنوان داده ورودی برای برنامه استفاده شده است .

[۱] Pre Process

فصل ۳– معماری منطقی و فیزیکی

۳–۱– مقدمه

آنچه در این قسمت تحت عنوان معماری منطقی و فیزیکی بحث خواهد شد، در مورد محیط و زبان برنامه‌نویسی، محیط اجرای برنامه و نحوه ی گرفتن خروجی از آن میباشد .

۳–۲– معماری منطقی

این پروژه با استفاده از زبان برنامه‌نویسی پایتون و همچنین با استفاده از کتابخوانه کراس[۱] در محیط گوگل کولب کد نویسی شده است.

یک کتابخانه رایگان منبع باز قدرتمند و با کاربرد آسان برای توسعه و ارزیابی مدل های یادگیری عمیق است . کراس دو کتابخانه یادگیری ماشین عددی Theano

و تنسورفلو [۲]را پوشش می دهد و به شما امکان می دهد فقط در چند خط کد، مدل های شبکه عصبی را تعریف و آموزش دهید.

۳–۳– مزایای کتابخانه keras در پایتون

- مدل های آماده
- پشتیبانی توسط شرکت های بزرگ نظیر Google, Microsoft, Amazon, Apple, Nvidia,
- منعطف و قابل تغییر
- قابلیت اجرا در پلتفرم هایiOS, Android, web API
- سریع بودن
- به وضوح بیان کردن خطاها

[۱] keras

[۲] TensorFlow

۳-۴- معماری فیزیکی

به دلیل محدودیت‌های فیزیکی و به خاطر اینکه آموزش شبکه عصبی با استفاده از GPU بسیار سریع‌تر از CPU هست، نیاز به GPU به مراتب سریع تر و به صرفه تر بود اما به دلیل اینکه قابلیت آموزش دادن با استفاده از کارت گرافیک‌های امکان‌پذیر نبود، بنابراین از محیط ابری گوگل کولب که ساختاری شبیه ژوپیتر [۱]نوت بوک دارد، برای آموزش شبکه، اجرای برنامه و گرفتن خروجی استفاده شد.

[۱] jupyter

فصل ۴– الگوریتم کلی برنامه و کدهای اجرایی مربوطه

۴–۱– مقدمه

در این فصل با الگوریتم کلی برنامه نحوه پیاده سازی و مفاهیم پشت آن همچنین کد های مربوطه به هر بخش به طور مجزا آشنا میشوید.

در ابتدا نیاز از مفاهیم کلی توضیح داده شوند و پس از بررسی به شرح حال کلی برنامه میپردازیم.

۴–۲– یادگیری ماشین چیست

یادگیری ماشین یک فرایند عملی استفاده از الگوریتم های مختلف جهت آنالیز و یادگیری داده ها برای مشخص کردن یا پیشبینی داده‌ی جدید میباشد

۴–۳– یادگیری عمیق چیست

یادگیری عمیق یک زیرشاخه از یادگیری ماشین است که از الگوریتم هایی استفاده میکند که از ساختار و عملکرد شبکه های عصبی مغز الگو گرفته اند.

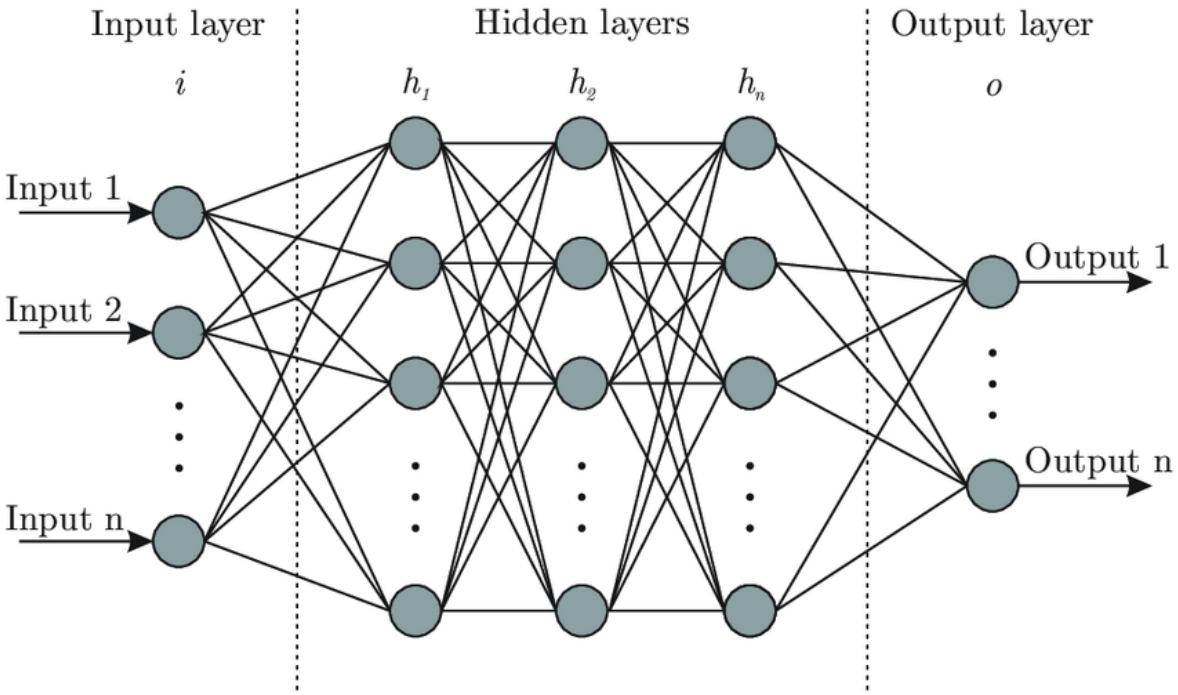
۴–۳–۱– شبکه های عصبی مصنوعی:

بهتر است بگوییم یک سیستم محاسباتی هستند که از شبکه های عصبی مغز الگو گرفته اند در ادامه به بررسی نحوه پیاده سازی، عملکرد و چند مدل معروف آن میپردازیم

۴-۳-۱-۱- شبکه های عصبی چه گونه کار میکنند ؟

در حالت کلی شبکه های عصبی داده ها را دریافت و در لایه‌های مخفی خود آن‌ها را تحلیل می‌کنند تا نهایتا یک خروجی ارائه بدهند. این داده‌ها می‌توانند گروهی از تصاویر، صداها، نوشته‌ها و... باشند که باید ترجمه و برای یک ماشین قابل درک بشوند. به کمک شبکه‌های عصبی، اطلاعات را طبقه‌بندی می‌کنیم؛ اطلاعات مختلف می‌توانند بر اساس شباهت به مثالی مشخص، گروه‌بندی شوند. آن‌ها حتی می‌توانند

امکانات و داده‌های لازم برای تغذیه به یک الگوریتم دیگر را هم فراهم و طبقه‌بندی کنند . معماری کلی آن ها از سه لایه ورودی[۱] ، مخفی[۲] و خروجی [۳]تشکیل شده است .



شکل ۴-۱ ساختار شبکه عصبی

۴-۳-۱-۲- روش‌های یادگیری شبکه‌های عصبی عمیق

شبکه‌های عصبی نمی‌توانند به طور مستقیم برای یک موضوع برنامه نویسی شوند. درست مانند رشد کردن مغز یک کودک که برای یادگیری به یک سری پیش‌نیاز لازم دارد. برای همین هم نمی‌توان گفت برای یادگیری شبکه‌های عصبی چقدر زمان لازم است.

۴-۳-۱-۳- یادگیری با نظارت:(Supervised Learning)

در روش یادگیری با نظارت که به آن روش دسته‌بندی [۴]هم می‌گویند، انسان اطلاعات دسته‌بندی‌شده خود را به صورت یک مجموعه داده، در اختیار شبکه عصبی قرار می‌دهد. در این حالت انتظار می‌رود که شبکه عصبی، رابطه‌ی خروجی و داده‌های ورودی را پیدا کند.

---

[۱] input

[۲] hidden

[۳] Output

[۴] Classification

از این روش برای اهداف مختلفی چون:

- تشخیص چهره فرد در یک تصویر و حالت صورت او (مثلا خوشحال یا عصبانی)
- تشخیص اشیای مختلف در تصاویر
- تشخیص حرکات در ویدیو
- شناسایی صدا، تبدیل صوت به متن، و حتی تشخیص احساسات در صدا
- دسته‌بندی ایمیل‌ها (مثلا مثلا اسپم بودن یا نبودن یک ایمیل)

استفاده می‌شود.

۴-۳-۱-۴- یادگیری بدون نظارت:(Unsupervised Learning)

در یادگیری بدون نظارت که به آن روش خوشه‌بندی[۱]هم می‌گویند، هدف پیدا کردن مشترکات است. در این روش، مشخصه معینی برای شناسایی شباهت‌ها وجود ندارد .در واقعیت هم اکثر داده‌های موجود در دنیا، به صورت کلاس‌بندی‌نشده[۲]هستند. یک قانون کلی در یادگیری ماشین می‌گوید: هر چقدر دیتای ورودی که یک الگوریتم می‌خواهد از آن پیروی کند بیشتر باشند، نتیجه دقیق‌تر است. بنابراین، روش یادگیری بدون نظارت پتانسیل این را دارد که مدلی با ضریب تقریب بالا تولید کند. این روش استفاده‌های مختلفی دارد؛ مانندِ:

- جست‌وجو و مقایسه‌ی اسناد، تصاویر و صداها با یکدیگر برای یافتن موارد مشابه
- تشخیص تضادها و پیدا کردن ناسازگاری‌ها در یک مجموعه داده

[۱] Clustering

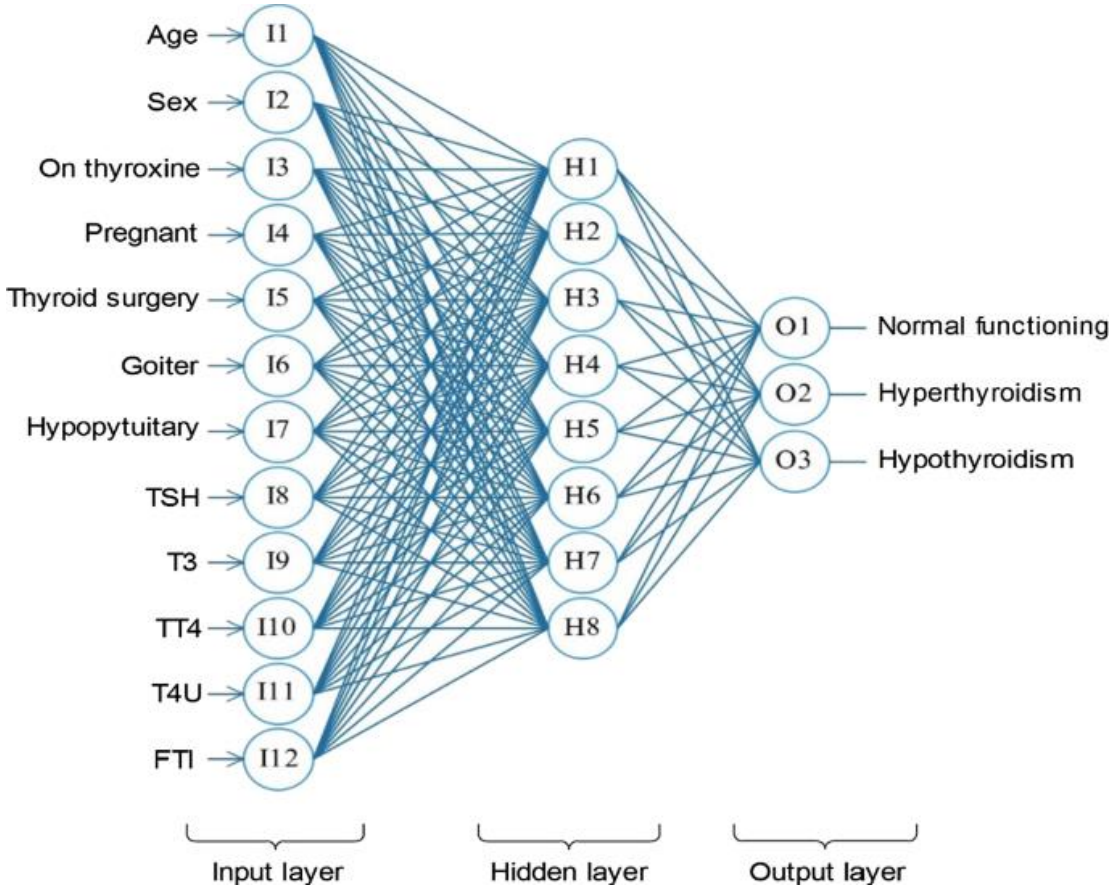
[۲] Unlabeled Data

۴-۳-۱-۵- انواع شبکه های عصبی

شبکه های عصبی انواع گوناگونی دارند که هر کدام از الگوریتم های متفاوتی استفاده میکنند در اینجا سعی شده به معرفی بعضی از آن ها بپردازیم که یادگیری آنها پیش نیازی برای درک پروژه میباشد .

پرسپترون چندلایه[۱]:

یکی از ساده ترین مدل های شبکه عصبی موجود میباشد این شبکه عصبی عملکردی مانند نحوه انتقال اطلاعات در مغز انسان دارد از آنجایی که در این نوع شبکه عصبی از رفتار لایه ای شبکه مغز انسان و روش انتشار سیگنال در آن الهام گرفته شده است به آن شبکه عصبی پیشخور هم میگویند. در این روش هر نرون یا همان سلول عصبی پس از دریافت یک داده آن را پردازش و به سلول دیگر منتقل میکند این روند تا گرفتن نتیجه مطلوب ادامه دارد



شکل ۴-۲ مثالی از شبکه عصبی چند لایه پرسپترون

[۱] Multi-Layer Perceptron

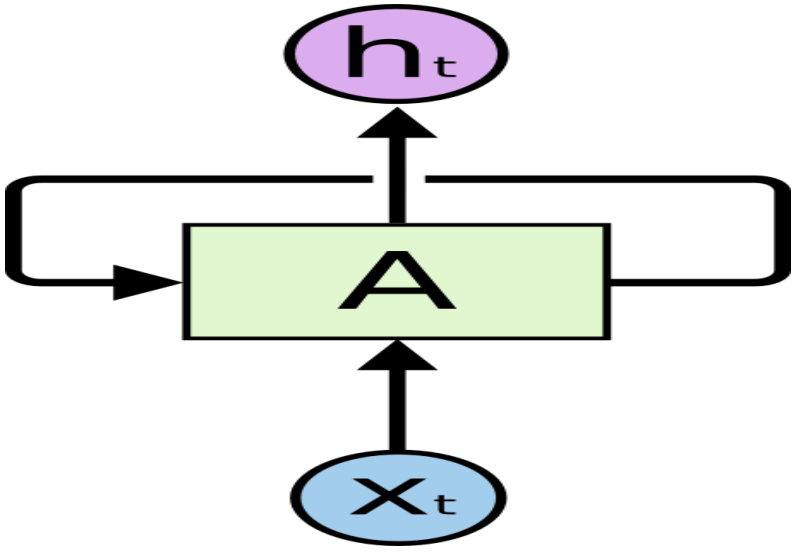
شبکه های عصبی بازگشی :

این شبکه و شبکه عصبی دیگری که بعدا در امتداد همین مطالب توضیح داده خواهد شد از اصلی ترین و مهمترین بخش پروژه بوده و تمام پروژه حول محور این دو شبکه پیاده سازی شده است. بهتر است برای درک بهتر مطالب با یک مثال توضیح را شروع کنیم تمام مطالب این بخش برگرفته از وبلاگ آقای Colah میباشد.

فکر کردن انسان ها به این گونه نیست که در هر ثانیه از ابتدا صورت بگیرد و روند جدیدی آغاز شود در همین لحظه که شما دارید این مقاله را مطالعه میکنید شما هر کلمه را با توجه به دانش و مفهومی که از خواندن کلمه های قبلی کسب کرده اید متوجه میشوید مغز شما همه چیز را دور نمی اندازد و فرایند فکر کردن جدیدی را در هر لحظه شروع نمیکند بلکه پیوسته در حال کسب اطلاعات جدید بر مبنای اطلاعات گذشته میباشد. به طور مثال معنی یک پاراگراف رو در انتهای آن متوجه میشوید .

یکی از نقص های بزرگ شبکه های عصبی متدوال این بود که نمی توانستند به این صورت همانند مغز انسان عمل کنند برای مثال فرض کنید مدلی که شما ساختید میخواهد مشخص کند چه اتفاقی در هر لحظه از فیلم در حال وقوع است مشخص نیست که شبکه های عصبی قدیمی چه طور میتوانند از اطلاعاتی که از صحنه های قبلی فیلم بدست آورده اند برای تشخیص نوع اتفاق در صحنه های بعدی فیلم استفاده کنند.

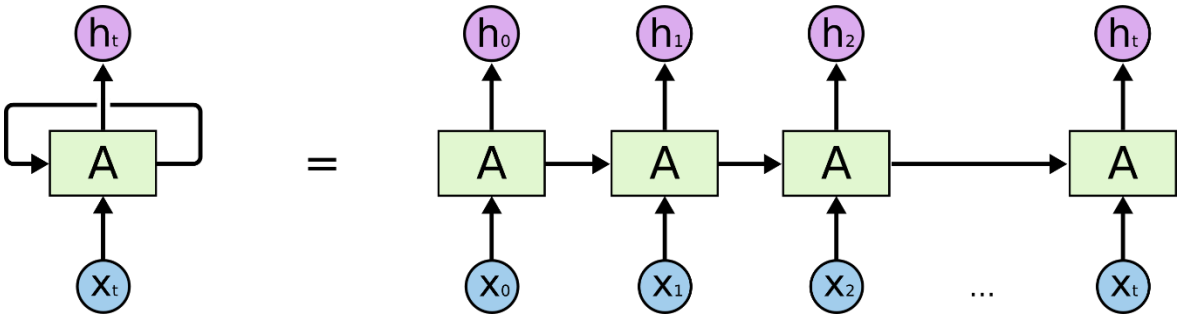
اینجا بود که شبکه های عصبی بازگشتی برای برطرف کردن این مشکل طراحی شدند. در حقیقت شبکه های عصبی بازگشتی همراه خود یک حلقه بازگشتی دارند که منجر میشوند اطلاعاتی که از لحظات قبلی بدست آوردیم از بین نرن و تو شبکه باقی بمونن.



شکل ۴-۳ شبکه عصبی بازگشتی

در شکل بالا یک بخش شبکه عصبی ، A ، مقدار x_t به عنوان ورودی میگیرد و مقدار h_t را به عنوان خروجی حلقه اجازه می دهد که اطلاعات اجازه میدهد که اطلاعات از یک مرحله به مرحله بعد در شبکه منتقل شوند.

برای درک بهتر موضوع خوب است بگوییم این شبکه ها عملا تفاوت خاصی با شبکه های عصبی معمولی نداشته و شبکه های عصبی بازگشتی را میتوان به صورت چندین کپی یکسان از یک شبکه عصبی در نظر گرفت که هرکدام اطلاعاتش را به شبکه بعدی منتقل میکند برای مثال وضعیت شبکه عصبی بازگشتی در صورت باز کردن حلقه نمایش داده شده است .



شکل ۴-۴ شبکه عصبی بازگشتی باز شده

با توجه به ذات ذنجیره مانند شبکه های عصبی بازگشتی میتوان تشخیص داد که این شبکه ها به مقدار زیادی به دنباله ها و لیست ها مرتبط هستند. در حقیقت شبکه های عصبی بازگشتی اولین انتخاب برای کار با چنین داده هایی است .

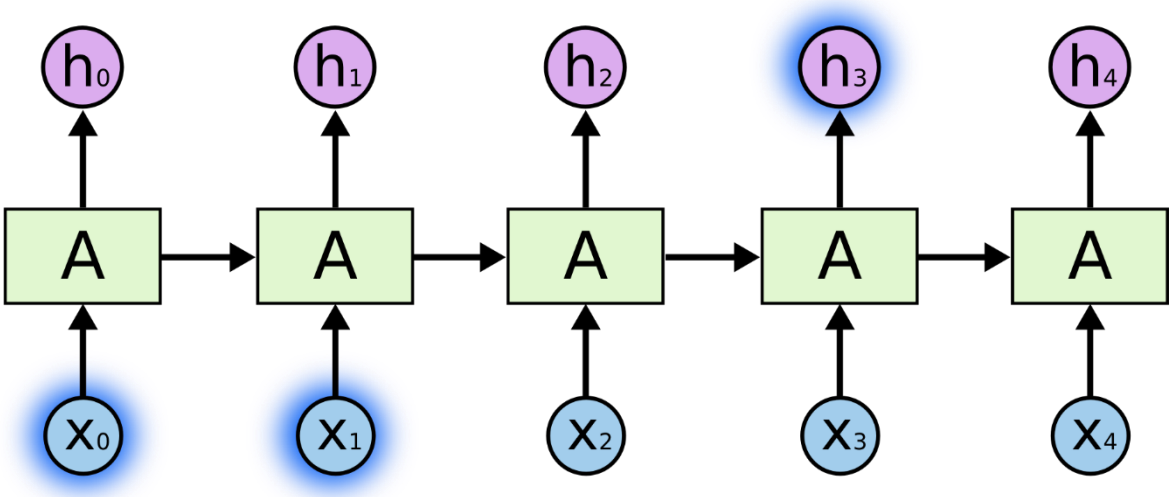
در سال های گذشته استفاده مکرر از این شبکه ها باعث کسب موفقیت های بسیاری از جمله استفاده از آن ها در حوزه های مختلفی مانند تشخیص صدا، مدل کردن زبان، درج خودکار توضیح برای تصاویر و .... شده است.

بیشتر این موفقیت‌ها مدیون استفاده از شبکه‌های حافظه طولانی کوتاه-مدت [۱] است. نوع خاصی از شبکه‌های عصبی بازگشتی که در بیشتر موارد عملکرد بهتری از شبکه‌های عصبی بازگشتی استاندارد دارد. تقریباً میشه گفت اکثر موفقیت‌های شبکه‌های عصبی بازگشتی وقتی بدست آمده که از حافظه طولانی کوتاه-مدت ها استفاده شده است.

[۱] LSTM : Long Term Short Memory

مشکلی به نام وابستگی‌های بلندمدت:

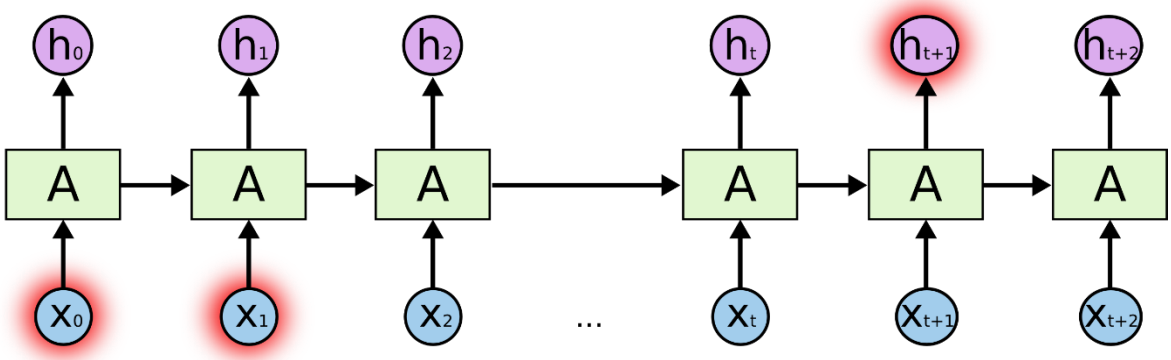
یکی از جذابیت‌های شبکه‌های عصبی بازگشتی این است که آن‌ها ممکن است بتوانند اطلاعات که قبلاً مشاهده شده را به کاری که در حال حاضر در حال انجام است مرتبط سازد، برای مثال استفاده از فریم‌های قبلی یک ویدئو می‌تواند در فهمیدن فریم کنونی کمک‌کننده باشد. اگر شبکه‌های عصبی بازگشتی بتوانند واقعاً این کار را انجام دهند، می‌توان آن‌ها را بسیار مفید دانست منتها بعضی از مواقع ما نیاز داریم فقط به اطلاعات گذشته نزدیک نگاه کنیم تا متوجه اطلاعات حال حاضر بشیم. برای مثال، فرض کنید ما مدلِ زبانی‌ای ساخته‌ایم که تلاش می‌کند کلمه بعدی را با توجه به کلمات قبلی‌ای که در اختیارش قرار دادیم پیش‌بینی کند. اگه ما می‌خوایم آخرین کلمه تو جمله «ابرها هستند در آسمان» رو پیش‌بینی کنیم، ما به اطلاعات اضافیِ دیگه‌ای نیاز نداریم و تقریباً میشه گفت واضحه که کلمه بعدی «آسمان» است. در موارد مشابه این مثال، که فاصله بین اطلاعات مرتبط و جایی که به این اطلاعات نیاز داریم خیلی کمه، شبکه‌های عصبی بازگشتی می‌تونن یاد بگیرن که از این اطلاعات استفاده کنند.



شکل ۴-۵ نحوه عملکرد یه شبکه عصبی بازگشتی

ولی ممکن است مواردی وجود داشته باشد که ما به اطلاعات بیشتری نیاز داشته باشیم. فرض کنید قصد داریم کلمه بعدی در جمله «من زبان انگلیسی را خیلی راحت صحبت می‌کنم... من به دنیا آمدم در آمریکا.» با توجه به اطلاعات اخیر (یعنی چهار پنج کلمه قبل از آخرین کلمه)، می‌توان گفت که کلمه آخر احتمالا اسم یک کشور است، ولی اگر بخواهیم دقیقاً متوجه بشیم چه کشوری است، ما نیاز داریم به اطلاعات دورتر (یعنی تا ده یا بیست کلمه قبل از آخرین کلمه) دسترسی داشته باشیم. به صورت کلی ممکن است فاصله

بین اطلاعات مرتبط و جایی که به این اطلاعات نیاز داریم زیاد باشد. متأسفانه، هر چه این فاصله افزایش پیدا می‌کند، شبکه‌های عصبی بازگشتی قدرت‌شان را در به یادآوردن و استفاده از اطلاعاتی که در گذشته دورتر یاد گرفته‌اند کاهش پیدا می‌کند و به عبارتی توانائی استفاده از اطلاعات گذشته دورتر را ندارند.



شکل ۴-۶ وابستگی بلند مدت در شبکه های عصبی بازگشتی

از نظر تئوری، شبکه‌های عصبی بازگشتی توانائی مدیریت وابستگی‌های بلندمدت رو باید داشته باشند. یک فرد متخصص می‌تونه با دقت پارامترهای شبکه رو طوری تعیین کنه که مسائل کوچک این شکلی را حل کنند. متأسفانه در عمل شبکه‌های عصبی بازگشتی توانائی یادگیری وابستگی بلندمدت رو ندارند.

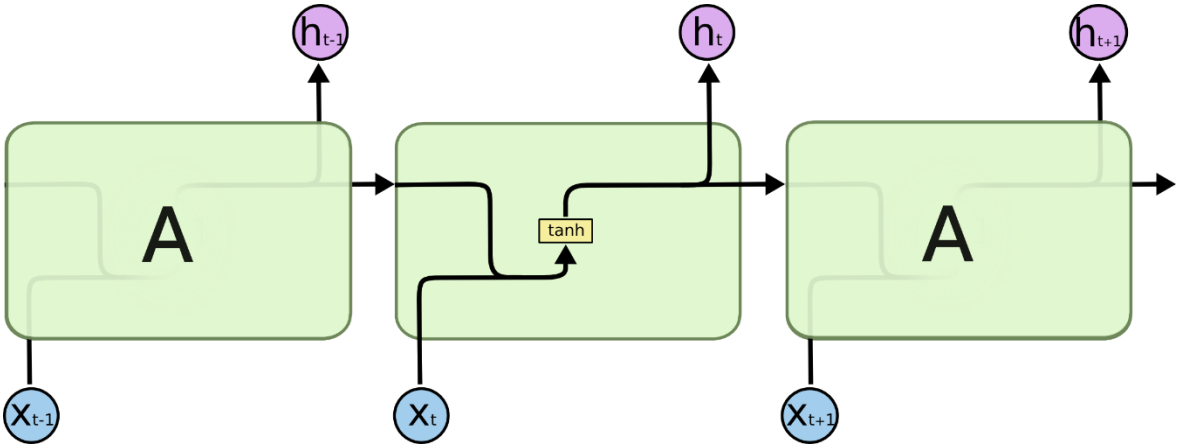
لازم به ذکر است که دو مشکل حاد محو شدگی و انفجار گرادیان توی شبکه های عصبی بازگشتی به وجود می آید که به وسیله شبکه عصبی حافظه طولانی کوتاه-مدت قابل برطرف شدن میباشد .

شبکه های عصبی حافظه طولانی کوتاه-مدت :

نوع خاصی از شبکه‌های عصبی بازگشتی هستند که توانائی یادگیری وابستگی‌های بلندمدت را دارند در حقیقت هدف از طراحی شبکه های عصبی حافظه طولانی کوتاه-مدت ، حل کردن مشکل وابستگی بلندمدت بود. به این نکته مهم توجه کنید که به یاد سپاری اطلاعات برای بازه‌های زمانی بلند مدت، رفتار پیش‌فرض و عادی این شبکه ها است و ساختار آن‌ها به صورتی است که اطلاعات خیلی دور را به خوبی یاد می‌گیرند که این ویژگی در ساختار آن‌ها نهفته است.

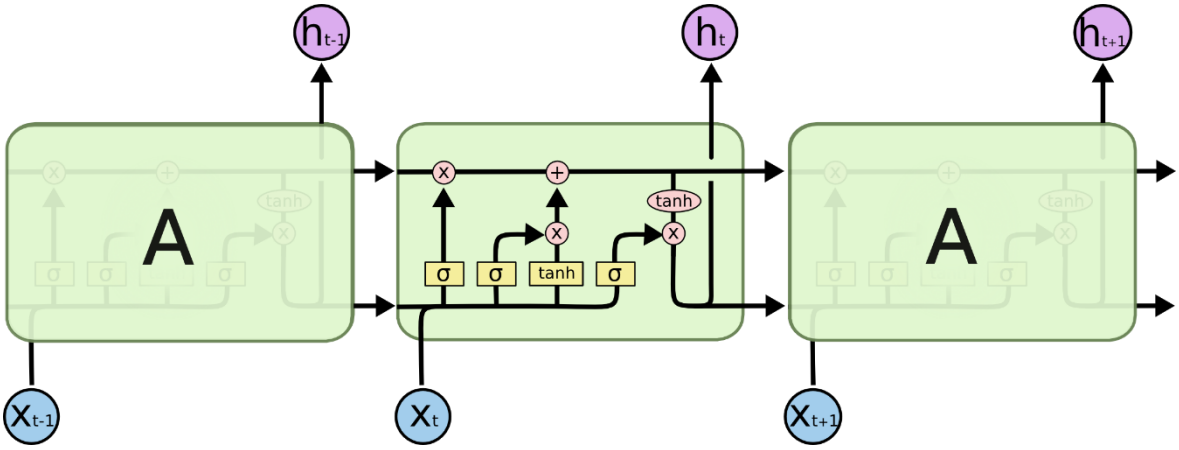
همه شبکه‌های عصبی بازگشتی به شکل دنباله‌ای (زنجیره‌ای) تکرار شونده از ماژول‌های (واحدهای) شبکه‌های عصبی هستند. در شبکه‌های عصبی بازگشتی استاندارد، این ماژول‌های تکرار شونده ساختار ساده‌ای دارند، برای مثال تنها شامل یک لایه تانژانت هایپربولیک [۱] هستند.

[۱] tanh

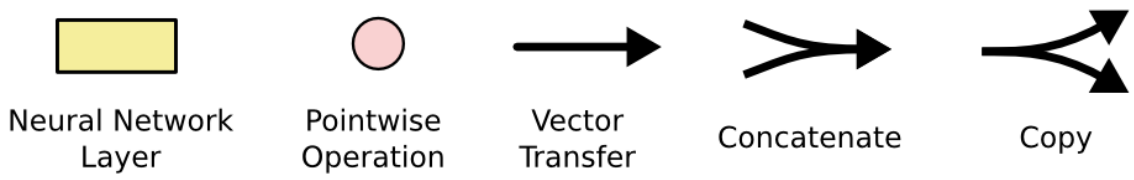


شکل ۴-۷ ماژول تکرار شونده در شبکه های بازگشتی استاندارد شامل یک لایه هستند

شبکه‌های عصبی حافظه طولانی کوتاه-مدت نیز چنین ساختار دنباله یا زنجیره‌مانندی دارند ولی ماژولِ تکرار شونده ساختار متفاوتی دارد. به جای داشتن تنها یک لایه شبکه عصبی، ۴ لایه دارند که طبق ساختار ویژه‌ای با یکدیگر در تعامل و ارتباط هستند.



شکل ۴-۸ ماژول های تکرار شونده در LSTM ها دارای ۴ لایه که با هم در تعامل هستند



شکل ۴-۹ کپی کردن | وصل کردن | بردار انتقال | عملیات نقطه به نقطه | یک لایه‌ی شبکه عصبی

تا اینجا با مفهموم کلی شبکه های عصبی بازگشتی و حافظه طولانی کوتاه-مدت آشنا شده ایم توضیح بیشتر این مفاهیم ما را از بحث اصلی این پایان نامه که توضیح فرایند و اجرای پروژه میباشد دور میکند لذا بهتر است به سراغ توضیح کد های مهم پروژه در این فصل رفته در ادامه بنا بر نیاز به توضیح بیشتر این مدل میپردازیم .

۴-۴- کد های اصلی برنامه

در این قسمت ما شروع به توضیح دادن هر قسمت از کد های برنامه میکنیم و سعی بر آن شده که ترتیب اجرایی کدها حفظ شده و در هر پارت به صورت جداگانه توضیح داده شوند.

به طور کلی تمام مراحل اجرای برنامه به صورت زیر خلاصه میشود

۱. تهیه و پیش پردازش دیتا
۲. اضافه و نصب تمام پیش نیاز های برنامه
۳. ساخت شبکه عصبی
۴. آماده سازی دیتا برای بخش های متفاوت پردازش در شبکه ی عصبی
۵. گرفتن خروجی نموداری و دقت عملکرد مدل برای هر مرحله از پردازش

۴-۴-۱- تهیه و پیش پردازش دیتا

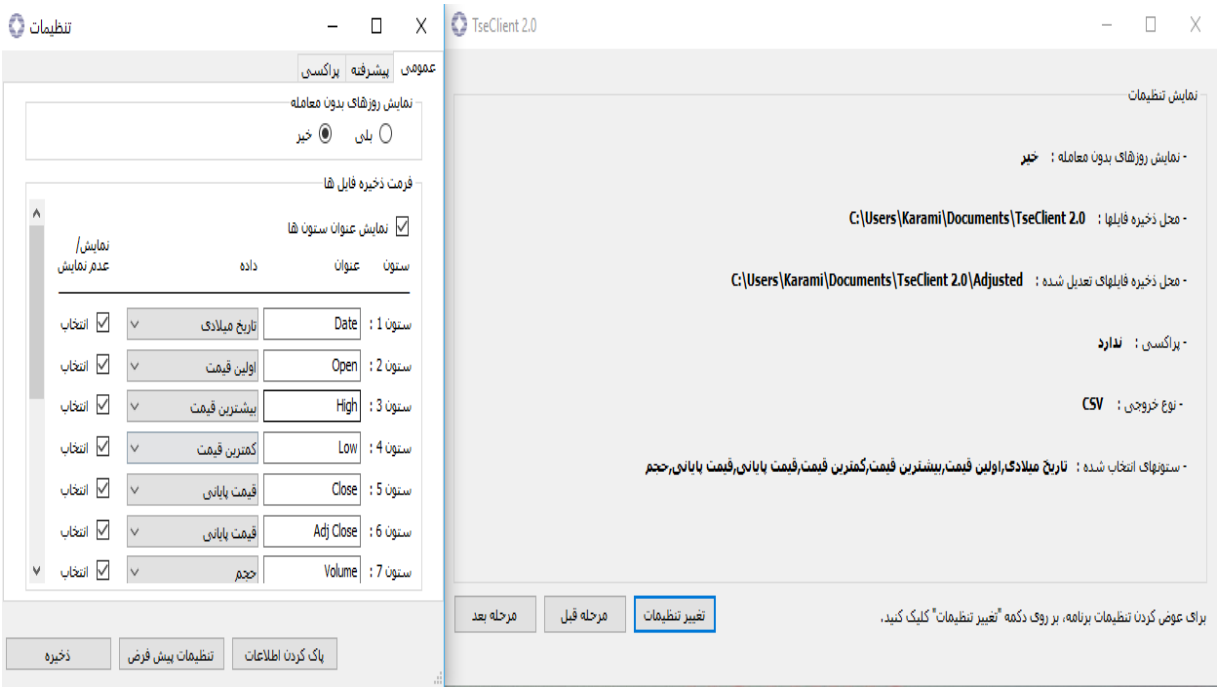
در این پروژه سعی بر آن شد که با استفاده از داده های بازار سهام ایران و انتخاب چند شرکت از این مجموعه شرکت های پذیرفته شده در سازمان بورس ایران داده های مالی مربوطه را تهیه کرده و برای عملکرد و ساخت مدل بسنجیم.

در انتخاب داده ی مورد نظر ، ما به عوامل متعددی از جمله نقدشوندگی شرکت ، محبوبیت آن ، سابقه ی فعالیت ، تاثیر گذاری عملکرد معاملاتی شرکت در شاخص کل و هموزن توجه ویژه ای داشته و در نهایت داده ی پنج شرکت انتخاب شد و از بین این ۵ شرکت بهترین شرکتی که عملکرد خیره کننده ای در مدل داشت را انتخاب کردیم .

داده های انتخاب شده متعلق به :

- ایران خودرو(خودرو)
- معدنی و صنعتی چادرملو(کچاد)
- ملی صنایع مس ایران (فملی)
- صنایع پتروشیمی خلیج فارس (فارس)
- فولاد مبارکه اصفهان (فولاد)

داده های مربوطه از نرم افزاری به نام Tse Client گرفته شده و نحوه ی چینش و فرمت آن در نرم افزار مشخص شده است .



شکل ۴-۱۰ شمایی از برنامه Tse Client

پس از تهیه دیتا فایل خروجی به ما پسوند csv بود که ستون تاریخ این فایل نیاز پردازش مجدد برای تبدیل به فرمتی بود که قابل محاسبه برای مدل باشد . لذا با استفاده از کد زیر تمام ستون اول از فایل استخراج شده و پس متناسب سازی در فایل دیگر ذخیر شد .

```
import csv
with open('IranKhodro-a.csv') as csvfile:
    readCSV = csv.reader(csvfile, delimiter=',')
    for row in readCSV:
        x=row[0]
        list1=[]
        list1.append(x[4:6])
        list1.append(x[6:8])
        list1.append(x[0:4])
        str1 = '/'.join(list1)
        with open('DateKhodro.csv', mode='a') as file_:
            file_.write("{}".format(str1))
            file_.write("\n")
```

شکل ۴-۱۱ قطعه کد مربوط به تبدیل ستون تاریخ به فرمت مورد نظر

۴-۴-۲- اضافه و نصب تمام پیش نیاز های برنامه

خوشبختانه به علت استفاده از گوگل کولب اضافه و نصب پیش نیاز های برنامه بسیار سریع و راحت بود و برای استفاده از کتابخانه یا فریم ورکی خاصی نیاز نبود فرایند نصب زمان بری طی بشود و با چند خط کد ساده تمام این پیشنیاز ها نصب و قابل استفاده می بود .

```
import math
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import keras

from keras.models import Model
from keras.layers import Dense, Input
from keras.layers import LSTM
from keras.layers import Dropout

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

شکل ۴-۱۲ وارد کردن کتابخانه های مورد نیاز

۴-۴-۳- نحوه‌ی دسترسی به داده‌های Google Drive از طریق گوگل کولب

```
from google.colab import drive
drive.mount('/content/drive')
```

شکل ۴-۱۳ کد دسترسی به داد های گوگل درایو از طریق گوگل کولب

۴-۴-۴- خواندن دیتا ها و اعمال تغیرات مورد نیاز

در این مرحله از پروژه ما با داشتن دسترسی به حافظه‌ی گوگل درایو اقدام به بارگذاری دیتا در پروژه میکنیم .

```
df = pd.read_csv('/content/drive/MyDrive/Project/Iran Khodro-a.csv')
df.rename(columns={'Close':'Trade Close', 'Volume':'Trade Volume'}, inplace=True) #Rename some column
print('Number of rows and columns:', df.shape)
length = df.shape[0] #number of rows
#print(len)

df.head() #Show us head and 5 first rows of file
```

```
Number of rows and columns: (4256, 7)
```

	Date	Open	High	Low	Trade Close	Adj Close	Trade Volume
0	3/25/2001	53.02	53	53	53.06	53.06	110870
1	3/26/2001	53.08	53	53	53.02	53.02	96613
2	3/27/2001	52.96	53	53	52.96	52.96	166600
3	3/28/2001	53.06	54	53	53.99	53.99	80676
4	3/31/2001	53.82	54	53	53.78	53.78	177362

شکل ۴-۱۴ کد بارگذاری دیتا در پروژه

همانطور که در شکل بالا مشخص شده است با استفاده از کتابخانه پاندا[۱] فایل مورد نظر را خوانده و در متغیری به نام df ذخیره کرده ایم سپس اقدام به تغیر نام یعضی از ستون های فایل کردیم سپس چند سطر اول به همراه نام هر ستون را با استفاده از تابع head نمایش دادیم .

[۱] panda

۴-۴-۵- آماده سازی و نرمال سازی داده ها

در هر شبکه عصبی ما نیاز به سه مرحله داریم تا بتوانیم یک مدل تولید کنیم سه مرحله یادگیری، اعتبار سنجی و تست لذا لازم است که برای هر سه بخش مورد نظر دیتای جداگانه ای را داشته باشیم.
بنابراین ما در پروژه داده های مورد نظر خود را به سه بخش تقسیم کردیم به این صورت که ۶۰ درصد از داده ها مربوط به مرحله یادگیری، ۲۰ درصد از بعدی دیتا مربوط به مرحله اعتبارسنجی و در نهایت ۲۰ درصد پایانی دیتا به مرحله تست صورت گرفت .

برای نرمال سازی داده ها ما از قیمت پایانی روز t ام تقسیم بر قیمت پایانی روز t-1 ام لگاریتم گرفتیم و برای پیشبینی ما از قیمت پایانی ۲۰ روز گذشته استفاده کردیم تا بتوانیم قیمت پایانی در روز t+24 ام را بدست آوریم . در نتیجه بعد از نرمال سازی دیتا ها ما به دنباله ای رسیدیم که به عنوان ورودی به مدل ما داده میشود برای درک بهتر این توضیحات آن ها را با زبان ریاضی در اینجا نمایش میدهیم

$$\log\left(\frac{Close\ t}{Close\ t-1}\right), \log\left(\frac{Close\ t+1}{Close\ t}\right), \log\left(\frac{Close\ t+2}{Close\ t+1}\right), \log\left(\frac{Close\ t+3}{Close\ t+2}\right), \log\left(\frac{Close\ t+4}{Close\ t+3}\right), \ldots.$$

این دنباله شمایی از ورودی ما به مدل بود البته ما در این مدل علاوه بر قیمت پایانی، حجم را هم در نظر گرفته ایم .

```
seq_length = 20
step_predict = 4

#create new coloumn (close_log)= log( (trade close) / (pervious day of trade close) )
df['close_log'] = np.log(df['Trade Close'] / df['Trade Close'].shift(1))

#create new coloumn (vol_log) = log( (trade volume) / (pervious day of trade volume) )
df['vol_log'] = np.log(df['Trade Volume'] / df['Trade Volume'].shift(1))

#create new coloumn (target_log) = log( (t+4th trade close) / (trade close))
df['target_log'] = np.log(df['Trade Close'].shift(-step_predict) / df['Trade Close'])

#pandas.core.frame.DataFrame
train_df = df[:int(len(df['Trade Close'])*0.6)].copy()
train_df.reset_index(inplace=True)
val_df = df[int(len(df['Trade Close'])*0.6):int(len(df['Trade Close'])*0.8)].copy()
val_df.reset_index(inplace=True)
test_df = df.loc[int(len(df['Trade Close'])*0.8):].copy()
test_df.reset_index(inplace=True)
train_df.shape, val_df.shape, test_df.shape
```

شکل ۴-۱۵ کد مربوط به نرمال سازی و تقسیم بندی داده ها

۴-۴-۶- پردازش دیتا برای مرحله یادگیری و اعتبارسنجی

در بخش قبل توضیح دادیم که نحوه تقسیم دیتا به چه صورت است حال باید در این مرحله مشخص کنیم داده های ورودی به مدل ما به چه صورت می باشند. قبل از آن بهتر است در مورد ورودی هایی که برای مدل LSTM داده میشوند را توضحی دهیم. مدل شبکه عصبی حافظه طولانی کوتاه-مدت از داده ها انتظار دارد تا در قالب مشخصی باشند که معمولا آرایه های سه بعدی است

```
#numpy.ndarray
X_train = []
y_train = []    #20         1702        -       4 => i in range (20,1698)
for i in range(seq_length, train_df.shape[0]-step_predict):
    X_train.append(train_df[i-seq_length+1: i][['close_log', 'vol_log']].values)
    y_train.append(train_df.loc[i-1, 'target_log'])
X_train, y_train= np.array(X_train), np.array(y_train)
#X_train.shape[0] arrays  that contains X_train.shape[0] arrays each with 2 elements
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 2))
X_train.shape, y_train.shape

X_val = []
y_val = []
for i in range(seq_length, val_df.shape[0]-step_predict):
    X_val.append(val_df[i-seq_length+1: i][['close_log', 'vol_log']].values)
    y_val.append(val_df.loc[i-1, 'target_log'])
X_val, y_val= np.array(X_val), np.array(y_val)
X_val = np.reshape(X_val, (X_val.shape[0], X_val.shape[1], 2))
```

شکل ۴-۴-۱۶ کد پردازش دیتا برای ورودی مدل

در نهایت با استفاده از کتابخانه nympy در پایتون داده های مربوطه X_train و X_val را به صورت یک آرایه سه بعدی برای انتقال به ورودی مدل تبدیل میکند.

۴-۴-۷- ساخت شبکه عصبی

یکی از مهمترین بخش ها در ساخت این پروژه مربوط به این مرحله بوده است که ما یک شبکه عصبی حافظه طولانی کوتاه-مدت را پیاده سازی کنیم بهتر است ابتدا کد های این بخش نشان داده شوند سپس به توضیحات مورد نیاز میپردازیم

```
# initialize network (1 layer)
                        #X_train.shape[1] =19
inputs = Input(shape=(X_train.shape[1], 2))
lstm1 = LSTM(units=50, return_sequences=False, recurrent_regularizer='l2')(inputs)

dropout1 = Dropout(rate=0.8)(lstm1)

# fully connected layers

fc1 = Dense(units=50)(dropout1)
dropoutfc1 = Dropout(rate=0.8)(fc1)
fc2 = Dense(units=25)(dropoutfc1)
dropoutfc2 = Dropout(rate=0.8)(fc2)
fc3 = Dense(units=10)(dropoutfc2)
dropoutfc3 = Dropout(rate=0.8)(fc3)
output = Dense(units=1)(dropoutfc3)

# fit lstm model

lstm_model = Model(inputs=inputs, outputs=output)
lstm_model.compile(optimizer = 'Adam', loss = 'mean_squared_error')
history = lstm_model.fit(X_train, y_train, epochs = 20, batch_size = 128, validation_data=(X_val, y_val))
```

شکل ۴-۴۱۷ کد ایجاد شبکه عصبی حافظه طولانی کوتاه-مدت

تابع Input برای ساخت کراس تنسور[۱] استفاده میشود یک شئ از تنسور است که ما با اضافه کردن ویژگی هایی به آن میتوانیم مدل شبکه عصبی را فقط با مشخص کردن ورودی و خروجی های مدل بسازیم .

برای درک بهتر فرض کنید سه متغیر a,b,c کراس تنسور باشند بنابراین ما میتوانیم یک مدل به این صورت model = Model(inputs=[a,b] , output=c) بسازیم.

[۱] Keras tensor

تابع LSMT برای ما یک لایه از این شبکه عصبی را درست میکند که شرح پارامتر های آن به صورت زیر میباشد:

- units: ابعاد فضای خروجی
- return_sequence: یک متغیر بولین میباشد که مشخص میکند آخرین خروجی در توالی خروجی بازگردانده شود و یا کل توالی
- recurrent_regularizer : انتخاب نوع تابع مورد نظر برای نظم دهی شبکه

در قسمت بعدی با Dropout مواجه میشویم که Dropout ها و Regularizer ها هر دو برای نظم دهی به شبکه استفاده میشوند یکجورایی برای جلوگیری از بیش‌پردازش [1] در شبکه استفاده میشوند که با مفهموم آن در قسمت های بعدی آشنا میشوید .

در خط بعد با استفاده از تابع Dens ما سه لایه شبکه تمام متصل درست کردیم که خروجی از لایه اول به این لایه ها پاس داده میشود

در بخش بعدی همانطور که قبلا توضیح دادیم حال میتوانیم با استفاده از تابع مدل و کراس تنسور هایی که قبلا تعریف کردیم به پیاده سازی مدل بپردازیم در حقیقت این تابع لایه ها را در یک شئ با ویژگی های آموزش و استنتاج قرار میدهد

تابع lstm_model.complie شما در این تابع پس از اینکه مدل را تولید کردید و قبل از آنکه مدل را آموزش بدهید باید تنظیمات مربوط به loss [2]و optimizer [3]را مشخص کنید و سپس مدل را کامپایل کنید.

ما در اینجا از تابع زیان Mean Squared Error استفاده کردیه ایم که برای درک آن فرمول ریاضی آن به صورت زیر میباشد.

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

شکل ۴-۴۱۸ فرمول میانگین مربع خطا

[1] Overfitting

[2] تابع زیان (خطا)

[3] بهینه ساز

همچنین از بهینه ساز Adam استفاده کردیم به طور خلاصه از بهینه سازها جهت بروز رسانی وزن ها در شبکه‌ی عصبی استفاده میشود.

در قسمت آخر تابع fit() مربوط به آموزش مدل است یا به اصطلاح train کردن مدل ، سپس، مدل برای اجرا روی ۲۰ دوره (epoch) و اندازه دسته ۱۲۰ (batch size) پردازش می‌شود

در نهایت خروجی این تابع به صورت زیر می‌باشد.

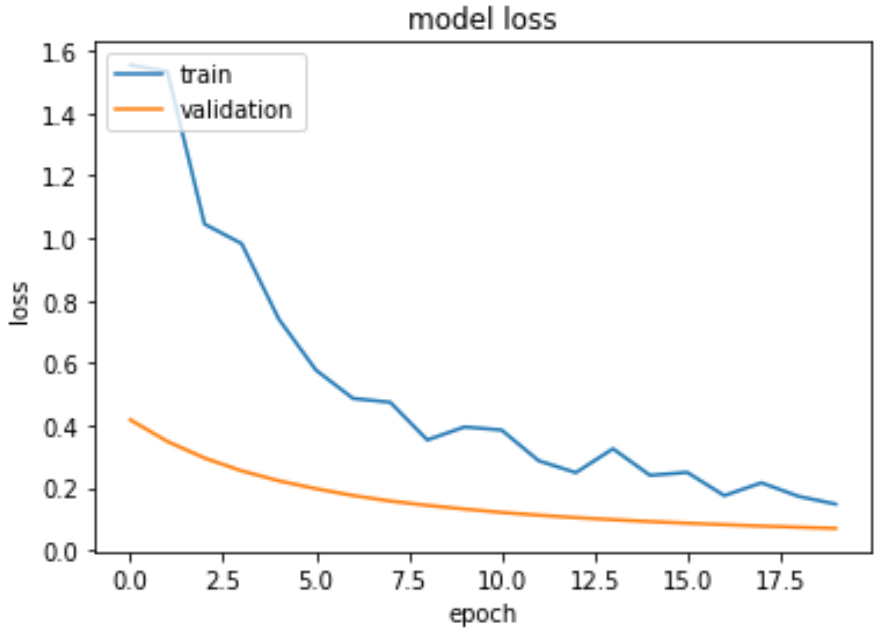
```
Epoch 1/20
14/14 [==============================] - 3s 61ms/step - loss: 1.5537 - val_loss: 0.4179
Epoch 2/20
14/14 [==============================] - 0s 25ms/step - loss: 1.5333 - val_loss: 0.3486
Epoch 3/20
14/14 [==============================] - 0s 25ms/step - loss: 1.0445 - val_loss: 0.2955
Epoch 4/20
14/14 [==============================] - 0s 28ms/step - loss: 0.9818 - val_loss: 0.2544
Epoch 5/20
14/14 [==============================] - 0s 27ms/step - loss: 0.7404 - val_loss: 0.2226
Epoch 6/20
14/14 [==============================] - 0s 26ms/step - loss: 0.5771 - val_loss: 0.1972
Epoch 7/20
14/14 [==============================] - 0s 25ms/step - loss: 0.4861 - val_loss: 0.1754
Epoch 8/20
14/14 [==============================] - 0s 27ms/step - loss: 0.4741 - val_loss: 0.1578
Epoch 9/20
14/14 [==============================] - 0s 25ms/step - loss: 0.3533 - val_loss: 0.1439
Epoch 10/20
14/14 [==============================] - 0s 30ms/step - loss: 0.3949 - val_loss: 0.1322
Epoch 11/20
14/14 [==============================] - 0s 25ms/step - loss: 0.3852 - val_loss: 0.1213
Epoch 12/20
14/14 [==============================] - 0s 24ms/step - loss: 0.2867 - val_loss: 0.1124
Epoch 13/20
14/14 [==============================] - 0s 25ms/step - loss: 0.2487 - val_loss: 0.1047
Epoch 14/20
14/14 [==============================] - 0s 25ms/step - loss: 0.3252 - val_loss: 0.0982
Epoch 15/20
14/14 [==============================] - 0s 24ms/step - loss: 0.2400 - val_loss: 0.0921
Epoch 16/20
14/14 [==============================] - 0s 26ms/step - loss: 0.2496 - val_loss: 0.0868
Epoch 17/20
14/14 [==============================] - 0s 24ms/step - loss: 0.1752 - val_loss: 0.0821
Epoch 18/20
14/14 [==============================] - 0s 23ms/step - loss: 0.2162 - val_loss: 0.0777
Epoch 19/20
14/14 [==============================] - 0s 23ms/step - loss: 0.1728 - val_loss: 0.0739
Epoch 20/20
14/14 [==============================] - 0s 25ms/step - loss: 0.1479 - val_loss: 0.0702
```

شکل ۴-۱۹ نمایش خروجی آموزش مدل

۴-۴-۸- نموداری از عملکرد مدل در موقع یادگیری

در شکل زیر کد مربوطه و نتیجه خروجی آن را مشاهده میکنید ، دو نمودار آبی و سبز که نمودار آبی که مشاهده میشود با نزدیک شدن به انتهای دوره ها (epoch) اختلاف بین نمودار یادگیری (آبی) و اعتبار سنجی(سبز) به طور چشمگیری کم شده که نشان از یادگیری خوب مدل میباشد .

```
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```



شکل ۴-۲۰ کد افزودن کتاب‌خانه‌های موردنیاز داده افزایی و خواندن تصویر

۴-۴-۹- پیشبینی مدل برای داده های Train

قبلا از اینکه به توضیح و تفسیر کد های مربوطه بپردازیم بهتر است بدانیم ما در هر مدلی که در شبکه های عصبی میسازیم پس از ساخت کامل آن نوبت به ارزیابی مدل میشود به بیان راحت تر مدل ما با داده هایی که از قبل داشته و داده هایی که برای مدل جدید هستند ارزیابی میشود، داده هایی که از قبل داشته داده هایی شامل Train و Validation به اصطلاح به این داده ها میگوییم InSample و داده هایی که برای Test استفاده میشود OutSample ، نکته ی قابل تامل این است که اگر میزان خطاها و دقت مدل ما در مورد داده های InSample به ترتیب بالا و کم بود ، نشان از این دارد که مدل ساخته شده کارایی نداشته و غیر قابل استفاده میباشد اما اگر نتایج مربوطه با این قسمت مورد قبول باشند انتظار ما این است که با اختلافی نه چندان زیاد نتایج برای داده های OutSample هم قابل قبول باشند یکی از راه های تشخیص بیش پردازش در مدل این است که نتایج داده های OutSample اختلاف معناداری با نتایج داده های InSample دارند .

با توجه به صحبت های بالا با نحوه عملکرد و ارزیابی مدل آشنا شدیم لذا به سراغ توضیح کد های مربوطه به این قسمت و رسم نمودار آن میرویم .

```
train_price = train_df[seq_length+step_predict:train_df.shape[0]]['Trade Close']
train_predicted = lstm_model.predict(X_train)
inv = train_df[seq_length:train_df.shape[0]-step_predict]['Trade Close'].values.reshape(-1, 1)
train_predicted = inv * np.exp(1)**train_predicted
```

شکل ۴-۲۱ کد پیشبینی مدل برای داده های Train

در این بخش ما داده های train را که مربوط به قیمت پایانی است به متغیری به نام train_price پاس داده و پیشبینی را توسط مدل با این داده ها انجام میدهیم همانطور که در قسمت ابدایی برنامه داده ها را به لگاریتم تبدل کرده و نسبت آن ها را به هم در ستون جدید ذخیره کردیم حال نوبت به آن رسیده که داده ها را به شکل اول تبدیل کرده و آنها را برای نمایش در نمودار مهیا کنیم .

۴-۴-۹-۱- رسم نمودار برای پیشبینی داده های Train

در تصویر زیر کد مربوط به رسم نمودار را مشاهده میکنید .

```
# visualising the results
date = train_df[seq_length+step_predict:train_df.shape[0]]['Date'].values
plt.figure(figsize=(20,10))
plt.plot(date, train_price, color = 'red', label = 'Khodro 2001-2021')
plt.plot(date, train_predicted, color = 'blue', label = 'Predicted Khodro 2001-2021')
plt.xticks(np.arange(0,len(train_predicted),100))
plt.title('TRAIN SET: Khodro Stock Price 2001 Till 2021')
plt.xlabel('Date')
plt.ylabel('Khodro 2001-2021 Stock Price')
plt.legend()
plt.show()
print(f'RMSE: {round(mean_squared_error(train_price, train_predicted, squared=False),4)}')
print(f'MAE: {round(mean_absolute_error(train_price, train_predicted),4)}')
print(f'r2_score: {round(r2_score(train_price, train_predicted),4)}')
```

شکل ۴-۲۲ کد رسم نمودار نتایج پیشبینی داده های Train

همانطور که مشاهده میکنید رسم نمودار به وسیله کتابخانه بسیار ارزشمند matplotlib و استفاده از تابع pyplot صورت گرفته است ابتدا مشخص میکنیم که دادهای ما متعلق به چه زمانی بودند سپس سایز نمودار را اعمال مکنیم و در مراحل بعدی اقدام به رسم نمودار و نام گذاری آن میپردازیم .

نکته‌ای که در این چند خط کد قابل بحث میباشد مربوط به سه خط آخر میباشد. بهتر است اینگونه شروع کنیم برای اعتبار از صحت و دقت مدل ما پارامتر هایی برای سنجش آن داریم که به دو دسته :

- الگوریتم های دسته بندی[۱]
- رگرسیون[۲]

[۱] Classification

[۲] Regression

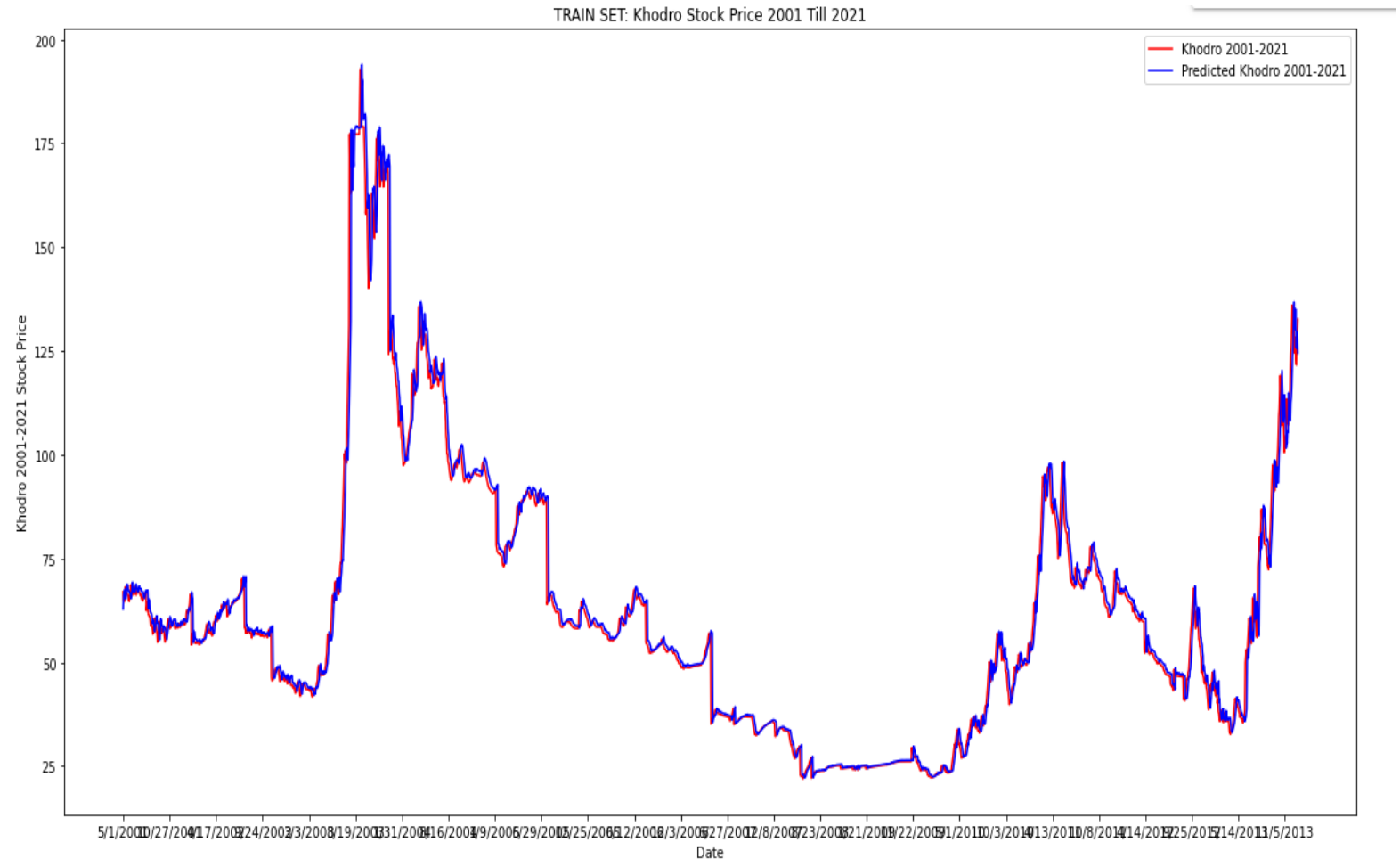
شکل ۴-۲۳ جدول مربوط به توابع زیان

بنابر توضیحات داده شده مدل ما یک مدل رگرسیون میباشد و برای سنجیدن دقت و کارایی مدل مجبوریم از پارامتر هایی مانند میانگین مربعات خطا (MSE) میانگین قدر مطلق خطا (MAE) استفاده کنیم در این جا از پارامتر r2_score هم استفاده کرده ایم. میانگین مربعات خطا را در سطر های بالا تر توضیح داده و فرمول ریاضی آن را نشان دادیم ، در اینجا مفهوم از RMSE همان مجذور میانگین مربعات خطا ها میباشد و فرمول میانگین قدر مطلق خطا به صورت زیر میباشد:

$$\mathbf{MAE} = \frac{\sum|\mathbf{y_i} - \hat{\mathbf{y}}_\mathbf{i}|}{\mathbf{n}}$$

که $y_i - \hat{y}_i$ نشان از اختلاف میان مقدار واقعی با مقدار پیشبینی شده دارد.

در مورد تابع r2_score ، خروجی آن یک پارامتر رتبه دهی به عملکرد میباشد که بهترین حالت آن ۱ و حتی میتواند در بدترین حالت هم منفی شود به هر حال ما از آن به عنوان یک پارامتری که وضعیت عملکرد مدل را نشان میدهد استفاده میکنیم




```
RMSE: 4.7665
MAE: 2.4645
r2_score: 0.9778
```

شکل ۴-۲۰ نمودار خروجی مربوط به پیشبینی مدل با داده های Train

۴-۴-۱۰- پیشبینی مدل برای داده های Validation

طبق توضیحات صورت گرفته در قبل لذا لازم است تا پیشبینی مدل را برای داده های Validation هم بسنجیم برای این کار ابتدا نیاز داریم که داده های validation را به متغیری اختصاص دهیم و بعد آن متغیر را به مدل داده تا بیشبینی را انجام دهد .

قطعه کد زیر مربوط به انجام پیشبینی برای داده های مورد نظر میباشد.

```
"""### Validation set"""

val_price = val_df[seq_length+step_predict:val_df.shape[0]]['Trade Close']
val_predicted = lstm_model.predict(X_val)
inv = val_df[seq_length:val_df.shape[0]-step_predict]['Trade Close'].values.reshape(-1, 1)
val_predicted = inv * np.exp(1)**val_predicted
```

شکل ۴-۲۱ کد پیشبینی مدل برای داده های Validation

۴-۴-۱۰-۱- رسم نمودار برای پیشبینی داده های Validation

قطعه کد زیر اقدام به ساخت نموداری برای رسم داده های پیشبینی شده و داده های Validation میکند .

```
# visualising the results
date = val_df[seq_length+step_predict:val_df.shape[0]]['Date'].values
plt.figure(figsize=(20,10))
plt.plot(date, val_price, color = 'red', label = 'Khodro 2001-2021')
plt.plot(date, val_predicted, color = 'blue', label = 'Predicted Khodro 2001-2021')
plt.xticks(np.arange(0,len(val_predicted),50))
plt.title('VAL SET: Khodro 2001-2021 Stock Price Prediction')
plt.xlabel('Date')
plt.ylabel('Khodro 2001-2021 Stock Price')
plt.legend()
plt.show()
print(f'RMSE: {round(mean_squared_error(val_price, val_predicted, squared=False),4)}')
print(f'MAE: {round(mean_absolute_error(val_price, val_predicted),4)}')
print(f'r2_score: {round(r2_score(val_price, val_predicted),4)}')
```

شکل ۴-۲۲ کد رسم نمودار پیشبینی داده های Validation

پس از اجرای قطعه کد بالا نمودار شماتیک آن برایمان رسم میشود که در تصویر زیر مشاهده میشود.

RMSE: 7.8963
MAE: 5.8778
r2_score: 0.9112

شکل ۴-۲۳ نمودار خروجی مربوط به پیشبینی مدل با داده های Validation

قابل مشاهده هست که میزان خطای ما نسبت به مرحله ای که مربوط به داده های Train بود بیشتر شده است البته اختلاف ناچیز آن مقداری طبیعی است و نحوه‌ی عملکرد مدل تا به اینجای کار بسیار چشمگیر بوده است .

۴-۴-۱۱- پیشبینی مدل برای داده های Test

به اصلی ترین بخش از اعتبارسنجی مدل رسیدیم در انیجا مشخص میشود که آیا واقعا مدل با کاربرد دارد یا خیر اینجاست که اگر اختلاف نتایج خطا ها با دو مرحله‌ی قبلی به طور معنا داری افزایش پیدا کند

میتوان گفت شاید مدل دچار بیش پردازش شده است و باید از راه های مختلفی این بیش پردازش را بر طرف کرد ، لذا در این مرحله اول به سراغ تهیه‌ی داده های تست برای پیشبینی مدل میرویم .

در قطعه کد زیر انجام عملیات پردازش داده ها و پیشبینی آن توسط مدل به نمایش داده شده است.

```
X_test = []
y_test = []
for i in range(seq_length, test_df.shape[0]-step_predict):
    X_test.append(test_df[i-seq_length+1: i][['close_log', 'vol_log']].values)
    y_test.append(test_df.loc[i-1, 'target_log'])
X_test, y_test= np.array(X_test), np.array(y_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 2))
#X_test.shape, y_test.shape

test_price = test_df[seq_length+step_predict:test_df.shape[0]]['Trade Close']
test_predicted = lstm_model.predict(X_test)
inv = test_df[seq_length:test_df.shape[0]-step_predict]['Trade Close'].values.reshape(-1, 1)
test_predicted = inv * np.exp(1)**test_predicted
```

۴-۲۴ کد تهیه‌ی داده های Test واجرای مدل برای داده های Test

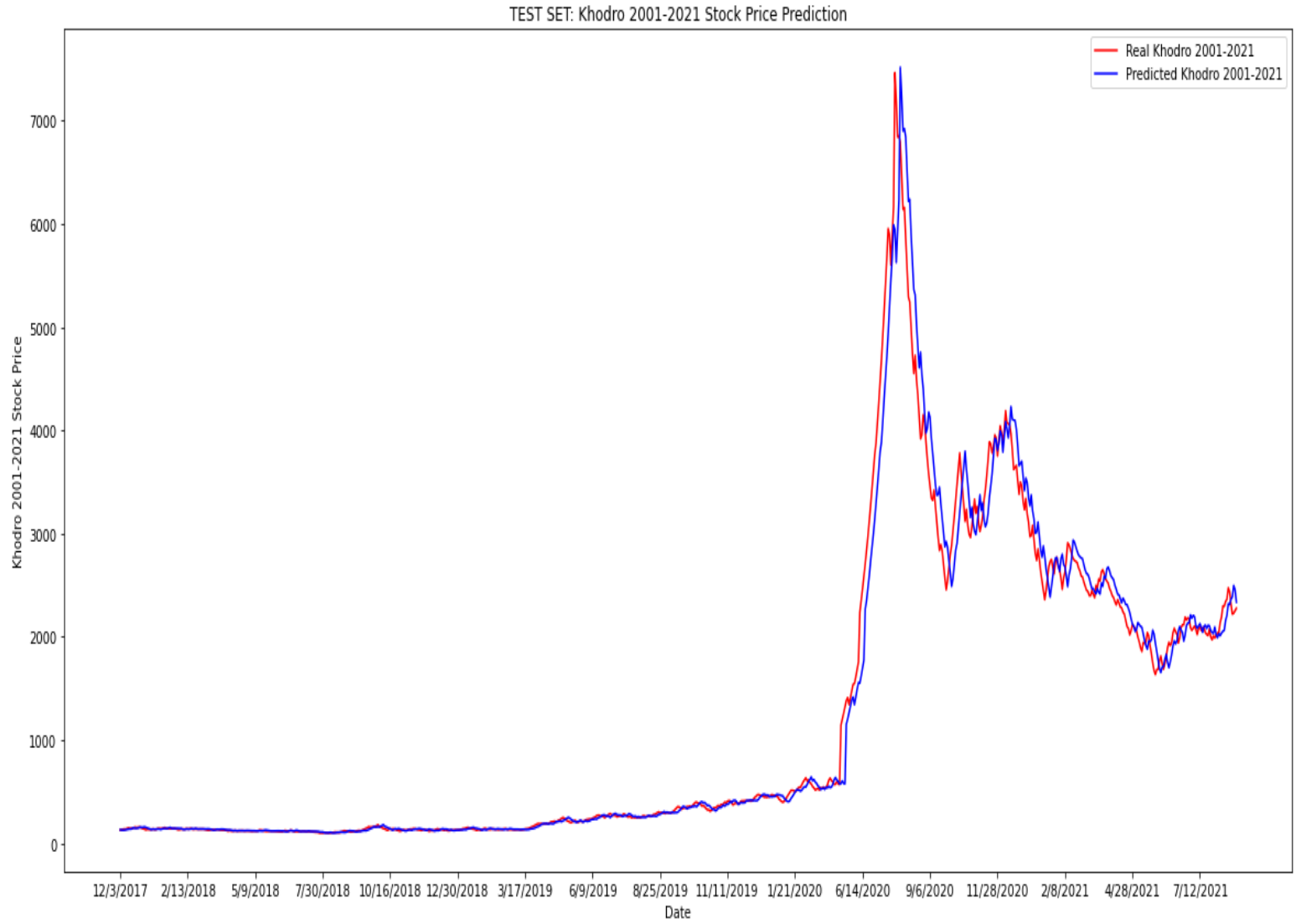
۴-۴-۱۱-۱- رسم نمودار برای پیشبینی داده های Test

قطعه کد زیر اقدام به ساخت نموداری برای رسم داده های پیشبینی شده و داده های Test میکند

```
# visualising the results
date = test_df[seq_length+step_predict:test_df.shape[0]]['Date'].values
plt.figure(figsize=(20,10))
plt.plot(date, test_price, color = 'red', label = 'Real Khodro 2001-2021')
plt.plot(date, test_predicted, color = 'blue', label = 'Predicted Khodro 2001-2021')
plt.xticks(np.arange(0,len(test_predicted),50))
plt.title('TEST SET: Khodro 2001-2021 Stock Price Prediction')
plt.xlabel('Date')
plt.ylabel('Khodro 2001-2021 Stock Price')
plt.legend()
plt.show()
print(f'RMSE: {round(mean_squared_error(test_price, test_predicted, squared=False),4)}')
#notice That MAE not RMAE
print(f'MAE: {round(mean_absolute_error(test_price, test_predicted),4)}')
print(f'r2_score: {round(r2_score(test_price, test_predicted),4)}')
```

۴-۲۵ کد رسم نمودار پیشبینی مدل با داده های Test

در شکل زیر خروجی قطعه کد بالا برایمان نمایان میشود .



۲۶-۴ نمودار خروجی مربوط به پیشبینی مدل با داده های Test

همانطور که در پارامترهای پایین نمودار مشاهده میکنید نتایج مربوط به آن اختلاف معناداری بین این مرحله و دو مرحله قبلی شکل گرفته است بطوری که در این مرحله RMSE :227.6522 وMAE : 105.7628

در وهله‌ی اول ما به پیدایش بیش پردازش در مدل مشکوک میشویم که پس از بررسی های گوناگون به این نتیجه رسیدیم که مدل ما داری بیش پردازش نیست و مشکل از داده های تست ما می باشد .

همانطور که شاهد بودیم در این چند سال اخیر با به وقوع افتادن عوامل متعددی در مسائل اقتصادی و سیاسی کشورمان توجه حاکمیت و شهروندان به بازار سهام و دارایی معطوف شد از یکسو تلاش حاکمیت

برای جبران کسری بودجه و کنترل نقدینگی و از طرف دیگر فعالیت شهروندان برای حفظ دارایی های خود در مقابل تورم و بی ارزشی پول رایج مملکت ، لذا ما شاهد افزایش و کاهش های بسیار شدیدی در بازار سهام شدیم که تقریبا برای اکثریت سهام های بازار اتفاق افتاد در نتیجه وجود چنین اتفاق خارج از پیشبینی و ناهماهنگ با سری های قبلی بازار سرمایه منجر به این شد که مدل برای این ۲ الی سه سال اخیر نتواند پیشبینی درستی انجام دهد و مقدار خطای خود را افزایش دهد البته قابل ذکر است که با وجود چنین اتفاقات ناگواری باز هم مدل توانسته به خطای قابل قبولی برای پیشبینی سهام ایرانخودرو برسد که بسیار حائز اهمیت است که این مدل به شکل چشمگیری برای این سهام خوب کار میکند .

با توجه به صحبت های اخیر ما تصمیم گرفتیم که ۲۰ درصد پایانی داده ها که مربوط به سه سال اخیر میباشد را به کلی از مدل حذف کرده و مجددا مدل را با دیتاهای جدید آموزش و تست کرده و نتایج آن را در فصل بعد به نمایش گذاشته ایم .

۴-۵- نتیجه

همانطور که مشاهده کردید ما در این فصل اقدام به ساخت ، کامپایل و تست مدل خود کرده ایم تمام عملیات مربوطه با داده های شرکت ایرانخودرو که مربوط به توالی سال های ۲۰۰۱ تا ۲۰۲۱ بود انجام دادیم و مشاهده کردیم لذا در فصل بعد اقدام به نمایش گزارش تصویری از نتایج خروجی مدل برای دیگر سهام های بازار بورس ایران میپردازیم

فصل ۵– نمونه ورودی‌ها و خروجی‌ها

۵–۱– مقدمه

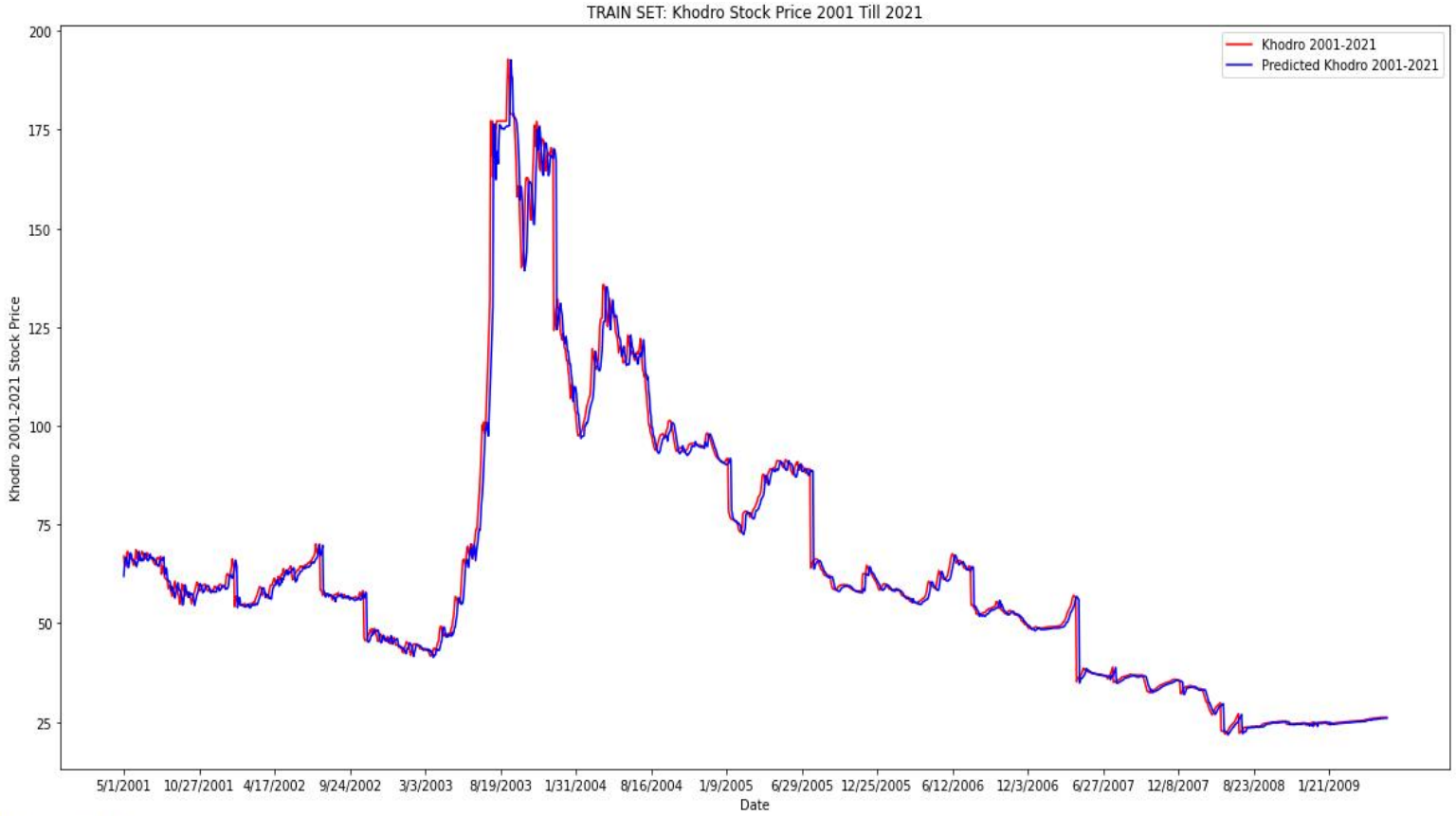
در این فصل نمونه ورودی‌ها و خروجی‌های بیشتری بررسی خواهد شد. ورودی برنامه شامل یک فایل مربوط به سوابق معاملات شرکت مورد نظر ، در اینجا منظور از شرکت مورد نظر سهام خاص در بازار بورس ایران میباشد و خروجی ها مربوط به نمودار رسم شده در هر مرحله میباشد .

همانطور که قبلا ذکر شد به دلیل بروز دستکاری های بسیاری در روند بازار در این سه سال اخیر ۲۰ درصد از داده های آخر تمام شرکت های مورد نظر را به کلی حذف کردیم و مدل را بر اساس سوابق باقی مانده پیاده سازی نمودیم .

۵–۲– شرکت ایرانخودرو

داده های این شرکت از سال ۲۰۰۱ تا ۲۰۲۱ موجود میباشد .

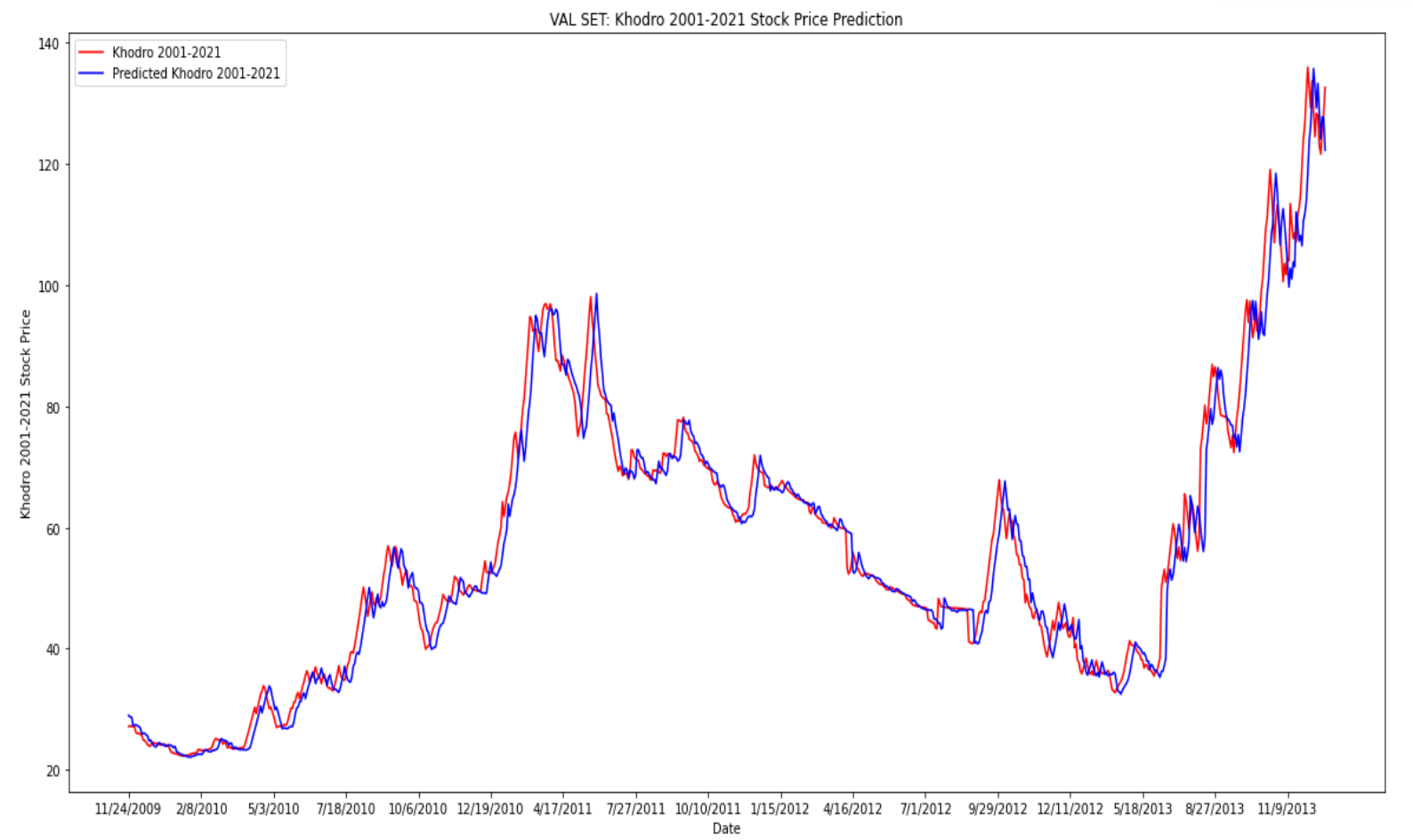
۵–۲–۱– نمودار پیشبینی مدل با داده های Train



RMSE: 4.9652
MAE: 2.1428
r2_score: 0.98

شکل ۵-۱ نمودار پیشبینی ایرانخودرو با داده های Train

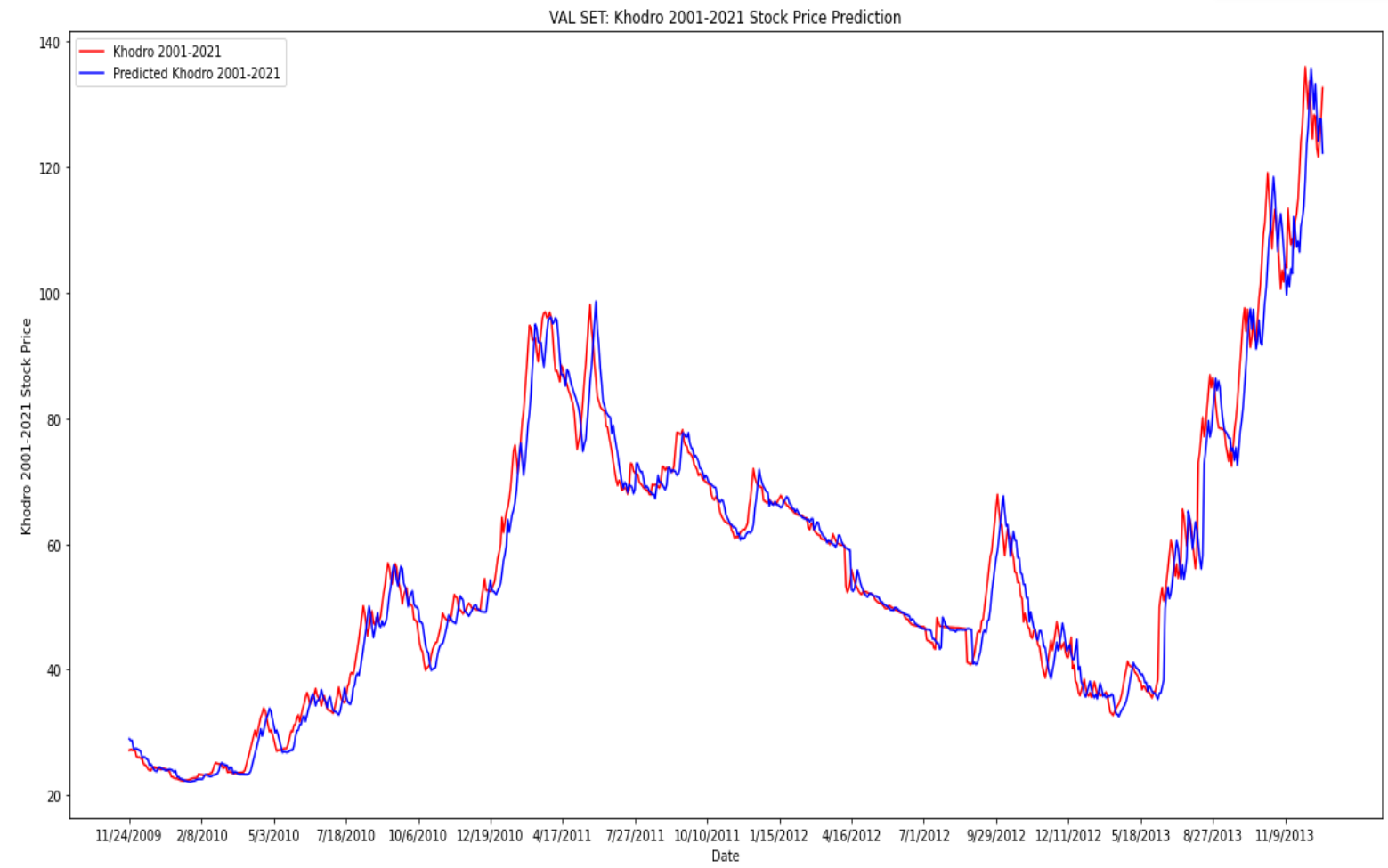
۵–۲–۲– نمودار پیشبینی مدل با داده های Validation



RMSE: 4.2697
MAE: 2.7852
r2_score: 0.9671

شکل ۵-۲ نمودار پیشبینی ایرانخودرو با داده های Validation

۵-۲-۳- نمودار پیشبینی مدل با داده های Test



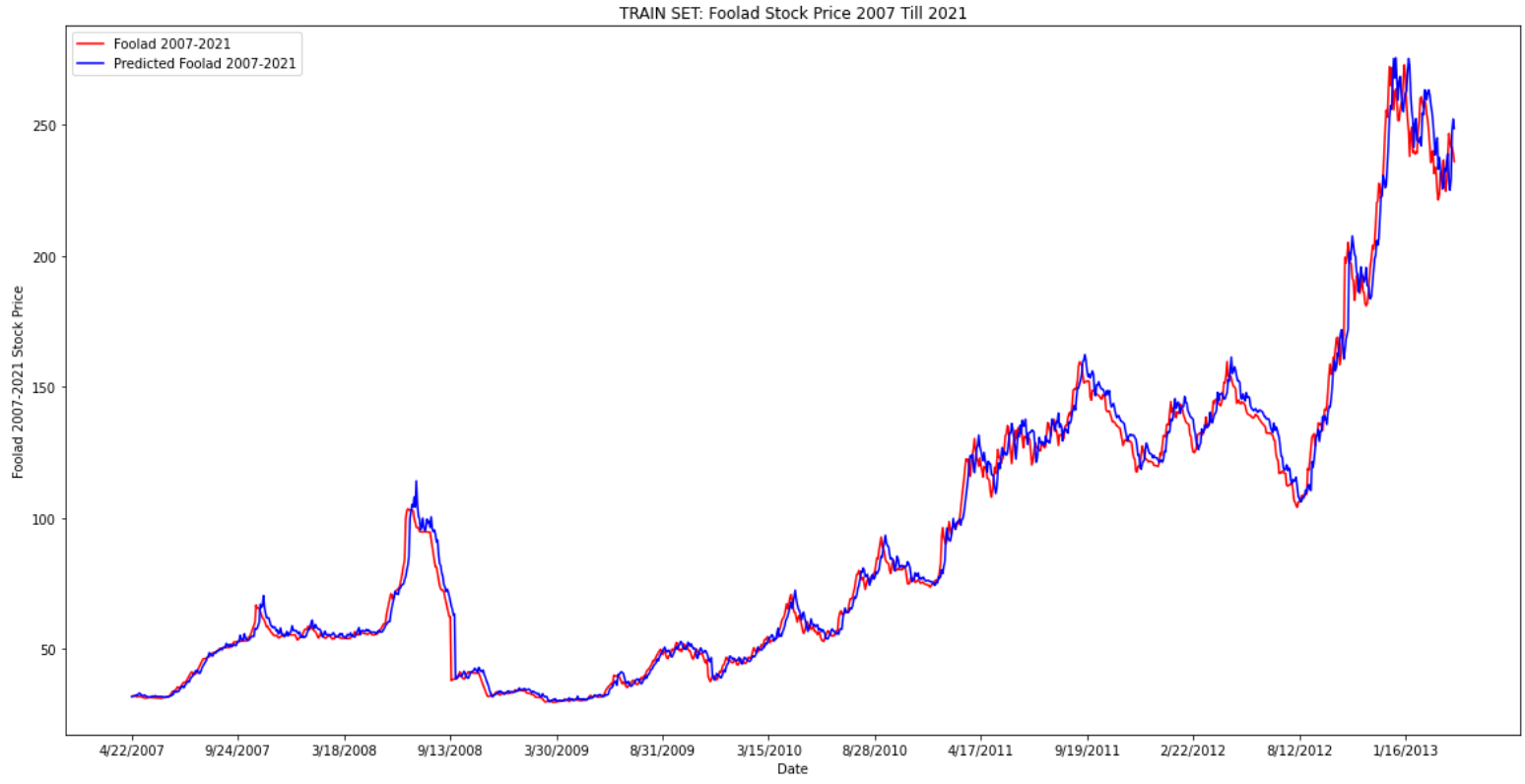
RMSE: 4.2697
MAE: 2.7852
r2_score: 0.9671

شکل ۵-۳ نمودار پیشبینی ایرانخودرو با داده های Test

۵–۳– شرکت فولاد مبارکه

داده های این شرکت از سال ۲۰۰۷ تا ۲۰۲۱ موجود می باشد

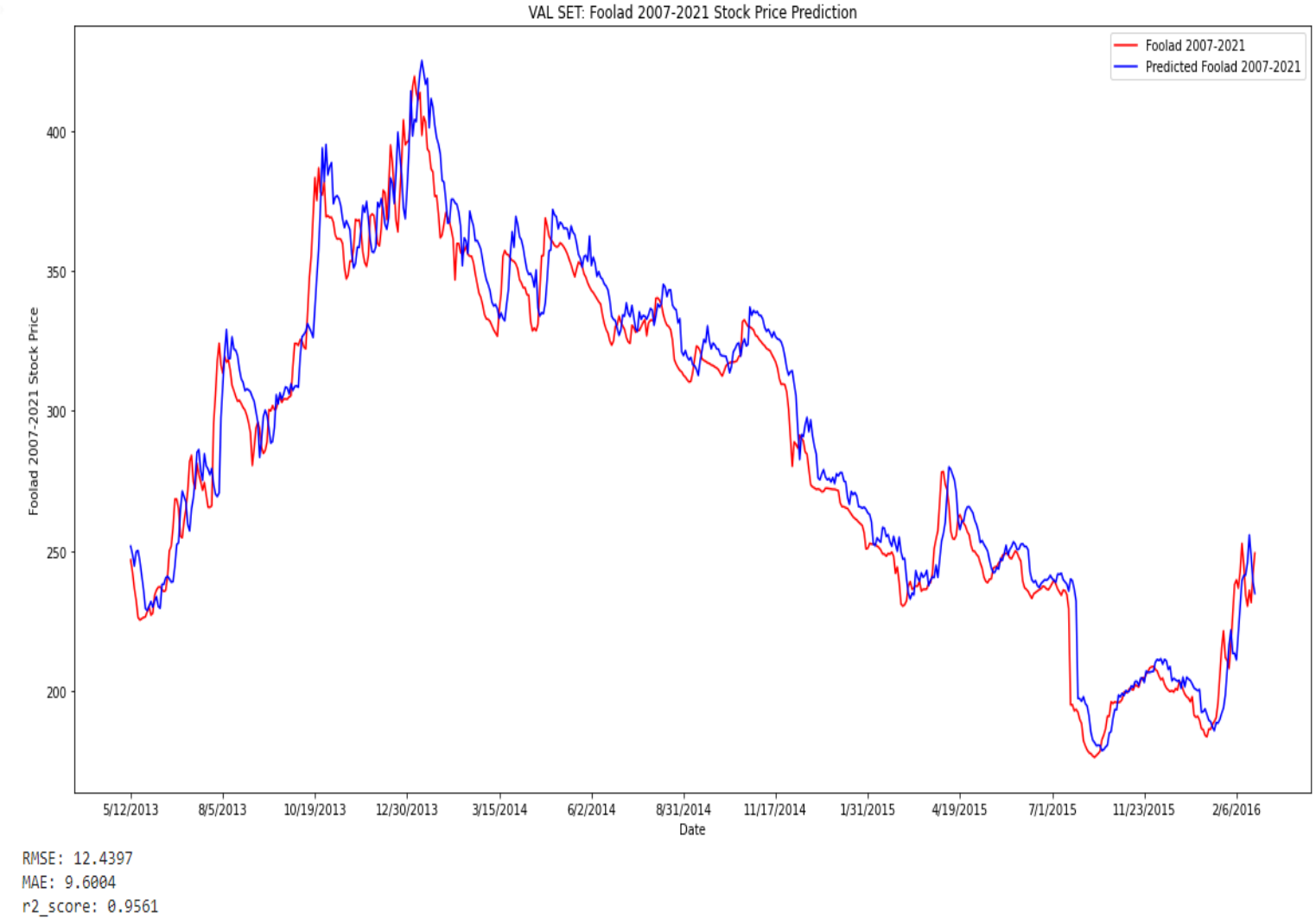
۵–۳–۱– نمودار پیشبینی مدل با داده های Train




```
RMSE: 5.9569
MAE: 3.8001
r2_score: 0.9894
```

شکل ۵-۴ نمودار پیشبینی فولاد مبارکه با داده های Train

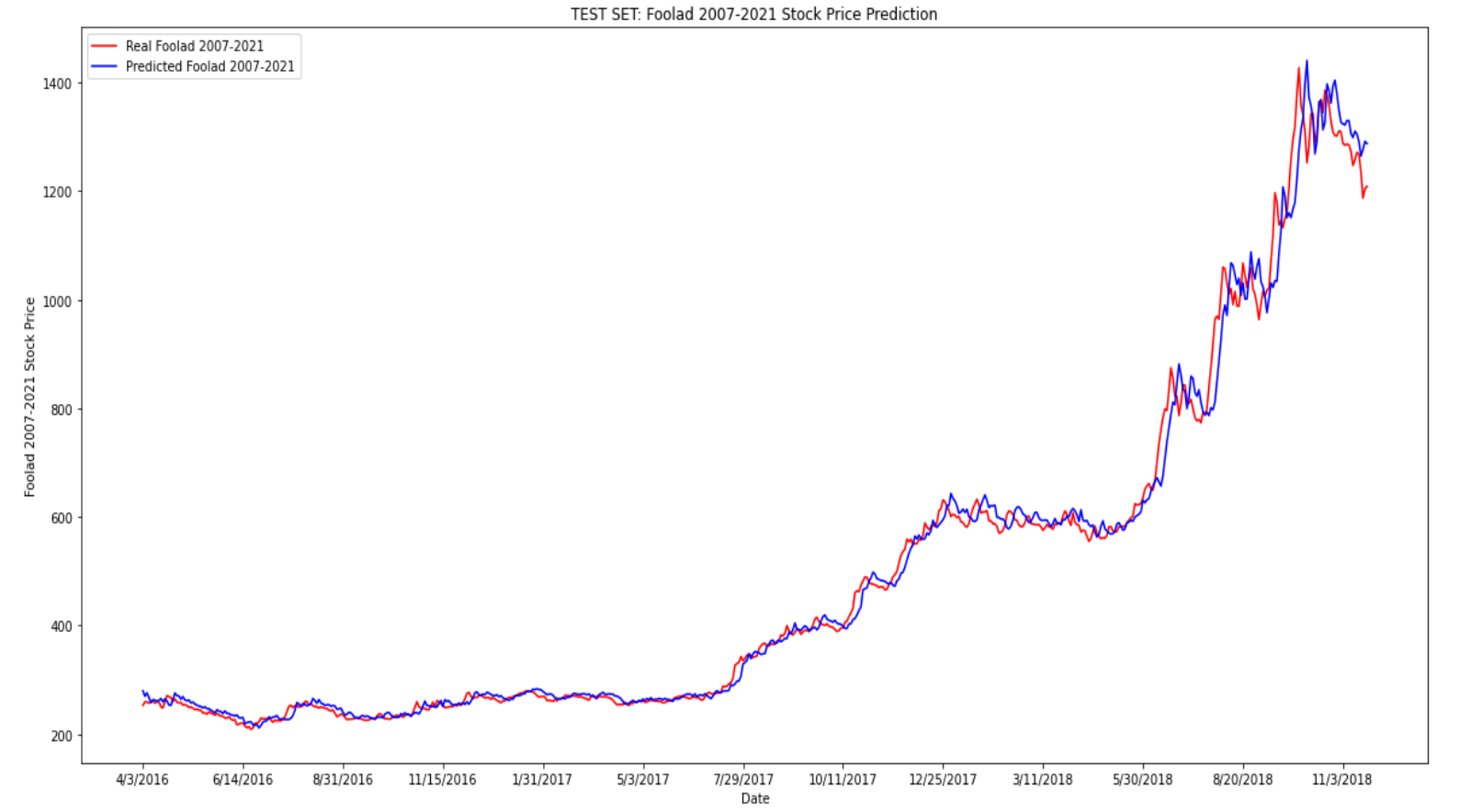
۵–۳–۲– نمودار پیشبینی مدل با داده های Validation



RMSE: 12.4397
MAE: 9.6004
r2_score: 0.9561

شکل ۵-۵ نمودار پیشبینی فولاد مبارکه با داده های Validation

۵–۳–۳– نمودار پیشبینی مدل با داده های Test



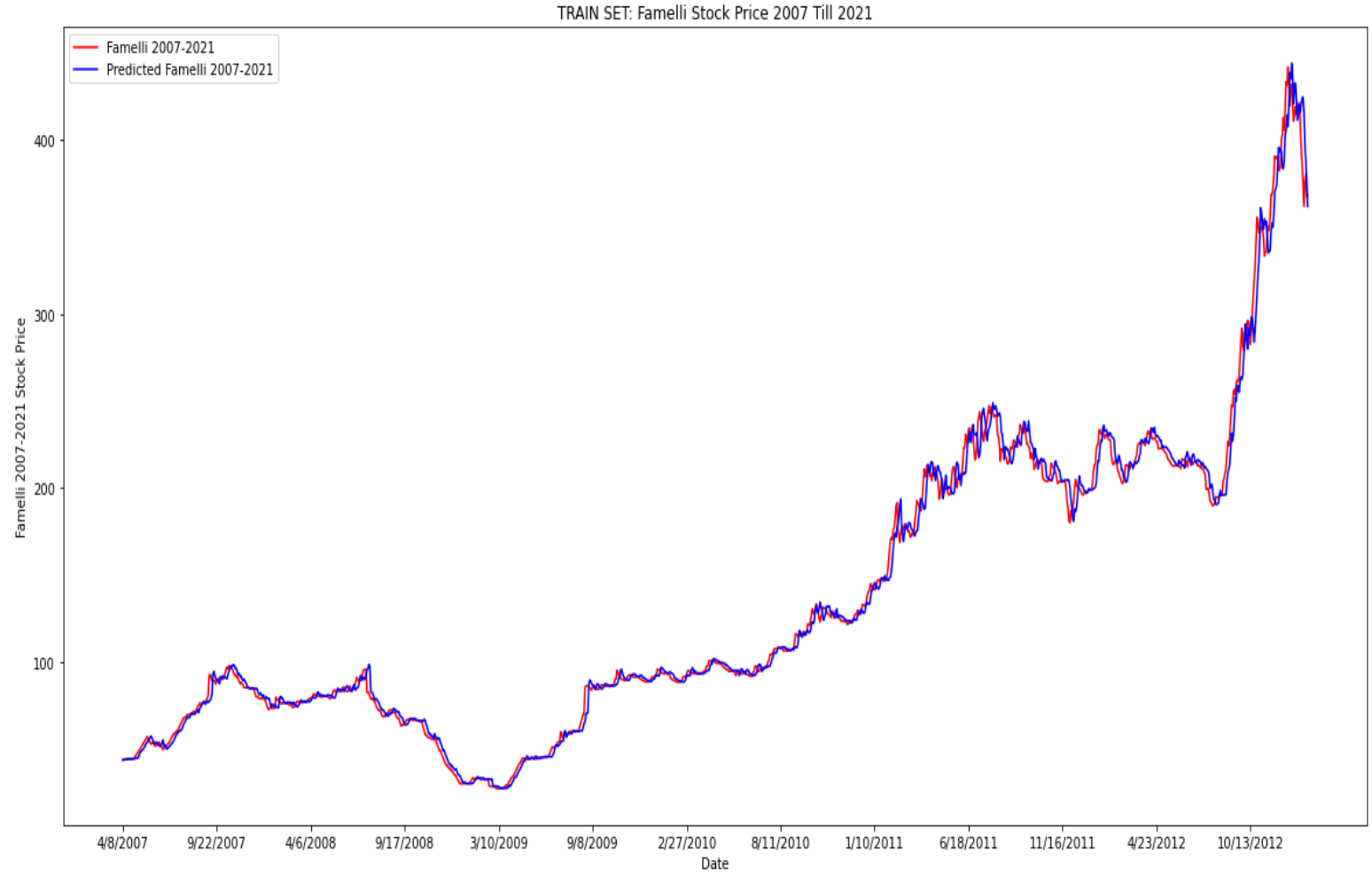

```
RMSE: 31.1012
MAE: 17.9333
r2_score: 0.9901
```

شکل ۵-۶ نمودار پیشبینی فولاد مبارکه با داده های Test

۵-۴- شرکت صنایع مس ایران

داده های این شرکت از سال ۲۰۰۷ تا ۲۰۲۱ موجود میباشد

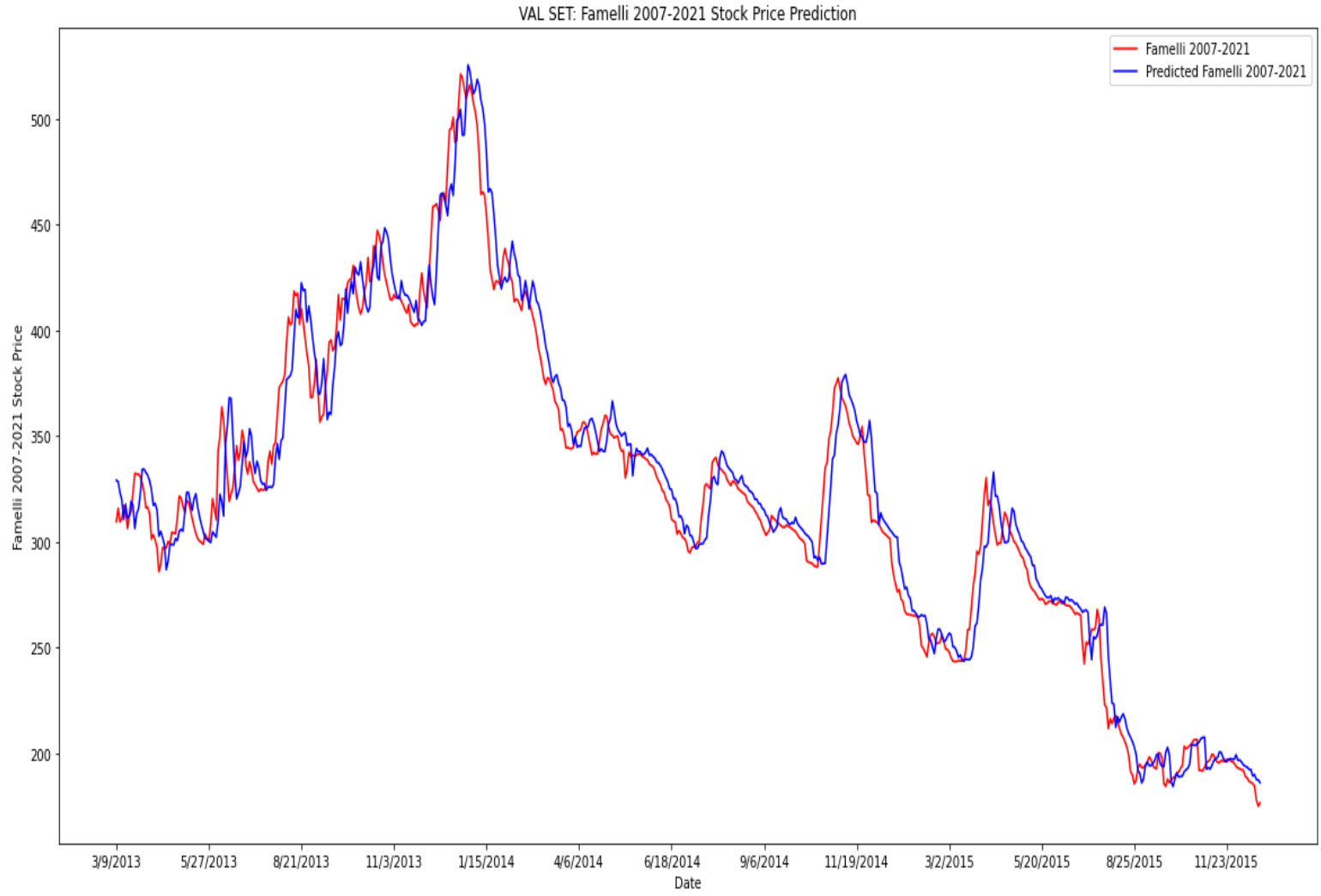
۵-۴-۱- نمودار پیشبینی مدل با داده های Train



RMSE: 7.1798
MAE: 4.3586
r2_score: 0.9931

شکل ۵-۷ نمودار پیشبینی صنایع مس ایران برای داده های Train

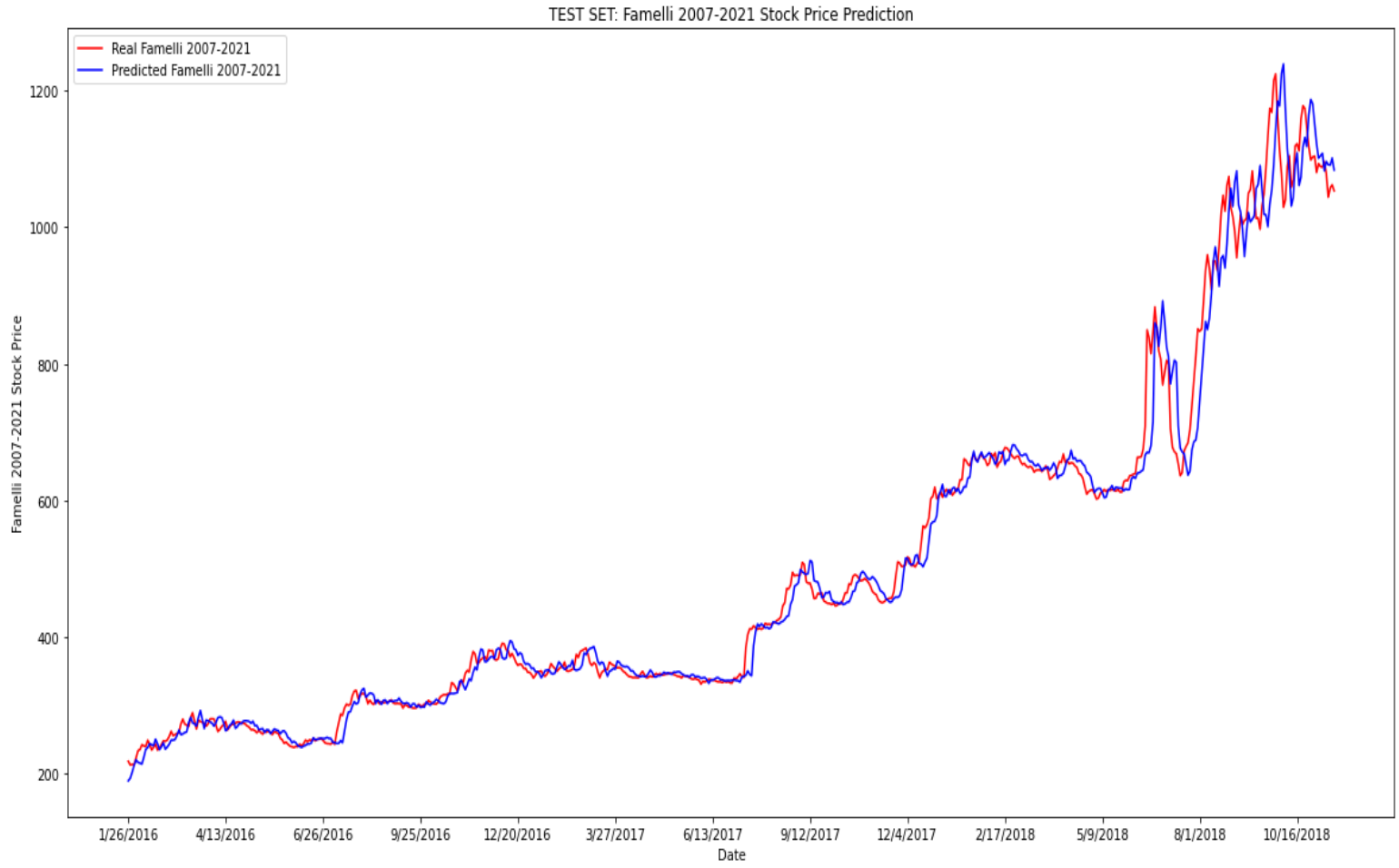
۵–۴–۲– نمودار پیشبینی مدل با داده های Validation



RMSE: 14.5477
MAE: 11.0681
r2_score: 0.9631

شکل ۵-۸ نمودار پیشبینی شرکت صنایع مس ایران برای داده های Validation

۵–۴–۳– نمودار پیشبینی مدل با داده های Test



RMSE: 33.2825
MAE: 18.6244
r2_score: 0.982

شکل ۵-۹ نمودار پیشبینی صنایع مس ایران برای داده های Test

۵-۵- نتیجه

در این پروژه نتایج سه شرکت از صنایع متفاوت که هر کدام جزو شرکت های بزرگ با ضریب نقد شوندگی قابل توجه بودند با پیاده سازی یک مدل شبکه عصبی بازگشتی حافظه طولانی کوتاه-مدت بررسی گردید عملکر مدل برای هر سه شرکت بسیار فراتر از انتظار بود اگر چه با توجه به توضیحات قبل، ما از ۲ الی ۳ سال آخر داده های مالی هر شرکت صرف نظر کردیم اما نتیجه‌ی عملکرد مدل درخشان بود در این فصل شاهد بودیم که بهترین عملکر در پیشبینی و کمترین خطا مربوط به شرکت ایرانخودرو بود، میتوان دلایل متعددی برای این که مدل برای این شرکت خاص عمکرده خیره کننده ای را داشته بیان نمود از مهمترین دلیل آن میتوان به بیشتر بودن داده های مالی برای آموزش مدل در نظر گرفت و همچنین میزان اقبالیت بازار به این سهم از قدیم الایام اشاره نمود، چرا که هر چه تراکنش های مربوطه به شرکتی بیشتر باشد در نتیجه عمق معاملات آن و نحوه ی روند آن بسیار منطقی تر از شرکتی است که توسط شرکت ها و حقوقی های بزرگ، روند آن دستکاری میشود لذا ما شرکت ایرانخودرو را به عنوان منبع برای ارائه مدل در نظر گرفتیم.

فصل ۶– نتیجه‌گیری و پیشنهادها

۶–۱– نتیجه‌گیری

همانطور که در فصل قبل بیان شد عملکرد مدل ما به وسیله ی سه شرکت متفاوت بررسی شده در جدول زیر میزان خطا و امتیاز را برای هر کدام از سه شرکت نشان داده است

نام شرکت	تعداد داده های آموزش	تعداد داده های تست	RMSE	MAE	r2_score
ایرانخودرو	۲۵۵۳	۸۵۱	۷,۷۶۳۷	۵,۶۴۰۴	۰,۹۱۴۱
صنایع مس ایران	۱۹۲۹	۶۴۳	۳۳,۲۸۲۵	۱۸,۶۲۴۴	۰,۹۸۲
فولاد مبارکه	۱۹۰۶	۶۳۵	۳۱,۱۰۱۲	۱۷,۹۳۳۳	۰,۹۹۰۱

جدول ۶-۱ نتایج کلی عملکرد مدل

۶-۲- پیشنهادها

این پروژه را میتوان ابتدای کار و شروعی بر آشنایی ، پیاده سازی و درک عمیق تر مباحث یادگیری عمیق و نحوه‌ی عمکرد آن برای سری های زمانی دانست لذا توصیه میشود برای درک و لمس مباحث مربوطه به این حوزه حتما اقدام به پیاده سازی مدلی بر مبنای دانسته های خود و سپس اقدام به اصلاح و ارتقای آن بپردازید که به موجب این فعالیت افق روشن و دیدگاه وسیعی بر ادامه‌ی روند پیشرفت شما قرار میگیرد.

۶-۲-۱- چشم انداز و افق پروژه

برای تکمیل و گسترش موضوع پیشبینی های مالی در نظر داریم که با بسط پروژه و پیاده سازی مدل های دیگر بر همین مبنا از جمله مدل های ترکیبی مانند CNN-LSTM اقدام به بهبود و خلق مدلی یکتا بیانجامیم که علاوه بر پیشبینی روند معاملاتی بتواند به طور خودکار معاملات الگورتمی هم انجام دهد تا بتوان از آن به عنوان یک سیستم سود ده در بازار های مالی یاد کرد .

فهرست مراجع

- Kim, T., & Kim, H. Y. (2019). Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. *PloS one*, *14*(2), e0212320.
- Christopher Olah. (n.d.). *Understanding LSTM Networks -- colah's blog*. Colah. Retrieved June 16, 2021, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Moghar, A., & Hamiche, M. (2020). Stock market prediction using LSTM recurrent neural network. *Procedia Computer Science*, *170*, 1168-1173.
- Brownlee, J. (2020, August 28). *How to Develop LSTM Models for Time Series Forecasting*. Machine Learning Mastery. https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
- *Keras: the Python deep learning API*. (n.d.). Keras. from https://keras.io/
- *deeplizard*. (n.d.). YouTube. from https://www.youtube.com/c/deeplizard
- Brownlee, J. (2020, August 28). *How to Develop LSTM Models for Time Series Forecasting*. Machine Learning Mastery. https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/

پایان