الگوریتم GMDH در پیش بینی قیمت سهام در بازار بورس

پروژه مقطع کارشناسی

ساجد زرین پور

استاد راهنما: فرشید مهردوست

تابستان ۱۳۹۵


بسم الله الرحمن الرحيم

چکیده

در این مقاله یک شبکه عصبی مدل GMDH و الگوریتم ژنتیک برای پیش بینی قیمت سهام ارائه شده است. برای پیش بینی قیمت سهام با شبکه عصبی مبتنی بر الگوریتم GMDH، ما از سود هر سهم (EPS)، سود پیش بینی شده هر سهم (PEPS)، سود نقدی هر سهم (DPS)، نسبت قیمت به درآمد (P/E) و نسبت درآمد به قیمت (E/P) به عنوان داده‌های ورودی و قیمت سهام به عنوان داده خروجی استفاده می کنیم. برای این کار، از یک پایگاه داده ساختگی استفاده کرده ایم. شبکه عصبی با ٪۸۵ داده های تجربی ساخته شده است. برای آزمودن تناسب بودن مدل، از باقی داده ها در شبکه استفاده می کنیم.

فهرست

مقدمه

پیش بینی قیمت سهام یکی از وظایف اصلی سرمایه گذاران حقیقی و حقوقی می باشد. این، یک مسئله مهم در تصمیم گیری های سرمایه گذاری مالی است و در حال حاضر توجه زیادی را از سوی جامعه محققین به خود جلب کرده است. گرچه، این مسئله با توجه به این‌که قیمت سهام / سود آن پرنوسان و غیرثابت است، به عنوان یکی از چالش برانگیزترین مسائل قلمداد می‌گردد (Hall,1994, Li et al 2003, Yaser and Atiya 1996).

تغییرات قیمت سهام یک سیستم بسیار دینامیک است که از تعدادی اصول مشتق شده است. دو روش اصلی تحلیلی، آنالیز بنیادی و آنالیز فنی اند. آنالیز بنیادی از فاکتورهای کلی اقتصادی مانند نرخ سرمایه گذاری، سرمایه مالی، نرخ های تورم، و نرخ مبادلات خارجی به عنوان وضعیت بنیادی مالی شرکت استفاده می‌کند. پس از تحلیل دقیق تمام این فاکتورها، تحلیل‌گر تصمیمی مبنی بر خرید یا فروش یک سهم می‌گیرد. یک تحلیل فنی مبتنی بر داده های سری‌های زمانی مالی است. گرچه، سری‌های زمانی مالی کاملا داده‌های پیچیده ای ارائه می‌دهند (برای مثال، روندها، تغییرات ناگهانی، فراریت دسته بندی) و این‌گونه سری‌ها اغلب غیرثابت-اند، جایی که یک متغییر گرایش روشنی برای حرکت به سمت یک مقدار ثابت یا یک روند خطی ندارد (Cheng and Liu 2008).

هدف این مقاله کاربرد شبکه‌های عصبی مبتنی بر الگوریتم GMDH برای پیش بینی قیمت سهام است. برای این‌کار، ما از سود هر سهم (EPS)، سود پیش بینی شده هر سهم (PEPS)، سود نقدی هر سهم (DPS)، نسبت قیمت به درآمد (P/E) و نسبت درآمد به قیمت (E/P) به عنوان داده‌های ورودی و قیمت سهام به عنوان داده خروجی استفاده می کنیم. دیدگاه تحلیل پیشنهادی در این مقاله از شبکه عصبی مبتنی بر الگوریتم GMDH برای پیش بینی قیمت سهام استفاده می‌کند و می‌تواند کمک خوبی برای سرمایه گذاران و تحلیل گران مالی باشد. این مقاله به صورت زیر سازمان‌دهی شده است.

بخش ۱ به معرفی مفاهیم اقتصادی می‌پردازد. بخش ۲ به معرفی اجمالی الگوریتم GMDH اقدام نموده و در بخش ۳ صورت ریاضی الگوریتم GMDH را بیان می‌نماید. بخش ۴ به توصیف اجمالی شبکه عصبی مبتنی بر الگوریتم GMDH پرداخته و در بخش ۵ توضیحات مربوط به پیاده سازی این الگوریتم در نرم افزار متلب آورده شده است. و در انتهاکد کامل برنامه در ضمیمه ۱ و اطلاعات مدل در ضمیمه ۲ آورده شده‌اند. کد کامل برنامه به همراه یک پایگاه داده تمثیلی برای مشاهده نحوه عملکرد برنامه در فایل های ضمیمه در اختیار علاقه‌مندان قرار گرفته است.

۱. نگاهی به مفاهیم کلی سهام

۱.۱ درآمد هر سهم (EPS): با محاسبه این رقم، سودی که شرکت در یک دوره مشخص به ازای یک سهم عادی به دست آورده معین می شود.

$$EPS = \frac{\text{سود پس از کسر مالیات}}{\text{تعداد سهام عادی منتشره}}$$

علت اصلی توجه به سود هر سهم (و نه سود کل شرکت) به هدف اصلی شرکت مربوط است که به حداکثر رسانیدن ثروت سهامداران می باشد.

۲.۱ درآمد پیش بینی شده هر سهم (PEPS)؛ آخرین پیش بینی از درآمد هر سهم است.

۳.۱سود نقدی هر سهم (DPS)؛ به مقدار سودی که شرکت تقسیم می‌کند و به طور نقدی به دست سهامدار می‌رسد، DPS گفته می شود. اینکه چه میزان از سود تحقق یافته هر سهم به صورت نقدی توزیع شود و چه میزان در شرکت باقی بماند تصمیمی است که در مجمع عادی سالانه توسط هیئت مدیره به مجمع پیشنهاد داده می‌شود و سپس به تصویب سهامداران می رسد. فرمول محاسبه سود نقدی توزیع شده (DPS) در ذیل نشان داده شده است.

DPS = EPS - (اندوخته قانونی + اندوخته طرح و توسعه + سود انباشته)

۴.۱ نسبت E/P و P/E سود هر سهم،E/P مبلغ سود به ازای ارزش اسمی سهام را نشان می دهد. با توجه به اینکه ارزش اسمی سهام کلیه شرکت های پذیرفته شده در بورس تهران ۱۰۰۰ ریالی بوده و ممکن است قیمت بازار انواع سهام متفاوت باشد، برای مقایسه شرکت ها با یکدیگر رقم سود نسبت به قیمت بازار تحت عنوان E/P سنجیده می شود. این نسبت بازده جاری سهام را نشان داده و در شرایط تساوی سایر عوامل، هرچه این نسبت بالاتر باشد سهام از موقعیت بهتری برخوردار خواهد بود. لازم به ذکر است هرچه میزان سود در هر سهم در مقایسه با قیمت بازار بیشتر باشد به خاطر تقاضایی که برای آن ایجاد می شود انتظار احتمال افزایش آتی قیمت بالاتر خواهد بود. خاطر نشان می شود که سهام داران در خرید سهام می توانند از نسبت قیمت به درآمد تحت عنوان P/E نیز استفاده کنند. نسبت P/E یکی از ابزارهای رایج و متداول جهت تحلیل وضعیت شرکت ها، صنعت و بازار است. این نسبت که از آن به عنوان ضریب سودآوری نیز نام برده می شود، حاصل تقسیم سهم بردرآمد هر سهم آن شرکت است و در واقع رابطه بین سهام شرکت با سود آن را نشان می دهد. همانطور که گفته شد، سود هر سهم، معیاری است که می توان با آن یک شرکت را طی چند سال متوالی از لحاظ افزایش سرمایه های شرکت مقایسه نمود ولی این متغییر، معیار مناسبی برای مقایسه چند شرکت نیست. این نقیصه در معیار P/E رفع می‌شود و با این معیار می‌توان شرکت های یک صنعت را مورد مقایسه قرار داد یا هر شرکت را با متوسط P/E صنعت مرتبط تطبیق داد. نسبت P/E بیانگر تعداد دفعاتی است که طول می کشد تا قیمت یک سهم از طریق سود آن بازیافت شود. البته تحلیل P/E براساس دوره بازگشت سرمایه باید توأم با احتیاط باشد. به دلیل این که شرکت ها در هر سال سود یکسانی ندارند و درآمد هر سهم آنها توأم با نوسان است. بنابراین در صورتی که P/E شرکتی برابر ۱۰ باشد لزوما به این معنی نیست که ۱۰ سال طول می کشد تا سرمایه بازگشت کند. اگر شرکت در سال های بعد سود بیشتری کسب کند، دوره بازگشت سرمایه کاهش می یابد. لازم به ذکر است که برای مقایسه P/E بین شرکت ها، مقایسه در درون یک صنعت انجام شود. با فرض ثابت

بودن سایر شرایط به نظر می رسد شرکتی که در گروه خود دارای P/E کوچکتری می باشد امکان رشد قیمت سهام آن در آینده نزدیک وجود دارد.

۲. نگاهی به الگوریتم GMDH

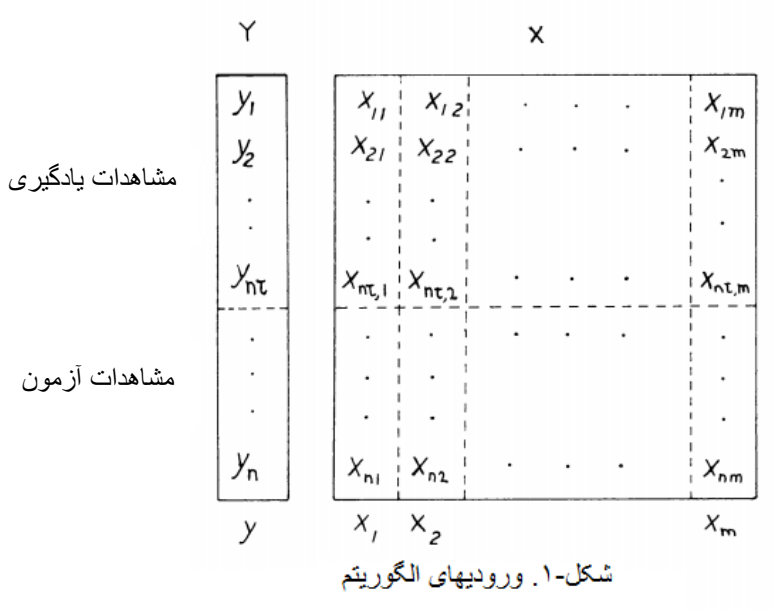
الگوریتم پایه ای GMDH یک روش ساخت چند جمله ای با مرتبه بالا به فرم

$$y = a + \sum_{i=1}^{m} b_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} d_{ijk} x_i x_j x_k + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m}\sum_{t=1}^{m} e_{ijkt} x_i x_j x_k x_t + \dots \qquad (۱.۲)$$

است که m متغیر ورودی $x_1, x_2, \dots, x_m$ را به یک متغییر خروجی y مرتبط می سازد.گرچه (۱.۲) به یک چند جمله ای با مرتبه بالای رگرسیون شباهت دارد، روشی که این چندجمله ای (که از این پس آن را چند جمله ای ایواخننکف می نامیم؛) طی آن ساخته می شود با تکنیک های متداول در آنالیز استاندارد رگرسیون تفاوت دارد. در واقع گفته می شود (اسکات و هاچینسون؛ ۱۹۷۶) که روش ساخت چندجمله ای ایواخننکف به روشی که طبیعت طی آن انتخاب طبیعی را انجام می دهد مشابه است. این روندی است که اکنون به توصیف آن خواهیم پرداخت.

۳. الگوریتم عددی GMDH

اطلاعات پایه ای که یک فرد باید برای ساخت چندجمله ای ایواخننکف جمع‌آوری کند یک مجموعه از n مشاهده مانند آنچه در شکل-۱ نمایش داده شده، است.



شکل-۱. ورودیهای الگوریتم

در شکل فوق :

nt : تعداد مشاهدات در مجموعه یادگیری،

nc=n-nt : تعداد مشاهدات در مجموعه آزمون،

n : تعداد کل مشاهدات و

m : تعداد متغییرها.

دلیل تقسیم مشاهدات به دو مجموعه مجزا به زودی بیان می شود. حال به توصیف گام های اصلی الگوریتم GMDH می پردازیم.

گام ۱. (ساخت متغییرهای جدید $z_1, z_2, \dots, z_{\binom{m}{2}}$)

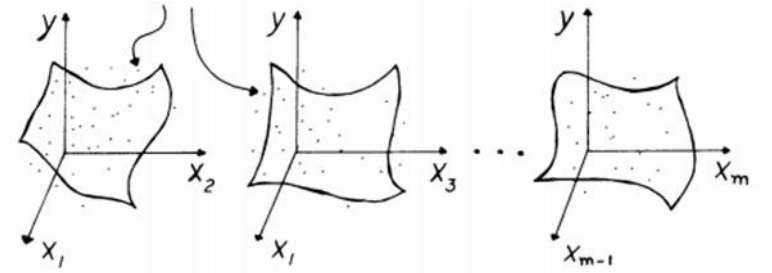
از تمام متغییرهای مستقل $x_1, x_2, \dots, x_m$ (ستون های آرایه X) هر بار ۲ متغییر انتخاب می کنیم و برای هریک از این $\binom{m}{2}$ ترکیب، چندجمله ای تقریب کمترین مربعات به فرم

$$y = A + Bu + Cv + Du^2 + Ev^2 + Fuv \quad (۱.۳)$$

را که بهترین تقریب برای مشاهده y_i در مجموعه یادگیری می باشد را پیدا می کنیم. بدین گونه $\binom{m}{2} = \frac{m(m-1)}{2}$ صفحه چندجمله ای مانند آنچه در شکل ۲ نشان داده شده است، می یابیم. حال،

برای هریک از این $\binom{m}{2}$ صفحه چندجمله ای نشان داده شده در شکل ۲، چند جمله ای را در n نقطه مقداریابی می کنیم.

مشاهدات مجموعه یادگیری



شکل-۲ ساختار متغییرهای جدید

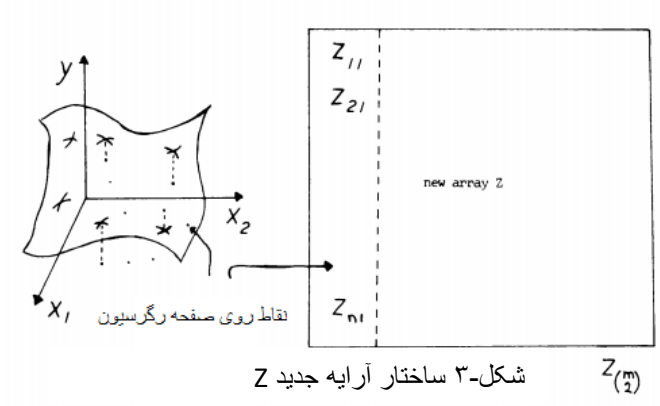
برای مثال برای اولین سطح نشان داده شده در بالا، ما چندجمله ای $y_k = A + Bx_i + Cx_j + Dx_i^2 + Ex_j^2 + Fx_ix_j$ را در نقاط $(x_{1i}, x_{1j}), (x_{2i}, x_{2j}), \dots, (x_{ni}, x_{nj})$ مقداریابی می کنیم و این n متغییر را در k-امین ستون از آرایه Z ذخیره می‌کنیم. $\binom{m}{2} - 1$ ستون باقی مانده به روش مشابهی ساخته می شوند (شکل ۳ را مشاهده کنید.).

می توان این $\binom{m}{2}$ ستون از Z را به عنوان $\binom{m}{2}$ مشاهده از متغییرهای $z_1, z_2, \dots, z_{\binom{m}{2}}$ که هریک از این متغییرهای جدید یک چندجمله ای از متغییرهای $x_1, x_2, \dots, x_m$ اند، در نظر گرفت. به عبارت دیگر ما متغیرهای جدیدی ساختیم که برخی از آنها با متغیرهای اصلی تعویض خواهند شد. هدف نگاه داشتن آن z_i هایی است که بهترین تخمین از متغییر خروجی y ارائه می دهند و دور ریختن اضافات است. این، جایی است که مجموعه آزمون وارد الگوریتم می شود.

گام ۲. (یافتن z-هایی که مهم نیستند.)

در این گام ستون هایی از X (متغییرهای قدیمی) را با ستون هایی از Z (متغییرهای جدید که چندجمله ای های مربعی از متغییرهای قدیمی اند) که بهترین پیش بینی از بردار مشاهده شده y ارائه می دهند، تعویض می کنیم. به طور دقیق تر، برای هر ستون j از Z، ما خطای کمترین مربعات d_j را توسط $d_j^2 = \sum_{i=nt+1}^{n} (y_i - z_{ij})^2$ (۲.۳) محاسبه می کنیم، سپس ستون های Z را براساس افزایش کمترین خطای مربعات، d_j ، مرتب می کنیم و پس از آن ستون هایی را که به ازای آنها $d_j < M$ (M یک عدد از پیش تعیین شده است.) برقرار است را انتخاب می کنیم تا با ستون های X تعویض شوند. (این ستون ها از Z ، آرایه X جدید را می سازند.)

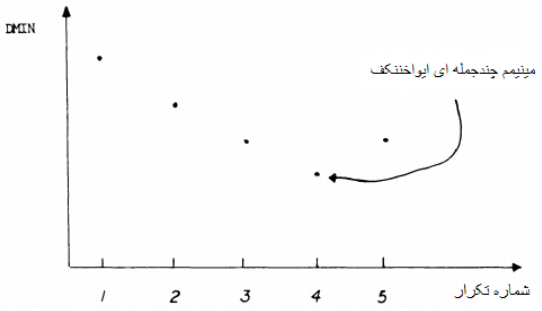
در این جا ذکر یک نکته ضروری است. تعداد این ستون های تعویض شده (مثلا m_i) ممکن است از تعداد ستون های اصلی (m) بیشتر یا کمتر باشد. همچنین توجه می کنیم که خطای d_j روش کمترین مربعات، برابر مجموع خطاهای مجموعه مشاهدات آزمون می باشد.



شکل-۳ ساختار آرایه جدید Z

گام ۳. (آزمون همگرایی)

از گام دوم، کوچکترین d_j را انتخاب می کنیم و آن را DMIN می نامیم و نمودار آن را مانند آنچه در شکل ۴ آمده، رسم می‌کنیم. اگر مقدار DMIN از مقدار DMIN قبلی کوچکتر باشد (اولین بار که وارد این گام می‌شویم درستی این مطلب را فرض می‌کنیم.)،



شکل-۴ معیار توقف

باز می گردیم و گام های اول و دوم را تکرار می کنیم. اما اگر مقدار جدید DMIN از مقدار قبلی آن بیشتر بود روند را متوقف می کنیم و از نتایج مربوط به DMIN قبلی استفاده می کنیم.

رگرسیون استاندارد تنها از میانگین خطای مربعات به عنوان معیار استفاده می کند و بنابراین تخمین اینکه مدل بسیار ساده یا بسیار پیچیده است غیرممکن می باشد. با محک زدن چندجمله ای ایواخننکف با مجموعه آزمون، بدست آوردن یک چندجمله ای یکتا با پیچیدگی بهینه ممکن است.گرچه تئوری الگوریتم GMDH به ما اجازه اثبات یکتایی مقدار مینیمم DMIN را نمی‌دهد، اما تمامی نتایج تجربی درستی این موضوع را تایید می کند. ذات منحنی DMIN اینگونه است که ابتدا به یک مقدار مینیمم کاهش می یابد و پس از آن شروع به افزایش می کند. برای مثال در شکل ۴ ما روند را پس از تکرار پنجم متوقف می‌کنیم. این، پایان الگوریتم ایواخننکف است. حال اولین ستون Z شامل مقادیر $\bar{y}_i$ از چندجمله ای ایواخننکف

$$\bar{y}_1 = a + \sum_{i=1}^{m} b_i x_{1i} + \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij} x_{1i} x_{1j} + \cdots$$

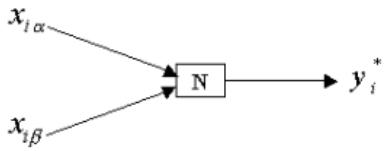
$$\vdots$$

$$\bar{y}_n = a + \sum_{i=1}^{m} b_i x_{ni} + \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij} x_{ni} x_{nj} + \cdots$$

مقداریابی شده در n نقطه داده اصلی می باشد. به عبارت دیگر ستون اول باید به میزان زیادی به متغییر وابسته y همگرا باشد. برای یافتن مقادیر مطلوب ضرایب $a, b_i, c_{ij}, \dots$ در چندجمله ای ایواخننکف (۱.۱) باید تمامی سطوح رگرسیون به فرم (۱.۳) را که در هر تکرار محاسبه می شوند، ذخیره شوند و به طور منظم، این چندجمله ای های متشکل از چندجمله ای های متشکل از چندجمله ای ها مقداریابی شوند، و این روند تا زمان رسیدن به چندجمله ای ایواخننکف ادامه داده شود. این، یک مسئله غیر بدیهی برنامه نویسی است اما بهرحال قابل حل است.

۴.شبکه عصبی GMDH

شبکه عصبی GMDH، شبکه ای خود سازمان ده و یک سویه است که از چندین لایه و هر لایه نیز از چندین نرون تشکیل شده است. تمامی نرون ها از ساختار مشابهی برخوردارند؛ به طوری که دارای دو ورودی و یک خروجی هستند. هر نرون با پنج وزن و یک بایاس عمل پردازش را میان داده های ورودی و خروجی براساس رابطه زیر برقرار می کند.



$$y_{ik}^* = N(x_{i\alpha}, x_{i\beta}) = b^k + w_1^k x_{i\alpha} + w_2^k x_{i\beta} + w_3^k x_{i\alpha}^2 + w_4^k x_{i\beta}^2 + w_5^k x_{i\alpha} x_{i\beta} \qquad (۱.۴)$$

که در آن

$i = ۱,۲, \dots, n$ (که n تعداد داده ورودی و خروجی است.)،

و $k = ۱,۲, \dots, \binom{m}{۲}$

$\alpha, \beta \in \{۱,۲,۳, \dots, m\}$ (که m تعداد نرون‌های لایه قبلی است.) .

وزن ها بر اساس روش خطای کمترین مربعات محاسبه شده و سپس به منزله مقادیر مشخص و ثابت، در داخل هر نرون جایگذاری می شود. ویژگی ممتاز این نوع شبکه آن است که نرون های مرحله قبلی یا لایه قبلی، عامل و مولد تولید نرون های جدید به تعداد $\binom{m}{۲} = \frac{m(m-۱)}{۲}$ هستند و از میان نرون های تولید شده، به اجبار تعدادی از آنها حذف شده تا بدین وسیله از واگرایی شبکه جلوگیری شود.

نرون هایی که برای ادامه و گسترش شبکه باقی می‌مانند، ممکن است برای ایجاد فرم همگرایی شبکه و عدم ارتباط آنها با نرون لایه آخر حذف شوند. به این لایه ها در اصطلاح نرون غیر فعال می گویند. معیار گزینش و حذف مجموعه ای از نرون ها در یک لایه، درصد مجموع مربعات خطا ($d_j^۲$) میان مقادیر خروجی واقعی (y_j) و خروجی نرون jـام (y_{ij}^*) است.

$$d_j^۲ = \sum_{i=nt+1}^{n} \left(y_i - y_{ij}^*\right)^۲ \qquad (۲.۴)$$

که در آن $j \in \{۱,۲,۳, \dots, \binom{m}{۲}\}$ و m تعداد نرون‌های گزینش شده در لایه قبلی است.

نگاشتی که بین متغییرهای ورودی و خروجی توسط این نوع از شبکه‌های عصبی برقرار می‌شود، به‌صورت تابع غیرخطی ولترا و به شکل رابطه شماره (۱.۳) است. ساختاری که برای نرون‌ها در نظر گرفته شده، به صورت فرم خلاصه شده دو متغییره درجه دوم زیر است.

$$y_i = f\left(x_{ip}, x_{iq}\right) = a_. + a_۱x_{ip} + a_۲x_{iq} + a_۳x_{ip}x_{iq} + a_۴x_{ip}^۲ + a_۵x_{iq}^۲ \qquad (۳.۴)$$

تابع f دارای شش ضریب مجهول است که به ازای تمام نمونه های دو متغییر وابسته به سیستم $\{\left(x_{ip}, x_{iq}\right), i = ۱,۲, \dots, n\}$، خروجی مطلوب $\{(y_i), i = ۱,۲, \dots, n\}$ را برآورد می کند. تابع f را براساس قاعده خطای کمترین مربعات پایه ریزی می‌کنیم.

$$\min \sum_{k=۱}^{n} \left[\left(f\left(x_{ki}, x_{kj}\right) - y_i\right)^۲\right] \qquad (۴.۴)$$

براین اساس دستگاه معادله ای را حل می‌کنیم که دارای شش مجهول و n معادله است.

$$\begin{cases} a_. + a_۱x_{۱p} + a_۲x_{۱q} + a_۳x_{۱p}x_{۱q} + a_۴x_{۱p}^۲ + a_۵x_{۱q}^۲ = y_۱ \\ a_. + a_۱x_{۲p} + a_۲x_{۲q} + a_۳x_{۲p}x_{۲q} + a_۴x_{۲p}^۲ + a_۵x_{۲q}^۲ = y_۲ \\ \vdots \\ a_. + a_۱x_{np} + a_۲x_{nq} + a_۳x_{np}x_{nq} + a_۴x_{np}^۲ + a_۵x_{nq}^۲ = y_n \end{cases} \qquad (۵.۴)$$

دستگاه معادله فوق را می توان به شکل ماتریسی $Aa=Y$ نمایش داد که در آن

$$A = \begin{bmatrix} ۱ & x_{۱p} & x_{۱q} & x_{۱p}x_{۱q} & x_{۱p}^۲ & x_{۱q}^۲ \\ ۱ & x_{۲p} & x_{۲q} & x_{۲p}x_{۲q} & x_{۲p}^۲ & x_{۲q}^۲ \\ \vdots & \vdots & \vdots & \vdots\vdots & \vdots & \vdots \\ ۱ & x_{\mathrm{n}p} & x_{nq} & x_{\mathrm{n}p}x_{\mathrm{n}q} & x_{np}^۲ & x_{nq}^۲ \end{bmatrix}$$

$a = [a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5]^T$ و $Y = [y_1 \quad y_2 \quad y_3 \quad \cdots \quad y_n]$.

برای تعیین دقت مدل تعدادی از معیارهای آماری استفاده شده است. این معیار ها عبارت اند از: جذر میانگین مربعات خطا (*RMSE*)، و قدرمطلق انحراف معیار (*MAD*).

$$\mathrm{RMSE} = \left[\frac{\sum_{i=0}^{M}\left(Y_{i(model)} - Y_{i(actual)}\right)^2}{M}\right]^{1/2} ,$$

$$MAD = \frac{\sum_{i=0}^{M}\left|Y_{i(model)} - Y_{i(actual)}\right|}{M} .$$

۵. پیاده سازی الگوریتم در نرم افزار متلب

برنامه از نقطه شروع استاندارد (تابع main) آغاز و هدایت می گردد. پس از بارگذاری پایگاه داده (در مثال پیاده سازی شده فایل Indice.mat) مجموعه های یادگیری و ناظر مشخص می شوند. در این مثال ۸۵٪ داده ها به مجموعه یادگیری (ptrain=0.85) و باقی به مجموعه آزمون داده شده است. پس از آن شبکه عصبی به کمک تابع GMDH و با استفاده از مجموعه یادگیری ساخته می‌شود، و در آخر مقداریابی ها صورت می گیرند و نتایج به صورت نمودارهایی نمایش داده می شوند.

بارگذاری پایگاه داده و مقداردهی اولیه (در پایگاه داده، آرایه 'y در جدول Targets و آرایه 'X در جدول Inputs ذخیره شده اند. سطرها به ترتیب حاوی مقادیر متغییرهای EPS,PEPS,DPS,E/P,P/E اند.)

```
data = load('Indice.mat');
Inputs = data.Inputs;
Targets = data.Targets;
nData = size(Inputs,2);
Perm = randperm(nData);
```

مشخص کردن مجموعه یادگیری

```
pTrain = 0.85;
nTrainData = round(pTrain*nData);
TrainInd = Perm(1:nTrainData);
TrainInputs = Inputs(:,TrainInd);
```

```
TrainTargets = Targets(:,TrainInd);
```

مشخص کردن مجموعه آزمون

```
pTest = 1 - pTrain;
nTestData = nData - nTrainData;
TestInd = Perm(nTrainData+1:end);
TestInputs = Inputs(:,TestInd);
TestTargets = Targets(:,TestInd);
```

ساخت شبکه عصبی با استفاده از مجموعه یادگیری

```
params.MaxLayerNeurons = 15;   % Maximum Number of Neurons in a Layer
params.MaxLayers = 4;          % Maximum Number of Layers
params.alpha = 0.6;            % Selection Pressure (in Layers)
params.pTrain = 0.85;          % Train Ratio (vs. Validation Ratio)
gmdh = GMDH(params, TrainInputs, TrainTargets);
```

ارزشیابی شبکه GMDH

```
Outputs = ApplyGMDH(gmdh, Inputs);
TrainOutputs = Outputs(:,TrainInd);
TestOutputs = Outputs(:,TestInd);
```

نمایش نتایج

```
disp('Type "gmdh.Layers" to see the layers'' info.');
disp(' ');

figure;
PlotResults(TrainTargets, TrainOutputs, 'Train Data');
```

```
figure;
PlotResults(TestTargets, TestOutputs, 'Test Data');

figure;
PlotResults(Targets, Outputs, 'All Data');

if ~isempty(which('plotregression'))
   figure;
   plotregression(TrainTargets, TrainOutputs, 'Train Data', ...
                  TestTargets, TestOutputs, 'TestData', ...
                  Targets, Outputs, 'All Data');
end
```

برای مشاهده کد کامل برنامه به انتهای مقاله مراجعه فرمایید. علاوه براین تمام کدها در فایل های ضمیمه موجود اند.

۶. نتیجه گیری

در این مقاله ما رابطه بین سود سهام و قیمت آن را بررسی کردیم. مدل کردن نرم افزاری پیش بینی قیمت سهام برای تمامی سرمایه گذاران و تحلیل‌گران برای کاهش ریسک سرمایه گذاری و افزایش منافع سهام داران مفید است.

ضمیمه ۱. کد کامل برنامه

محتوای فایل main.m

```
%
% Copyright (c) 2015, Yarpiz (www.yarpiz.com)
% All rights reserved. Please read the "license.txt" for license
terms.
%
% Project Code: YPML113
% Project Title: Implementation of Group Method of Data Handling
in MATLAB
% Publisher: Yarpiz (www.yarpiz.com)
%
% Developer: S. Mostapha Kalami Heris (Member of Yarpiz Team)
%
% Contact Info: sm.kalami@gmail.com, info@yarpiz.com
%

clc;
clear;
close all;

%% Load Data

data = load('Indice.mat');
Inputs = data.Inputs;
Targets = data.Targets;

nData = size(Inputs,2);
Perm = randperm(nData);

% Train Data
pTrain = 0.85;
nTrainData = round(pTrain*nData);
TrainInd = Perm(1:nTrainData);
TrainInputs = Inputs(:,TrainInd);
TrainTargets = Targets(:,TrainInd);

% Test Data
pTest = 1 - pTrain;


nTestData = nData - nTrainData;
```

```
TestInd = Perm(nTrainData+1:end);
TestInputs = Inputs(:,TestInd);
TestTargets = Targets(:,TestInd);

%% Create and Train GMDH Network

params.MaxLayerNeurons = 15;    % Maximum Number of Neurons in a
Layer
params.MaxLayers = 4;           % Maximum Number of Layers
params.alpha = 0.6;             % Selection Pressure (in Layers)
params.pTrain = 0.85;          % Train Ratio (vs. Validation Ratio)
gmdh = GMDH(params, TrainInputs, TrainTargets);

%% Evaluate GMDH Network

Outputs = ApplyGMDH(gmdh, Inputs);
TrainOutputs = Outputs(:,TrainInd);
TestOutputs = Outputs(:,TestInd);

%% Show Results

disp('Type "gmdh.Layers" to see the layers'' info.');
disp(' ');

figure;
PlotResults(TrainTargets, TrainOutputs, 'Train Data');

figure;
PlotResults(TestTargets, TestOutputs, 'Test Data');

figure;
PlotResults(Targets, Outputs, 'All Data');

if ~isempty(which('plotregression'))
    figure;
    plotregression(TrainTargets, TrainOutputs, 'Train Data', ...
                   TestTargets, TestOutputs, 'TestData', ...
                   Targets, Outputs, 'All Data');
end
```

محتوای فایل GMDH.m

```
function gmdh = GMDH(params, X, Y)

    disp('Training GMDH:');

    MaxLayerNeurons = params.MaxLayerNeurons;
    MaxLayers = params.MaxLayers;
    alpha = params.alpha;

    nData = size(X,2);

    % Shuffle Data
    Permutation = randperm(nData);
    X = X(:,Permutation);
    Y = Y(:,Permutation);

    % Divide Data
    pTrainData = params.pTrain;
    nTrainData = round(pTrainData*nData);
    X1 = X(:,1:nTrainData);
    Y1 = Y(:,1:nTrainData);
    pTestData = 1-pTrainData;
    nTestData = nData - nTrainData;
    X2 = X(:,nTrainData+1:end);
    Y2 = Y(:,nTrainData+1:end);

    Layers = cell(MaxLayers, 1);

    Z1 = X1;
    Z2 = X2;

    for l = 1:MaxLayers

        L = GetPolynomialLayer(Z1, Y1, Z2, Y2);

        if l>1
            if L(1).RMSE2 > Layers{l-1}(1).RMSE2
                break;
            end
        end

        ec = alpha*L(1).RMSE2 + (1-alpha)*L(end).RMSE2;
        ec = max(ec, L(1).RMSE2);
        L = L([L.RMSE2] <= ec);
```

```
        if numel(L) > MaxLayerNeurons
            L = L(1:MaxLayerNeurons);
        end

        if l==MaxLayers && numel(L)>1
            L = L(1);
        end

        Layers{l} = L;

        Z1 = reshape([L.Y1hat],nTrainData,[])';
        Z2 = reshape([L.Y2hat],nTestData,[])';

        disp(['Layer ' num2str(l) ': Neurons = ' num2str(numel(L))
', Min Error = ' num2str(L(1).RMSE2)]);

        if numel(L)==1
            break;
        end

    end

    Layers = Layers(1:l);

    gmdh.Layers = Layers;

    disp(' ');

end
```

محتوای فایل ApplyGMDH.m

```
function Yhat = ApplyGMDH(gmdh, X)

    nLayer = numel(gmdh.Layers);

    Z = X;
    for l=1:nLayer
        Z = GetLayerOutput(gmdh.Layers{l}, Z);
    end
    Yhat = Z;

end

function Z = GetLayerOutput(L, X)

    m = size(X,2);
    N = numel(L);
    Z = zeros(N,m);

    for k=1:N
        vars = L(k).vars;
        x = X(vars,:);
        Z(k,:) = L(k).f(x);
    end

end
```

محتوای فایل PlotResualt.m

```
function PlotResults(Targets, Outputs, Title)

    Errors = Targets - Outputs;
    MSE = mean(Errors.^2);
    RMSE = sqrt(MSE);
    ErrorMean = mean(Errors);
    ErrorStd = std(Errors);

    subplot(2,2,[1 2]);
    plot(Targets,'r');
    hold on;
    plot(Outputs);
    legend('Targets','Outputs');
    ylabel('Targets and Outputs');
    grid on;
    title(Title);

    subplot(2,2,3);
    plot(Errors);
    title(['MSE = ' num2str(MSE) ', RMSE = ' num2str(RMSE)]);
    ylabel('Errors');
    grid on;

    subplot(2,2,4);
    histfit(Errors, 50);
    title(['Error Mean = ' num2str(ErrorMean) ', Error StD = '
num2str(ErrorStd)]);

end
```

محتوای فایل FitPolynomaial.m

```
function p = FitPolynomial(x1, Y1, x2, Y2, vars)
    X1 = CreateRegressorsMatrix(x1);
    c = Y1*pinv(X1);

    Y1hat = c*X1;
    e1 = Y1- Y1hat;
    MSE1 = mean(e1.^2);
    RMSE1 = sqrt(MSE1);
    MAD1=mad(abs(e1));%%

    f = @(x) c*CreateRegressorsMatrix(x);

    Y2hat = f(x2);
    e2 = Y2- Y2hat;
    MSE2 = mean(e2.^2);
    RMSE2 = sqrt(MSE2);
    MAD2=mad(abs(e2));%%

    p.vars = vars;
    p.c = c;
    p.f = f;
    p.Y1hat = Y1hat;
    p.MSE1 = MSE1;
    p.RMSE1 = RMSE1;
    p.Y2hat = Y2hat;
    p.MSE2 = MSE2;
    p.RMSE2 = RMSE2;
   %-------------------------
    p.MAD1=MAD1;
    p.MAD1=MAD1;

end

function X = CreateRegressorsMatrix(x)

    X = [ones(1,size(x,2))
         x(1,:)
         x(2,:)
         x(1,:).^2
         x(2,:).^2
         x(1,:).*x(2,:)];

end
```

محتوای فایل GetPolynomialLayer.m

```
function L = GetPolynomialLayer(X1, Y1, X2, Y2)

    n = size(X1,1);

    N = n*(n-1)/2;

    template                                                    =
FitPolynomial(rand(2,3),rand(1,3),rand(2,3),rand(1,3),[]);

    L = repmat(template, N, 1);

    k = 0;
    for i=1:n-1
        for j=i+1:n
            k = k+1;
            L(k) = FitPolynomial(X1([i j],:), Y1, X2([i j],:), Y2,
[i j]);
        end
    end

    [~, SortOrder] = sort([L.RMSE2]);

    L = L(SortOrder);

end
```

ضمیمه ۲. اطلاعات مدل

۱.۲ اطلاعات آماری و جداول

مدل ما از سه لایه تشکیل شده است.اطلاعات هر کدام از این لایه ها به قرار زیر است:

لایه اول

$$y_۱ = ۵۹۳.۱۹۳۳ + ۱۰.۶۱۷۲\, x_۱ - ۱۰.۹۱۱۸\, x_۲ - ۰.۲۶۱۸\, x_۱x_۲ - ۰.۲۴۱۳\, x_۱^۲ + ۰.۵۰۳۴\, x_۲^۲$$

$$y_۲ = ۶۲۳.۴۹۸۵ - ۰.۳۰۹۸\, x_۲ - ۱.۰۶۰۷\, x_۴ + ۰.۰۰۰۲\, x_۲x_۴ + ۰.۰۰۷۹\, x_۲^۲ - ۰.۰۰۰۱\, x_۴^۲$$

$$y_۳ = ۶۲۲.۶۴۴۴ - ۰.۳۰۴۵\, x_۱ - ۱.۰۵۷۲\, x_۴ + ۰.۰۰۰۲\, x_۱x_۴ + ۰.۰۰۷۸\, x_۱^۲ - ۰.۰۰۰۱\, x_۴^۲$$

$$y_۴ = ۵۹۲.۴۵۹۷ - ۰.۲۴۱۸\, x_۳ - ۱.۰۷۲۷\, x_۴ + ۰.۰۰۰۲\, x_۳x_۴ + ۰.۰۰۷۹\, x_۳^۲ - ۰.۰۰۰۱\, x_۴^۲$$

نتیجه اعمال بر مجموعه داده های یادگیری

تابع	Mad	RMSE
$y_۱$	۲۲,۹۶۲۲	۲۶,۹۱۱۱
$y_۲$	۲۲,۶۴۵۳	۲۹,۲۴۲۱
$y_۳$	۲۲,۴۷۸۶	۲۹,۴۵۸۳
$y_۴$	۲۲,۱۹۷۴	۲۹,۷۸۷۲

نتیجه اعمال بر مجموعه داده های ناظر

تابع	Mad	RMSE
$y_۱$	۲۲,۰۵۹۸	۳۰,۷۰۶۵
$y_۲$	۲۰,۸۳۳۴	۳۲,۰۰۳۷
$y_۳$	۲۰,۶۲۷۸	۳۲,۰۸۲۲
$y_۴$	۲۰,۵۰۳۶	۳۲,۳۴۳۷

لایه دوم

$$y_۱ = [-۵.۲۲۴۰ + ۰.۰۰۹۴\, x_۱ + ۰.۰۱۱۲\, x_۲ + ۰.۰۰۰۰\, x_۱x_۲ + ۰.۰۰۰۰\, x_۱^۲ - ۰.۰۰۰۱\, x_۲^۲] * ۱۰^۳$$

$$y_۱ = [-۵.۲۸۶۶ + ۰.۰۰۹۸\, x_۱ + ۰.۰۱۰۹\, x_۳ + ۰.۰۰۰۰\, x_۱x_۳ + ۰.۰۰۰۰\, x_۱^۲ - ۰.۰۰۰۱\, x_۳^۲] * ۱۰^۳$$

$$y_۱ = [-۵.۰۵۷۳ + ۰.۰۱۰۲\, x_۱ + ۰.۰۰۹۷\, x_۴ + ۰.۰۰۰۰\, x_۱x_۴ + ۰.۰۰۰۰\, x_۱^۲ - ۰.۰۰۰۱\, x_۴^۲] * ۱۰^۳$$

نتیجه اعمال بر مجموعه داده های یادگیری

RMSE	Mad	تابع
۱۸,۵۲۸۵	۳۰,۴۰۰۸	$y_۱$
۱۸,۵۳۳۸	۳۰,۳۸۷۸	$y_۲$
۱۹,۹۷۰۵	۳۰,۰۶۴۸	$y_۳$

نتیجه اعمال بر مجموعه داده های ناظر

RMSE	Mad	تابع
۲۱,۲۷۳۱	۲۶,۲۸۳۴	$y_۱$
۲۱,۲۸۵۱	۲۶,۳۸۶۳	$y_۲$
۲۱,۳۹۴۹	۲۶,۶۷۸۷	$y_۳$

لایه سوم

$$y_۱ = [-۱.۱۵۴۷ + ۰.۱۴۶۷\, x_۱ - ۰.۱۴۱۳\, x_۲ - ۰.۰۱۰۲\, x_۱x_۲ - ۰.۰۰۹۹\, x_۱^۲ + ۰.۰۲۰۱\, x_۲^۲] * ۱۰^۳$$

نتیجه اعمال بر مجموعه داده های یادگیری

RMSE	Mad	تابع
۱۷,۰۱۱۵	۳۱,۲۵۲۳	$y_۱$

نتیجه اعمال بر مجموعه داده های ناظر

RMSE	Mad	تابع
۱۹,۶۰۲۷	۲۷,۰۹۰۹	$y_۱$

و در نهایت چندجمله ای ایواخننکو عبارت است از

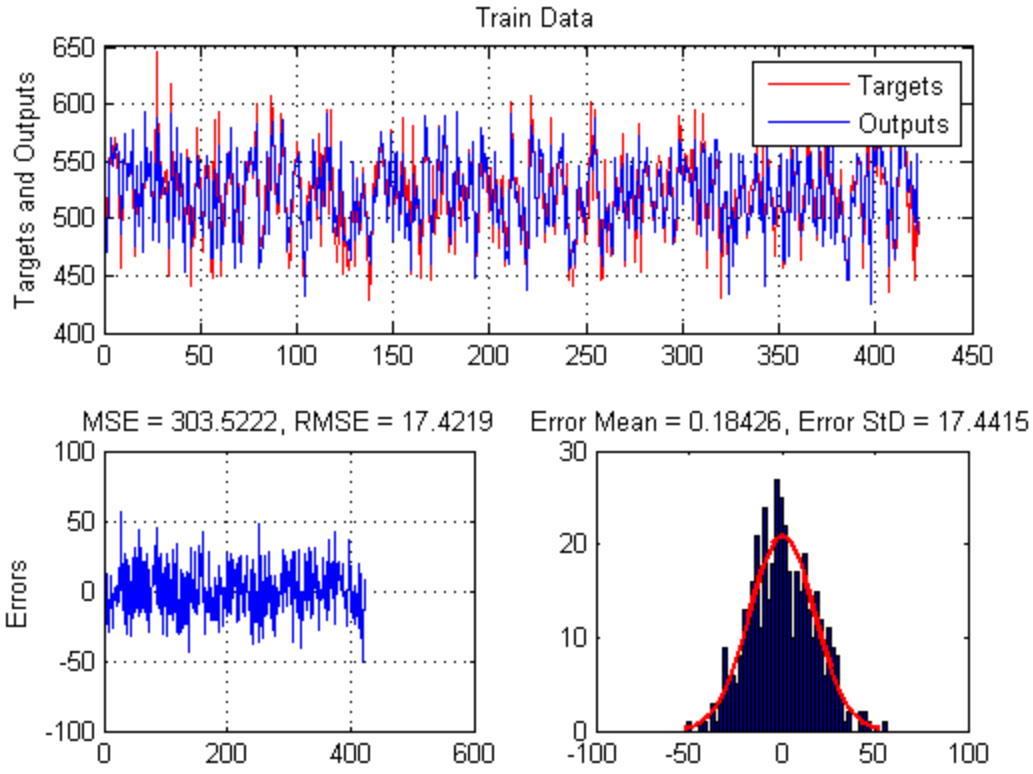
$$y \quad = ۰.۰۰۱۹ + ۰.۹۲۳۱\, x_۱ - ۰.۹۲۳۱\, x_۲ + ۶۰۷.۶۴۶۰\, x_۱x_۲ + ۶۰۷.۵۵۱۱\, x_۱^۲ + ۶۰۷.۷۵۵۹\, x_۲^۲$$

که متغییرها براساس لایه آخر شبکه و ضرایب از حل دستگاه (۳.۵) توسط قطعه کد زیر بدست آمده اند.

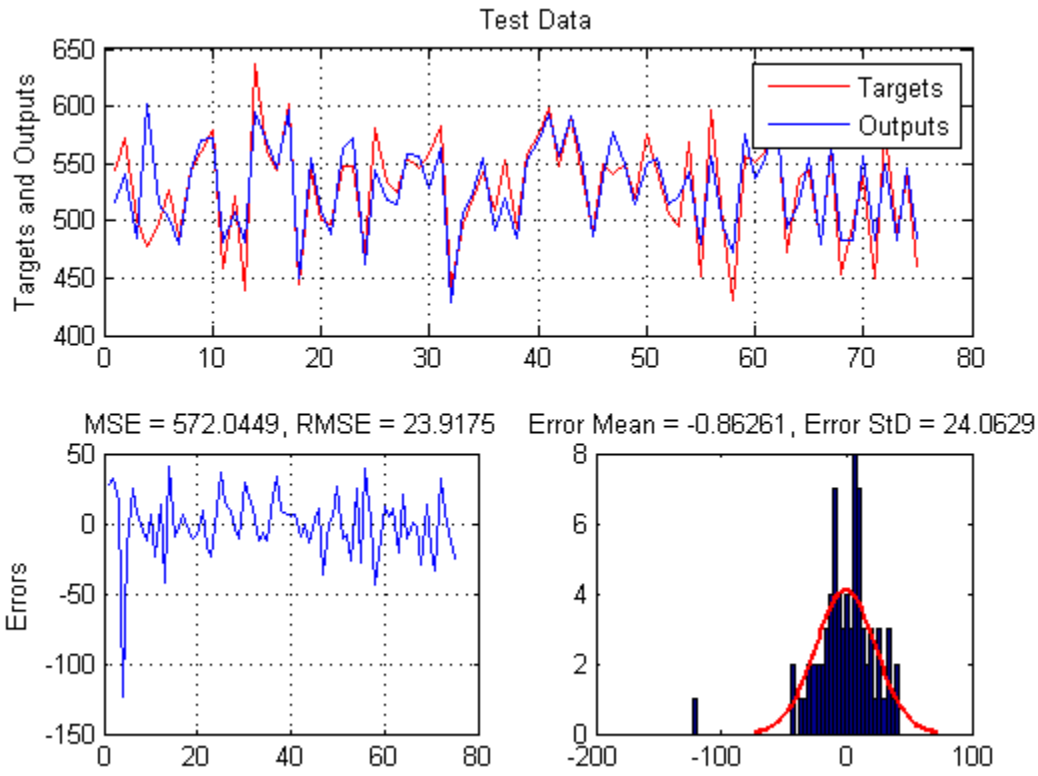
```
A=[ones(1,498)',Inputs(1,:)',Inputs(2,:)',Inputs(1,:)'.*Inputs(2,:)',Inputs(1
,:)'.^2,Inputs(2,:)'.^2]
disp('coefficients :')
a=Outputs'\A
```

RMSE	Mad	تابع
۲۰,۱۴۴۴	۹,۵۳۹۹	y

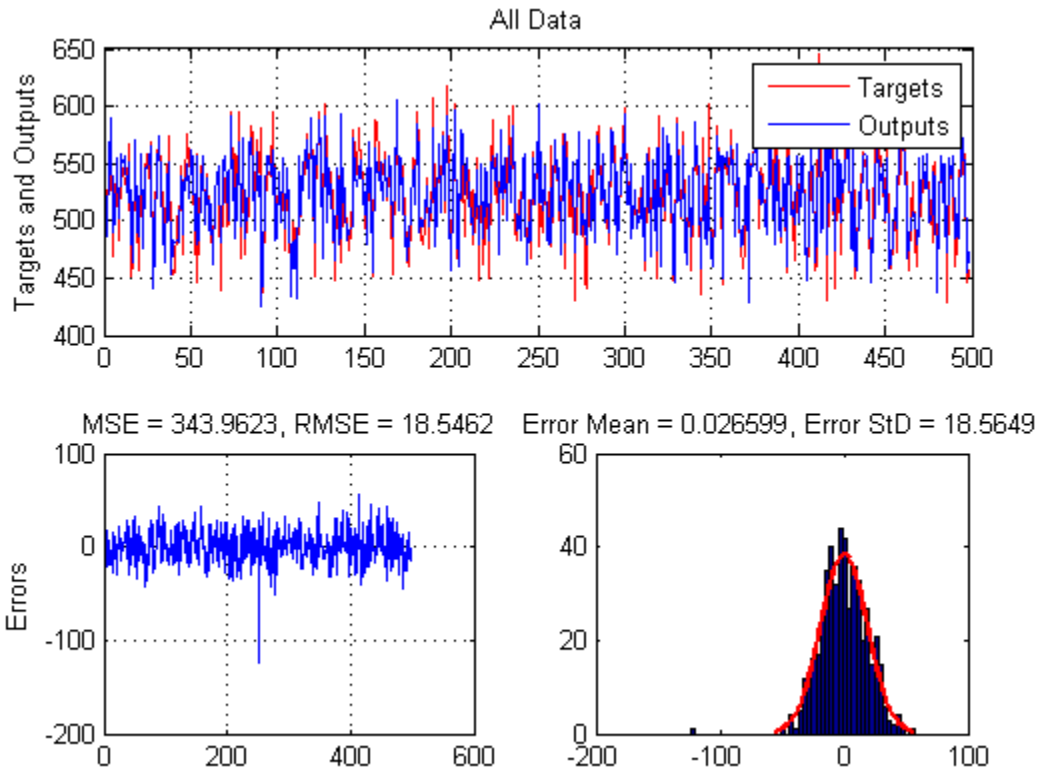
۲.۲ نمودارهای مدل



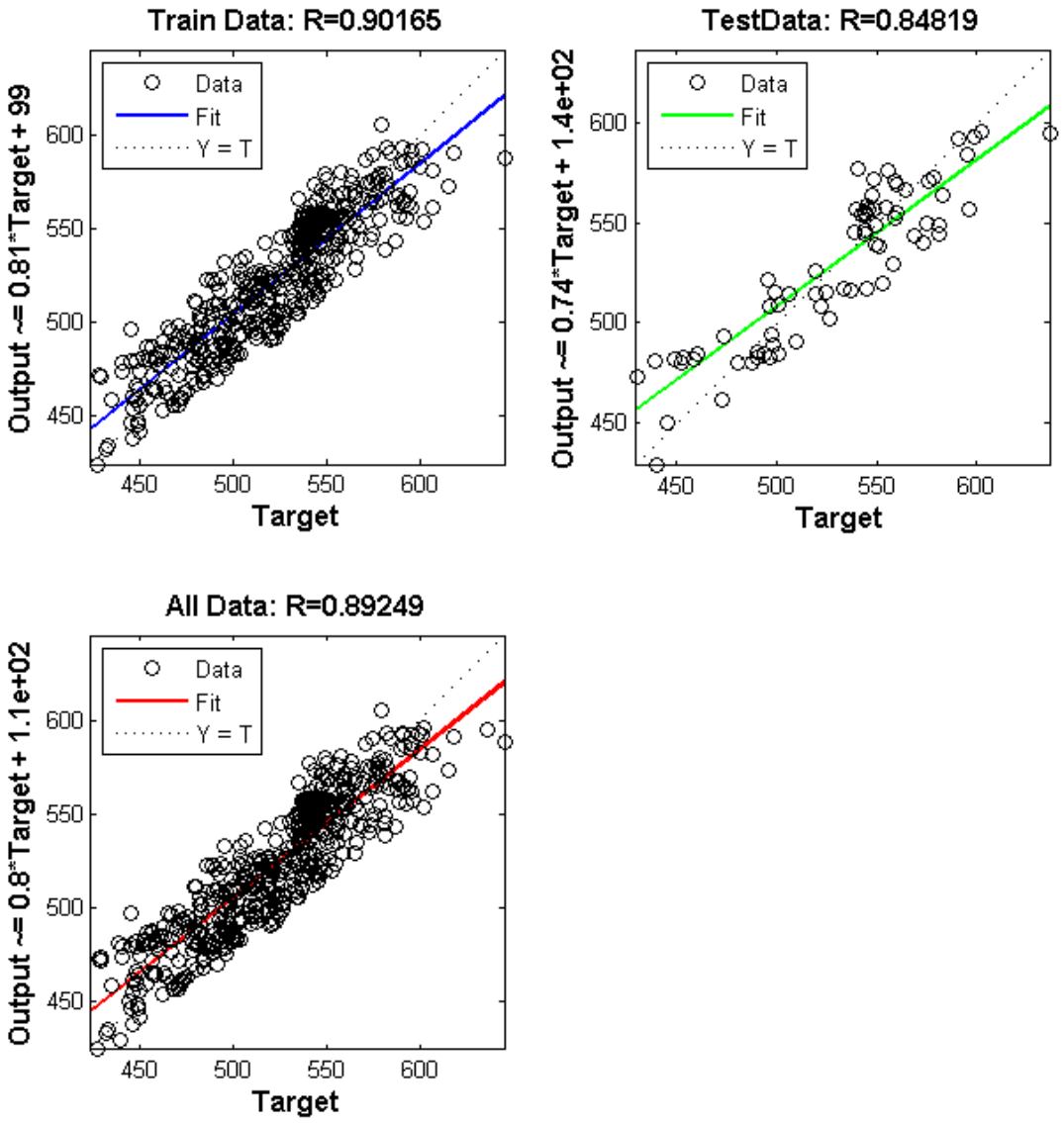
نمودار۱.۲.۲ (نمودار بالا) مقادیر واقعی (رنگ قرمز) و مدلسازی شده (رنگ آبی) قیمت سهام به ازای مجموعه اندیس، (نمودار پایین-چپ) خطای مدل به ازای مجموعه اندیس، (نمودار پایین-راست) توزیع خطا؛ برای مجموعه یادگیری.



نمودار ۲.۲.۲ (نمودار بالا) مقادیر واقعی (رنگ قرمز) و مدلسازی شده (رنگ آبی) قیمت سهام به ازای مجموعه اندیس، (نمودار پایین-چپ) خطای مدل به ازای مجموعه اندیس، (نمودار پایین-راست) توزیع خطا؛ برای مجموعه ناظر.



نمودار۳.۲.۲ (نمودار بالا) مقادیر واقعی (رنگ قرمز) و مدلسازی شده (رنگ آبی) قیمت سهام به ازای مجموعه اندیس، (نمودار پایین-چپ) خطای مدل به ازای مجموعه اندیس، (نمودار پایین-راست) توزیع خطا؛ برای کل داده ها.



نمودار۴.۲.۲ نمودار رگرسیون مقادیر خروجی مدل به ازای مجموعه های یاگیری (بالا-چپ)، ناظر (بالا-راست) و کل داده ها (پایین).

مراجع :

۱. The GMDH Algorithm of Ivakjnenko, Stanle J.Farlow, The American Statistician, Vol.35, No.4(Nov.,1981),210-215

۲. Applying GMDH-Type Nural Network and Genetic Algorithm for Stock Price Prediction Of Iranian Cement Sector, Saeed Fallahi, Meysam Shaverdi, Vahab Bashiri, Applications and Applied Mathematics, Vol.6, Issue 2(December 2011), pp.572-591, ISSN: 1932-9466

۳. الگوسازی و پیش بینی EPS شرکت های پذیرفته شده در بورس اوراق بهادار تهران با رویکرد شبکه عصبی GMDH، علی اصغر انواری رستمی،عادل آذر، محمد نوروزی. مجله بررسی‌های حسابداری و حسابرسی، دانشکده مدیریت دانشگاه تهران، دوره ۲۰، شماره ۱، بهار ۱۳۹۲، صص ۱۸-۱.