دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده صنایع

پایان‌نامه کارشناسی

معاملات الگوریتمی مالی ترکیبی با استفاده از الگوریتم‌های جنگل تصادفی، ماشین‌های بردار پشتیبان، شبکه‌های عصبی و نزدیک‌ترین همسایه

نگارش
مسعود هادی نجف‌آبادی

استاد راهنما
دکتر مسعود ماهوتچی

خرداد ماه ۱۴۰۰

صفحه فرم ارزیابی و تصویب پایان‌نامه- فرم تأیید اعضاء کمیته دفاع

به نام خدا

تعهدنامه اصالت اثر

تاریخ:

اینجانب مسعود هادی نجف‌آبادی متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آن‌ها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان‌نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مآخذ بلامانع است.

مسعود هادی نجف‌آبادی

امضا

چکیده

روند تغییرات شاخص همواره به عنوان یکی از معیارهای سرمایه‌گذاری مدنظر قرار می‌گیرد. این پژوهش یک سیستم پشتیبان تصمیم برای معاملات الگوریتمی در بازارهای مالی ارائه می‌دهد که از رویکردی جدید در اتخاذ تصمیمات خودکار استفاده می‌کند. این رویکرد با ترکیب الگوریتم‌های جنگل تصادفی، ماشین بردار پشتیبان، شبکه عصبی و نزدیک‌ترین همسایه به پیش‌بینی سیگنال‌های خرید، نگهداری یا فروش می‌پردازد. داده‌های ورودی مدل، قیمت و حجم معاملات پانصد سهم شاخص S&P500 و قیمت سهام صندوق SPY به عنوان معیار شاخص S&P500 از ابتدای ماه ژانویه سال ۲۰۱۵ تا پایان ماه ژوئن سال ۲۰۲۱ بوده است. همچنین به جهت کاهش نویز ورودی به سیستم اندیکاتورهای میانگین متحرک وزن‌دار پنج روزه و شاخص قدرت نسبی نیز مورد استفاده قرار گرفته است. در مرحله نخست از الگوریتم جنگل تصادفی برای تعیین صد سهم دارای بیشترین اهمیت استفاده شده است، سپس با توجه به نتیجه‌ی مرحله‌ی اول هر یک از الگوریتم‌ها بر اساس داده‌های مربوط به ابتدای سال ۲۰۱۵ تا یک روز قبل از روز مورد پیش‌بینی، به تولید سیگنال خرید، نگهداری یا فروش می‌پردازند و سپس نتایج پیش‌بینی هر یک از الگوریتم‌ها با معیار دقت مدل سنجیده می‌شود. در مرحله‌ی پایانی با استفاده از تکنیک بگینگ[1] که یک متاالگوریتم برای ترکیب نتایج چند پیش‌بینی‌کننده بر اساس رای اکثریت است، استفاده می‌شود. به این نحو که نتایج پیش‌بینی الگوریتم‌های مرحله‌ی قبل در هر روز به عنوان متغیر ورودی در اختیار الگوریتم بگینگ قرار گرفته و این الگوریتم سیگنال نهایی را بر اساس رای اکثریت برای روز آتی تولید می‌کند.

واژه‌های کلیدی:

معاملات الگوریتمی، بازار سهام، پیشبینی مالی، الگوریتم ترکیبی.

[1] Bagging

فهرست مطالب

صفحه

فهرست اشکال

صفحه

فهرست جداول

صفحه

فهرست علائم و اختصارات

علائم لاتین

RF	جنگل تصادفی
MLP	پیش‌بینی‌کننده‌ی چندلایه
SVM	ماشین بردار پشتیبان
KNN	K نزدیک‌ترین همسایه
WMA5	میانگین متحرک وزن‌دار پنج روزه
RSI	شاخص قدرت نسبی
ACC	دقت
TP	مثبت صحیح
TN	منفی صحیح
FP	مثبت غلط
FN	منفی غلط

فصل اول

مقدمه

یکی از مهم‌ترین دستاوردهای علم قابلیت پیش‌بینی بخشیدن به متغیرها و پدیده‌ها می باشد. پژوهشگران علوم مالی نیز با استفاده از ابزارهای مختلف به دنبال طراحی مدل‌هایی هستند که به وسیله آن متغیرها و حوادث مدنظرشان در بازارهای مالی را پیش‌بینی کنند. پیش‌بینی سود (جوگ و مک کنومی[1]، ۲۰۰۳) پیش‌بینی ورشکستگی (وندا[2]، ۲۰۰۴) و پیش‌بینی جریانات نقدی (براچت[3]، ۲۰۰۷) از این دست پژوهش‌هاست [۱].

به جرات می‌توان گفت که پیش‌بینی قیمت سهام از مهم‌ترین اهداف در علوم مالی و سرمایه‌گذاری است (هاموز[4]، ۲۰۱۵). امروزه پیش‌بینی قیمت سهام نه تنها بسیار چالش برانگیز، بلکه بسیار مورد علاقه سرمایه‌گذاران می‌باشد. پیش‌بینی قیمت سهام از دو جنبه حائز اهمیت است. اول آنکه پیش‌بینی دقیق قیمت سهام برای بهینه‌سازی پرتفوهای سرمایه‌گذاری و اجرای راهبردهای سرمایه‌گذاری بسیار تأثیرگذار است (چانگ و چین[5]، ۲۰۰۵). دوم آنکه با استفاده از روش‌های پیش‌بینی مطلوب می‌توان بازدهی را در سطح مشخصی از ریسک افزایش داد و بالعکس (کیومر[6]، ۲۰۰۶) [۱].

در این میان شاخص به عنوان یک معیار آماری، قابلیت مقایسه وضعیت کنونی را نسبت به گذشته فراهم آورده و بررسی و تحلیل آن اطلاعات مفیدی را به کارشناسان و افراد ذی‌ربط در آن حوزه ارائه می‌دهد. شاخص‌های قیمت سهام در تمام بازارهای مالی دنیا، به مثابه یکی از مهم‌ترین معیارهای سنجش عملکرد بازار سهام، اهمیت زیادی دارند و شاید مهم‌ترین دلیل توجه روزافزون به آن‌ها، این نکته باشد که شاخص‌ها از تجمیع حرکت‌های قیمتی سهام تمام شرکت‌ها یا طبقه‌ی خاصی از شرکت‌های موجود در بازار به دست می‌آیند (ژیان ژو وانگ، ژنگ، گو و ژو ژی وانگ[7]، ۲۰۱۱) [۲].

---

[1] Jog, V. & McConomy, B. J.

[2] Wallace Wanda, A.

[3] Brochet, F.

[4] Al-Hmouz, R.

[5] Cheung, Y. W., Chinn, M. D.

[6] Kumar, M.

[7] Wang, J. Z., Wang, J. J., Zhang, Z. G., & Guo, S. P.

داده‌های تاریخی نشـــان می‌دهد ویژگی‌های پیچیده شـــاخص کل قیمت، مانند غیرخطی بودن، عدم قطعیت، نوسان و پویایی، پیش‌بینی آن را دشوار می‌کند و نتایج پیش‌بینی را با عدم قطعیت زیادی مواجه می‌ســازد که خود تاثیر شــایان توجهی در بازده ســرمایه‌گذاران، صــندوق‌های ســرمایه‌گذاری، نهادهای سرمایه‌گذاری و سایر فعالان این حوزه به همراه دارد (خسروی‌نژاد و شعبانی، ۱۳۹۳) [۲].

سه نوع تحلیل شناخته‌شده در بازارهای مالی مورد استفاده قرار می‌گیرد. تحلیل بنیادی بر پایه‌ی عملکرد شرکت‌ها و رشد سودآوری آن‌ها بنا شده است و تحلیل مالی-رفتاری حوزه‌ای از دانش مالی است که از نظریه‌های مبتنی بر روانشناسی برای توضیح رفتار بازارهای مالی بهره می‌گیرد. تحلیل تکنیکال سومین روش اســت که بر پایه ســابقه معاملات یک دارایی مالی از طریق نمودار قیمت و فرمول‌های ریاضــی که اندیکاتورهای تکنیکال نامیده می‌شــوند، بنا شــده اســت. در ســال‌های اخیر از هوش مصــنوعی نیز برای پیش‌بینی بازار استفاده شده است که ترکیب آن با تحلیل تکنیکال می‌تواند منجر به ایجاد سیستم‌های خودکار معاملاتی و الگوریتمی شـــود. از اولین کارها که با ترکیب تحلیل تکنیکال و الگوریتم‌های بهینه‌سازی و هوش مصنوعی سعی در ایجاد یک سیستم معاملاتی خودکار داشته است، می‌توان به اسکابار و کلوته[1] (۲۰۰۲) اشاره کرد [۳].

دراین پژوهش تلاش شده است تا با بهره‌گیری از اندیکاتورهای تحلیل تکنیکال در مرحله آغازین قابلیت پیش‌بینی‌پذیری را در اثر کاهش نویز ورودی به مدل افزایش داده و ســـپس با ارائه‌ی ترکیب جدیدی از الگوریتم‌های داده‌کاوی، در یک فرآیند ســـه مرحله‌ای به تولید ســـیگنال‌های معاملاتی برای روز آتی پرداخته شده است.

در مدل پیشــنهاد شــده در مرحله اول ســهم‌های دارای بیشــترین تاثیر انتخاب می‌شــوند، ســپس از چهار الگوریتم مختلف برای پیش‌بینی اســـتفاده می‌شـــود و در نهایت با اســـتفاده از یک الگوریتم چند لایه سیگنال‌های تولید شده در مرحله قبل ترکیب شده و سیگنال نهایی ارائه می‌شود.

بررسی‌های صورت گرفته نشان‌دهنده‌ی صحت قابل قبول سیگنال‌های تولید شده و بازده مناسب‌تر مدل از استراتژی خرید و نگهداری است.

[1] Skabar and Cloete

فصل دوم

مبانی نظری و مروری بر ادبیات پژوهش

در این قسمت، ما الگوریتم‌های دسته‌بندی با ناظر را که برای پیش‌بینی جهت حرکت قیمت استفاده شده است، مورد بحث و بررسی قرار می‌دهیم. دسته‌بندی، شکلی از تحلیل داده‌ها تلقی می‌شود که در آن مدل‌هایی جهت توصیف کلاس‌های مهمی از داده‌ها استخراج می‌شود. دسته‌بندی داده‌ها فرآیندی است که کار خود را در دو گام انجام می‌دهد: گام یادگیری (که در آن مدل ساخته می‌شود) و گام دسته‌بندی (که در آن جهت پیشگویی برچسب‌های کلاس از مدل ساخته شده در گام اول استفاده می‌شود). به دلیل آن که برچسب کلاس هر یک از تاپل‌های آموزشی مشخص شده‌اند، این گام همچنین به عنوان یادگیری با ناظر شناخته می‌شوند. در مقابل یادگیری بی‌ناظر (یا خوشه‌بندی) است، که در آن برچسب کلاس تاپل‌های آموزشی شناخته شده نیست و ممکن است تعداد یا مجموعه دسته‌هایی که در نهایت به دست خواهند آمد نیز از قبل مشخص نباشند [۴].

۲-۱- شبکه عصبی (ANN)

الگوریتم ANN از سیستم عصبی بیولوژیکی الهام گرفته شده است و می‌توان آن را به عنوان یک ارگانیسم کامل متشکل از تعداد زیادی واحد محاسباتی در نظر گرفت که برای حل مسئله با یکدیگر تعامل دارند. هر نورون سیگنال‌های سلول‌های عصبی همسایه را جمع‌آوری کرده و آن‌ها را به لایه بعدی منتقل می‌کند و در نتیجه سیگنال‌های "تحریک کننده" یا "بازدارنده" ایجاد می‌کند (گورونسکو[1]، ۲۰۱۱). از این رو، هر نورون می‌تواند به عنوان یک پردازنده دیده شود که محاسبه ساده ای انجام می‌دهد؛ مانند تصمیم گیری در مورد ارسال کردن یا نکردن سیگنال به سلول‌های عصبی دیگر. یادگیری زمانی اتفاق می‌افتد که اثرات سیناپس‌ها تغییر کند، به عنوان مثال تأثیر یک نورون بر روی یک نورون دیگر تغییر می‌کند (تونچان[2]، ۲۰۰۸).

مدل‌های ANN دارای سه ویژگی اصلی هستند:

۱) یک یا چند لایه از سلول‌های عصبی پنهان که بخشی از لایه‌های ورودی یا خروجی شبکه نیستند و امکان یادگیری را فراهم می‌کنند.

[1] Gorunescu

[2] Tunçhan

۲) غیر خطی بودن قابل تمایز در فعالیت عصبی.

۳) مدل اتصال شبکه از اتصال درجه بالایی برخوردار است.

این ویژگی‌ها همراه با یادگیری از طریق آموزش به حل مشکلات دشوار و متنوع کمک می‌کند. یادگیری از طریق آموزش در یک مدل ANN تحت نظارت را الگوریتم پس‌انتشار خطا[1] نیز می‌نامند. الگوریتم BPN شبکه را براساس نمونه‌های ورودی-خروجی آموزش می‌دهد و یک سیگنال خطا پیدا می‌کند که تفاوت خروجی محاسبه شده و خروجی مورد نظر است. الگوریتم وزن سیناپسی گره‌های شبکه را متناسبا تنظیم می‌کند. بر اساس این اصل، یادگیری پس‌انتشار خطا در دو جهت رخ می‌دهد:

حرکت رو به جلو[2]: در اینجا، یک ماتریس ورودی به شبکه ارائه می‌شود. هر ورودی به یک گره متصل می‌شود که یک سیگنال خروجی موقتی ایجاد می‌کند که در لایه بعدی توسط یک عملکرد انتقال به گره دیگری منتقل می‌شود. سیگنال ورودی از طریق لایه‌های شبکه به جلو انتشار می‌یابد و در انتهای خروجی شبکه به عنوان سیگنال خروجی ظاهر می‌شود. خروجی که در لایه خروجی محاسبه می‌شود با پاسخ مورد نظر مقایسه می‌شود و مقدار تفاوت، خطای آن گره را تعریف می‌کند. وزن‌های سیناپسی شبکه در طول حرکت رو به جلو ثابت می‌مانند.

حرکت رو به عقب[3]: سیگنال خطایی که از نورون خروجی از آخرین لایه نشات می‌گیرد، از طریق شبکه به عقب پخش می‌شود. این، شیب محلی هر نورون را در هر لایه محاسبه می‌کند و اجازه می‌دهد تا وزن سیناپسی شبکه مطابق با قانون دلتا تغییر کند. محاسبه بازگشتی با حرکت رو به جلو که توسط حرکت رو به عقب دنبال می‌شود، برای هر الگوی ورودی تا زمان همگرایی شبکه ادامه می‌یابد (گوا[4]، ۲۰۰۷، مجومدر و هوسیان[5]، ۲۰۰۷، مانتری و همکاران[6]، ۲۰۱۴).

[1] error back-propagation (BPN)

[2] Forward Pass

[3] Backward Pass

[4] Quah

[5] Majumder & Hussian

[6] Mantri et al

BPN راه حل‌هایی را برای چندین مسـئله خطی و غیر خطی مانند طبقه‌بندی، کنترل گیاه، پیشـگویی، پیش‌بینی و علم رباتیک شناسایی می‌کند (مهرآرا و همکاران، ۲۰۱۰) [۵].

۲-۲- ماشین بردار پشتیبان (SVM)

SVM برای اولین بار توسـط کورتس و وپنیک[1] (۱۹۹۵) در زمینه تئوری یادگیری آماری و به حداقل رساندن ریسک ساختاری ارائه شده است. SVM در شناسایی الگو و مشکلات تخمین رگرسیون استفاده می‌شود و در حل مسائل تخمین وابستگی، پیش‌بینی و ساخت ماشین‌های هوشمند کاربرد دارد (اسمولا و شولکوف[2]، ۲۰۰۴).

الگوریتم SVM به کمک یک نگاشت غیرخطی داده‌های آموزشی اولیه را به یک بعد بالاتر تبدیل می‌کند. در این بعد جدید به دنبال ابرصـفحه‌ای [3] بهینه می‌گردد، که تاپل‌های یک کلاس را از دیگر کلاس به صورت خطی تفکیک می‌کند. با یک نگاشت غیرخطی مناسب به یک بعد بالای کافی، داده‌های دو کلاس را همیشـه می‌توان به کمک یک ابرصـفحه تفکیک نمود. الگوریتم SVM این ابرصـفحه را با کمک بردارهای پشـتیبان[4] (که اساسا تاپل‌های آموزشی هستند) و حاشیه‌ها[5] (که با کمک بردارهای پشـتیبان تعریف می‌شوند) پیدا می‌کند.

اگر داده‌ها به صـورت خطی غیر قابل تفکیک باشـند، یک طبقه‌بندی‌کننده‌ی SVM غیرخطی اعمال می‌شود. SVM بردارهای ورودی را با استفاده از یک تحول غیرخطی Φ به یک فضای مشخصه‌ی چند بعدی تبدیل می‌کند و با استفاده از یک تابع هسته $K(x, y)$ که در آن $\{x1 \dots xn\}$ بردارهای ورودی و $\{y1 \dots yn\}$ برچسب‌های آن‌ها $\{-1، 1\}$ هستند، در فضای مشخصه یک جداکننده‌ی خطی به عمل می‌آورد. هسته تابعی است که محصول نقطه ای $(\varphi(x).\varphi(y))$ دو تصویر برداری در فضای مشخصه را

[1] Cortes & Vapnik

[2] Smola & Schölkopf

[3] Hyperplane

[4] Support Vectors

[5] Margine

برمی‌گرداند (جونگ و رجیا[1]، ۲۰۰۸). اشـکال مختلفی برای K(x, y) در ادبیات وجود دارد (نصـرآبادی، ۲۰۰۷). در زیر چند نمونه آورده شده است.

خطی:

$$K(x,y) = x.y$$

چند جمله ای:

$$K(x,y) = (x.y)^d \text{ یا } K(x,y) = (1 + x.y)^d$$

گوسی

$$K(x,y) = \exp\left[\frac{\left|\left|x-y\right|\right|^2}{2\delta^2}\right]$$

به دلیل آن که الگوریتم‌های SVM دارای قابلیت مدل‌سازی کران‌های تصمیم‌گیری غیرخطی پیچیده هستند، حتی سریع‌ترین آن‌ها نیز می‌تواند دارای سرعت پایینی در زمان آموزش باشد، اما صحت آن‌ها بسیار بالا است. در ضمن آن‌ها نسبت به دیگر روش‌ها کمتر دچار مشکل بیش‌برازش داده‌ها می‌شوند. بردارهای پشتیبان توصیف فشرده‌ای از مدل یادگیری‌شده را ارائه می‌دهند. از الگوریتم‌های SVM هم می‌توان برای پیشگویی عددی و هم برای دسته‌بندی داده‌ها استفاده کرد.

SVM همچنین با اسـتفاده از رویکردهای "یکی در برابر یکی" یا "یکی علیه همه" برای مسـائل چند طبقه‌ای[2] اعمال می‌شود (مهرآرا و همکاران، ۲۰۱۰) [۵].

۲-۳- درخت تصمیم

یک درخت تصمیم همانطور که از نام آن مشخص است، یک ساختار درختی شبیه به فلوچارت دارد. هر گره داخلی (گره غیربرگ) در این درخت آزمونی را بر روی یک صفت خاصه نشان می‌دهد و هر شاخه نتیجه‌ی آزمون را نمایش می‌دهد و در هر گره برگ (یا گره پایانی) یک برچسب کلاس نگهداری می‌شود. برای ساخت درختان تصمیم به هیچ دانش خاص یا تنظیم پارامتری نیاز نیست. بنابراین برای یافتن اکتشافی دانش مناسب است. درختان تصمیم می‌توانند داده‌های چندبعدی را کنترل کنند. از نقطه نظر

[1] Jung & Reggia

[2] multi-class

بصری، هضم دانش ارائه شده در درختان تصمیم برای انسان راحت است. گام‌های دوگانه‌ی یادگیری و دسته‌بندی در استقراء درختان تصمیم ساده و سریع است و به طور کلی دارای صحت مناسبی هستند.

اواخر سال‌های ۱۹۷۰ و اوایل سال‌های ۱۹۸۰، محققی در حوزه‌ی یادگیری ماشین، الگوریتم ID3 را برای استقراء درخت تصمیم طراحی نمود. این کار تعمیم کارهای قبلی بر روی سیستم‌های یادگیری مفهوم[1] تلقی می‌شد که توسط گروهی از محققین شرح داده شده بود. پس از این الگوریتم C4.5 (که نسخه‌ی بعدی ID3 محسوب می‌شود) ارائه شد. اغلب از این الگوریتم برای سنجش الگوریتم‌های جدیدتر در یادگیری با ناظر استفاده می‌شد. در سال ۱۹۸۴، گروهی از پژوهشگران در حوزه‌ی آمار کتابی را با نام درختان رگرسیون و دسته‌بندی[2] (CART) منتشر کردند که در آن تولید درختان تصمیم دودویی بحث می‌شود. الگوریتم‌های ID3 و CART در زمان‌های تقریبا یکسانی و به صورت مستقل طراحی و پیشنهاد شدند و از رویکرد مشابهی برای یادگیری درختان تصمیم از تاپل‌های آموزش استفاده می‌کنند. این دو الگوریتم پایه و اساسی، باعث انجام سریع پژوهش‌هایی بر روی موضوع استقراء درخت تصمیم شدند [۴].

۲-۴- یادگیرنده‌های کُند (یادگیری از طریق همسایه‌ها)

در رویکرد کند، یادگیرنده تا آخرین دقیقه انتظار می‌کشد و قبل از آن مدلی برای دسته‌بندی یک تاپل آزمایشی ساخته نمی‌شود. بنابراین در مواجهه با یک تاپل آموزشی یک یادگیرنده‌ی کند آن را ذخیره می‌کند (یا تنها پردازش کمی را انجام می‌دهد) و تا دریافت یک تاپل آزمایشی صبر می‌کند. تنها این یادگیرنده هنگامی که با تاپل آزمایشی روبه‌رو می‌شود، به تلاش برای دسته‌بندی تاپل مذبور بر اساس شباهت آن با تاپل‌های آموزشی ذخیره شده می‌پردازد. بر خلاف روش‌های یادگیری مشتاق، یادگیرنده‌های کند کار کمتری را هنگام آموزش و کار بیشتری در هنگام دسته‌بندی یا پیشگویی عددی انجام می‌دهند.

ساختار داده‌ها در عملکرد آن‌ها تقریبا بی‌تاثیر است و به طور طبیعی از یادگیری افزایشی پشتیبانی می‌کند. آن‌ها قادر هستند فضای تصمیم پیچیده‌ای که شامل شکل‌هایی نظیر ابرچندضلعی‌ها می‌شود را مدل‌سازی کنند، که ممکن است این کار به سادگی توسط الگوریتم‌های دیگر یادگیری انجام نشود [۴].

[1] Concept Learning Systems

[2] Classification and Regression Trees

۲-۴-۱- دسته‌بندی‌های k همسایه‌ی نزدیک[1]

روش k نزدیک‌ترین همسایه برای اولین بار در اوایل سال‌های ۱۹۵۰ معرفی شد. این روش در مواجهه با داده‌های آموزشی حجیم با حجم عملیات بالایی روبه‌رو بود و به همین دلیل تا سال‌های ۱۹۶۰ که قدرت محاسبات کامپیوترها هنوز به اندازه‌ی کافی افزایش پیدا نکرده بود، از محبوبیت خوبی برخوردار نبود. از این الگوریتم به صورت گسترده‌ای در حوزه‌ی تشخیص الگو استفاده می‌شود.

دسته‌بندی‌های نزدیک‌ترین همسایه، یادگیری خود را بر اساس تشابه انجام می‌دهند، این کار با مقایسه‌ی تاپل آزمایشی و تاپل‌های آموزشی مشابه با آن صورت می‌گیرد. تاپل‌های آموزشی با کمک n صفت خاصه توصیف می‌شوند و هر تاپل در واقع نمایش نقطه‌ای در فضای n بعدی است. بدین ترتیب تمام تاپل‌های آموزشی در یک فضای n بعدی ذخیره می‌شوند. هرگاه با یک تاپل ناشناخته روبه‌رو می‌شوید، دسته‌بند k نزدیک‌ترین همسایه به دنبال k تاپل آموزشی است که شبیه‌ترین تاپل‌ها به تاپل ناشناخته هستند. این k تاپل آموزشی k همسایه‌ی نزدیک تاپل ناشناخته هستند.

دسته‌بندهای نزدیک‌ترین همسابه از مقایسه‌های مبتنی بر فاصله استفاده می‌کنند که در آن به طور طبیعی وزن هر یک از صفات خاصه برابر در نظر گرفته می‌شود. بنابراین هنگامی که داده‌ها حاوی نویز و صفات خاصه‌ی نامرتبط باشند، ممکن است با کاهش صحت دسته‌بند روبه‌رو شوید. به هرحال روش را می‌توان با وزن دهی صفات خاصه و حذف داده‌های نویز اصلاح نمود. انتخاب سنجه‌ای برای محاسبه‌ی فاصله می‌تواند امر مهمی باشد و ممکن است از سنجه‌های دیگری به غیر از فاصله‌ی اقلیدسی برای این کار استفاده شود [۴].

۲-۵- تکنیک‌هایی جهت بهبود صحت دسته‌بندی

در این روش‌ها مدلی برای دسته‌بندی انتخاب می‌شود که ترکیبی از چندین دسته‌بند است. هر دسته‌بند رای خود را صادر می‌کند و نتیجه‌ی نهایی در مورد برچسب کلاس بر اساس این رای‌ها صادر می‌شود.

[1] k-nearest-neighbor

۲-۵-۱- روش Bagging

مجموعه داده‌های D با تعداد d تاپل را در نظر بگیرید. روش bagging این گونه عمل می‌کند: در تکرار i ام الگوریتم (i=1,2,…,k) مجموعه آموزشی D_i با کمک روش نمونه گیری با جایگزینی از مجموعه داده‌های D انتخاب می‌شود. به دلیل آن که از نمونه گیری با جایگزینی استفاده می‌شود، ممکن است برخی از تاپل‌های موجود در D در مجموعه‌ی D_i قرار نگیرند، در حالی که برخی دیگر، بیش از یک بار انتخاب شوند. با کمک هر یک از مجموعه‌های آموزشی D_i مدل M_i تولید می‌شود. برای دسته‌بندی تاپل جدیدی مانند X هر یک از دسته‌بندهای M_i برچسب کلاس پیشنهادی خود را به عنوان یک رای ارائه می‌دهند. دسته‌بند تلفیقی M* با شمارش آراء، رای اکثریت را برای برچسب کلاس تاپل X انتخاب می‌کند. از روش bagging می‌توان جهت پیش گویی مقادیر پیوسته نیز استفاده کرد. برای این کار کافی است میانگین مقادیر برگردانده شده توسط دسته‌بندها را محاسبه کنیم [۴].

۲-۵-۲- روش Boosting و الگوریتم AdaBoost

در روش boosting به هر یک از تاپل‌های آموزشی نیز وزنی تخصیص داده می‌شود. پس از ساخت دسته‌بند M_i وزن‌ها تغییر خواهند کر تا دسته‌بند M_{i+1} که پس از M_i تولید می‌شود، توجه بیشتری را بر روی تاپل‌هایی که به درستی توسط M_i دسته‌بندی نشده‌اند، داشته باشد. دسته‌بند نهایی M* رای نهایی را با ترکیب رای‌های هر یک از دسته‌بندهای پایه محاسبه می‌کند، جایی که وزن رای هر یک از این دسته‌بندها تابعی از صحت آن است. در واقع وزن یک تاپل، دشواری دسته‌بندی آن تاپل را منعکس می‌کند. به صورتی که وزن بالاتر نشان می‌دهد اغلب این تاپل به درستی بسته‌بندی نشده است [۴].

۲-۵-۳- جنگل‌های تصادفی

در این بخش به سراغ روشی دیگر از روش‌های تلفیقی به نام جنگل‌های تصادفی می‌رویم. تصور کنید دسته‌بندهای استفاده شده در روش تلفیقی همگی از نوع درخت تصمیم هستند؛ بدین ترتیب این مجموعه تشکیل یک جنگل را خواهند داد. هر یک از درختان تصمیم با استفاده از یک انتخاب تصادفی صفات خاصه‌ی موجود در هر گره جهت تعیین انشعاب ساخته می‌شوند. به عبارت دیگر هر درخت بر اساس مقادیر یک بردار تصادفی ساخته می‌شود. این مقادیر دارای توزیع یکسانی برای تمام درختان

موجود در جنگل هستند و به صورت مستقلی نمونه گیری می‌شوند. برای دسته‌بندی نیز هر درخت رای خود را صادر و نتیجه نهایی با رای اکثریت تعیین می‌شود [۴].

فصل سوم

مروری بر پیشینه پژوهش

پنج گروه از پژوهشگران معتقد هستند که پیش‌بینی قیمت سهام امکان‌پذیر نیست. اولین گروه کسانی هستند که به فرضیه بازار کارا اعتقاد دارند. در بازار کارای سرمایه، اعتقاد بر این است که قیمت سهام انعکاسی از تمام اطلاعات مربوط به آن سهم است و تغییرات قیمت سهام دارای الگوی خاص قابل پیش‌بینی نیست.

دومین گروه کسانی هستند که به علت موثر بودن عوامل متعدد بر تغییرات قیمت سهام و رفتار آشوب‌گونه و غیر خطی تغییرات قیمت سهام پیش‌بینی قیمت سهام را امری غیرممکن می‌دانند (منجمی و همکاران ۱۳۸۸). در حقیقت پراکندگی قیمت سهام تحت تاثیر عوامل کلان اقتصادی مانند نرخ بهره و نرخ تورم، وقایع سیاسی مانند جنگ و تهدیدات بین‌المللی و منطقه‌ای، وقایع اجتماعی مانند اعتصاب‌ها و آشوب‌ها و عوامل رفتاری و روانی سرمایه‌گذاران قرار دارد. اگر قیمت یا بازده در بازارهای مالی با دقت بالایی قابل پیش‌بینی باشد، سیستم‌های پیش‌بینی کننده‌ی قیمت تبدیل به یک دستگاه چاپ پول شده که ثروت زیادی را نصیب سرمایه‌گذاران می‌کند که این امر، در یک اقتصاد پایدار امکان پذیر نیست (گرنجر و تیمرمن[1]، ۲۰۰۴). به بیان دیگر اگر تغییرات قیمت سهام در بازار قابل پیش‌بینی باشد (به طور تصادفی نباشد) احتمال وجود یک ثروت نامحدود برای سرمایه‌گذاران وجود خواهد داشت که با توجه به وضعیت حاضر و نتایج پژوهش‌های موجود این امکان وجود ندارد (گرانگر[2]، ۱۹۹۱).

سومین گروه، نوع رویدادها در بازارهای مالی را دلیل اصلی عدم پیش‌بینی پذیری قیمت می‌دانند. هر رویدادی دارای دو ویژگی احتمال و شدت است. مدل‌های پیش‌بینی کننده قیمت، مبتنی بر احتمالات است و اساسا محققان به شدت رویدادها توجهی نمی‌کنند. طبق نظر این گروه به دلایل رفتاری، پیش‌بینی تغییرات قیمت با احتمال پایین ولی شدت بالا غیرقابل امکان است (نیکلاس تالب[3]، ۲۰۱۲).

چهارمین گروه ماهیت بازارهای مالی را دلیل اصلی عدم پیش‌بینی پذیری قیمت سهام می‌دانستند. به نظر این گروه، ماهیت بازی به جمع صفر بودن در بازارهای مالی دلیل اصلی دشواری پیش‌بینی در این حوزه است. پیش‌بینی دقیق تر هوا باعث تغییر رفتار هوا نمی‌شود ولی پیش‌بینی دقیق قیمت سهام بر

[1] Timmermann, A., & Granger, C. W.

[2] Granger, C. W. J.

[3] Nassim Nicholas Taleb

خود قیمت سهام تاثیرگذار است. یک اقتصاددان با نام لوکاس چیزی را مطرح کرده بود که به نام خودش به عنوان نقد لوکاس معروف شـد. نقد لوکاس اظهار داشـت که شـاید پیش گویی اقتصـاددانان بر روند اقتصاد تاثیر بگذارد که اثر آن پیش گویی را خنثی سازد. فرض کنیم اقتصاددانان تورم را پیش‌بینی کنند، خزانه داری و بانک مرکزی در واکنش به گفته‌های ایشان وارد عمل شده و با اعمال سیاست‌های پولی و مالی تورم را پایین می‌آورند (تانگ و لین[1]، ۲۰۰۷)

پنجمین گروه عامل مهمی به نام *اقتصـاد اطلاعات* را مطرح می‌سـازند. به نظر این گروه شـکل گیری اطلاعات نامتقارن عامل مهمی در پیش‌بینی ناپذیری در بازارهای مالی اسـت. همیشـه کسـانی (مدیران ارشد شرکت ها) هستند که دارای اطلاعات محرمانه باشند (نیکواقبال و همکاران، ۱۳۹۲).

با این حال گروهی از پژوهشگران با توجه به نتایج حاصل از تحقیقات‌شان پیش‌بینی در بازارهای مالی و قیمت سـهام را امری بسـیار دشـوار ولی امکان پذیر می‌دانند (مالکی[2]، ۲۰۰۳). ماهیت پیچیده، تکاملی، داشـتن خاصـیت دینامیکی و غیرخطی که به دلیل تعامل حوادث و شـرایط اقتصـادی بوجود می‌آید و انتظارات غیرعقلانی سرمایه‌گذاران پیش‌بینی قیمت سهام را به امری دشوار مبدل می‌سازد نه امری محال (ناکاموری، ۲۰۰۵). کشـف و بهبود الگوریتم‌های پیش‌بینی کننده و امکان انجام محاسـبات پیچیده به وسیله رایانه‌ها، پژوهشگران را در این امر دشوار یاری می‌رساند [۱].

برای پیش‌بینی قیمت سـهام راه‌های متفاوتی وجود دارد. یک راه کاهش پیچیدگی با اسـتخراج بهترین صفات یا انتخاب از بین ویژگی‌ها می‌باشد [۶]. این روش با کاهش پیچیدگی به پیش‌بینی قیمت با صحت بیشتر کمک می‌کند. بررسی رابطه میان شرکت‌های هم صنعت نیز به ایجاد مدل‌های یادگیری ماشینی کمک می‌کند. در اینجا به بررسـی کوتاهی پیرامون کاربرد شـبکه‌های عصـبی، ماشـین بردار پشـتیبان، درخت تصمیم و نزدیک‌ترین همسایه در پیش‌بینی قیمت سهام می‌پردازیم.

[1] Tang, C. F., & Lean, H. H.

[2] Malkiel, B. G.

برگس[1] و همکاران (۲۰۰۰) از شـــبکه عصـــبی با دو لایه پنهان برای پیش‌بینی معاملات آتی یورو/دلار با بهره گیری از بالاترین قیمت، پایین‌ترین قیمت، قیمت باز گشـایی و قیمت بسـته شـدن، اسـتفاده کردند [۷].

تیلاکاراتن[2] و همکاران (۲۰۰۷) با بهره گیری از الگوریتم شـــبکه‌های عصـــبی به پیش‌بینی ســـیگنال معاملاتی روز آتی شاخص سهام معمولی استرالیا[3] با استفاده بازده روز فعلی در قیمت پایانی S&P 500، شاخص FTSE 100 (انگلیس) و شاخص CAC 40 (فرانسه) به عنوان ورودی پرداخته اند. نویسـندگان دریافتند که شبکه‌های عصبی رو به جلو[4] عملکرد بهتری نسبت به شبکه‌های احتمالی[5] دارند [۸].

تاکور و کومار[6] (۲۰۱۷) روشی جدید از ترکیب الگوریتم‌های ماشین بردار پشتیبان وزنی و جنگل تصادفی برای ایجاد ســـیگنال‌های خرید / نگهداری / فروش ارائه کرده لند به گونه ای که در ابتدا با اســـتفاده از جنگل تصادفی زیرمجموعه‌ای بهینه از طیف گسترده ای از ابزارهای تکنیکال انتخاب می‌شود و سپس از الگوریتم ماشین بردار پشتیبان وزن دار جهت پیش‌بینی استفاده می‌شود [۹].

پژوهش فونته و همکاران[7] (۲۰۰۶) با به کار بردن الگوریتم ژنتیک ســـعی در بهینه کردن پارامترهای اندیکاتورهای مختلف تکنیکال داشـته اسـت. در بازار داخلی در خصـوص طراحی یک سـیسـتم معاملات الگوریتمی با استفاده از اندیکاتورهای تکنیکال، می‌توان به کار دستپاک و رستگار (۱۳۹۴) اشاره کرد. در این پژوهش با اســتفاده از اطلاعات اندیکاتورهای تکنیکال به پیش‌بینی روند قیمت ســهم و تعیین میزان خریدنی، فروختنی و یا نگهداشـتنی بودن آن سهم پرداخته شده است. این مقاله که در بازار بورس تهران انجام شده است، حاکی از بهتر بودن سیستم معاملات الگوریتمی طراحی شده مبتنی بر تکنیکال نسبت به اســتراتژی خرید و نگهداری اســت. در پژوهشـــی دیگر، فلاح پور و حکیمیان (۱۳۹۵) به طراحی یک

[1] Burgess

[2] Tilakaratne

[3] Australian All Ordinary (AORD) index

[4] ANN feedforward

[5] probabilistic networks

[6] Manoj Thakur & Deepak Kumar

[7] de la Fuente, D., Garrido, A., Laviada, J. and Gómez, A.

سیستم معاملات الگوریتمی از نوع معاملات زوجی در بورس اوراق بهادار تهران پرداخته اند که مطابق ادعای نویسندگان بازدهی چشمگیری نسبت به بازدهی معمولی سهام در مدت مشابه دارد [۳].

فلاح پور و علی پور (۱۳۹۳) به پیش بینی شاخص کل سهام بورس اوراق بهادار تهران با استفاده از شبکه های عصبی موجکی پرداختند. ابتدا از تبدیل موجک گسسته برای نویززدایی داده‌ها در سری زمانی استفاده کردند؛ سپس به کمک شبکه های عصبی به پیشبینی شاخص سهام پرداختند. بر اساس نتایج، عملکرد شبکهٔ عصبی موجکی سطح خطای کمتری نسبت به شبکهٔ عصبی معمولی در پیشبینی شاخص سهام داشت [۲].

مشاری و همکاران (۱۳۹۶) با کمک الگوریتم ژنتیک به بهینه سازی متغیرهای پژوهش پرداخته و سپس مدلی جهت پیش‌بینی نقاط طلایی ارائه نموده اند [۱].

سزار و همکارش (۲۰۱۸) با استفاده از شبکه عصبی حلقوی، اطلاعات سری‌های زمانی مالی را به تصاویر دو بعدی تبدیل کرده و با استفاده از آن به تحلیل و تشخیص نقاط خرید، نگهداری و فروش پرداخته اند. در واقع با استفاده از ۱۵ نوع تحلیل تکنیکال (با پارامترها و فواصل زمانی متفاوت برای هر سهم) بر روی شاخص داوجونز یک تصویر دو بعدی ایجاد کرده و سپس با تحلیل تصاویر سیگنال خرید، نگهداری و فروش ایجاد کرده‌اند [۱۰].

فصل چهارم

روش‌شناسی پژوهش

۴-۱- تعریف مساله

این پژوهش از نظر رویکرد کاربردی بوده و مبتنی بر پژوهش‌های میدانی است. جامعه آماری آن، اطلاعات جمع آوری شده از قیمت ۵۰۰ سهام موجود در شاخص S&P500[۱] در بازه زمانی ابتدای سال ۲۰۱۵ تا پایان ماه ژوئن سال ۲۰۲۱ می‌باشد که به جهت ارزیابی بهتر مدل، در دو مرتبه به دو سری داده آموزش و آزمون مختلف دسته‌بندی شدند. این پژوهش درصدد است تا مدل پیش‌بینی جدیدی برای پیش‌بینی قیمت سهام صندوق SPY[۲] به عنوان معیار شاخص S&P500 ارائه نماید، به گونه‌ای که در ابتدا از میان پانصد سهم موجود در شاخص S&P500 که اطلاعات هر یک در بازه زمانی آموزش ورودی مدل به شمار می‌رود، صد سهام دارای بیشترین اهمیت در پیش‌بینی را به عنوان ورودی‌های اصلی انتخاب کرده و با استفاده از ترکیب چند الگوریتم، نتیجه مطلوب‌تری ارائه دهد. مدل ترکیبی مدنظر در ابتدا با استفاده از هر یک از الگوریتم‌های MLP، RF، SVM و KNN به پیش‌بینی تغییر قیمت در روز آتی پرداخته و در مرحله دوم نتایج به دست آمده از الگوریتم‌های مذکور را با استفاده از متاالگوریتم بگینگ ترکیب نموده و بر اساس رای اکثریت الگوریتم‌ها، سیگنال نهایی برای روز آتی را تولید می‌کند. لازم به ذکر است که پارامترهای هریک از الگوریتم‌های مورد استفاده، با استفاده از الگوریتم اعتبارسنجی متقابل[۳] جست‌وجوی شبکه‌ای[۴] در کتابخانه Scikit learn زبان برنامه نویسی پایتون، تعیین شده است.

---

[۱] S&P 500 به عنوان یکی از معیارهای اصلی بازار سهام ایالات متحده عمل می‌کند و سلامت مالی و ثبات اقتصاد را نشان می‌دهد.

[۲] SPDR S&P 500 Trust ETF که با عنوان SPY ETF نیز شناخته می‌شود، یکی از محبوب ترین صندوق‌هایی است که هدف آن دنبال‌کردن شاخص S&P 500 است.

[3] Cross Validation

[4] Grid Search

۴-۲- مراحل اجرای پژوهش

۴-۲-۱- مرحله اول: شناسایی و جمع‌آوری متغیرهای مسئله

اطلاعات قیمت پایانی[1] و حجم معاملات روزانه مربوط به ۵۰۰ سهام موجود در شاخص S&P500 و قیمت نماد SPY در بازه زمانی ابتدای سال ۲۰۱۵ تا پایان ماه ژوئن سال ۲۰۲۱ از طریق کتابخانه مالی یاهو[2] در زبان برنامه‌نویسی پایتون[3] استخراج گردید و در راستای کاهش نویز ورودی به مدل، با استفاده از این اطلاعات اندیکاتورهای شاخص قدرت نسبی[4] و میانگین متحرک وزنی پنج روزه[5] نیز برای هر یک از نمادها در بازه‌ی زمانی ذکر شده محاسبه شد.

۴-۲-۲- مرحله دوم: انتخاب صد متغیر دارای بیشترین اهمیت

در این مرحله با استفاده از قابلیت تعیین اهمیت متغیرها[6] در الگوریتم جنگل تصادفی، صد سهام دارای بیشترین اهمیت در پیش‌بینی سیگنال‌های معاملاتی انتخاب می‌شوند. سپس پایگاه داده ورودی مدل که شامل قیمت پایانی، حجم معاملات، شاخص قدرت نسبی و میانگین متحرک وزنی پنج روزه برای صد سهم انتخاب شده، است به عنوان مجموعه متغیرهای مورد استفاده در پیش‌بینی، تشکیل می‌شود.

۴-۲-۳- مرحله سوم: پیش‌بینی و ارزیابی اولیه

در این گام داده‌ها به دو دسته آموزش و آزمایش تقسیم بندی می‌شوند، با توجه به اجرای مدل در دو بازه آموزش و آزمون متفاوت، در مرتبه اول داده‌های مربوط به ابتدای سال ۲۰۱۵ تا ابتدای سال ۲۰۲۰

[1] Adjusted Close

[2] Yahoo Finance

[3] Python

[4] Relative Strength Index (RSI)

[5] 5 Day Weighted Moving Average (WMA5)

[6] Feature Importance

به عنوان داده آموزش[1] و داده‌های مربوط به ابتدای ۲۰۲۰ به بعد به عنوان داده آزمایش[2] درنظر گرفته می‌شود و در مرتبه دوم داده‌های مربوط به ابتدای سال ۲۰۱۵ تا ابتدای سال ۲۰۲۱ به عنوان داده آموزش[3] و داده‌های مربوط به ابتدای ماه ژانویه سال ۲۰۲۰ تا پایان ماه ژوئن سال ۲۰۲۱ به عنوان داده آزمایش[4] درنظر گرفته می‌شود. در ادامه هر یک از الگوریتم‌های MLP، RF، SVM و KNN توسط داده‌های آموزشی برای پیش‌بینی سیگنال روز آتی آموزش داده می‌شوند و نتایج عملکرد هریک از الگوریتم‌ها برای پیش‌بینی توسط داده‌های آموزشی، با استفاده از ماتریس اختلال[5] و دو معیار دقت[6] و اطمینان[7] سنجیده خواهد شد.

$$Precision\ (positive) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

$$Precision(negative) = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}}$$

$$Recall(positive) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$Recall(negative) = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

۴-۲-۴- مرحله چهارم: پیش‌بینی بر اساس داده‌های آزمایشی و ارزیابی مدل‌ها

در این مرحله هر یک از الگوریتم‌ها بر اساس داده‌های مربوط به ابتدای سال ۲۰۱۵ تا یک روز قبل از روز مورد پیش‌بینی، به تولید سیگنال خرید، نگهداری یا فروش می‌پردازند (همانند معامله‌گری که در

[1] Train

[2] Test

[3] Train

[4] Test

[5] Confusion Matrix

[6] Precision

[7] Recall

پایان روز فعلی برای پیش‌بینی ســـیگنال روز بعد، اطلاعات مربوط به روز جاری را نیز در اختیار خواهد داشت). در پایان نتایج پیش‌بینی هر یک از الگوریتم‌ها با معیار دقت[1] مدل سنجیده می‌شود.

$$Accuracy(ACC) = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}}$$

۴-۲-۵- مرحله پنجم: ترکیب نتایج پیش‌بینی الگوریتم‌ها و تولید سیگنال نهایی

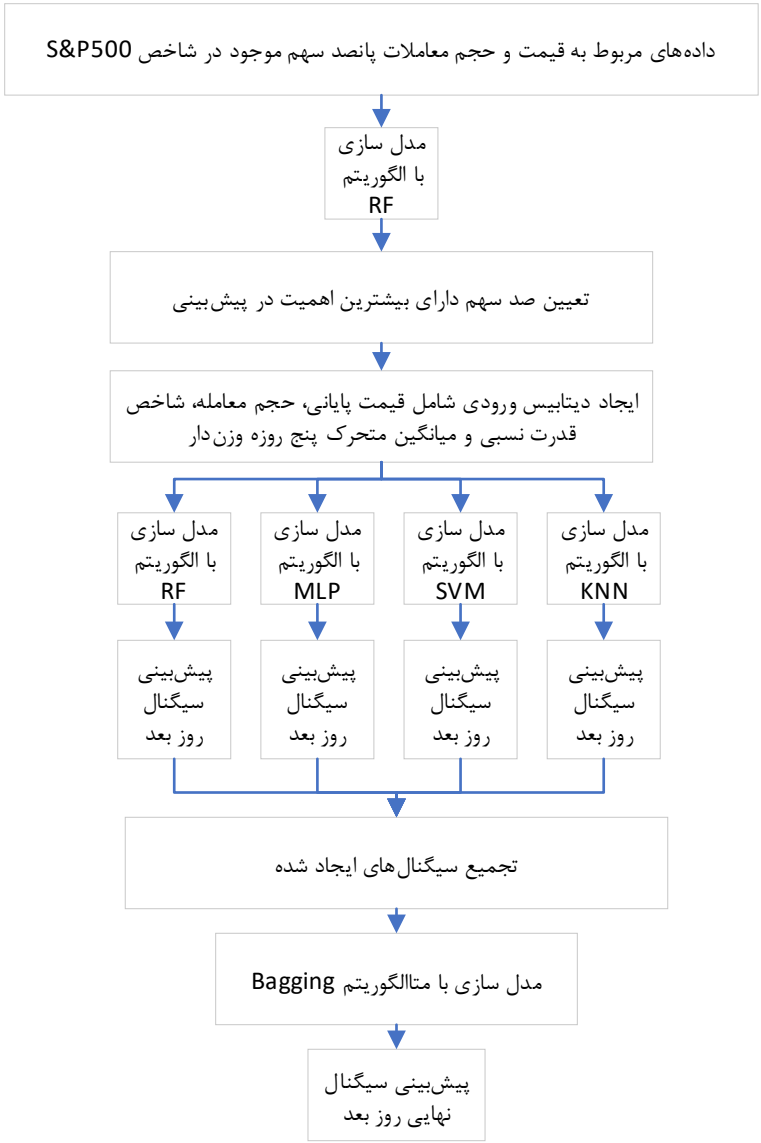
در این مرحله نتایج الگوریتم‌های مرحله قبل با اســتفاده از متاالگوریتم[2] بگینگ[3]، که یک الگوریتم ترکیبی است، سیگنال نهایی برای روز آتی را بر اساس رای اکثریت تولید می‌کند.

شکل ذیل مراحل اجرای مدل را نشان می‌دهد [شکل ۴-۱]:

[1] Accuracy

[2] Meta-Algorithm

[3] Bagging (Bootstrap aggregating)



شکل ۴-۱: مدل پیش‌بینی

فصل پنجم

بررسی نتایج

۵-۱- جزئیات پیاده سازی مدل

تمامی الگوریتم‌ها از داده‌های مربوط به سهام شرکت‌های موجود در شاخص S&P500 به عنوان داده ورودی و از اطلاعات مربوط به قیمت سهام صندوق SPY به عنوان داده مورد پیش‌بینی و در بازه زمانی ابتدای سال ۲۰۱۵ الی ابتدای ماه جولای سال ۲۰۲۱ استفاده می‌کنند.

در این مطالعه با توجه به بررسی‌های موریس[1] در رابطه با توزیع داده‌ها و مرزبندی تعیین سیگنال‌ها، فرض می‌شود که ۰.۵٪ افزایش (یا کاهش) در قیمت پایانی به اندازی کافی معقول است که حرکت مربوطه به عنوان سیگنال خرید (یا فروش) در نظر گرفته شود همچنین درصورتی که تغییر قیمت کمتر از ۰.۵٪ افزایش (یا کاهش) باشد، سیگنال نگهداری تولید خواهد شد [۵].

پارامترهای اصلی الگوریتم‌ها از جمله MLP و RF توسط قابلیت جستجوی شبکه‌ای[2] کتابخانه سایکیت‌لرن[3] و با اعتبارسنجی متقابل[4] پنجگانه بیش از هزار و پانصد حالت مختلف تعیین خواهند شد.

۵-۲- سنجش عملکرد مدل

در مرحله آموزش هر یک از الگوریتم‌ها بر اساس دو معیار Precision و Recall مورد ارزیابی قرار گرفته‌اند. نتایج هر یک از الگوریتم‌ها در بازه‌های زمانی آموزش و آزمون متفاوت که در ادامه آورده شده است، حاکی از آن است که الگوریتم جنگل تصادفی دارای دقت صددرصد در پیش‌بینی بر اساس داده‌های آموزش است که این می‌تواند نشانه بیش برازش[5] نیز باشد. همچنین واضح است که متاالگوریتم بگینگ نیز خود را با نتایج الگوریتم جنگل تصادفی مطابقت داده و از دقت صددرصد در بازه زمانی آموزش برخوردار است.

[1] S. A. Morris

[2] Grid Search

[3] Scikit Learn

[4] Cross Validation

[5] Over Fitting

جدول ۵-۱: عملکرد الگوریتم‌ها در پیش‌بینی با استفاده از داده‌های آموزشی در بازه‌ی زمانی ۲۰۱۵ الی ۲۰۲۰

Signal	MLP		RF		SVC		KNN		**FINAL MODEL**		Support
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	**Precision**	**Recall**	
Sell	0.68	0.41	1.00	1.00	0.86	0.03	0.49	0.46	**1.00**	**1.00**	213
Hold	0.75	0.81	1.00	1.00	0.61	0.99	0.71	0.88	**1.00**	**1.00**	735
Buy	0.56	0.56	1.00	1.00	0.58	0.07	0.67	0.32	**1.00**	**1.00**	296
Accuracy	70%		100%		60%		67%		**100%**		

جدول ۵-۲: عملکرد الگوریتم‌ها در پیش‌بینی با استفاده از داده‌های آموزشی در بازه‌ی زمانی ۲۰۱۵ الی ۲۰۲۱

Signal	MLP		RF		SVC		KNN		**FINAL MODEL**		Support
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	**Precision**	**Recall**	
Sell	0.97	0.82	1.00	1.00	0.67	0.01	0.49	0.52	**1.00**	**1.00**	284
Hold	0.87	1.00	1.00	1.00	0.59	0.97	0.70	0.84	**1.00**	**1.00**	818
Buy	0.95	0.79	1.00	1.00	0.50	0.17	0.58	0.32	**1.00**	**1.00**	396
Accuracy	90%		100%		57%		64%		**100%**		

در مرحله آزمون نمره دقت هر یک از الگوریتم‌ها، بر اساس درستی یا نادرستی پیش‌بینی سیگنال‌های روزانه، در دو مرحله و برای بازه‌های زمانی آموزش و آزمون متفاوت محاسبه شده است. نتایج حاکی از آن است که صحت پیش‌بینی توسط متاالگوریتم بگینگ که در انتهای مدل قرار دارد، از صحت پیش‌بینی هریک از الگوریتم‌های دیگر بیشتر بوده و در نتیجه سیگنال‌های ایجاد شده توسط آن از صحت بالاتری برخوردار است.

جدول ۵-۳: عملکرد الگوریتم‌ها در پیش‌بینی با استفاده از داده‌های آزمایشی در بازه‌ی زمانی ابتدای ۲۰۲۰ الی ابتدای جولای ۲۰۲۱

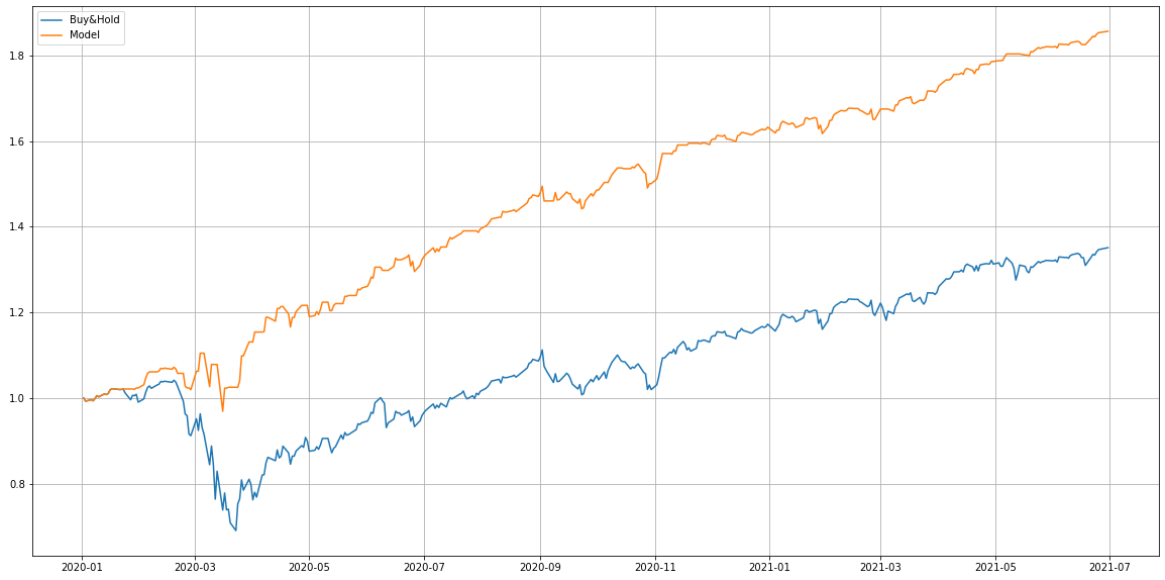
Signal	MLP		RF		SVC		KNN		**FINAL MODEL**		Support
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	
Sell	0.22	0.25	0.30	0.24	0.35	0.06	0.26	0.35	0.31	0.27	93
Hold	0.43	0.38	0.51	0.66	0.43	0.92	0.52	0.60	0.52	0.66	146
Buy	0.30	0.31	0.41	0.34	0.46	0.17	0.41	0.25	0.42	0.34	138
Accuracy	**32%**		**44%**		**43%**		**41%**		**45%**		

جدول ۵-۴: عملکرد الگوریتم‌ها در پیش‌بینی با استفاده از داده‌های آزمایشی در بازه‌ی زمانی ابتدای ۲۰۲۱ الی ابتدای جولای ۲۰۲۱

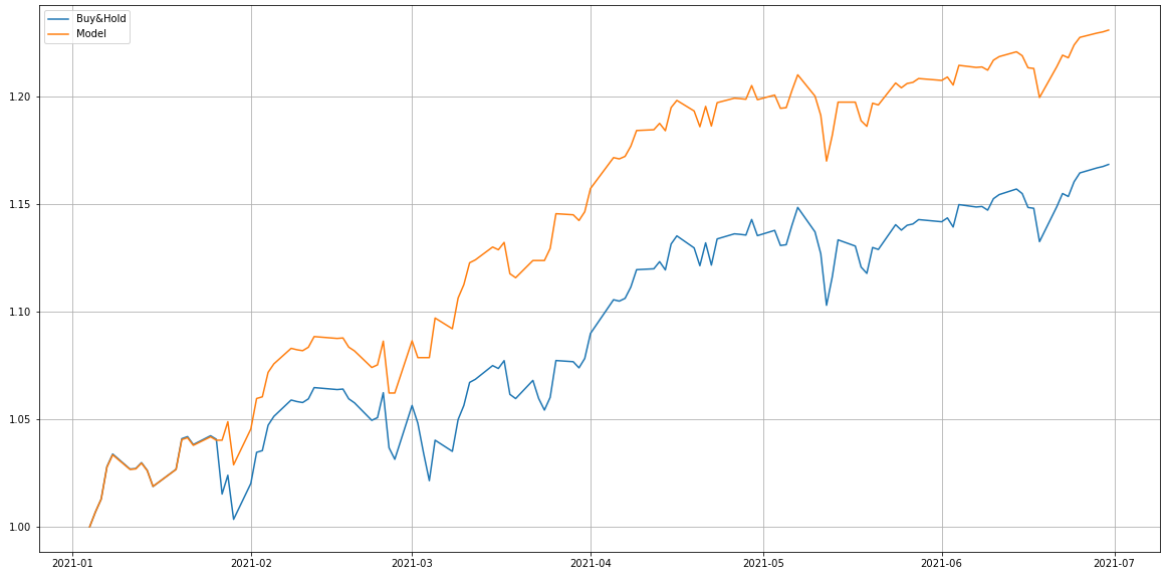
Signal	MLP		RF		SVC		KNN		**FINAL MODEL**		Support
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	
Sell	0.00	0.00	0.43	0.13	0.00	0.00	0.25	0.26	0.44	0.15	23
Hold	0.52	0.94	0.57	0.92	0.55	0.98	0.64	0.67	0.59	0.93	63
Buy	0.40	0.11	0.53	0.21	0.67	0.21	0.35	0.32	0.53	0.21	38
Accuracy	**50%**		**56%**		**56%**		**48%**		**57%**		

۵-۳- مقایسه بازده

نمودارهای ذیل بازده مدل را در مقایسه با بازده استراتژِی خرید و نگهداری برای داده‌های آزمون و در دو بازه زمانی متفاوت نمایش می‌دهد که حاکی از عملکرد بهتر مدل می‌باشد.



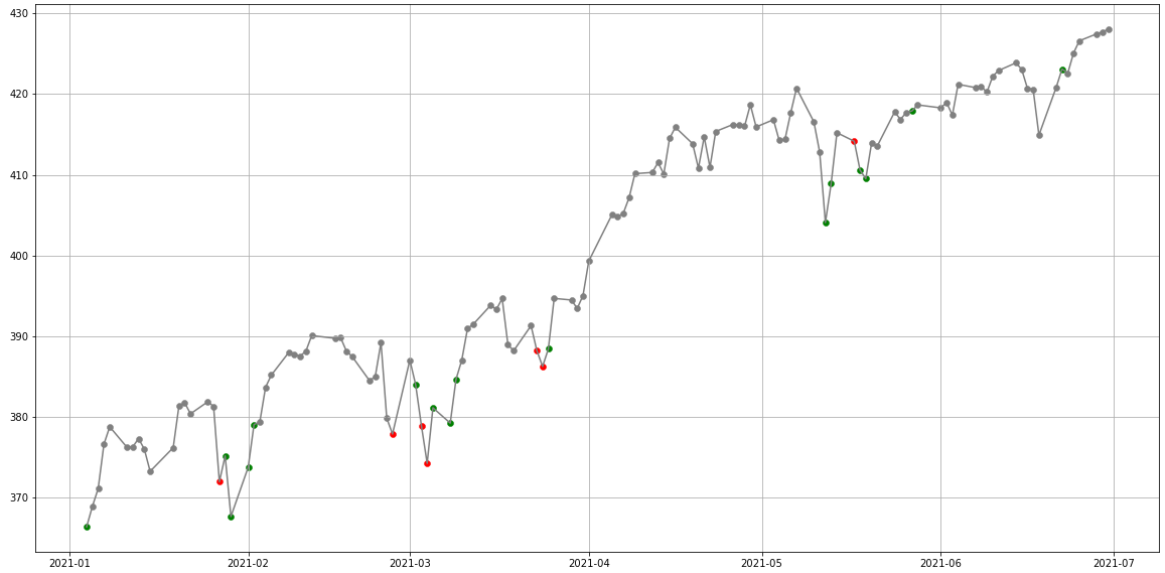
شکل ۵-۱: مقایسه‌ی بازده مدل و بازده استراتژی خرید و نگهداری در بازه‌ی زمانی ابتدای ژانویه‌ی ۲۰۲۰ تا ابتدای جولای ۲۰۲۱



شکل ۵-۲: مقایسه‌ی بازده مدل و بازده استراتژی خرید و نگهداری در بازه‌ی زمانی ابتدای ژانویه۲ی ۲۰۲۱ تا ابتدای جولای ۲۰۲۱

نمودارهای ذیل سیگنال‌های تولید شده توسط مدل را بر روی نمودار قیمت سهام SPY و در دو بازه زمانی آزمون متفاوت نمایش می‌دهد. قابل ذکر است که سیگنال خرید با رنگ سبز، سیگنال فروش با رنگ قرمز و سیگنال نگهداری با رنگ خاکستری نمایش داده شده است.

شکل ۵-۳: سیگنال‌های ایجاد شده توسط مدل در بازه‌ی زمانی آزمون (ابتدای ژانویه‌ی ۲۰۲۰ تا ابتدای جولای ۲۰۲۱)



شکل ۵-۴: سیگنال‌های ایجاد شده توسط مدل در بازه‌ی زمانی آزمون (ابتدای ژانویه‌ی ۲۰۲۱ تا ابتدای جولای ۲۰۲۱)

بررسی سیگنال‌های تولید شده توسط مدل، بیانگر عملکرد مناسب مدل در مواجهه با تلاطمات قیمت می‌باشد.

فصل ششم

جمع‌بندی، نتیجه‌گیری و پیشنهادات

۶-۱- جمع‌بندی و نتیجه‌گیری

هدف اصلی این پژوهش دستیابی به نرخ سود بالاتر در بازار سرمایه است و همانطور که اشاره شد، تلاش شده است بر اساس تغییرات قیمت سهم‌های زیر مجموعه شاخص S&P500 تغییرات شاخص در روز معاملاتی بعد پیش‌بینی شود. مدل ارائه شده در این پژوهش در سه مرحله و با استفاده از الگوریتم‌های جنگل تصادفی، شبکه عصبی، ماشین بردار پشتیبان و نزدیکترین همسایه به پیش‌بینی سیگنال خرید، نگهداری و یا فروش می‌پردازد. به این صورت که در ابتدا با استفاده از الگوریتم جنگل تصادفی صد سهم دارای بیشترین اهمیت در پیش‌بینی انتخاب می‌شوند، سپس هر یک از الگوریتم‌های ذکر شده به پیش‌بینی سیگنال معاملاتی روز آتی می‌پردازند و در نهایت با استفاده از الگوریتم شبکه عصبی سیگنال نهایی بر پایه سیگنال‌های تئلید شده در مرحله قبل ایجاد می‌شود.

صحت پیش‌بینی توسط الگوریتم نهایی از صحت پیش‌بینی هریک از الگوریتم‌های دیگر بیشتر بوده و میزان بازده ناشی از معامله بر اساس سیگنال‌های روزانه تولید شده توسط مدل برای داده‌های آزمون بیشتر از سود از ناشی استراتژِی خرید و نگهداری است که بیانگر عملکرد مناسب مدل پیشنهادی است.

۶-۲- پیشنهادات

پیشنهاد می‌شود برای توسعه‌ی این روش از داده‌های تکمیلی همچون دیگر شاخص‌های بازارسهام ایالات متحده، شاخص بازار سهام دیگر کشورها و همچنین اندیکاتورهای دیگر استفاده شود. همچنین می‌توان با استفاده از الگوریتم‌های داده کاوی نوین و استفاده از روش‌های دیگر در جهت تعیین پارامترهای الگوریتم‌ها، عملکرد مدل را بهبود بخشید. همچنین به‌کارگیری متاالگوریتم Boosting به عنوان الگوریتم نهایی می‌تواند مورد بررسی قرار گیرد.

منابع و مراجع

[۱] مشاری، محمد؛ دیده خانی، حسین؛ خلیلی دامغانی، کاوه؛ عباسی، ابراهیم؛ "طراحی مدل هوشمند ترکیبی جهت پیش‌بینی نقاط طلایی قیمت سهام"، *فصلنامه علمی پژوهشی دانش سرمایه‌گذاری*، سال هشتم، شماره ۲۹، صفحات ۶۶-۴۵، بهار ۱۳۹۸.

[۲] درودی، دیاکو؛ ابراهیمی، سید بلبک؛ "ارائه‌ی روش هیبریدی نوین برای پیش‌بینی شاخص کل قیمت بورس اوراق بهادار"، *تحقیقات مالی*، دوره ۱۸، شماره ۴، صفحات ۶۳۲-۶۱۳، زمستان ۱۳۹۵.

[۳] رستگار، محمدعلی؛ صداقتی پور، امین؛ "ارائه سیستم معاملات الگوریتمی برای قرارداد آتی سکه طلا مبتنی بر داده‌های درون-روزی"، *فصلنامه علمی پژوهشی دانش سرمایه‌گذاری*، سال هفتم، شماره ۲۸، صفحات ۶۷-۴۹، زمستان ۱۳۹۷.

[۴] هان، ژیاوی؛ کمبر، میشلین؛ پی، ژان؛ اسماعیلی، مهدی؛ *داده‌کاوی (مفاهیم و تکنیک‌ها)*، انتشارات نیازدانش، تهران، ۱۳۹۳.

[5] ERKARTAL, Bugra & Ozdamar, Linet. (2018). Generating Buy/Sell Signals for an Equity Share Using Machine Learning. Eurasian Journal of Business and Economics. 11. 85-105. 10.17015/ejbe.2018.022.04.

[6] Mantri, J. K., Gahan, P., & Nayak, B. B. “Artificial neural networks--an application to stock market volatility. Soft-Computing in Capital Market: Research and Methods of Computational Finance for Measuring Risk of Financial Instruments”, 179, 2014.

[7] Burgess, A. N., & others. “A computational methodology for modelling the dynamics of statistical arbitrage”, PHD Thesis: University of London, 2000.

[8] Tilakaratne, C. D., Morris, S. A., Mammadov, M. A., & Hurst, C. P. “Predicting stock market index trading signals using neural networks”, In Proceedings of the 14th Annual Global Finance Conference (GFC'07), Pages 171–179, 2007.

[9] Manoj Thakur, Deepak Kumar. “A hybrid financial trading support system using multi-category classifiers and random forest”, Applied Soft Computing, Volume 67, Pages 337-349, 2017.

[10] Omer Berat Sezer, Ahmet Murat Ozbayoglu. “Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach”, Applied Soft Computing, Volume 70, Pages 525-538, 2018.

پیوست‌ها

جدول پ-۱: شرح کد مدل ارائه شده

```
#Feature Importance

y = spy['Signal'][spy.index.isin(data['Adj Close'].index)]
X = data['Adj Close'][data['Adj Close'].index.isin(spy.index)]

x_train = X.loc[X.index < '2020-01-01']
y_train = y.loc[y.index < '2020-01-01']

from sklearn.ensemble import RandomForestClassifier
fi_forest = RandomForestClassifier()
fi_forest.fit(x_train, y_train)
FI = pd.DataFrame()

FI['Stock'] = data['Adj Close'].columns
FI['Importance'] = fi_forest.feature_importances_

FI = FI.sort_values('Importance',ascending=False)
FI = FI.reset_index()
VIP = FI['Stock'].loc[:100].values
print (FI['Stock'].loc[:100].values)


# Change daily price to WMA5
df_WMA5 = pd.DataFrame()
for stock in data['Adj Close'].columns :
    df_WMA5[stock] = ta.WMA(data['Adj Close'][stock],timeperiod = 5)
df_WMA5.dropna(inplace=True)

# Change daily price to RSI
df_RSI = pd.DataFrame()
for stock in data['Adj Close'].columns :
    df_RSI[stock] = ta.RSI(data['Adj Close'][stock],timeperiod = 14)
df_RSI.dropna(inplace=True)

# Make a new dataset with WMA5 , Vol , RSI
df_data = pd.DataFrame()
df_data = pd.merge(df_WMA5[VIP]
                   ,data['Volume'][VIP]

,right_index=True,left_index=True,suffixes=('_wma','_vol'))
df_data = pd.merge(df_data
                   ,df_RSI[VIP]

,right_index=True,left_index=True,suffixes=('','_rsi'))
df_data = pd.merge(df_data
                   ,spy[['Adj Close','Volume']]

,right_index=True,left_index=True,suffixes=('','_spy'))
```

```
df_data.dropna(axis=0,inplace=True)


#Make a Function to measure performance

from sklearn.model_selection import cross_val_score,
cross_val_predict
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    '''
    print the accuracy score, classification report and confusion
matrix of classifier
    '''
    if train:
        '''
        training performance
        '''
        print("Train Result:\n")
        print("accuracy score:
{0:.4f}\n".format(accuracy_score(y_train, clf.predict(X_train))))
        print("Classification Report: \n
{}\n".format(classification_report(y_train, clf.predict(X_train))))
        print("Confusion Matrix: \n
{}\n".format(confusion_matrix(y_train, clf.predict(X_train))))
        """
        res = cross_val_score(clf, X_train, y_train, cv=10,
scoring='accuracy')
        print("Average Accuracy: \t {0:.4f}".format(np.mean(res)))
        print("Accuracy SD: \t\t {0:.4f}".format(np.std(res)))
        print('\n')
        """
    elif train==False:
        '''
        test performance
        '''
        print("Test Result:\n")
        print("accuracy score:
{0:.4f}\n".format(accuracy_score(y_test, clf.predict(X_test))))
        print("Classification Report: \n
{}\n".format(classification_report(y_test, clf.predict(X_test))))
        print("Confusion Matrix: \n
{}\n".format(confusion_matrix(y_test, clf.predict(X_test))))


from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.preprocessing import normalize

test_date = '2020-01-01'
y = spy['Signal'][spy.index.isin(df_data.index)]
X = df_data[df_data.index.isin(spy.index)]

result = pd.DataFrame()
result['Real'] = spy['Signal'][spy.index.isin(df_data.index)]
result['MLP'] = 0
```

```
result['RF'] = 0
result['SVM'] = 0
result['KNN'] = 0


#Train Part


#x_train = normalize(X.loc[X.index < test_date],axis=1)
x_train = X.loc[X.index < test_date]
y_train = y.loc[y.index < test_date]
#x_test = normalize(X.loc[X.index > test_date],axis=1)
x_test = X.loc[X.index > test_date]
y_test = y.loc[y.index > test_date]


mlp = MLPClassifier(hidden_layer_sizes = (10,),
                    activation = 'identity',
                    solver ='lbfgs',
                    alpha = 0.001)
mlp.fit(x_train, y_train)
result['MLP'].loc[result.index < test_date] = mlp.predict(x_train)


forest = RandomForestClassifier(max_depth =  5,
                                max_features = 'log2',
                                min_samples_leaf = 10,
                                min_samples_split = 7,
                                n_estimators = 20)
forest = RandomForestClassifier()
ada_rf = AdaBoostClassifier(base_estimator=forest, n_estimators=100,
learning_rate=0.5)
ada_rf.fit(x_train, y_train.ravel())
#forest.fit(x_train, y_train)
result['RF'].loc[result.index < test_date] = ada_rf.predict(x_train)


svc = SVC()
svc.fit(x_train, y_train)
result['SVM'].loc[result.index< test_date] = svc.predict(x_train)


knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
result['KNN'].loc[result.index< test_date] = knn.predict(x_train)


print('MLP :')
print_score(mlp, x_train, y_train, x_test, y_test, train=True)
print_score(mlp, x_train, y_train, x_test, y_test, train=False)
print('RF with Ada :')
print_score(ada_rf, x_train, y_train,x_test, y_test, train=True)
print_score(ada_rf, x_train, y_train,x_test, y_test, train=False)
print('SVC :')
print_score(svc, x_train, y_train,x_test, y_test, train=True)
print_score(svc, x_train, y_train,x_test, y_test, train=False)
print('KNN :')
print_score(knn, x_train, y_train,x_test, y_test, train=True)
print_score(knn, x_train, y_train,x_test, y_test, train=False)


# Test Part
```

```
for i in y.loc[y.index >= test_date].index :

        x_train = X.loc[X.index < i]
        y_train = y.loc[y.index < i]
        x_test = X.loc[X.index == i]
        y_test = y.loc[y.index == i]

        mlp = MLPClassifier(hidden_layer_sizes = (10,),
                            activation = 'identity',
                           solver ='lbfgs',
                            alpha = 0.001)

        mlp.fit(x_train, y_train)
        result['MLP'].loc[result.index == i] =
mlp.predict(x_test)[0]

        forest = RandomForestClassifier(max_depth =  5,
                                        max_features = 'log2',
                                        min_samples_leaf = 10,
                                        min_samples_split = 7,
                                        n_estimators = 20)

        ada_rf = AdaBoostClassifier(base_estimator=forest,
n_estimators=10, learning_rate=0.5)
        ada_rf.fit(x_train, y_train.ravel())
        #forest.fit(x_train, y_train)
        result['RF'].loc[result.index == i] =
ada_rf.predict(x_test)[0]

        svc = SVC()
        svc.fit(x_train, y_train)
        result['SVM'].loc[result.index == i] =
svc.predict(x_test)[0]

        knn = KNeighborsClassifier()
        knn.fit(x_train, y_train)
        result['KNN'].loc[result.index == i] =
knn.predict(x_test)[0]


#GridSearch for RF

from sklearn.model_selection import GridSearchCV

x_train =X.loc[X.index < test_date]
y_train = y.loc[y.index < test_date]

forest = RandomForestClassifier()

params_grid = {'n_estimators' : [10,20,50,100,200],
               'max_depth' : [5,10,15,20],
               'min_samples_split' : [2,5,7,10,20],
               'min_samples_leaf' : [1,3,5,7,10],
               'max_features' : ["auto", "sqrt", "log2"]}

grid_search = GridSearchCV(forest, params_grid,
                           n_jobs=-1, cv=5,
                           verbose=1, scoring='accuracy')
```

```
grid_search.fit(x_train, y_train)

print(grid_search.best_score_)

grid_search.best_estimator_.get_params()

#GridSearch for MLP

from sklearn.model_selection import GridSearchCV

x_train =X.loc[X.index < test_date]
y_train = y.loc[y.index < test_date]

mlp_grid = MLPClassifier()

params_grid =
{'hidden_layer_sizes':[(10,),(100,),(20,),(200,),(50,),(500,)],
                'activation':['identity', 'logistic', 'tanh',
'relu'],
                'solver':['lbfgs', 'sgd', 'adam'],
                'alpha':[0.001,0.0001,0.00001]}

grid_search = GridSearchCV(mlp_grid, params_grid,
                           n_jobs=-1, cv=5,
                           verbose=1, scoring='accuracy')

grid_search.fit(x_train, y_train)

print(grid_search.best_score_)

grid_search.best_estimator_.get_params()

def CLF_Report (Model,y_test,y_model):
    print(Model)
    print("accuracy score:
{0:.4f}\n".format(accuracy_score(y_test,y_model)))
    print("Classification Report: \n
{}\n".format(classification_report(y_test,y_model)))
    print("Confusion Matrix: \n
{}\n".format(confusion_matrix(y_test,y_model)))

y_test = result['Real'].loc[y.index > test_date]
rf_test = result['RF'].loc[y.index > test_date]
mlp_test = result['MLP'].loc[y.index > test_date]
svc_test = result['SVM'].loc[y.index > test_date]
knn_test = result['KNN'].loc[y.index > test_date]

CLF_Report('RF',y_test, rf_test)
CLF_Report('MLP',y_test, mlp_test)
CLF_Report('SVM',y_test, svc_test)
CLF_Report('KNN',y_test, knn_test)


result['past Real'] = result['Real'].shift(1)
result.dropna(axis=0,inplace=True)


#Final model with MLP
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.neural_network import MLPClassifier

#Train
y = result['Real']
X = result[['RF','MLP','SVM','past Real']]

x_train = X.loc[X.index < test_date]
y_train = y.loc[y.index < test_date]
x_test = X.loc[X.index > test_date]
y_test = y.loc[y.index > test_date]

result['Final'] = 0

#knn = KNeighborsClassifier()
#knn.fit(x_train, y_train)
#result['Final'].loc[result.index < test_date] =
knn.predict(x_train)

#mlp_clf = MLPClassifier()
#mlp_clf.fit(x_train, y_train)
#result['Final'].loc[result.index < test_date] =
mlp_clf.predict(x_train)

forest_clf = RandomForestClassifier()
ada_rf = AdaBoostClassifier(base_estimator=forest_clf,
n_estimators=10, learning_rate=0.5)
ada_rf.fit(x_train, y_train.ravel())
result['Final'].loc[result.index < test_date] =
ada_rf.predict(x_train)

y_train = result['Real'].loc[y.index < test_date]
final_train = result['Final'].loc[y.index < test_date]
CLF_Report('Final Train',y_train, final_train)

#Test
for i in y.loc[y.index >= test_date].index :
        x_train = X.loc[X.index < i].values
        y_train = y.loc[y.index < i]
        x_test = X.loc[X.index == i].values
        y_test = y.loc[y.index == i]

        #knn = KNeighborsClassifier()
        #knn.fit(x_train, y_train)
        #result['Final'].loc[result.index == i] =
knn.predict(x_test)[0]

        #mlp_clf = MLPClassifier()
        #mlp_clf.fit(x_train, y_train)
        #result['Final'].loc[result.index == i] =
mlp_clf.predict(x_test)[0]

        forest_clf = RandomForestClassifier()
        ada_rf = AdaBoostClassifier(base_estimator=forest_clf,
n_estimators=10, learning_rate=0.5)
        ada_rf.fit(x_train, y_train.ravel())
        result['Final'].loc[result.index == i] =
ada_rf.predict(x_test)[0]
```

```
y_test = result['Real'].loc[y.index > test_date]
final_test = result['Final'].loc[y.index > test_date]
CLF_Report('Final Test :' , y_test , final_test)


x_test = normalize(X.loc[X.index > test_date],axis=0)
y_test = y.loc[y.index > test_date]

Return = pd.DataFrame()
Return['Return'] = spy['Return'].loc[y_test.index]
Return['Buy&Hold'] = (1 + Return['Return']).cumprod()
Return['Buy&Hold'].iloc[0] = 1

Return['model'] = 1
for idx,val in enumerate(y_test.index) :
    if idx==0:
        Return['model'].iloc[idx] = 1
    else:
        if result['Final'].loc[val] == 0 :
            Return['model'].loc[val] = Return['model'].iloc[idx-1] +
spy['Next Return'].loc[val]
        if result['Final'].loc[val] == 1 :
            Return['model'].loc[val] = Return['model'].iloc[idx-1] +
spy['Next Return'].loc[val]
        if result['Final'].loc[val] == -1 :
            Return['model'].loc[val] = Return['model'].iloc[idx-1]

plt.figure(figsize=(20,10))
plt.plot(Return['model'],label='model')
plt.plot(Return['Buy&Hold'],label='Buy&Hold')
plt.legend()
plt.grid()
plt.show()


chart = pd.DataFrame()
chart['Price'] = spy['Adj Close'].loc[spy.index > test_date]
chart['Signal'] = result['Final'].loc[result.index > test_date]

chart['Buy'] = chart['Price'].loc[chart['Signal']==1]
chart['Sell'] = chart['Price'].loc[chart['Signal']==-1]
chart['Hold'] = chart['Price'].loc[chart['Signal']==0]

plt.figure(figsize=(20,10))
plt.plot(chart['Price'],c='gray')
plt.scatter(chart.index,chart['Hold'],c='gray',linewidths=0.5)
plt.scatter(chart.index,chart['Buy'],c='green',linewidths=0.5)
plt.scatter(chart.index,chart['Sell'],c='red',linewidths=0.5)
plt.grid()
plt.show()
```