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

آبان ماه 1400

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

به نام خدا

تعهدنامه اصالت اثر

تاریخ:

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

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 استفاده از الگوریتم‌های جنگل تصادفی، ماشین بردار پشتیبان، شبکه عصبی و نزدیک‌ترین همسایه و ترکیب آن‌ها با استفاده از متاالگوریتم بگینگ، 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 2015 تا پایان ماه اکتبر سال 2021 بوده است. همچنین به جهت کاهش نویز ورودی به سیستم علاوه بر نرمال‌سازی داده‌ها، از اندیکاتورهای میانگین متحرک وزن‌دار پنج روزه و شاخص قدرت نسبی نیز استفاده شده‌است. 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 2015 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 می شود. 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
ANN	شبکه عصبی مصنوعی
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 دســـتاوردهای علم قابلیت پیش‌بینی بخشـــیدن به متغیرها و پدیده‌ها می باشـــد. پژوهشگران علوم مالی نیز با استفاده از ابزارهای مختلف به دنبال طراحی مدل‌هایی هستند که به وسیله آن متغیرها و حوادث مدنظرشـــان در بازارهای مالی را پیش‌بینی کنند. پیش‌بینی ســـود (جوگ و مک کنومی[1]، 2003) پیش‌بینی ورشکستگی (وندا[2]، 2004) و پیش‌بینی جریانات نقدی (براچت[3]، 2007) از این دست پژوهش‌هاست [1].

به جرات می‌توان گفت که پیش‌بینی قیمت سهام از مهم‌ترین اهداف در علوم مالی و سرمایه‌گذاری است (هاموز[4]، 2015). امروزه پیش‌بینی قیمت ســـهام نه تنها بســـیار چالش برانگیز، بلکه بســـیار 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]، 2005). دوم آنکه با استفاده از روش‌های پیش‌بینی مطلوب می‌توان بازدهی را در سطح مشخصی از ریسک افزایش داد و بالعکس (کیومر[6]، 2006) [1].

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 ســـهام، اهمیت زیادی دارند و شـــاید مهم‌ترین دلیل توجه روزافزون به آن‌ها، این نکته باشـــد 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]، 2011) [2].

---

[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، 1393) [2].

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 بهینه سازی و هوش مصنوعی سعی در ایجاد یک سیستم معاملاتی خودکار داشته است، می‌توان به اسکابار و کلوته[1] (2002) اشاره کرد [3].

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 پیشنهاد شده در مرحله اول سهم‌های دارای بیشترین تاثیر انتخاب می شوند، سپس 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[1] 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[2](که در آن با استفاده از داده‌های ورودی و برچسب آن‌ها، مدل ساخته می‌شود) و گام دسته‌بندی[3] (که در آن از مدل ساخته شده در گام اول جهت پیشگویی برچسب داده‌های جدید استفاده می‌شود). به دلیل آن که برچسب کلاس هر یک از ردیف‌های آموزشی مشخص شده‌اند، این گام همچنین به عنوان یادگیری با ناظر[4] شناخته می شوند. در مقابل یادگیری بی‌ناظر[5] (یا خوشه‌بندی) است، که در آن برچسب کلاس تاپل[6]های آموزشی شناخته شده نیست و ممکن است تعداد یا مجموعه دسته‌هایی که در نهایت به دست خواهند آمد نیز از قبل مشخص نباشند [4].

2-1- شبکه عصبی (ANN)

الگوریتم ANN[7] 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[8]، 2011). از این رو، هر نورون می‌تواند به عنوان یک پردازنده دیده شود که محاسبه ساده ای انجام

---

[1] Classifier

[2] Learning Step

[3] Classification Step

[4] Supervised Learning

[5] Un Supervised Learning

[6] Tuple(در لغت به معنای «چندتایی» بوده و در جایگاه—های مختلف معانی متفاوتی دارد و این در حالی است در زبان—هایی مانند پایتون، لیسپ و غیره، تاپل به مجموعهٔ مقادیری گفته می‌شود که به صورت مرتب شده پشت سر هم آمده باشند.)

[7] Artificial Neural Network

[8] Gorunescu

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[1]، 2008).

مدل‌های ANN دارای سه ویژگی اصلی هستند:

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

2) غیر خطی بودن قابل تمایز در فعالیت عصبی.

3) 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[2] نیز می‌نامند. الگوریتم BPN شبکه را برا ساس 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[3]: 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.

[1] Tunçhan

[2] error Back Propagation Neural Network (BPN)

[3] Forward Pass

حرکت رو به عقب[1]: 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[2]، 2007، مجومدر و هوسیان[3]، 2007، مانتری و همکاران[4]، 2014).

BPN راه حل‌هایی را برای چندین مسئله خطی و غیر خطی مانند طبقه‌بندی، کنترل گیاه، پیشگویی، پیش‌بینی و علم رباتیک شناسایی می‌کند (مهرآرا و همکاران، 2010) [5].

2-2- ماشین بردار پشتیبان (SVM)

SVM[5] برای اولین بار توسط کورتس و وپنیک[6] (1995) 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[7]، 2004).

الگوریتم SVM به کمک یک نگاشت غیرخطی داده‌های آموزشی اولیه را به یک بعد بالاتر تبدیل می‌کند. در این بعد جدید به دنبال ابرصفحه‌ای[8] 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 همیشه می‌توان به کمک یک ابرصفحه تفکیک نمود. الگوریتم SVM این ابرصفحه را با کمک

[1] Backward Pass

[2] Quah

[3] Majumder & Hussian

[4] Mantri et al

[5] Support Vector Machine

[6] Cortes & Vapnik

[7] Smola & Schölkopf

[8] Hyperplane

بردارهای پشتیبان[1] (که اساسا تاپل‌های آموزشی هستند) و حاشیه‌ها[2] (که با کمک بردارهای پشتیبان تعریف می‌شوند) پیدا می‌کند.

اگر داده‌ها به صورت خطی غیر قابل تفکیک باشند، 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 ... xn} بردارهای ورودی و {y1… 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 برمی‌گرداند (جونگ و رجیا[3]، 2008). اشکال مختلفی برای K(x, y) در ادبیات وجود دارد (نصرآبادی، 2007). 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

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

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.

[1] Support Vectors

[2] Margine

[3] Jung & Reggia

SVM همچنین با اســـتفاده از رویکردهای "یکی در برابر یکی" یا "یکی علیه همه" برای مســـائل چند طبقه‌ای[1] اعمال می‌شود (مهرآرا و همکاران، 2010) [5].

2-3- درخت تصمیم (DT)

یک درخت تصمیم[2] 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 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 1970 و اوایل سال‌های 1980، محققی در حوزه‌ی یادگیری ماشین، الگوریتم ID3 را برای استقراء درخت تصمیم طراحی نمود. این کار تعمیم کارهای قبلی بر روی سیستم‌های یادگیری مفهوم[3] 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 1984، گروهی از پژوهشگران در حوزه‌ی آمار کتابی را با نام درختان رگرسیون و دسته‌بندی[4] (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 [4].

[1] multi-class

[2] Decision Tree

[3] Concept Learning Systems

[4] Classification and Regression Trees

2-4- یادگیرنده‌های کُند[1] (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[2]، یادگیرنده‌های کند کار کمتری را هنگام آموزش و کار بیشتری در هنگام دسته‌بندی یا پیشگویی عددی انجام می‌دهند. این در حالی است که در یادگیرنده‌های مشتاق، قبل از دریافت یک تاپل جدید برای دسته‌بندی، با کمک مجموعه تاپل‌های آموزشی یک مدل کلی ساخته می‌شود.

ساختار داده‌ها در عملکرد آن‌ها تقریبا بی‌تاثیر است و به طور طبیعی از یادگیری افزایشی[3] پشتیبانی می‌کند که در آن داده‌های ورودی به طور مداوم برای گسترش دانش مدل موجود و آموزش بیشتر مدل استفاده می شود. هدف از یادگیری افزایشی این است که مدل یادگیری با داده های جدید سازگار شود بدون اینکه دانش موجود خود را فراموش کند. همچنین 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 [4].

2-4-1- دسته‌بندی‌های k همسایه‌ی نزدیک[4]

روش k نزدیک‌ترین همسایه برای اولین بار در اوایل سال‌های 1950 معرفی شد. این روش در مواجهه با داده‌های آموزشی حجیم با حجم عملیات بالایی روبه‌رو بود و به همین دلیل تا سال‌های 1960 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 داده آزمایشی و داده‌های آموزشی مشابه با آن صورت می‌گیرد. داده‌های آموزشی با کمک n صفت مشخص توصیف می‌شوند و هر داده در واقع نمایش نقطه‌ای در فضای n بعدی است. بدین ترتیب تمام داده‌های

[1] Lazy Learners

[2] Eager Learners

[3] Incremental Learning

[4] k-nearest-neighbor

آموزشی در یک فضای n بعدی ذخیره می‌شوند. هرگاه با یک داده ناشناخته روبه‌رو می‌شوید، دسته‌بند k نزدیک‌ترین همسایه به دنبال k تاپل آموزشی است که شبیه‌ترین داده‌ها به داده ناشناخته هستند. این k داده آموزشی k همسایه‌ی نزدیک داده 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 [4].

2-5- 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.

2-5-1- 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 می شود. به دلیل آن که از نمونه گیری با جایگزینی ا ستفاده می شود، ممکن ا ست 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 د سته‌بندهای M_i برچ سب کلاس پی شنهادی 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 [4].

2-5-2- 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 کرد 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 [4].

2-5-3- جنگل‌های تصادفی (RF)

در این بخش به سراغ روشی دیگر از روش‌های تلفیقی به نام جنگل‌های تصادفی[1] 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 می شوند. 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 [4].

[1] Random Forest

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 1388). 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]، 2004). 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]، 1991).

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 به این دلیل که احتمال پایین و شدت بالایی دارد، امکان پذیر نیست (نیکلاس تالب[3]، 2012).

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]، 2007)

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، 1392).

با این حال گروهی از پژوهشگران با توجه به نتایج حاصل از تحقیقات شان پیش‌بینی در بازارهای مالی و قیمت سـهام را امری بسـیار دشـوار ولی امکان پذیر می‌دانند (مالکی[2]، 2003). 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، 2005). کشـف و بهبود الگوریتم‌های پیش‌بینی کننده و امکان انجام محاسـبات پیچیده به وسیله رایانه‌ها، پژوهشگران را در این امر دشوار یاری می‌رساند [1].

برای پیش‌بینی قیمت سـهام راه‌های متفاوتی وجود دارد. یک راه کاهش پیچیدگی با اسـتخراج بهترین صفات یا انتخاب از بین ویژگی‌ها می‌باشد [6]. 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 (2000) از شـــبکه عصـــبی با دو لایه پنهان برای پیش‌بینی معاملات آتی یورو/دلار با بهره گیری از بالاترین قیمت، پایین‌ترین قیمت، قیمت باز گ شایی و قیمت ب سته شدن، ا ستفاده کردند [7].

تیلاکاراتن[2] و همکاران (2007) 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. نویسندگان دریافتند که شبکه‌های عصبی رو به جلو[4] عملکرد بهتری نسبت به شبکه‌های احتمالی[5] دارند [8].

تاکور و کومار[6] (2017) روشـــی جدید از ترکیب الگوریتم‌های ماشـــین بردار پشـــتیبان وزنی و جنگل تصادفی برای ایجاد سیگنال‌های خرید / نگهداری / فروش ارائه کرده اند به گونه ای که در ابتدا با استفاده 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 [9].

پژوهش فونته و همکاران[7] (2006) 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 (1394) اشاره کرد. در این پژوهش با اســتفاده از اطلاعات اندیکاتورهای تکنیکال به پیش‌بینی روند قیمت ســهم و تعیین میزان خریدنی، فروختنی و یا نگهداشتنی 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 (1395) 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 [3].

فلاح پور و علی پور (1393) 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 [2].

مشاری و همکاران (1396) با کمک الگوریتم ژنتیک به بهینه سازی متغیرهای پژوهش پرداخته و سپس مدلی جهت پیش‌بینی نقاط طلایی ارائه نموده اند [1].

سزار و همکارش (2018) 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 15 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 [10].

فصل چهارم

روش‌شناسی پژوهش

4-1- تعریف مساله

این پژوهش از نظر رویکرد کاربردی بوده و مبتنی بر پژوهش‌های میدانی اســـت. هدف این پژوهش ارائه مدل پیش‌بینی جدیدی برای شاخص S&P500 می‌باشد.

همانطور که در پژوهش تاکور و کومار[1] (2017)، در ابتدا توســـط الگوریتم جنگل تصـــادفی به انتخاب ویژگی‌های مهم پرداخته شده‌ا ست، این پژوهش نیز در ابتدا تو سط الگوریتم جنگل ت صادفی صد سهام دارای بی‌شترین اهمیت در پی‌شبینی را از میان پان‌صد سهم موجود در شاخص S&P500 انتخاب می‌کند. در مرحله بعد تلاش شـــده اســـت با اســـتفاده از ترکیب چند الگوریتم، نتیجه مطلوب‌تری ارائه شـــود. مدل ترکیبی مدنظر در ابتدا با اســـتفاده از هر یک از الگوریتم‌های MLP، RF، SVM و KNN به پیش‌بینی تغییر قیمت در روز آتی پرداخته و در مرحله آخر، نتایج به د ست آمده از الگوریتم‌های مذکور را با ا ستفاده از متاالگوریتم بگینگ ترکیب نموده و بر اســـاس رای اکثریت الگوریتم‌ها، ســـیگنال نهایی برای روز آتی را تولید می‌کند. لازم به ذکر ا ست که پارامترهای هریک از الگوریتم‌های مورد ا ستفاده، با ا ستفاده از الگوریتم اعتبار سنجی متقابل[2] جست‌وجوی شبکه‌ای[3] در کتابخانه Scikit learn زبان برنامه نویسی پایتون، تعیین شده است.

4-2- مراحل اجرای پژوهش

4-2-1- مرحله اول: شناسایی و جمع‌آوری متغیرهای مسئله

اطلاعات قیمت پایانی[4] و حجم معاملات روزانه مربوط به 500 ســـهام موجود در شـــاخص S&P500 و قیمت نماد SPY در بازه زمانی ابتدای ســـال 2015 تا پایان ماه اکتبر ســـال 2021 از طریق کتابخانه مالی

[1] Manoj Thakur & Deepak Kumar

[2] Cross Validation

[3] Grid Search

[4] Adjusted Close

یاهو[1] در زبان برنامه‌نویسی پایتون[2] استخراج گردید و در راستای کاهش نویز ورودی به مدل، با استفاده از این اطلاعات اندیکاتورهای شاخص قدرت نسبی (14روزه)[3] و میانگین متحرک وزنی (پنج روزه)[4] نیز، همانطور که در پژوهش آدریان (2011)[5] توصیه شده است، برای هر یک از نمادها در بازه‌ی زمانی ذکر شده محاسبه شد. همچنین به دلیل تفاوت در مقیاس داده‌ها، در انتها تمامی داده‌ها نرمال می‌شوند.

4-2-2- مرحله دوم: انتخاب صد متغیر دارای بیشترین اهمیت

در این مرحله با استفاده از قابلیت تعیین اهمیت متغیرها[6] در الگوریتم جنگل تصادفی، ابتدا با استفاده از چهار ورودی مختلف که عبارت‌اند از قیمت پایانی، حجم معاملات، شاخص قدرت نسبی (14روزه) و میانگین متحرک وزنی (پنج روزه)، مدلی جهت تعیین اهمیت هر یک از 500 سهام در پیشبینی شاخص ایجاد شده و بر اساس این مدل صد سهام دارای بیشترین اهمیت در پیش‌بینی، انتخاب می شوند. سپس پایگاه داده ورودی مدل اصلی که شامل قیمت پایانی، حجم معاملات، شاخص قدرت نسبی و میانگین متحرک وزنی پنج روزه برای صد سهم انتخاب شده، است، تشکیل می‌شود.

4-2-3- مرحله سوم: پیش‌بینی و ارزیابی اولیه

در این گام الگوریتمی برای انتخاب بازه‌های زمانی تصادفی پانصد روزه ایجاد می‌شود، به گونه‌ای که چهارصد روز ابتدایی هر بازه زمانی به عنوان داده آموزش[7] و صد روز پایانی بازه به عنوان داده آزمایش[8] درنظر گرفته می‌شود. در ادامه هر یک از الگوریتم‌های MLP، RF، SVM و KNN توسط داده‌های آموز شی برای پیش‌بینی سیگنال روز آتی آموزش داده می شوند و نتایج عملکرد هریک از الگوریتم‌ها برای

[1] Yahoo Finance

[2] Python

[3] Relative Strength Index (RSI)

[4] 5 Day Weighted Moving Average (WMA5)

[5] Adrian

[6] Feature Importance

[7] Train

[8] Test

پیش‌بینی تو سط داده‌های آموز شی، با ا ستفاده از ماتریس اختلال[1] و دو معیار دقت[2] و اطمینان[3] سنجیده خواهد شد.

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

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

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

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

4-2-4- مرحله چهارم: پیش‌بینی بر اساس داده‌های آزمایشی و ارزیابی مدل‌ها

در این مرحله هر یک از الگوریتم‌ها بر ا ساس داده‌های مربوط به ابتدای سال 2015 تا یک روز قبل از روز مورد پیش‌بینی، به تولید سیگنال خرید، نگهداری یا فروش می‌پردازند (همانند معامله‌گری که در پایان روز فعلی برای پیش‌بینی سیگنال روز بعد، اطلاعات مربوط به روز جاری را نیز در اختیار خواهد داشت). در پایان نتایج پیش‌بینی هر یک از الگوریتم‌ها با معیار دقت[4] مدل سنجیده می‌شود.

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

[1] Confusion Matrix

[2] Precision

[3] Recall

[4] Accuracy

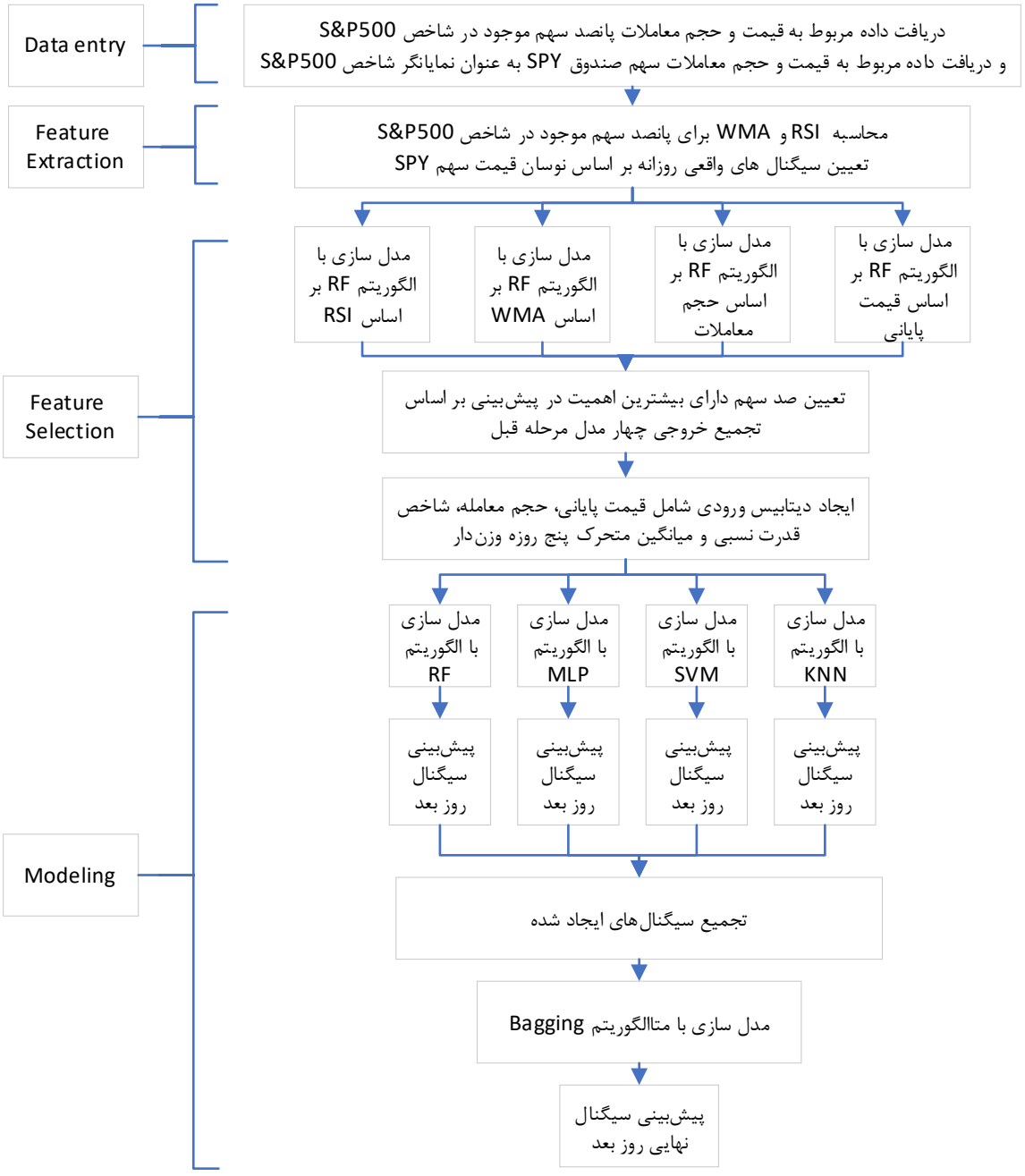
4-2-5- مرحله پنجم: ترکیب نتایج پیش‌بینی الگوریتم‌ها و تولید سیگنال نهایی

در این مرحله نتایج الگوریتم‌های مرحله قبل با اســـتفاده از متاالگوریتم[1] بگینگ[2]، که یک الگوریتم ترکیبی اســـت، ســـیگنال نهایی برای روز آتی را بر اســـاس رای اکثریت تولید می‌کند. به این صـــورت که متاالگوریتم بگینگ ابتدا توســـط خروجی الگوریتم‌های MLP، RF، SVM و KNN در بازه زمانی آموزش، آموزش می‌بیند و سپس در بازه زمانی آزمون با دریافت خروجی مدل‌های مذکور در هر روز، تصمیم‌گیری نهایی را انجام دهد.

شکل ذیل مراحل اجرای مدل را نشان می‌دهد [شکل 4-1]:

[1] Meta-Algorithm

[2] Bagging (Bootstrap aggregating)



شکل 4-1: مدل پیش‌بینی

فصل پنجم

بررسی نتایج

5-1- جزئیات پیاده سازی مدل

جامعه آماری این پژوهش، اطلاعات جمع آوری شـده از قیمت و حجم معاملات 500 سـهام موجود در شـاخص S&P500 در بازه زمانی ابتدای سـال 2015 تا پایان ماه اکتبر سـال 2021 می‌باشـد. این پژوهش در صدد ا ست تا مدل پیش‌بینی جدیدی برای پیش‌بینی قیمت سهام صندوق SPY به عنوان معیار شاخص S&P500 ارائه نماید.

در این مطالعه با توجه به بررسـی‌های موریس[1] در رابطه با توزیع داده‌ها و مرزبندی تعیین سـیگنال‌ها، فرض می‌شـود که 0.5٪ افزایش (یا کاهش) در قیمت پایانی به اندازی کافی معقول اسـت که حرکت مربوطه به عنوان سیگنال خرید (یا فروش) در نظر گرفته شود همچنین درصورتی که تغییر قیمت کمتر از 0.5٪ افزایش (یا کاهش) باشد، سیگنال نگهداری تولید خواهد شد [5].

پارامترهای اصلی الگوریتم‌های MLP و RF توسط قابلیت جستجوی شبکه‌ای[2] کتابخانه سایکیت‌لرن[3] و با اعتبار سنجی متقابل[4] پنجگانه (که بیش از هزار و پانصد حالت مختلف برای پارامترها را بررسی کرده و بهترین پارامترها جهت پیش‌بینی را تعیین می‌کند) انجام شده‌است.

5-2- سنجش عملکرد مدل

برای سنجش مدل، تابعی طراحی شد تا سری های زمانی تصادفی به طول 500 روز کاری انتخاب شوند. سپس چهار صد روز ابتدای هر بازه زمانی به عنوان داده آموزش و صد روز پایانی به عنوان داده آزمایش در نظر گرفته شـده اند. ابتدا داده‌های آموزش به هر یک از چهار مدل MLP، RF، SVC و KNN داده شـده و دقت هریک از مدل‌ها در بازه آموزش سـنجیده شـد. سـپس هریک از مدل‌ها توسـط داده‌های آزمایش بررسی شده و دقت آن‌ها سنجیده شده ا ست. در ادامه خروجی هر یک از چهار مدل مذکور به

[1] S. A. Morris

[2] Grid Search

[3] Scikit Learn

[4] Cross Validation

عنوان ورودی به متاالگوریتم بگینگ داده شده و خروجی آن به عنوان سیگنال نهایی در نظر گرفته شده است که دقت آن نیز مورد بررسی قرار گرفته است.

جدول 5-1: عملکرد الگوریتم‌ها در پیش‌بینی با استفاده از داده‌های آموزشی و آزمایشی در بازه های زمانی مختلف

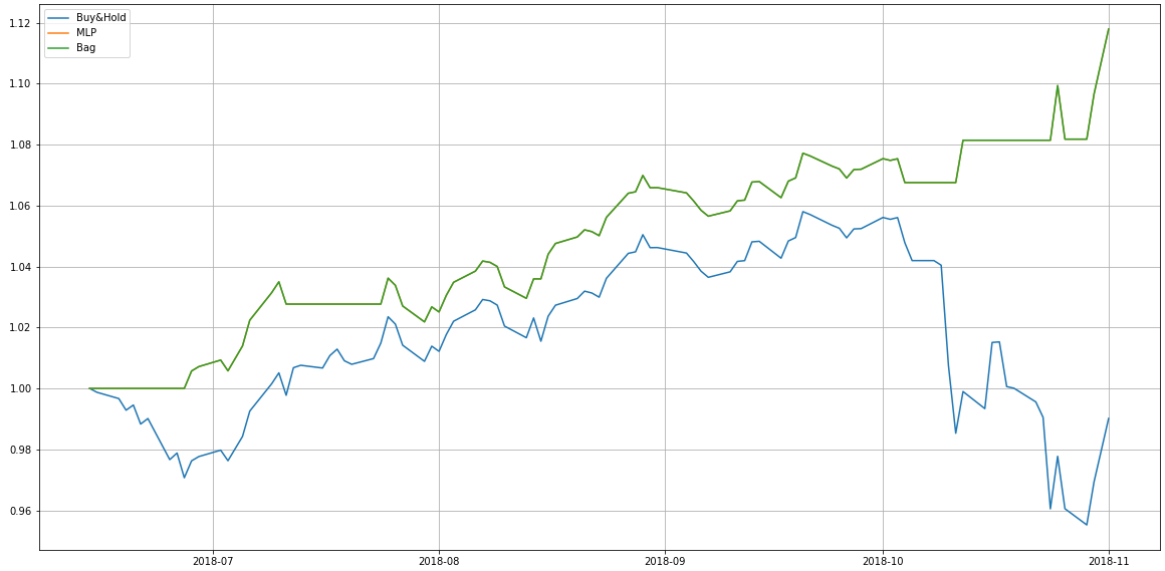
Run	Start	End	Train				Test					Return	
			MLP	RF	SVC	KNN	MLP	RF	SVC	KNN	BAG (Final Model)	Buy&Hold	BAG
1	10/14/2016	10/10/2018	82.75%	100.00%	69.50%	73.25%	61.00%	61.00%	66.00%	55.00%	60.61%	3.00%	6.86%
2	3/21/2018	3/17/2020	63.25%	100.00%	52.25%	67.25%	54.00%	51.00%	57.00%	54.00%	50.51%	-15.50%	-1.75%
3	9/19/2016	9/13/2018	80.00%	100.00%	69.00%	73.00%	56.00%	56.00%	64.00%	51.00%	55.56%	10.81%	11.92%
4	7/2/2019	6/25/2021	67.50%	100.00%	49.75%	61.75%	50.00%	50.00%	53.00%	51.00%	50.51%	11.19%	17.71%
5	11/14/2017	11/11/2019	62.25%	100.00%	53.75%	65.00%	58.00%	57.00%	61.00%	58.00%	56.57%	5.49%	11.68%
6	8/18/2015	8/11/2017	68.50%	100.00%	58.25%	66.00%	76.00%	74.00%	78.00%	78.00%	73.74%	4.82%	3.45%
7	6/28/2019	6/23/2021	62.50%	100.00%	50.25%	62.00%	54.00%	52.00%	53.00%	50.00%	51.52%	11.48%	12.02%
8	11/7/2016	11/1/2018	75.75%	100.00%	69.00%	72.75%	58.00%	52.00%	63.00%	50.00%	51.52%	-0.99%	11.79%
9	2/10/2016	2/5/2018	78.75%	100.00%	69.00%	72.75%	76.00%	69.00%	76.00%	73.00%	68.69%	6.59%	9.89%
10	6/17/2019	6/10/2021	58.75%	100.00%	50.25%	62.00%	50.00%	41.00%	54.00%	47.00%	50.51%	10.70%	12.25%
Avg			**70.00%**	**100.00%**	**59.10%**	**67.58%**	**59.30%**	**56.30%**	**62.50%**	**56.70%**	**56.97%**	**4.76%**	**9.58%**
Std			**8.18%**	**0.00%**	**8.50%**	**4.70%**	**8.97%**	**9.14%**	**8.50%**	**9.90%**	**7.87%**	**7.80%**	**5.17%**
Min			**58.75%**	**100.00%**	**49.75%**	**61.75%**	**50.00%**	**41.00%**	**53.00%**	**47.00%**	**50.51%**	**-15.50%**	**-1.75%**
Max			**82.75%**	**100.00%**	**69.50%**	**73.25%**	**76.00%**	**74.00%**	**78.00%**	**78.00%**	**73.74%**	**11.48%**	**17.71%**

به جهت سنجش بهتر مدل پیشبینی، ده مرتبه و در بازه‌های زمانی متفاوت مدل را اجرا کرده و میانگین، انحراف از معیار، کمترین مقدار و بی شترین مقدار دقت مدل محا سبه شده‌ا ست. همچنین در انتها بازده مدل نیز با استراتژی خرید و نگهداری مقایسه شده است.

5-3- مقایسه بازده

در بازه‌های زمانی متفاوت بازده معاملات بر مبنای مدل پیشبینی با استراتژی "خرید - نگهداری" سهام مورد مقایسه واقع شده است و میانگین، انحراف از معیار، کمترین مقدار و بیشترین مقدار بازده هریک از استراتژی‌ها نیز محاسبه است.

نتایج نشان دهنده این موضوع است که مدل پیشنهادی نه تنها بصورت میانگین بازده بهتری از خود نشان می‌دهد، بلکه انحراف از معیار کمتری نیز در مقایسه با استراتژی خرید و نگهداری از خود به‌جا گذاشته است. در شکل5-1 بازده پیشبینی مدل (سبز رنگ) با بازه استراتژی خرید و نگهداری (آبی رنگ) در صد روز پایانی بازه زمانی 2016/7/11 تا 2018/1/11 (داده آزمایش) مورد مقایسه قرار گرفته است.



شکل 5-1: مقایسه‌ی بازده مدل پیشنهادی و بازده استراتژی خرید و نگهداری

نمودارهای ذیل سیگنال‌های تولید شده توسط مدل را بر روی نمودار قیمت سهام SPY در صد روز پایانی بازه زمانی 2016/7/11 تا 2018/1/11 (داده آزمایش) 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 5-2: سیگنال‌های ایجاد شده توسط مدل در بازه‌ی زمانی آزمون

برر سی سیگنال‌های تولید شده تو سط مدل، بیانگر عملکرد منا سب مدل در مواجهه با تلاطمات قیمت می‌باشد.

فصل ششم

جمع‌بندی، نتیجه‌گیری و پیشنهادات

6-1- 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 متاالگوریتم بگینگ سیگنال نهایی بر پایه سیگنال‌های تولید شده در مرحله قبل ایجاد می‌شود.

نتایج نشان دهنده این موضوع است که مدل پیش‌بینی از دقت مناسبی در تعیین سیگنال‌های روزانه برخوردار است که موجب شده است میزان بازده ناشی از معامله بر اساس سیگنال‌های روزانه تولید شده توسط مدل برای داده‌های آزمون، بیشتر از سود از ناشی استراتژِی "خرید - نگهداری" باشد که بیانگر عملکرد مناسب مدل پیشنهادی است.

6-2- پیشنهادات

پیشنهاد می‌شود برای توسعه‌ی این روش از داده‌های تکمیلی همچون دیگر شاخص‌های بازارسهام ایالات متحده، شاخص بازار سهام دیگر کشورها و اندیکاتورهای پیشرفته‌تر استفاده شود. همچنین می‌توان با استفاده از سایر الگوریتم‌های داده کاوی و استفاده از روش‌های دیگر در جهت تعیین پارامترهای الگوریتم‌ها، عملکرد مدل را بهبود بخشید. همچنین به‌کارگیری روش‌های دیگر متاالگوریتم مثل Boosting ممکن است باعث بهبود نتایج پیش بینی شود.

منابع و مراجع

[1] 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 29، صفحات 66-45، بهار 1398.

[2] درودی، دیاکو؛ ابراهیمی، سید بابک؛ "ارائه‌ی روش هیبریدی نوین برای پیش‌بینی شاخص کل قیمت بورس اوراق بهادار"، *تحقیقات مالی*، دوره 18، شماره 4، صفحات 632-613، زمستان 1395.

[3] 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 28، صفحات 67-49، زمستان 1397.

[4] هان، ژیاوی؛ کمبر، میشلین؛ پی، ژان؛ اسماعیلی، مهدی؛ *داده‌کاوی (مفاهیم و تکنیک‌ها)*، انتشارات نیازدانش، تهران، 1393.

[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.

[11] Moroșan, Adrian. (2011). The relative strength index revisited. African journal of business management. 5. 5855.

پیوست‌ها

جدول پ-1: شرح کد مدل ارائه شده

```
#Import S&P500 Stocks price
resp =
requests.get('http://en.wikipedia.org/wiki/List_of_S%26P_500_companies
')
soup = bs.BeautifulSoup(resp.text, 'lxml')
table = soup.find('table', {'class': 'wikitable sortable'})
tickers = []
for row in table.findAll('tr')[1:]:
    ticker = row.findAll('td')[0].text
    tickers.append(ticker)

tickers = [s.replace('\n', '') for s in tickers]
start = datetime.datetime(2015,1,1)
end = datetime.datetime(2021,11,1)
data = yf.download(tickers, start=start, end=end)


#Import SPY as S&P500 Index
start = datetime.datetime(2015,1,1)
end = datetime.datetime(2021,11,1)
spy = yf.download('SPY', start=start, end=end)


#Preprocesesing Handel nulls
data = data.loc[spy.index]

if data.isnull().values.any() :
    data.fillna(method='backfill',inplace=True)
    data.fillna(method='ffill',inplace=True)
    data.dropna(axis=1,inplace=True)
    print("Total N/A count is : {}".format(data.isnull().sum().sum()))

# Feature extraction
spy['Return'] = spy['Adj Close']/spy['Adj Close'].shift(1) - 1
spy['Next Price'] = spy['Adj Close'].shift(-1)
spy['Next Return'] = spy['Return'].shift(-1)
spy['Cumulative Return'] = (1 + spy['Return']).cumprod()
spy['Signal'] = spy['Next Return'].apply(lambda x : 1 if x>(0.005)
else (-1 if x<-0.005 else 0 ))

#Normalize data
from sklearn.preprocessing import MinMaxScaler

data_scaled = pd.DataFrame()
data_scaled = data.copy()

scaler = MinMaxScaler()
data_scaled.loc[:,:] = scaler.fit_transform(data_scaled)
```

```
#Change daily price to WMA5 & RSI
# 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)

# WMA5 MinMaxScaler
df_WMA5_scaled = pd.DataFrame()
df_WMA5_scaled = df_WMA5.copy()

df_WMA5_scaler = MinMaxScaler()
df_WMA5_scaled.loc[:,:] = df_WMA5_scaler.fit_transform(df_WMA5_scaled)

# RSI MinMaxScaler
df_RSI_scaled = pd.DataFrame()
df_RSI_scaled = df_RSI.copy()

df_RSI_scaler = MinMaxScaler()
df_RSI_scaled.loc[:,:] = df_RSI_scaler.fit_transform(df_RSI_scaled)

#New approach to make Test & Train datasets
def Test_Train_Maker (x,y):

    length = x.index.shape[0]
    rand = int(np.random.rand() * (length - 500))

    x_train = x.iloc[rand:rand+400]
    x_test = x.iloc[rand+401:rand+501]

    y_train = y.iloc[rand:rand+400]
    y_test = y.iloc[rand+401:rand+501]

    return x_train , x_test , y_train , y_test

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

x_train , x_test , y_train , y_test = Test_Train_Maker(X , y)

from sklearn.ensemble import RandomForestClassifier

fi_forest = RandomForestClassifier(random_state=42)
fi_forest.fit(x_train['Adj Close'], y_train)
FI_Adj = pd.DataFrame()
FI_Adj['Stock'] = data['Adj Close'].columns
FI_Adj['Importance'] = fi_forest.feature_importances_

fi_forest = RandomForestClassifier(random_state=42)
fi_forest.fit(x_train['Volume'], y_train)
```

```
FI_Vol = pd.DataFrame()
FI_Vol['Stock'] = data['Volume'].columns
FI_Vol['Importance'] = fi_forest.feature_importances_


fi_forest = RandomForestClassifier(random_state=42)
fi_forest.fit(df_RSI_scaled.loc[x_train.index], y_train)
FI_RSI = pd.DataFrame()
FI_RSI['Stock'] = df_RSI_scaled.columns
FI_RSI['Importance'] = fi_forest.feature_importances_


fi_forest = RandomForestClassifier(random_state=42)
fi_forest.fit(df_WMA5_scaled.loc[x_train.index], y_train)
FI_WMA = pd.DataFrame()
FI_WMA['Stock'] = df_WMA5.columns
FI_WMA['Importance'] = fi_forest.feature_importances_

FI_Total = pd.DataFrame()
FI_Total['Stock'] = data['Adj Close'].columns
FI_Total['Importance'] = 0
for c in FI_Total['Stock'] :
  a = FI_Adj['Importance'].loc[FI_Adj['Stock']== c]
  b = FI_Vol['Importance'].loc[FI_Vol['Stock']== c]
  r = FI_RSI['Importance'].loc[FI_RSI['Stock']== c]
  w = FI_WMA['Importance'].loc[FI_WMA['Stock']== c]
  FI_Total['Importance'].loc[FI_Total['Stock']== c] = a + b + r + w

print('start :',x_train.index[0])
print('end :',x_test.index[-1])

VIP =
FI_Total.sort_values('Importance',ascending=False).reset_index()[:100]
['Stock'].array
Score =
FI_Total.sort_values('Importance',ascending=False).reset_index()[:100]
['Importance'].array
print(VIP)

#Make a new dataset with WMA5 , Vol , RSI
df_data = pd.DataFrame()

df_data = pd.merge(data_scaled['Adj Close'][VIP]
                   ,data_scaled['Volume'][VIP]

,right_index=True,left_index=True,suffixes=('_adj','_vol'))

df_data = pd.merge(df_data
                   ,df_WMA5_scaled[VIP]

,right_index=True,left_index=True,suffixes=('','wma'))

df_data = pd.merge(df_data
                   ,df_RSI_scaled[VIP]

,right_index=True,left_index=True,suffixes=('','_rsi'))

# SPY MinMaxScaler
```

```
SPY_scaled = pd.DataFrame()
SPY_scaled = spy[['Adj Close','Volume']].copy()
SPY_scaler = MinMaxScaler()
SPY_scaled.loc[:,:] = SPY_scaler.fit_transform(SPY_scaled)

df_data = pd.merge(df_data
                   ,SPY_scaled

,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

result = pd.DataFrame()
result['Real'] = spy['Signal'][spy.index.isin(x_train.index)]
result['MLP'] = 0
result['RF'] = 0
result['SVM'] = 0
result['KNN'] = 0

#Train Part

mlp = MLPClassifier()
mlp.fit(x_train, y_train)
result['MLP'] = mlp.predict(x_train)

forest = RandomForestClassifier()
ada_rf = AdaBoostClassifier(base_estimator=forest, n_estimators=100,
learning_rate=0.5)
ada_rf.fit(x_train, y_train.ravel())
result['RF'] = ada_rf.predict(x_train)

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

knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
result['KNN'] = 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

result_test = pd.DataFrame()
result_test['Real'] = spy['Signal'][spy.index.isin(x_test.index)]
result_test['MLP'] = 0
result_test['RF'] = 0
result_test['SVM'] = 0
result_test['KNN'] = 0

for i in y_test.index :

        x_train_loop =
df_data.loc[x_train.index.union(x_test.index[x_test.index < i ])]
        y_train_loop =
y.loc[y_train.index.union(y_test.index[y_test.index < i ])]
        x_test_loop = df_data.loc[x_test.index[x_test.index == i ]]
```

```
        y_test_loop = y.loc[y_test.index[y_test.index == i ]]

        mlp = MLPClassifier()
        mlp.fit(x_train_loop, y_train_loop)
        result_test['MLP'].loc[result_test.index == i] =
mlp.predict(x_test_loop)[0]

        forest = RandomForestClassifier()
        ada_rf = AdaBoostClassifier(base_estimator=forest,
n_estimators=10, learning_rate=0.5)
        ada_rf.fit(x_train_loop, y_train_loop.ravel())
        result_test['RF'].loc[result_test.index == i] =
ada_rf.predict(x_test_loop)[0]

        svc = SVC()
        svc.fit(x_train_loop, y_train_loop)
        result_test['SVM'].loc[result_test.index == i] =
svc.predict(x_test_loop)[0]

        knn = KNeighborsClassifier()
        knn.fit(x_train_loop, y_train_loop)
        result_test['KNN'].loc[result_test.index == i] =
knn.predict(x_test_loop)[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()

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_test['Real']
rf_test = result_test['RF']
mlp_test = result_test['MLP']
svc_test = result_test['SVM']
knn_test = result_test['KNN']

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)

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

#Final model with MLP
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier

result_test['Final'] = 0
result_test['Bag'] = 0

#Train
y_train_final = result['Real']
y_test_final = result['Real']
x_train_final = result[['RF','MLP','SVM','past Real']]
x_test_final = result_test[['RF','MLP','SVM','past Real']]

mlp_clf = MLPClassifier()
mlp_clf.fit(x_train_final, y_train_final)
result_test['Final'] = mlp_clf.predict(x_test_final)

clf = DecisionTreeClassifier(random_state=42)
bag_clf = BaggingClassifier(base_estimator=clf, n_estimators=1000,
                            bootstrap=True, n_jobs=-1,
                            random_state=42)
bag_clf.fit(x_train_final, y_train_final.ravel())
result_test['Bag'] = bag_clf.predict(x_test_final)


CLF_Report('Final(MLP)',result_test['Real'], result_test['Final'])
CLF_Report('Final(BAG)',result_test['Real'], result_test['Bag'])

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

Return['model'] = 1
for idx,val in enumerate(result_test.index) :
    if idx==0:
        Return['model'].iloc[idx] = Return['Buy&Hold'].iloc[0]
    else:
        if result_test['Final'].loc[val] == 1 :
```

```
            Return['model'].loc[val] = Return['model'].iloc[idx-1] +
Return['Return'].loc[val]
        if result_test['Final'].loc[val] == -1 :
            Return['model'].loc[val] = Return['model'].iloc[idx-1]
        if result_test['Final'].loc[val] == 0 :
            for i in range(idx):
                if result_test['Final'].iloc[idx-i] == 1 :
                    Return['model'].loc[val] =
Return['model'].iloc[idx-1] + Return['Return'].loc[val]
                    break
                if result_test['Final'].iloc[idx-i] == -1 :
                    Return['model'].loc[val] =
Return['model'].iloc[idx-i]
                    break


Return['Bag'] = 1
for idx,val in enumerate(result_test.index) :
    if idx==0:
        Return['Bag'].iloc[idx] = Return['Buy&Hold'].iloc[0]
    else:
        if result_test['Bag'].loc[val] == 0 :
            for i in range(idx):
                if result_test['Bag'].iloc[idx-i] == 1 :
                    Return['Bag'].loc[val] = Return['Bag'].iloc[idx-1]
+ Return['Return'].loc[val]
                    break
                if result_test['Bag'].iloc[idx-i] == -1 :
                    Return['Bag'].loc[val] = Return['Bag'].iloc[idx-i]
                    break
        if result_test['Bag'].loc[val] == 1 :
            Return['Bag'].loc[val] = Return['Bag'].iloc[idx-1] +
Return['Return'].loc[val]
        if result_test['Bag'].loc[val] == -1 :
            Return['Bag'].loc[val] = Return['Bag'].iloc[idx-1]

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

plt.legend()
plt.grid()
plt.show()

chart = pd.DataFrame()
chart['Price'] = spy['Adj Close'].loc[result_test.index]
chart['Signal'] = result_test['Bag'].loc[result_test.index]

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()
```