رایانش سریع از طریق ارتقای جنگل تصادفی با استفاده از دو تکنیک فشرده‌سازی و موازی‌سازی

نعیمه محمدکریمی، محمد قاسم‌زاده، مهدی یزدیان دهکردی و امین نظارات

چکیده: در این پژوهش به دنبال ارتقای یکی از الگوریتم‌های کارامد در یادگیری ماشین، به نام جنگل تصادفی هستیم. برای این منظور از تکنیک‌های فشرده‌سازی و موازی‌سازی بهره می‌بریم. چالش اساسی مورد توجه در این پژوهش، در رابطه با به کارگیری جنگل تصادفی در پردازش و تحلیل داده‌های حجیم می‌باشد. در چنین مواردی، این الگوریتم به دلیل مراجعات پرشمار به حافظه، کارایی معمول و مورد نیاز را ندارد. این پژوهش نشان می‌دهد که چگونه می‌توان با به کارگیری یک شیوه فشرده‌سازی ابتکاری، در کنار تکنیک‌های موازی‌سازی به هدف مورد نظر دست یافت. در این رابطه، اجزای مشترک درختان در جنگل تصادفی با یکدیگر به اشتراک گذاشته می‌شوند. علاوه بر این، روش موازی‌سازی مبتنی بر دستورات برداری‌سازی به همراه روش موازی‌سازی مبتنی بر حافظه اشتراکی در جریان پردازش داده‌ها به کار می‌روند. به منظور ارزیابی عملکرد روش پیشنهادی، آن را بر روی مجموعه داده‌های محک Kaggle که در رقابت‌های مربوط به الگوریتم‌های یادگیری به وفور به کار می‌روند، اجرا نمودیم. نتایج به‌دست‌آمده حاکی از آن است که به کارگیری روش فشرده‌سازی پیشنهادی، ٦٦ درصد بهبود در سرعت پردازش داده‌ها به دنبال داشته است. همچنین به کارگیری فشرده‌سازی به همراه موازی‌سازی یادشده، ۹٦ درصد بهبود را به همراه داشته است. به طور کلی نتایج آزمایشی و تحلیل‌ها دلالت بر این دارند که راهکارهای پیشنهادی، قدمی مؤثر در راستای رسیدن به رایانش سریع برای جنگل تصادفی در اختیار می‌گذارد.

کلیدواژه: **یادگیری ماشین، جنگل تصادفی، رایانش سریع، فشرده‌سازی، موازی‌سازی، داده حجیم.**

۱- مقدمه

الگوریتم‌های یادگیری ماشین در حل مسایل مختلف از اهمیت بسزایی برخوردار می‌باشند. با رشد روزافزون حجم داده، این الگوریتم‌ها با چالش‌های جدی همراه می‌شوند، لیکن باید این الگوریتم‌ها به شیوه‌هایی متفاوت از روش‌های سنتی، پردازش شوند. از چالش‌های موجود برای پردازش الگوریتم‌های یادگیری ماشین با کلان داده، پیچیدگی فضایی و زمانی است [۱] تا [۴]. در رایانش سریع و یا به اختصار HPC[1] سعی می‌شود تا با فشرده‌سازی هرچه بهتر داده‌ها در حافظه و همچنین موازی‌سازی الگوریتم‌ها، سرعت و پیچیدگی آنها کاهش داده شود [۵] تا [۷].

به طور کلی در روش‌های مبتنی بر رایانش سریع سه رویکرد مختلف موازی‌سازی مبتنی بر حافظه وجود دارد. رویکرد اول حافظه اشتراکی[2] است که در آن پردازنده‌های مختلف یک کامپیوتر از یک حافظه مشترک برای خواندن و نوشتن استفاده می‌کنند، مانند روش OpenMP[3] [۸]. رویکرد دوم، حافظه توزیع‌شده[4] است که هر سیستم، حافظه مختص به خود را داشته و تبادل داده بین پردازنده سیستم‌ها از طریق ارسال و دریافت پیام انجام می‌شود، مانند روش MPI [۹]. در نهایت رویکرد سوم، حافظه ترکیبی[5] است که از ترکیب دو رویکرد قبلی نشأت می‌گیرد، مانند روش PGAS [۱۰] که از ترکیب OpenMP و MPI استفاده می‌کند.

علاوه بر موازی‌سازی مبتنی بر حافظه، موازی‌سازی در سطح دستورات نیز وجود دارد. کامپایلرها از این رویکرد برای موازی‌سازی استفاده می‌کنند و سعی می‌کنند دستوراتی را که با یکدیگر وابستگی ندارند در یک کلاک (سیکل) اجرا نمایند. از جمله روش‌های موازی‌سازی در سطح دستورات می‌توان به روش‌های محاسباتِ دستورالعمل‌های صریحِ موازی[6]، عدمِ پذیرشِ تقاضا برای تغییر نام ثبات/ اجرا[7]، روش اجرای احتمالی[8] و برداری‌سازی[9] اشاره کرد. در روش محاسبات دستورالعمل‌های صریح موازی، چند دستورالعمل به صورت هم‌زمان قابل اجرا است که هرچه تعداد دستورات غیر وابسته بیشتر باشد، درصد بالاتری از موازی‌سازی ایجاد می‌شود [۱۱]. در روشِ عدمِ پذیرشِ تقاضا برای تغییر نام ثبات/ اجرا، امکان اجرای دستورات بیشتری در یک سیکل وجود دارد که با تغییر نام ثبات‌ها می‌توان به این حالت دست یافت [۱۲]. در روش اجرای احتمالی برای افزایش سرعت اجرا، تمام یا بخشی از دستورات، قبل از این که زمان اجرای ترتیبی آنها فرارسیده باشد، اجرا می‌شوند [۱۳]. روش برداری‌سازی، یکی از حالت‌های خاص مدل Flynn است که در آن یک دستور می‌تواند روی چندین داده به صورت هم‌زمان کار کند [۱۴].

دلیل استفاده از روش‌های مبتنی بر رایانش سریع، سرعت بالا و دقت زیاد آن در محاسبات پیچیده همچون پیش‌بینی وضعیت آب و هوا و محاسبات عددیِ احتمال تصادف خودرو و دیگر زمینه‌های پیچیده محاسباتی است [۱۵]. به عنوان نمونه ضرب دو ماتریس به کمک معماری Xeon Phi از شرکت اینتل و دستورالعمل‌های AVX-۵۱۲، با به کارگیری ۶۸ هسته موازی‌سازی شده و بهبود ۹۰ درصدی نسبت به

این مقاله در تاریخ ۹ مرداد ماه ۱۳۹۸ دریافت و در تاریخ ۱۵ فروردین ماه ۱۳۹۹ بازنگری شد.

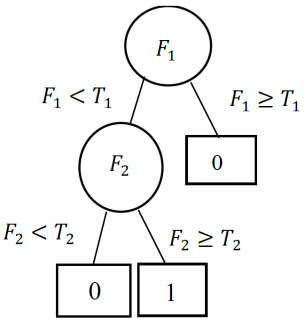
نعیمه محمدکریمی، دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران، (email: n.m.karimi.h.a@stu.yazd.ac.ir).

محمد قاسم‌زاده (نویسنده مسئول)، دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران، (email: m.ghasemzadeh@yazd.ac.ir).

مهدی یزدیان دهکردی، دانشکده مهندسی کامپیوتر، دانشگاه یزد، یزد، ایران، (email: yazdian@yazd.ac.ir).

امین نظارات، دانشکده مهندسی کامپیوتر، دانشگاه ماساریک، برنو، جمهوری چک، (email: nezarat@ics.muni.cz).

1. High Performance Computing
2. Shared Memory
3. Open Multi-Processing
4. Distributed Memory
5. Hybrid Memory
6. Explicitly Parallel Instruction Computing
7. Out of Order Execution/Register Renaming
8. Speculative Execution
9. Vectorization

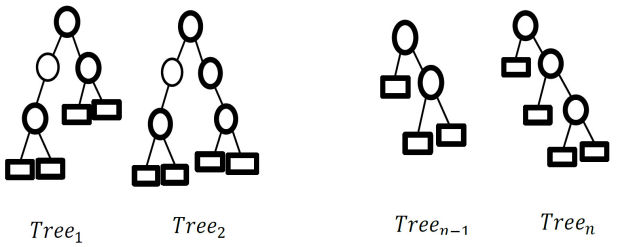


شکل ۱: ساختار درخت.

معماری‌های گذشته حاصل شده است [۱۶]. الگوریتم مرتب‌سازی ادغامی موازی‌سازی با استفاده از n پردازنده، سرعت اجرا را از $O(n \log n)$ به $O(n)$ افزایش داده است [۱۵]. یکی از کاربردهای رایانش سریع در الگوریتم‌های یادگیری ماشین، کار با کلان داده است که باعث افزایش سرعت چند برابری الگوریتم و در مواردی باعث افزایش دقت آنها می‌شود. یادگیری تقویتی نمونه‌ای از این روش‌هاست که با بهره‌گیری از روش‌های مبتنی بر رایانش سریع، زمان حل روابط پیچیده در کاربرد داروسازی، از چندین ساعت به چند دقیقه کاهش داده شده است [۱۷]. الگوریتم k-means یکی از الگوریتم‌های پرکاربرد برای دسته‌بندی داده‌ها در یادگیری ماشین است. بایدون و همکاران [۱۸] با ارائه یک پیاده‌سازی موازی برای k-means، سرعت اجرای آن را به طور چشم‌گیری افزایش دادند. با استفاده از MPI [۱۵]، سرعت اجرای الگوریتم k-means، P (تعداد پردازنده‌ها) برابر افزایش می‌یابد.

یکی از الگوریتم‌های محبوب یادگیری ماشین که برای دسته‌بندی داده‌ها استفاده می‌شود، جنگل تصادفی است [۱۹] تا [۲۱]. این الگوریتم انعطاف‌پذیری خوبی بر روی داده‌ها، به خصوص داده‌های کلان داشته و به داده‌ها، بیش‌برازش نمی‌شود. از جمله کاربردهای جنگل تصادفی می‌توان به دسته‌بندی، رگرسیون، تخمین چگالی و یادگیری manifold اشاره کرد [۲۱]. یک جنگل تصادفی از تعدادی درخت تصمیم تشکیل شده که برگ‌های این درختان برچسب کلاس مربوط به داده را نشان می‌دهند [۲۲]. در هنگام دسته‌بندی یک داده، درخواست‌های زیاد حافظه و رفت و برگشت بین حافظه نهان، حافظه اصلی و دیسک باعث کندشدن زمان پیش‌بینی برچسب داده می‌شود. هر اندازه بتوان تعداد درختان بیشتری را در حافظه نهان جای داد، در نتیجه تعداد خطاهای کش کمتری رخ داده و سرعت اجرای الگوریتم افزایش خواهد یافت.

ابراهیم کامل و همکاران [۲۳] مفهومی با عنوان forest packing ارائه دادند تا با بهینه‌کردن ترتیب قرارگیری گره‌های یک درخت بر روی دیسک، تعداد دفعات مراجعه به دیسک را کاهش دهند. اخیراً در سال ۲۰۱۹ برون و همکاران، روشی ارائه داده‌اند که علاوه بر بهینه‌کردن ترتیب قرارگیری گره‌های درختان در حافظه، از اشتراک‌گذاری گره‌ها برای کاهش حافظه استفاده می‌کنند. هر برگ از یک درخت، یک گره در حافظه به خود اختصاص می‌دهد که برچسب کلاس را نشان می‌دهد و این برچسب بین برگ‌های زیادی مشابه می‌باشد. روش برون و همکاران، درختان را در چند دسته تقسیم کرده و برای هر کلاس یک برچسب در هر دسته در نظر می‌گیرد. در این پژوهش روش Compressed Bin (CBin) برای فشرده‌سازی بیشتر درختان در حافظه پیشنهاد شده است. این روش از ایده اشتراک‌گذاری برچسب‌ها بین تمام Binها استفاده می‌کند تا با فشرده‌سازی بیشتر درختان، امکان قرارگیری تعداد درختان بیشتری در حافظه را فراهم کند. ارزیابی‌های انجام‌شده نشان می‌دهد که



شکل ۲: جنگل تصادفی.

روش پیشنهادی CBin سرعت اجرای الگوریتم جنگل تصادفی را بهبود داده است. علاوه بر این، روش دیگری با نام Parallel CBin (PCBin) برای افزایش سرعت ارائه شده است. در این روش از ایده پیشنهادی فشرده‌سازی در روش CBin به همراه ایده موازی‌سازی مبتنی بر حافظه (OpenMP) و همچنین موازی‌سازی مبتنی بر دستور (برداری‌سازی) استفاده شده است. ارزیابی‌های انجام‌شده نشان می‌دهند که از نظر سرعت اجرای الگوریتم، روش PCBin سریع‌تر از روش CBin نیز عمل کرده است.

این مقاله از پنج بخش تشکیل شده است. در بخش دوم به بررسی کارهای پیشین در زمینه جنگل تصادفی، فشرده‌سازی جنگل تصادفی و موازی‌سازی می‌پردازیم. در بخش سوم، روش پیشنهادی تشریح می‌گردد و در بخش چهارم به ارزیابی نتایج حاصل از آزمایش‌ها و مقایسه با روش پیشین پرداخته می‌شود. در بخش پنجم، نتیجه‌گیری و پیشنهادهایی برای کارهای آینده ارائه خواهد شد.

۲- دانش پس‌زمینه

در این بخش به تشریح مفاهیم پایه‌ای مرتبط با مبحث پژوهشی این مقاله می‌پردازیم. در این رابطه، ابتدا درخت تصمیم و جنگل تصادفی معرفی می‌شوند. در ادامه، پیمایش‌های مختلف در درخت تشریح می‌گردد و سپس روش‌های موازی‌سازی برداری‌سازی و OpenMP، معرفی و مورد بحث قرار می‌گیرند.

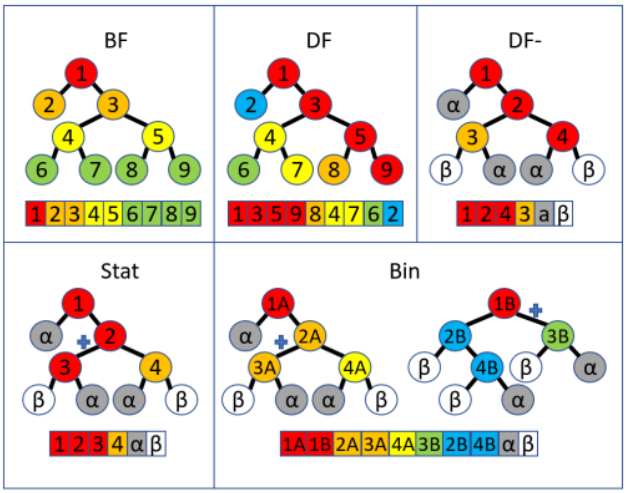
۲-۱ درخت تصمیم و جنگل تصادفی

درخت تصمیم یک طبقه‌بند با ناظر است که با استفاده از داده‌های آموزشی برچسب‌دار، آموزش داده می‌شود [۲۲]. داده‌های آموزشی شامل بردار ویژگی و یک برچسب برای هر نمونه داده می‌باشند. همان طور که در شکل ۱ مشاهده می‌شود، در درخت تصمیم گره‌ها مربوط به ویژگی‌ها بوده و برگ‌ها دربردارنده برچسب کلاس‌ها می‌باشند. برای طبقه‌بندی یک نمونه داده، مقدار ویژگی در هر گره با یک حد آستانه مقایسه شده و نمونه در یکی از شاخه‌ها پیش می‌رود. در نهایت با رسیدن به برگ، برچسب نمونه مشخص می‌شود.

جنگل تصادفی برای اولین بار توسط لئو بریمان ارائه شده است [۲۲]. جنگل از مجموعه‌ای از درخت‌های تصمیم با ساختارهای ساده ساخته شده که هر درخت با انتخاب تصادفی نمونه‌ها از داده‌های آموزشی ساخته می‌شود (شکل ۲). برای طبقه‌بندی، نمونه داده به تمام درخت‌ها داده شده و نتیجه نهایی بر اساس رأی اکثریت آنها مشخص می‌شود.

۲-۲ بررسی درختان با پیمایش‌های مختلف

مشکل گلوگاه در حافظه یکی از عمده‌ترین مشکلات در پردازش داده‌های حجیم است. فشرده‌سازی (Forest Packing)، این امکان را فراهم می‌کند که بتوان تعداد درخت‌های بیشتری را در فضای حافظه



شکل ۳: نمایش مختلف درختان در حافظه [۱۹].

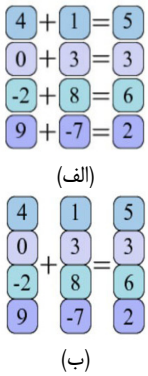
نگهداری کرد. این امر باعث کاهش تعداد خطاهای کش و افزایش سرعت اجرای دستورالعمل‌ها می‌شود.

با توجه به شکل ۳، در یک درخت پیمایش‌های مختلفی وجود دارد که از نظر سرعت دسترسی به برگ با هم متفاوت هستند. پیمایش سطحی (BF) و پیمایش عمقی (DF) برای هر برگ درخت، یک گره مجزا تخصیص می‌دهند (شکل ۳). روش DF^- درخت را به صورت عمقی پیمایش می‌کند و برای فشرده‌سازی داده‌ها، گره برچسب را بین همه برگ‌ها با برچسب یکسان به اشتراک می‌گذارد (شکل ۳). باید در نظر داشت که با این کار ممکن است بهبود تا تقریباً ۵۰ درصد در حافظه حاصل شود زیرا نیمی از داده‌ها در برگ‌ها هستند. روش Stat [۱۹] برای فشرده‌سازی، مشابه DF^- عمل می‌کند با این تفاوت که در ابتدا چند سطح محدود از درخت را به صورت سطحی پیمایش کرده و بعد از انتخاب محتمل‌ترین مسیر برای داده ورودی، بقیه درخت را به صورت عمقی پیمایش می‌کند (شکل ۳).

روش دیگری که اخیراً پیشنهاد شده است، روش Bin می‌باشد [۱۹]. در این روش برای پردازش جنگل در حافظه، درختان در چند دسته تقسیم می‌شوند که به هر دسته یک Bin گفته می‌شود. برای فشرده‌سازی بیشتر در یک Bin، برچسب کلاس‌ها بین درختان آن Bin به اشتراک گذاشته می‌شود تا حافظه کمتری مصرف شود. همچنین برای افزایش سرعت پردازش درختان، هر Bin به صورت موازی پردازش می‌شود. اگر T تعداد کل درختان و B تعداد درخت در هر Bin باشد، آن گاه تعداد Binها برابر با T/B خواهد بود. اندازه بسته‌ها به میزان RAM سیستم بستگی دارد. هر اندازه RAM بزرگ‌تر باشد می‌توان تعداد درختان موجود در یک Bin را افزایش داد. شکل ۳، روش Bin را نشان می‌دهد.

۲-۳ برداری‌سازی

برداری‌سازی حلقه‌ها به معنای اجرای هم‌زمان یک عمل بر روی تعدادی از عناصر می‌باشد که می‌تواند باعث افزایش کارایی برنامه‌ها شود. به عنوان مثال در صورتی که یک آرایه به صورت سریال جمع شود، جمع عناصر به صورت شکل ۴-الف، یکی پس از دیگری در N کلاک (N تعداد عناصر آرایه) انجام می‌شود. در حالی که با برداری‌سازی، جمع عناصر آرایه به صورت یک جا در یک کلاک مشابه شکل ۴- ب انجام خواهد شد.



شکل ۴: برداری‌سازی [۲۴]، (الف) جمع دو آرایه به صورت سریالی و (ب) جمع دو آرایه به صورت موازی.

۴-۲ OpenMP

OpenMP یک رابط برنامه کاربردی است که به منظور موازی‌سازی بین چند پردازنده یا هسته با استفاده از حافظه اشتراکی طراحی شده است [۲۵]. با استفاده از کتابخانه‌های نخ در OpenMP و کامپایلر OpenMP، به راحتی عمل موازی‌سازی حلقه انجام می‌گردد. نخ‌ها می‌توانند وظیفه‌های جدیدی از تکرارهای پردازش‌نشده حلقه به دست آوردند.

مشخصه اصلی موازی‌سازی با OpenMP، بهره‌گیری از حافظه مشترک می‌باشد که در مدت زمان اجرا، داده‌ها بین هسته‌های پردازنده به اشتراک گذاشته می‌شوند (شکل ۵). OpenMP شامل یک مجموعه دستورات کامپایلری است که در محیط‌های فرترن، C و C++ فراخوانی می‌شوند.

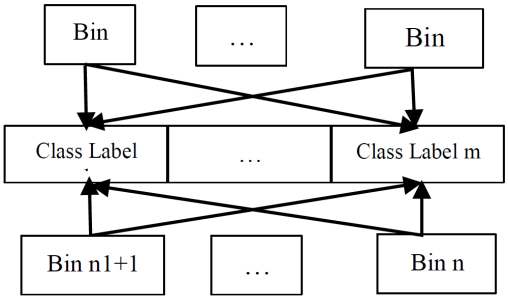
۳- روش‌های پیشنهادی

ابتدا در بخش ۳-۱ روش پیشنهادی Compressed Bin (CBin) که از فشرده‌سازی برای بهبود کارایی روش Bin [۱۹] بهره می‌برد، توضیح داده شده و سپس در بخش ۳-۲ رویکرد موازی‌سازی با عنوان Paralled CBin (PCBin) جهت افزایش سرعت پیش‌بینی در داده‌های تست، توضیح داده شده است.

۱-۳ روش پیشنهادی CBin

در روش CBin از تکنیک فشرده‌سازی برای کاهش مصرف حافظه و افزایش سرعت پیش‌بینی الگوریتم جنگل تصادفی استفاده شده است. در روش برون و همکاران [۱۹] چند درخت در یک دسته (Bin) قرار داده می‌شوند و به جای این که هر برگ درخت شامل یک برچسب کلاس باشد، هر Bin فقط شامل یک برچسب به ازای هر کلاس است. هرچند این رویکرد تعداد برچسب‌ها را کاهش می‌دهد اما برای هر کلاس در هر Bin همچنان ایجاد یک برچسب لازم است. در روش پیشنهادی CBin از این ایده استفاده شده که برای تمام Binها فقط یک برچسب به ازای هر کلاس در نظر گرفته شود. در Error! Reference source not found.، نمای کلی از این روش به تصویر کشیده شده است. یک لیست شامل برچسب کلاس‌ها در مرکز قرار دارد و تمام Binها (که شامل چندین درخت هستند) به صورت ستاره‌ای با این لیست در ارتباط هستند.

با به کارگیری روش CBin میزان حافظه مورد استفاده به اندازه $(\#Bin-1)\times\#class$ نسبت به روش برون و همکاران [۱۹] کاسته شده و زمان پیش‌بینی نیز کاهش داده خواهد شد.



شکل ۵: نمای کلی از روش ارائه‌شده CBin.

جدول ۱: مشخصات پایگاه داده و Random Forest.

	Allstate	Higgs
Training Observations	۵۰۰۰۰۰	۲۵۰۰۰۰
Test Observation	۵۰۰۰۰	۲۵۰۰۰
Number of Features in Dataset	۳۳	۳۰
Avg Number of Internal Nodes	۱۷۸۶۳۲	۴۷۹۱۵
Number of Trees in Forest	۳۲	۳۲

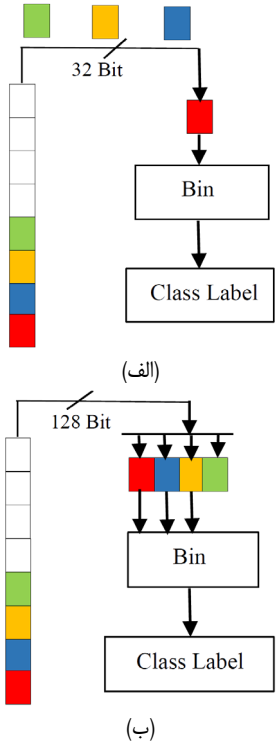
۳-۲ روش پیشنهادی PCBin

در معماری کامپایلر اینتل در حالت پردازش سریالی، تبادل و پردازش اطلاعات در قالب یک کلمه (۳۲بیتی) مشابه شکل ۶- الف انجام می‌شود. با به کارگیری حالت پردازشی برداری، مطابق شکل ۶- ب چهار کلمه به صورت هم‌زمان به هسته پردازشی منتقل شده و سرعت پردازش تا حدودی افزایش می‌یابد.

ابتدا به کمک معماری کامپایلر اینتل و برداری‌سازی، سعی در بهبود سرعت پیش‌بینی الگوریتم جنگل تصادفی داریم. همان طور که در شکل ۶-الف مشاهده می‌شود تبادل و پردازش اطلاعات در قالب یک کلمه است اما در شکل ۶-ب، بر حسب سخت‌افزار موجود بیش از یک کلمه در یک زمان پردازش می‌شود. تمام رخدادهای شکل ۶ در یک هسته رخ می‌دهد. بنابراین به کمک ویژگی‌های موجود در شکل ۶-ب، بهبود حاصل می‌گردد اما به خاطر خاصیت ذاتی درخت‌های تصادفی این بهبود چشم‌گیر نیست زیرا در بیشتر مواقع برای پیمایش درختان در یک Bin، خطاهای کش زیادی رخ می‌دهد و بنابراین بقیه Binها منتظر اتمام کار آن Bin هستند. در روش پیشنهادی PCBin برای دستیابی به سرعت بالا در پیش‌بینی داده‌های تست از ترکیب پردازش برداری و معماری OpenMP [۲۵] استفاده شده است.

در سیستم‌هایی که جریان داده‌ها به صورت آنلاین وارد سیستم می‌شود باید نرخ ورود داده با نرخ پردازش داده، فاصله زیادی نداشته باشد. زیرا سیستم‌های آنلاین دارای صفی هستند که داده‌های ورودی در آن قرار می‌گیرند تا برای پردازش، فراخوانی شوند. اگر سرعت ورود داده‌ها خیلی بیشتر از سرعت پردازش آنها باشد، این صف پر شده و ممکن است داده‌ای گم شود یا سیستم با شکست مواجه شود. بنابراین سرعت در چنین سیستم‌هایی از اهمیت بسزایی برخوردار است. می‌توان به کمک روش‌های نرم‌افزاری و سخت‌افزاری در جهت موازی‌سازی استفاده کرد و سرعت پردازش را افزایش داد. در این مقاله به کمک ترکیب دو روش برداری‌سازی و OpenMP، موازی‌سازی انجام شده است.

اگر تعداد پردازنده‌ها، تعداد داده‌ها و زمان مورد نیاز برای پردازش هر داده به ترتیب با P، n و t نشان داده شود، پیچیدگی زمانی پردازش



شکل ۶: برداری‌سازی، (الف) Scalar Instruction for prediction Bin و (ب) Vector Instruction for prediction Bin.

جدول ۲: اطلاعات سیستمی.

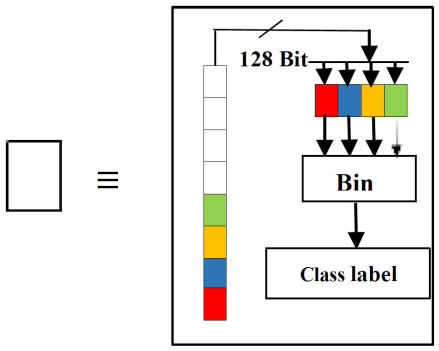
System Tools	Specification
CPU	Intel (R) Xeon (R) CPU X۵۶۷۰ @ ۲٫۹۳ GHz
RAM Size	۳۲ GB
#core	۲۴
Linux	Centos ۶٫۹

این داده‌ها از مرتبه $O(t\times n)$ به مرتبه $O(t\times n/P)$ کاهش خواهد یافت. با افزایش تعداد هسته پردازنده تا حد قابل قبولی در کاهش زمان اجرای الگوریتم، بهبود حاصل می‌شود.

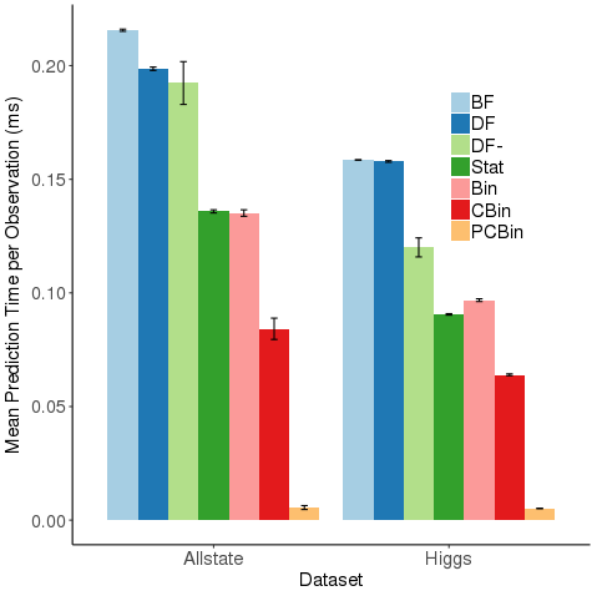
٤- نتایج آزمایش‌ها و تحلیل

برای ارزیابی روش‌ها از دو پایگاه داده Allstate [۲۶] و Higgs [۲۷] از مجموعه داده‌های رقابتی Kaggle که به وفور از این دو پایگاه داده، در پژوهش‌های مختلف استفاده شده و مشخصات این پایگاه داده‌ها در جدول ۱ نشان داده شده است. داده‌ها شامل دو قسمت تست و آموزش می‌باشند که از داده آموزش برای به دست آوردن احتمال پیمایش گره‌ها و آموزش جنگل و از داده تست برای به دست آوردن سرعت پیش‌بینی استفاده شده است. متوسط تعداد گره‌های داخلی و تعداد درختان در هر جنگل، برای هر پایگاه داده نیز در دو سطر انتهایی جدول ۱ نشان داده شده که این پارامترها برای همه روش‌های مقایسه‌شده یکسان می‌باشد. با توجه به سیستم پردازشی موجود (جدول ۲)، تعداد درختان در جنگل تصادفی، ۳۲ در نظر گرفته شده است.

برای ارزیابی منصفانه در شرایط یکسان، تمام پیاده‌سازی‌ها در زبان برنامه‌نویسی C++ انجام شده است [۲۸]. اطلاعات سخت‌افزاری و نرم‌افزاری سیستم که آزمایش‌ها روی آن انجام شده است، در جدول ۲ نشان داده شده است.



شکل ۷: مدل پیشنهادی به کمک OpenMP+Vectorization.



شکل ۸: تأثیر روش‌های مختلف با پایگاه داده‌های مختلف.

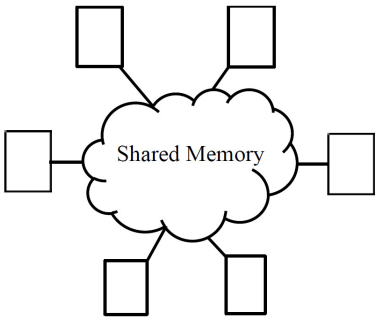
٤-١ مقایسه روش‌ها از نظر زمان پیش‌بینی

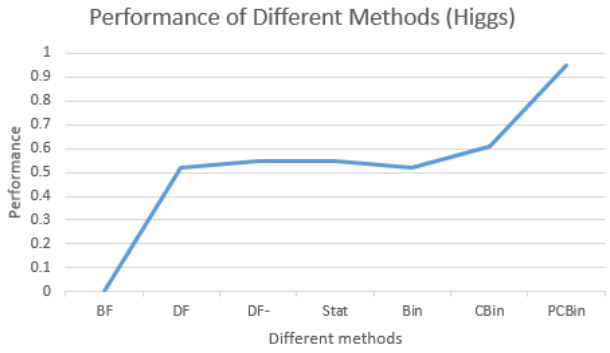
در این بخش دو روش پیشنهادی که با نام‌های Compressed Bin (CBin) و Paralled Compressed Bin (PCBin) نمایش داده می‌شوند با روش‌های پیمایش BF، DF، DF⁻، Stat و Bin که در بخش ۲-۲ در مورد آنها توضیح داده شد، مقایسه می‌شوند. متوسط زمان پیش‌بینی برای روش‌ها در شکل ۷ نشان داده شده است. نتایج نشان می‌دهد که روش‌های پایه BF و DF زمان پیش‌بینی بیشتری نسبت به سایر روش‌ها داشته‌اند. روش پیشنهادی CBin با اشتراک‌گذاری برچسب‌های یکسان به فشرده‌سازی بهتر و در نتیجه زمان پیش‌بینی بهتری نسبت به روش Bin و سایر روش‌ها دست یافته است. به علاوه روش PCBin که از موازی‌سازی نیز بهره می‌برد، زمان پیش‌بینی را به صورت قابل توجهی کاهش داده است.

برای مقایسه بهتر روش‌ها، نسبت بهبود هر روش نسبت به بدترین نتیجه (BF) نیز محاسبه و در شکل ۸ نشان داده شده است. بر روی داده Allstate، روش CBin به بهبود ۶۱ درصد و روش PCBin به بهبود ۹۵ درصد رسیده و بر روی داده Higgs نیز روش CBin به بهبود ۷۱ درصد و روش PCBin به بهبود ۹۷ درصد دست یافته است.

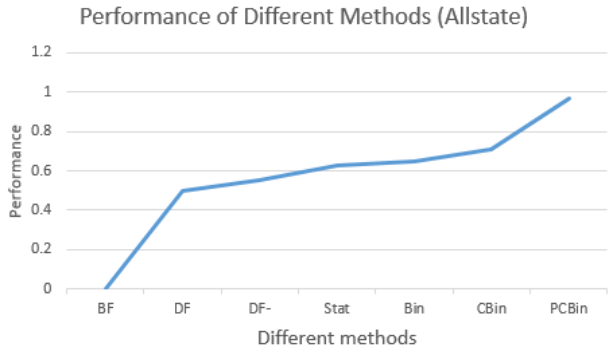
٤-٢ مقایسه زمان تخمین زده شده با زمان اندازه‌گیری‌شده

در این بخش زمان تخمین زده شده [۱۹] برای اجرای روش‌ها و زمان





(الف)



(ب)

شکل ۹: مقایسه نسبت بهبود زمان روش‌ها نسبت به روش BF، (الف) پایگاه داده Higgs و (ب) پایگاه داده Allstate.

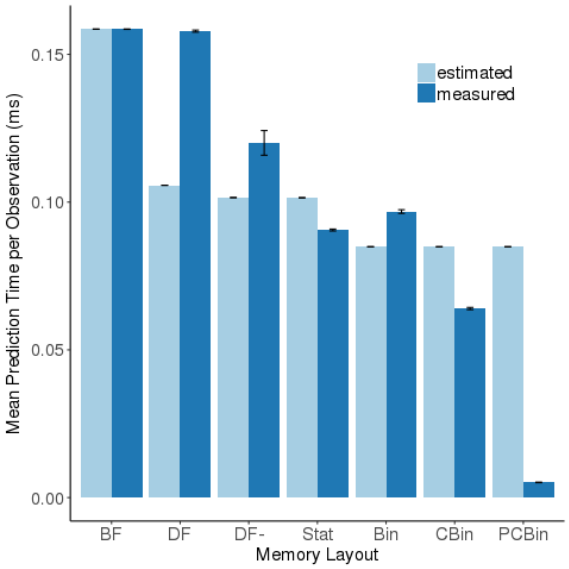
اندازه‌گیری شده در شکل ۹ مقایسه شده‌اند. زمان تخمین زده شده با فرض در اختیار داشتن، یک مدل کش ساده و یکسان‌بودن توزیع ورود داده‌ها به داخل حافظه طبق رابطه زیر محاسبه می‌شود

$$E\{RT\} = \overline{T}_m \times \frac{\overline{d} - \#N}{EU} \qquad (۱)$$

که در این رابطه $E\{RT\}$ متوسط زمان اجرا، $\overline{T}_m$ متوسط زمان خطای کش، $\overline{d}$ متوسط عمق جنگل، EU تعداد منابعی که هم‌زمان به یک کش دسترسی دارند و $\#N$ تعداد گره‌هایی است که انتظار می‌رود دسترسی به آنها در خطاهای کش بعدی مورد نیاز باشد و بنابراین بهتر است در کش، باقی بمانند. $\overline{T}_m$ متوسط زمان خطاهای کش را نشان می‌دهد که از رابطه زیر محاسبه می‌شود [۱۹]

$$\overline{T}_m = \frac{RT_{BF}}{\overline{d}} \qquad (۲)$$

در این رابطه RT_{BF}، زمان اجرای سطحی و $\overline{d}$ متوسط عمق جنگل می‌باشد. با توجه به نتایج شکل ۱۰، مشاهده می‌شود که بین زمان تخمین زده شده و زمان آزمایش‌شده، فاصله وجود دارد. این فاصله به دلایل



شکل ۱۰: زمان پیش‌بینی تخمین زده شده و اندازه‌گیری شده.

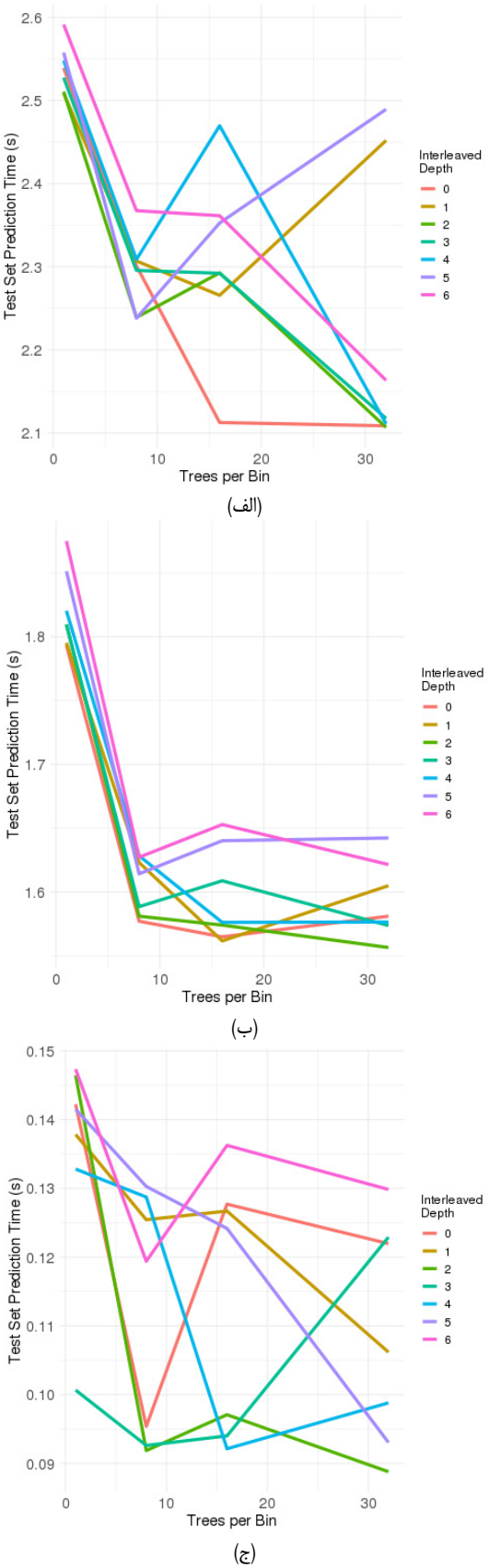
مختلفی در هنگام بارگذاری هر درخت در حافظه کش بستگی دارد؛ از جمله اندازه درخت، مشخص‌نبودن توزیع برگ‌ها در درخت و همچنین مشخص‌نبودن این که کدام گره‌های درخت miss و کدام hit می‌شوند. نتایج نشان می‌دهند که زمان اندازه‌گیری شده برای دو روش پیشنهادی Cbin و PCBin کمتر از زمان تخمینی می‌باشند. این مسئله نشان می‌دهد که با در اختیار داشتن یک حافظه ثابت، روش پیشنهادی توانسته است با فشرده‌سازی بیشتر درخت‌ها، گره‌های بیشتری از درخت را در حافظه بارگذاری کرده و تعداد missها را کاهش دهد و در نتیجه به زمان اندازه‌گیری بهتری دست یابد.

٤-۳ تأثیر تعداد درخت هر Bin در زمان پیش‌بینی

آزمایش‌ها نشان می‌دهند که تعداد درخت در هر Bin و عمق interleaved در پیش‌بینی زمان تست مؤثر است. عمق interleaved به عمقی از درخت گفته می‌شود که تا قبل از آن پیمایش به صورت سطحی و بعد از آن به صورت عمقی انجام می‌شود. شکل ۱۱ نتایج مربوط به این ارزیابی برای روش Bin و دو روش پیشنهادی CBin و PCBin به ازای عمق‌های interleaved از صفر تا ۶ را نشان می‌دهد. در این نمودارها محور افقی تعداد درخت در هر Bin و محور عمودی زمان پیش‌بینی برای داده‌های تست را نشان می‌دهد. همان طور که در نمودارها مشاهده می‌شود، دو روش پیشنهادشده، بهبود قابل توجهی در زمان داشته‌اند و همان گونه که انتظار می‌رود روش PCBin که از موازی‌سازی استفاده کرده، زمان بسیار کمتری را داشته است. به علاوه می‌توان گفت که تغییرات زمان در روش CBin و به خصوص PCBin نسبت به روش Bin کمتر بوده است.

۵- نتیجه‌گیری

این پژوهش در رابطه با دستیابی به دانش و فناوری رایانش سریع می‌باشد و در آن به ارائه روشی کارامد برای فشرده‌سازی و موازی‌سازی جنگل تصادفی پرداختیم. روش پیشنهادی از دو بخش CBin و PCBin تشکیل شده که روش CBin برای فشرده‌سازی جنگل تصادفی و روش PCBin برای موازی‌سازی روش CBin ارائه شده است. روش پیشنهادی به زبان C++ بر روی یک کلاستر ۲۴هسته‌ای با ۳۲ گیگابایت حافظه پیاده‌سازی گردید. با به کارگیری دو روش CBin و PCBin در جنگل



شکل ۱۱: تأثیر تعداد درخت در هر Bin و عمق interleaved، (الف) Bin، (ب) CBin و (ج) PCBin methods.

تصادفی، پیچیدگی زمانی و حافظه جنگل تصادفی، کاهش یافت. نتایج به دست آمده حاکی از آن است که به کارگیری روش فشرده‌سازی پیشنهادی، به طور متوسط ۶۶ درصد بهبود در سرعت پردازش داده‌ها به دنبال داشته است. همچنین به کارگیری فشرده‌سازی به همراه

موازی‌سازی یادشده، ۹۶ درصد بهبود را به همراه داشته است. به طور کلی نتایج آزمایشی و تحلیل‌ها دلالت بر این دارند که راهکارهای پیشنهادی، قدمی مؤثر در راستای رسیدن به رایانش سریع برای جنگل تصادفی در اختیار می‌گذارد. در ادامه این پژوهش می‌توان عملکرد روش MPI در پردازش داده حجیم را بررسی نمود.

٦- اقرار و تصدیق

ضمن اجرای این پژوهش، از قابلیت‌های پردازشی Astek Cluster [۲۹] بهره‌برداری گردید. از همکاری و مساعدت مدیریت آن مجموعه، تقدیر به عمل می‌آید.

مراجع

[۱] ی. صالحی و ن. دانشپور، "یک روش بدون پارامتر مبتنی بر نزدیکی برای تشخیص داده‌های پرت،" *نشریه مهندسی برق و مهندسی کامپیوتر ایران، ب- مهندسی کامپیوتر*، سال ۱۷، شماره ۱، صص. ۲۴-۱۶، بهار ۱۳۹۸.

[2] A. Majeed, "Improving time complexity and accuracy of the machine learning algorithms through selection of highly weighted top k features from complex datasets," *Annals of Data Science*, vol. 6, no. 4, pp. 1-23, Dec. 2019.

[3] M. K. Leung, A. Delong, B. Alipanahi, and B. J. Frey, "Machine learning in genomic medicine: a review of computational problems and data sets," *Proceedings od the IEEE*, vol. 104, no. 1, pp. 176-197, Jan. 2016.

[4] R. C. Edgar, "MUSCLE: a multiple sequence alignment method with reduced time and space complexity," *BMC Bioinformatics*, vol. 5, no. 1, p. 113, Aug. 2004.

[5] C. W. Hsu and C. J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. on Neural Networks*, vol. 13, no. 2, pp. 415-425, Mar. 2002.

[6] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. European Conf. on Computer Vision*, vol. 1, pp. 430-443, May 2006.

[7] M. L. Wallace, *et al*., "Multidimensional sleep and mortality in older adults: a machine-learning comparison with other risk factors," *The J. of Gerontology: Series A*, vol. 74, no. 12, pp. 1903-1909, Dec. 2019.

[8] E. Gabriel, et al., Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation, Springer, Berlin, Heidelberg, 2004.

[9] M. Snir, S. Otto, S. Huss-Lederman, J. Dongarra, and D. Walker, *MPI--the Complete Reference: The MPI Core*, Massachusetts Institute of Technology, 1998.

[10] Y. Zheng, A. Kamil, M. B. Driscoll, H. Shan, and K. Yelick, "UPC++: a PGAS extension for C++," in *Proc. IEEE 28th Int. Parallel and Distributed Processing Symp.,* pp. 1105-1114, Phoenix, AZ, USA, 19-23 May 2014.

[11] M. S. Schlansker and B. R. Rau, "EPIC: explicitly parallel instruction computing," *IEEE Computer*, vol. 33, no. 2, pp. 37-45, Feb. 2000.

[12] J. Kadomoto, *et al*., "An area-efficient out-of-order soft-core processor without register renaming," in *Proc. Int. Conf. on Field-Programmable Technology, FPT'18*, pp. 374-377, Naha, Okinawa, Japan, 10-14 Dec. 2018.

[13] K. B. Theobald, G. R. Gao, and L. Hendren, "Speculative execution and branch prediction on parallel machines," in *Proc. of the 7th Int. Conf. on Supercomputing*, pp. 77-86, Aug. 1993.

[14] Z. N. Wang, J. Tyacke, P. Tucker, and P. Boehning, "Parallel computation of aeroacoustics of industrially relevant complex-geometry aeroengine jets," *Computers & Fluids*, vol. 178, pp. 166-178, Nov. 2019.

[15] F. Nielsen, *Introduction to HPC with MPI for Data Science*, Switzerland: Springer Nature, 2016.

[16] R. Lim, Y. Lee, R. Kim and C. Jaeyoung, "An implementation of matrix–matrix multiplication on the Intel KNL processor with AVX-512," *Cluster Computing*, vol. 21, no. 4, pp. 1785-1795, Dec. 2018.

[17] D. Lecina, J. F. Gilabert, and V. Guallar, "Adaptive simulations, towards interactive protein-ligand modeling," *Scientific Reports*, vol. 7, no. 1, p. 8466, Aug. 2017.

[18] M. Baydoun, H. Ghaziri, and M. Al-Husseini, "CPU and GPU parallelized kernel K-means," *The J. of Supercomputing*, vol. 74, no. 8, pp. 3975-3998, May 2018.

[19] J. Browne, T. Tomita, D. Mhembere, R. Burns, and J. T. Vogelstein, "Forest packing: fast, parallel decision forests," in *Proc. of the SIAM Int. Conf. on Data Mining*, pp. 46-54, May 2019.

[20] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, Oct. 2001.

[21] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning," *Foundations and Trends in Computer Graphics and Vision*, vol. 7, no. 2-3, pp. 81-227, Feb. 2012.

[22] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random forests and decision trees," *International J. of Computer Science Issues*, vol. 9, no. 5, pp. 272-278, Sept. 2012.

[23] I. Kamel and C. Faloutsos, "On packing R-trees," in *Proc. of the 2nd Int. Conf. on Information and Knowledge Management*, pp. 490-499, Dec. 1993.

[24] I. K. Landing, "Guide to Automatic Vectorization with Intel AVX-512 Instructions in Knights Landing Processors," 11 May 2016. [Online]. Available: https://colfaxresearch.com/knl-avx512/.

[25] P. Alonso, R. Cortina, F. J. Matrines-Zaldivar, and J. Ranilla, "Neville elimination on multi-and many-core systems: OpenMP, MPI and CUDA," *The J. of Supercomputing*, vol. 58, no. 2, pp. 215-225, Nov. 2011.

[26] *Allstate Claim Prediction Challenge*, Accessed Feb. 13 2018. [Online]. Available: https://www.kaggle.com/c/ClaimPredictionChallenge.

[27] *Higgs Boson Machine Learning Challenge*, Accessed Feb. 13 2018. [Online]. Available: https://www.kaggle.com/c/higgs-boson.

[28] J. Browne, *Forestpacking*, 11 Oct 2018. [Online]. Available: https://github.com/jbrowne6/forestpacking.

[29] A. Nezarat, *Astek HPC Big Data*, [Online]. Available: http://astek.ir/. [Accessed 22 July 2019].

نعیمه محمدکریمی سال ۱۳۸۷ مدرک کارشناسی مهندسی کامپیوتر (نرم‌افزار) خود را از دانشگاه پیام نور اردکان دریافت نمود. از مهر ماه ۱۳۹۱ الی دی‌ماه ۱۳۹۳ نامبرده جهت انجام دوره کارشناسی ارشد مهندسی کامپیوتر (هوش مصنوعی) در دانشگاه یزد مشغول به تحصیل و پژوهش بودند. زمینه‌های علمی مورد علاقه نامبرده عبارتند از: یادگیری ماشین، یادگیری عمیق، داده‌های حجیم، سیستم¬های فازی.

محمد قاسم‌زاده در سال ۱۳۶۸ مدرک کارشناسی مهندسی کامپیوتر خود را از دانشگاه شیراز و در سال ۱۳۷۴ مدرک کارشناسی ارشد مهندسی کامپیوتر (هوش ماشین و رباتیک) خود را از دانشگاه صنعتی امیرکبیر دریافت نمود. از بهمن ماه ۱۳۸۰ الی بهمن ماه ۱۳۸۴ نامبرده جهت انجام دوره دکتری (علوم کامپیوتر) در دانشگاه تربیر و دانشگاه پتسدام هر دو در کشور آلمان مشغول به تحصیل و پژوهش بودند. ایشان در سال ۱۳۸۴ موفق به اخذ درجه دکتری علوم کامپیوتر گردید. محمد قاسم‌زاده از سال ۱۳۷۵ تاکنون به عنوان عضو هیأت علمی در دانشگاه یزد مشغول به تدریس و تحقیق می‌باشند. زمینه‌های علمی مورد علاقه نامبرده عبارتند از: طراحی و تحلیل الگوریتم‌ها، پردازش زبان طبیعی، سیستم‌های هوشمند و محاسبات نرم.

مهدی یزدیان دهکردی مدرک کارشناسی مهندسی کامپیوتر گرایش نرم‌افزار را در سال ۱۳۸۵ از دانشگاه یزد و مدرک کارشناسی ارشد و دکترای خود را به ترتیب در سال‌های ۱۳۸۸ و ۱۳۹۴ در رشته مهندسی کامپیوتر گرایش هوش مصنوعی از دانشگاه شیراز اخذ کرد. ایشان از سال ۱۳۹۴ در بخش هوش‌مصنوعی دانشکده مهندسی کامپیوتر دانشگاه یزد مشغول به فعالیت گردید و اینک نیز عضو هیأت علمی این دانشکده است. زمینه‌های پژوهشی مورد علاقه ایشان یادگیری ماشین، یادگیری عمیق و بینایی ماشین است.

امین نظارت در سال ۱۳۸۱ مدرک کارشناسی علوم کامپیوتر خود را از دانشگاه شهید باهنر کرمان و در سال ۱۳۹۰ مدرک کارشناسی ارشد مهندسی فناوری اطلاعات خود را از دانشگاه شیراز دریافت نمود. از سال ۱۳۹۱ الی سال ۱۳۹۵ نامبرده جهت انجام دوره دکتری مهندسی کامپیوتر (نرم‌افزار) در دانشگاه شیراز مشغول تحصیل و پژوهش بودند. ایشان در سال ۱۳۹۵ موفق به اخذ درجه دکتری مهندسی کامپیوتر گردید. دکتر نظارت از سال ۱۳۹۶ تا ۱۳۹۸ عضو هیأت علمی دانشگاه پیام نور یزد بودند. در حال حاضر ایشان در حال گذراندن دوره پسادکتری در دانشگاه ماساریک در جمهوری چک می‌باشند. زمینه‌های علمی مورد علاقه نامبرده عبارتند از: بازیابی هوشمند اطلاعات، داده‌های حجیم، محاسبات اَبری و محاسبات با کارایی بالا.