نشریه علمی‌ترویجی محاسبات نرم
سال پنجم/ شمارهٔ اول/ بهار و تابستان ۹۵/ صفحهٔ ۳۶ـ۶۵

دریافت مقاله: ۱۳۹۳/۱۱/۲۴
پذیرش مقاله: ۱۳۹۵/۳/۳۱

مروری بر روش‌های تخمین هزینهٔ نرم‌افزار مبتنی بر یادگیری ماشین

صبا بیرانوند[۱*]، محمدعلی زارع چاهوکی[۲]

[۱] دانشجوی کارشناسی ارشد، دانشکده برق و کامپیوتر، دانشگاه یزد، یزد، ایران
saba.beiranvand@stu.yazd.ac.ir
[۲] استادیار، دانشکده برق و کامپیوتر، دانشگاه یزد، یزد، ایران
chahooki@yazd.ac.ir

چکیده

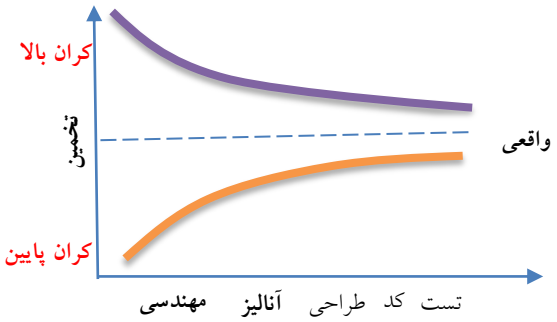
مدیریت پروژهٔ نرم‌افزاری از مهم‌ترین فعالیت‌ها در توسعهٔ محصول نرم‌افزاری است؛ زیرا تمامی فرایند توسعهٔ نرم‌افزار، از ابتدا تا انتها را شامل می‌شود. تخمین هزینهٔ نرم‌افزار، یک فعالیت چالشی در مدیریت پروژهٔ نرم‌افزاری است. مفهوم تخمین هزینهٔ نرم‌افزار، همزمان با شروع صنعت کامپیوتر در سال ۱۹۴۰ مورد توجه قرار گرفته و همچنان پژوهش در این حوزه ادامه دارد. بااینکه تلاش، فقط دربرگیرندهٔ بخشی از هزینه‌های توسعهٔ یک پروژهٔ نرم‌افزاری است، عامل اساسی برای تعیینِ هزینه محسوب می‌شود. ازاین‌رو، در پژوهش‌های این حوزه، دو اصطلاح تخمینِ تلاش و تخمینِ هزینه به‌صورت معادل به‌کار می‌روند. مدل تخمین هزینهٔ نرم‌افزاری درصورتی‌که که قبل از عقد قرارداد، دقت و اطمینان زیادی برای پیش‌بینی هزینهٔ پروژه‌های نرم‌افزاری فراهم کند، مناسب است. به‌علت ذات غیرقطعیِ تخمین و برای افزایش دقت، به‌مرور توجه پژوهشگران به استفاده از روش‌های یادگیری ماشین در این حوزه معطوف شده است. در پژوهش حاضر، به بررسی مطالعات صورت‌گرفته در تخمین هزینهٔ نرم‌افزار با روش‌های یادگیری ماشین پرداخته و روش‌های تخمین ارائه‌شده، معیارهای ارزیابی دقت این روش‌ها و دادگانِ مورداستفاده در کارهای پژوهشی و همچنین پژوهش‌های آتی در این حوزه را معرفی کرده‌ایم.

واژه‌های کلیدی: تخمین هزینهٔ نرم‌افزار، تخمین تلاش نرم‌افزار، یادگیری ماشین.

۱. مقدمه

مسئلهٔ مهم برای مدیران پروژه، تخمین‌های دقیق و مطمئنِ تلاشِ توسعهٔ نرم‌افزار، به‌خصوص در مراحل اولیهٔ چرخهٔ توسعهٔ نرم‌افزار است [۱ و ۲]. دلیل آن، ذاتِ غیرقطعی تخمین و همچنین تغییرات سریع در روش‌های توسعهٔ نرم‌افزار است [۲]. شکست پروژه، مسئله‌ای است که امروزه بسیار بدان توجه شده است و تخمین نادرست هزینهٔ نرم‌افزار یکی از عوامل مؤثر بر شکست پروژه محسوب می‌شود. دو اصطلاح تخمین هزینه و تخمین تلاشِ موردنیاز برای توسعهٔ پروژهٔ نرم‌افزاری (SDEE[1]) در پژوهش‌های مهندسی نرم‌افزار و مدیریت پروژه، معادل هم، مورداستفاده قرار می‌گیرند. درواقع تلاش، عامل اصلی تعیین‌کنندهٔ هزینه است و این واقعیت وجود دارد که تخمینِ هزینهٔ نرم‌افزار، درنهایت یک مسئلهٔ تخمینِ تلاشِ نرم‌افزار است [۳ـ۶].

هزینه و تلاش به‌شدت به‌هم وابسته‌اند. انتخاب بهترین مدل مبتنی بر میزان داده‌های در دسترس، روش‌های توسعه، روش‌های استفاده شده و عوامل دیگری از این قبیل است. شکل (۱) بیانگر رابطهٔ تخمین دقیق، با تخمین انجام‌شده براساس مراحل مختلف توسعهٔ پروژه است.



شکل (۱): رابطهٔ تخمین دقیق با مراحل مختلف توسعه [۷]

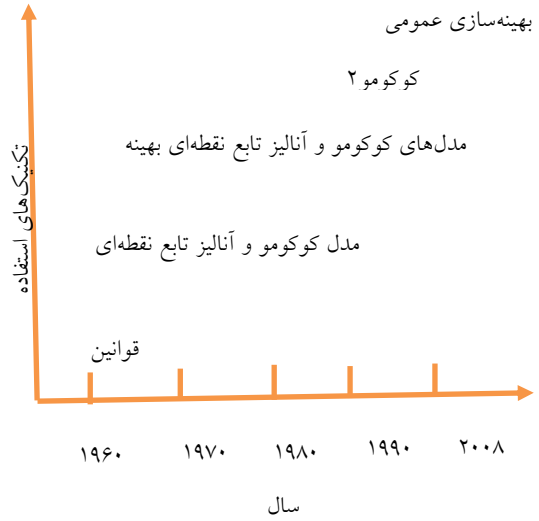
۱. ۱. تاریخچه

تخمین هزینهٔ نرم‌افزار[2] فرایندی قدیمی است که با شروع صنعت کامپیوتر در سال ۱۹۴۰ به‌وجود آمد و چندین‌بار توسعه یافت. از این فرایند در صنعت توسعهٔ نرم‌افزار برای پیش‌بینی تلاش، منابع و هزینهٔ هر فرایند توسعه استفاده می‌شود. بنابراین، عمل تخمین هزینهٔ نرم‌افزار، برای مدیریت و نظارت بر توسعهٔ نرم‌افزار موردنیاز است. تا قبل از ارائهٔ روش‌های تخمین، فرایند تخمین هزینهٔ نرم‌افزار به برخی الگوریتم‌های ساده وابسته بود [۸].

درواقع، تخمین هزینهٔ نرم‌افزار از سال ۱۹۶۰ شروع شد؛ زمانی‌که فرانک فریمن[3] مفاهیم مدل‌های تخمین پارامتریک را توسعه داد و به توسعهٔ مدل قیمت[4] برای سخت‌افزار منجر شد. در سال ۱۹۷۰، محققان در این زمینه برای خیلی از پروژه‌ها، از روش‌های آماری، برای شناخت فاکتورهای مؤثر روی هزینهٔ توسعهٔ نرم‌افزار، با استفاده از روش‌های رگرسیون و وابستگی استفاده کردند. با اتمام این دهه، مدل COCOMO توسط باری بوهم[5] و آبتس[6] فرموله شد. مدل پارامتریک قیمت هم با اتمام این دوره ارائه شد. سپس تجزیه‌وتحلیل تابع نقطه‌ای (FPA[7])، توسط چند تن از کارکنان شرکت IBM برای تخمین هزینه و تلاشِ نرم‌افزار ارائه شد [۸].

به‌طور کلی، می‌توان گفت خیلی از تحولات روی مدل‌ها و روش‌های ارائه‌شده، در سال ۱۹۸۰ صورت گرفت. در همین سال بود که یکی از محققانِ پیشرو در تخمین هزینهٔ نرم‌افزار، باری بوهم و همکارانش، روی مدل COCOMO که قبلاً خودشان ارائه کرده بودند، تغییراتی اعمال نمودند که حاصل آن مدل جدیدی به نام COCOMOII بود. از دههٔ ۹۰ به‌بعد، تحقیقات به‌سمت بهبودنمایی در صنعت نرم‌افزار و فناوری اطلاعات کشیده شد و بیشتر محققان در بهبود دقت الگوریتم‌های مختلفِ ارائه‌شده در این زمینه سعی کرده و روی این بحث متمرکز شده‌اند [۸]. عبدالله زید و همکارانش در مقالهٔ پژوهشی خود [۸] روند کلی تحولات تاریخی فرایند تخمین هزینهٔ نرم‌افزار را در شکل (۲) به تصویر کشیده‌اند.

1. Software Development Effort Estimation
2. Software Cost Estimation
3. Frank Freiman
4. PRICE
5. Barry W. Boehm
6. C. Abts
7. Function point analysis



شکل (۲): سیر تاریخی تحولات روش‌های تخمین هزینهٔ نرم‌افزار [۸]

۱. ۲. روش‌های مختلف تخمین هزینهٔ نرم‌افزار

روش‌های مختلفی از آغاز تاکنون در زمینهٔ تخمین هزینه و تلاش نرم‌افزار ارائه شده و مقالات مختلف، دسته‌بندی‌های متفاوتی برای این روش‌ها ارائه داده‌اند. برایان[1] و همکارانش در [۴] روش‌های ارائه‌شده را در شش دستهٔ مبتنی بر مدل، مبتنی بر خبره، مبتنی بر یادگیری، پویایی، رگرسیون و بیزینِ مرکب دسته‌بندی کرده‌اند. عبدالله زید و همکارانش در [۸]، تقسیم‌بندیِ روش‌های تخمین هزینه و تلاش در دو دستهٔ کمی و کیفی را توصیه کرده‌اند که براساس آن، روش‌های کیفی بیشتر به شباهت و تفاوت وابسته‌اند و روش‌های کمّی مثل روش‌های پارامتریک و تحلیلی، به جزئیات وابسته بوده و می‌توانند در فاز پیشرفته‌ای از اجرا استفاده شوند.

عمدهٔ مقالات مانند [۶ و ۹] روش‌های تخمین هزینهٔ نرم‌افزار را به سه دستهٔ مدل‌های الگوریتمی، روش‌های مبتنی بر قضاوت کارشناسان[2] و روش‌های یادگیری ماشین (ML[3]) تقسیم کرده‌اند. البته دسته‌ای از مقالات، ازجمله مقالات ارائه‌شده در [۷ و ۱۰] مدل مبتنی بر قضاوت کارشناسان را هم از دسته مدل‌های الگوریتمی می‌دانند و در اصل، روش‌های تخمین هزینهٔ نرم‌افزار را در دو دستهٔ کلی الگوریتمی و غیرالگوریتمی معرفی کرده‌اند. شکل (۳ نمایش تصویری این دسته‌بندی است.

شکل (۳): دسته‌بندی روش‌های مختلف تخمین هزینهٔ نرم‌افزار

در جدول (۱)، مقایسه‌ای بین دو دستهٔ تعیین‌شده انجام شده است.

جدول (۱) مقایسهٔ روش‌های الگوریتمی و غیرالگوریتمی [۷]

روش‌های الگوریتمی	روش‌های غیرالگوریتمی
انعطاف‌پذیری کم	انعطاف‌پذیری بالا
مبتنی بر مدل	مبتنی بر یادگیری
به اطلاعات زیادی نیاز دارد.	برخی از ویژگی‌های پروژه کافی‌اند.
روش‌های آماری	روش‌های مختلف
فرایند تخمین ساده	ممکن است فرایند پیچیده باشد.
نیازمند به‌روزرسانی است.	سازگار با تغییرات جدید است.
تخمین سریع	اغلب، تخمین زمان‌بر است.
هیچ انسانی نمی‌تواند در مدل مداخله داشته باشد.	کارشناسان می‌توانند روش را تنظیم کنند.
برآورد غلط در مراحل اولیهٔ پروژه	تخمین دقیق در مراحل اولیه
قابل‌استفاده با پارامترهای مخصوص	قابل استفاده با پارامترهای مختلف

به‌طور کلی، تخمین تلاش مبتنی بر مدل‌های الگوریتمی، براساس رابطهٔ (۱) طراحی می‌شود [۹].

$$EFFORT = F(x_1, x_2, \dots.) \quad (۱)$$

در این رابطه، منظور از $x_1, x_2, \dots$ ویژگی‌هایی است که برای هر پروژه بیان می‌شود. تخمین تلاش مبتنی بر مدل‌های الگوریتمی مختلف، معمولاً به‌طور قابل‌توجهی، متفاوت از سایر مدل‌هاست. درواقع، در این روش‌ها مدلی براساس یک الگوریتم

1. Briand
2. Expert Judgment
3. Machine learning

مشخص فرموله شده و از آن فرمول، برای تخمین هزینهٔ نرم‌افزار استفاده می‌شود. تاکنون مدل‌های الگوریتمی مختلفی برای تخمین هزینهٔ نرم‌افزار ارائه شده است (نک: شکل (۳). ازجمله معایب مدل‌های الگوریتمی، یکی این است که کار را برای مدیران سخت کرده‌اند؛ چراکه تصمیم‌گیری دربارهٔ میزان منابع واقعی، برای آن‌ها سخت شده و دیگر اینکه مبتنی بر داده‌های قدیمی‌اند و نمی‌توانند پیشرفت‌های فعلی در زبان‌های برنامه‌نویسی، سخت‌افزار و مهندسی نرم‌افزار را بازتاب دهند [۶].

مدل‌های غیرالگوریتمی بر استفاده از آزمایش‌ها و روش‌های خاص خود مبتنی هستند [۹]. معمولاً داده‌های قدیمی برای قضاوت خبرگان لازم نیستند. قضاوت خبرگان اغلب مبتنی بر استفادهٔ مجدد از موارد تعیین‌کنندهٔ پروژه‌های قبلی است که حال ممکن است مستند هم نشده باشند. نتایج به‌دست‌آمده بیانگر آن است که ۶۲ درصد از تخمین هزینهٔ پروژه‌های نرم‌افزاری در سازمان‌ها، از این روش استفاده می‌کنند. مزیتی این نوع از روش‌ها این است که برای فرهنگِ سازمانی خاصی، سفارشی شده‌اند. پس در مقایسه با رویکرد الگوریتمی، دقیق‌تر خواهند بود و در خیلی از موارد ثابت شده است که دقت بهتری نسبت‌به سایر مدل‌ها ارائه داده‌اند. ولی این تخمین، ذهنی و براساس منطق خبره است و درواقع، مزیت آن را می‌توان به‌عنوان عیب آن هم در نظر گرفت؛ زیرا هر خبره فقط براساس تجربیات و فرهنگی که در آن سازمانِ خاص وجود دارد، هزینه را تخمین می‌زند و همان خبره در سازمان دیگر ممکن است هزینهٔ نرم‌افزار را با دقتی کمتر تخمین بزند. پس لزوماً در محیط‌های مختلف، همان دقت را نخواهند داشت [۶].

روش‌های یادگیری ماشین، الگوهایی را از داده‌های پروژه‌های قدیمی یاد می‌گیرند و از این الگوها برای پیش‌بینی تلاش استفاده می‌کنند. ایدهٔ اساسی استفاده از روش‌های یادگیری ماشین برای تخمین و پیش‌بینی تلاش، این است که دادگان[1] قدیمی شامل پروژه‌های قدیمی زیادی هستند که با خصوصیات باارزششان برای توصیف آن پروژه تشریح شده‌اند. پروژه‌های با خصوصیاتی دارای مقادیر مشابه، ممکن است شامل تلاش‌های پروژهٔ تقریباً مشابهی باشند. کار روش‌های یادگیری ماشین برای یادگیری الگوهای ذاتی ویژگی‌های پروژه‌ها، ارتباطات بین آن‌ها و تلاش‌های پروژه است که می‌تواند برای پیش‌بینی تلاش پروژهٔ جدید استفاده شود [۶]. مهم‌ترین دلایل اهمیت تخمین هزینهٔ نرم‌افزار به این شرح است که به طبقه‌بندی پروژه‌های بهبودیافته، با توجه به طرح جامع اقتصادی کمک می‌کند، در استفادهٔ بهینه از منابع تأثیرگذار است، در ارزیابی تغییرات مورداستفاده است و مشتریان انتظار دارند هزینه در راستای هزینهٔ واقعی باشد [۹].

با توجه به وجود عدم قطعیت در تخمین نرم‌افزاری، استفاده از روش‌های غیرقطعی و انعطاف‌پذیر یادگیری ماشین، می‌تواند نقش بسزایی در افزایش دقت تخمین داشته باشد. مزیت‌های این نوع از روش‌های تخمین، شامل توانایی مدل‌های هوش محاسباتی برای مدل کردن مجموعه‌ای پیچیده از ارتباطات بین تلاش و افزایندگان آن و همین‌طور توانایی آن‌ها در یادگیری از داده‌های پروژه‌های قدیمی است [۱۱]. بدین منظور از پژوهش‌هایی که به مرور محاسبات نرم [۱۲] و یادگیری ماشین [۱۳] در این زمینه پرداخته‌اند، استفاده می‌شود.

۱. ۳. روش تحقیق

مقالات فراوانی تاکنون در راستای کاربرد یادگیری ماشین در این زمینه شناسایی شده‌اند که در طول دوران پژوهش، به مطالعهٔ این مقالات پرداخته شده است. دسته‌ای از این مقالات به مرور کارهای مختلف در این زمینه پرداخته‌اند. از میان مقالات انگلیسی، مقاله‌های [۴ و ۱۳] مقالاتی هستند که ازنظر حوزهٔ موردبررسی، مشابه مقالهٔ حاضرند. به همین دلیل، در مرور انجام‌شده، از آن‌ها نیز استفاده شده است. این مقالات، کارهای انگلیسی در محدودهٔ سال‌های ۱۹۹۱ تا۲۰۱۰ را دربرمی‌گیرند. در میان مقالات فارسی هم، مقالات مروری در زمینهٔ تخمین هزینهٔ نرم‌افزار ارائه شده است، اما این مقالات روی روش‌های یادگیری ماشین در این زمینه تمرکز نکرده‌اند. تفاوت مقالهٔ حاضر با سایر مقالات مروری ارائه‌شده در این است که اولین مقالهٔ مروری فارسی، در زمینهٔ تخمین هزینهٔ نرم‌افزار با روش‌های یادگیری ماشین است؛ ضمن اینکه مقالات سال‌های ۲۰۱۰ به‌بعد را هم در بردارد. در مرور

1. Dataset

صورت‌گرفته مقالات فارسی هم بررسی شده‌اند.

در مرور صورت‌گرفته در این مقاله، به‌دنبال یافتن پاسخی برای پرسش‌های پژوهشی مطرح در زمینهٔ تخمین هزینهٔ نرم‌افزار با استفاده از رویکردهای یادگیری ماشین هستیم؛ پرسش‌هایی نظیر:

- کدام‌یک از روش‌های ML در SDEE استفاده شده‌اند؟
- دقت کلی تخمین مدل‌های ML چقدر است؟ (دادگان استفاده‌شده، متدهای اعتبارسنجی، متریک‌های استفاده‌شده برای اندازه‌گیری دقت در تخمین و میزان دقت)
- آیا مدل‌های MLبهتر از مدل‌های غیر ML هستند؟
- کدام مدل ML بهتر از سایر مدل‌های ML است؟ (مدل‌های ML در مقایسه باهم، درجه بهبود)
- هر روش ML در چه وضعیتی بهتر است؟ (نقاط قوّت و ضعف مدل‌های ML ازنظر SDEE)

این پرسش‌های پژوهشی همچنین در [۴] که مقالهٔ مروری خوبی در این حوزه است، استفاده شده است. در [۴] مقالات بین سال‌های ۱۹۹۱ تا ۲۰۱۰ براساس پرسش‌های پژوهشی انتخاب و مرور شده‌اند. در این مقاله هم براساس همان پرسش‌های پژوهشی، مقالات ارائه‌شده در سال‌های بعد از ۲۰۱۰ انتخاب و مرور شده‌اند. البته برای تکمیل مطالب، از برخی مقالات مربوط‌به سال‌های قبل هم استفاده شده است.

در بخش ۲، به بررسی الگوریتم‌های یادگیری ماشین مختلفی که در زمینهٔ تخمین هزینهٔ نرم‌افزار استفاده شده‌اند، خواهیم پرداخت. دادگان مختلفی که در این زمینه استفاده شده‌اند، در بخش ۳ بررسی می‌شوند. در بخش ۴ معیارهای ارزیابی پرکاربرد در تخمین هزینهٔ نرم‌افزار معرفی شده و در بخش ۵ به جمع‌بندی و ارائهٔ پیشنهادهایی برای پژوهش‌های آتی در زمینهٔ مورد بحث پرداخته می‌شود.

۲. یادگیری ماشین در تخمین هزینهٔ نرم‌افزار

تخمین دقیق تلاش موردنیاز در مراحل اولیهٔ توسعهٔ نرم‌افزار، نقش بسیار مهمی در مدیریت پروژه بر عهده دارد [۴]. از سال ۱۹۸۰، براساس پژوهش [۸] یعنی برهه‌ای از زمان که تحولات اساسی در زمینهٔ تخمین تلاش نرم‌افزار رخ داده است، بسیاری از روش‌های تخمین در زمینهٔ تخمین هزینهٔ نرم‌افزار ارائه شده‌اند که از میان آن‌ها از روش مبتنی بر قضاوت متخصصان و روش مقایسه‌ای[1] با بیشتری استفاده شده است. در سال‌های اخیر، به روش‌های مبتنی بر یادگیری ماشین در این زمینه، بیشتر توجه شده است [۴].

ون[2] و همکارانش در [۴] مروری سامان‌مند روی مقالاتی که از سال ۱۹۹۱ تا ۲۰۱۰ در زمینهٔ تخمین هزینهٔ نرم‌افزار با استفاده از روش‌های یادگیری ماشین ارائه شده‌اند، انجام داده‌اند. در این مقالهٔ مروری، اشاره شده است که برخی از محققان حتی نشان داده‌اند که روش مبتنی بر یادگیری ماشین، یکی از سه دستهٔ اصلی روش‌های تخمین (دو دستهٔ دیگر شامل مدل‌های الگوریتمی و قضاوت هستند) است.

مقالات مختلفی تاکنون در این زمینه ارائه شده‌اند که دقت‌های مختلفی را با الگوریتم‌های یادگیری ماشین گزارش داده‌اند. اما معمولاً دقت‌های گزارش‌شده در کارهای مختلف، با هم متفاوت بوده و همین اختلاف در مطالعات تجربی موجود که علت آن هم هنوز قابل‌درک نیست، ممکن است باعث ممانعت از استفادهٔ این الگوریتم‌ها توسط متخصصان در این زمینه شود. در مقایسه با سایر حوزه‌هایی که الگوریتم‌های یادگیری ماشین کاربرد دارند و با موفقیت پیاده‌سازی شده‌اند، در حوزهٔ تخمین هزینهٔ نرم‌افزار، چالش‌های زیادی ازجمله دادگان کوچک، عدم قطعیت اطلاعات، معیارهای کیفی و عوامل انسانی وجود دارند. علاوه بر این، باید گفت که این نوع استفاده از روش‌های ML بسیار پیچیده‌تر از روش‌های تخمین معمولی است [۴].

بااینکه در سال‌های اخیر با هدف افزایش دقت، استفاده از مدل‌های یادگیری ماشین در تخمین تلاش توسعهٔ نرم‌افزار، بیشتر مورد توجه قرار گرفته است، هیچ‌یک از مدل‌های موجود در تمام شرایط مناسب نیستند و از دادگانی به دادگان دیگر، دقت متفاوتی را ارائه می‌دهند. بنابراین، نیاز به ساختن مدلی تخمینی وجود دارد که مطمئن باشد و دقت بیشتری را فراهم کند. با همین هدف، ترکیبات مختلفی از این الگوریتم‌ها چه در ترکیب با الگوریتم‌های

1. Analogy
2. Jianfeng Wen

یادگیری ماشینِ دیگر و چه در ترکیب با الگوریتم‌های غیریادگیری ماشین، تاکنون ارائه شده است. محمود الیش[1] و همکارانش در [۱۱] ترکیبات همگن[2] و ناهمگن[3] متفاوتی از الگوریتم‌های مختلف یادگیری ماشین ارائه داده و آزمایش‌های گوناگونی را روی آن‌ها انجام داده‌اند. ترکیبات ناهمگن شامل اعضایی با داشتن الگوریتم‌های یادگیری پایه و ترکیبات همگن شامل اعضایی با الگوریتم یادگیری تک‌نوع هستند. در ادامه، مروری بر کارهایی ازاین‌دست که می‌توان در زمینهٔ تخمین تلاش و روش‌های یادگیری ماشین انجام داد، خواهیم داشت.

۲. ۱. معرفی روش‌ها

براساس مروری که در این پژوهش بر مطالعات صورت‌گرفته در زمینهٔ تخمین هزینهٔ نرم‌افزار ازجمله [۴] انجام شده است، از سال ۱۹۹۱ تاکنون ۸ دسته الگوریتم یادگیری ماشین در حوزهٔ تخمین هزینهٔ نرم‌افزار استفاده شده‌اند. در جدول (۲) این دسته‌بندی‌ها ارائه شده و در ادامه، شرح مختصری از آن‌ها را خواهیم داشت. روش‌های پرکاربرد با * نشان داده شده‌اند.

جدول (۲) روش‌های یادگیری ماشین مورداستفاده در تخمین هزینهٔ نرم‌افزار

مخفف	عنوان	روش
استدلال مبتنی بر مورد*	Case-Based Reasoning	CBR
شبکه‌های عصبی مصنوعی*	Artificial Neural Networks	ANN
درخت تصمیم*	Decision Trees	DT
شبکه‌های بیزین	Bayesian Networks	BN
رگرسیون بردار پشتیبان	Support Vector Regression	SVR
الگوریتم ژنتیک	Genetic Algorithms	GA
برنامه‌نویسی ژنتیک	Genetic Programming	GP
قوانین وابستگی	Association Rules	AR

البته نکتهٔ مهم این است که علاوه بر دسته‌بندی ارائه‌شده در جدول (۲)، روش‌های یادگیری ماشین دیگری هم طی سال‌های اخیر ارائه شده و به‌تبع، در این حوزه به‌کار رفته‌اند. نکتهٔ گفتنی اینکه در برخی مطالعات، الگوریتم‌های یادگیری ماشین در ترکیب با الگوریتم‌های دیگر مورداستفاده قرار گرفته‌اند؛ مثلاً روش‌هایی مثل منطق فازی و GA اغلب برای ترکیب با سایر روش‌های یادگیری ماشین به‌کار گرفته شده‌اند.

GA با توجه به مروری که انجام شده است، فقط در فرم ترکیبی استفاده می‌شود. در این ترکیبات، روش GA تنها به‌منظور وزن‌دهی یا انتخاب ویژگی[4]، استفاده شده است. منطق فازی هم برای مقابله با عدم اطمینان و عدم صحت اطلاعات استفاده می‌شود. در عمل، برای بهبود عملکرد مدل با پیش‌پردازش ورودی‌های مدل، از روش منطق فازی استفاده شده است [۴].

البته شایان ذکر است که برخی مقالات مثل [۳] به‌جای رگرسیون بردار پشتیبان، روش‌های رگرسیون را نام برده‌اند. در مطالعات مرورشده، روش‌های ANN و CBR به‌عنوان پرکاربردترین روش‌ها نسبت‌به سایر روش‌ها معرفی شده‌اند. در ادامهٔ کار، با انواع دسته‌های الگوریتم‌های یادگیری ماشین آشنا خواهیم شد. در ضمیمهٔ الف، شرح مختصری از مقالاتی که از روش‌های یادگیری ماشین در تخمین هزینهٔ نرم‌افزار استفاده و در این پژوهش بررسی شده‌اند، به‌صورت جدول آمده است.

۲. ۱. ۱. استدلال مبتنی بر مورد

روش استدلال مبتنی بر مورد، یکی از روش‌های یادگیری ماشین در تخمین هزینهٔ نرم‌افزار است که امکان تطبیق با موارد مشابهی را که در گذشته رخ داده، برای متخصصان این زمینه فراهم می‌کند. CBR منبع استدلال قیاسی، حافظهٔ پویا و نقش موقعیت‌های قبلی است که توجه بسیاری را در سال‌های اخیر به خود جلب کرده است. برخی از پژوهشگران، CBR را به‌صورت دوره‌ای و متشکل از چهار مرحله توصیف کرده‌اند: بازیابی مورد یا موارد خیلی مشابه مثل پروژه‌های توسعه‌یافتهٔ قبلی؛ استفادهٔ مجدد از موارد بازیابی‌شده برای یافتن یک راه‌حل برای مسئله؛ دریافت راه‌حل یا تجدیدنظر در راه‌حل پیشنهادی؛ نگهداری راه‌حل به‌عنوان یک راه‌حل جدید.

در سال‌های اخیر، برخی تحقیقات ملموس در کاربرد CBR برای تخمین تلاش، نشان داده‌اند که CBR می‌توان امکان ویژه‌ای برای مدیریت توسعهٔ نرم‌افزار فراهم کند [۳]. در این روش، از الگوریتم‌های مختلف برای محاسبهٔ شباهت از معیارهایی مثل

1. Mahmoud O. Elish
2. Homogeneous
3. Heterogeneous
4. Feature selection

فاصلهٔ مَنهتن[1]، اقلیدسی[2] و معیارهایی ازاین‌دست استفاده می‌شود. در تألیفات گوناگون، این دسته از الگوریتم‌های یادگیری ماشین را با عنوان الگوریتم‌های غیر پارامتریک یاد می‌کنند [۱۴]. در مطالعهٔ پژوهشی‌ـآزمایشی صورت‌گرفته در [۱۵]، از الگوریتم تعیین فاصلهٔ اقلیدوسی، برای محاسبهٔ فاصله یا میزان شباهت نمونه‌ها، برای استفاده از کاربرد الگوریتم CBR در تخمین هزینهٔ نرم‌افزار استفاده شده است.

۲. ۱. ۲. شبکه‌های عصبی مصنوعی

ANNها تعداد زیادی از عناصر پردازشیِ به‌شدت به‌هم‌پیوسته به نام نورون‌ها[3] دارند که معمولاً در عمل به‌صورت موازی هستند و تحت معماری منظمی، پیکربندی شده‌اند. هر نورون به‌وسیلهٔ یک لینکِ ارتباطی[4]، با سایر نورون‌ها در ارتباط است. هر لینکِ ارتباطی وزنی دارد که بیانگر اطلاعاتی دربارهٔ سیگنال ورودی است. نورون، یک مجموع از وزن‌های ورودی را محاسبه می‌کند و اگر مجموع وزن‌ها بالاتر از یک حد آستانه[5] باشد، یک خروجی را تولید می‌کند. این فرایند تا زمانی که یک یا چند خروجی تولید شوند، ادامه می‌یابد. این مدل‌های تخمین می‌توانند با استفاده از داده‌های آموزشی قدیمی، برای تولید نتایج، با تنظیم دقیق مقادیر پارامترهای الگوریتم، آموزش ببینند تا اختلاف بین مقدار واقعی شناخته‌شده با مدل پیش‌بینی را کاهش دهند [۳].

همچنین پژوهشگرانی در این زمینه مثل [۱۶] شبکه‌های عصبی مورداستفاده در تخمین نرم‌افزار را در انواع شبکهٔ عصبی روبه‌جلو[6]، شبکهٔ عصبی مکرر[7]، شبکهٔ عصبی تابع پایه‌ای شعاعی (RBF)[8]، شبکهٔ عصبی‌ـفازی[9] خلاصه کرده‌اند. پژوهشگران در برخی مقالات ازجمله [۱۵]، از دو روش شبکهٔ عصبی چندلایه (MLP) و شبکهٔ عصبی تابع پایه‌ای شعاعی را به‌عنوان نمایندگانی از الگوریتم‌های شبکهٔ عصبی در مطالعهٔ آزمایشی خود، در حوزهٔ تخمین هزینهٔ نرم‌افزار استفاده کرده و دقت آن‌ها را مورد ارزیابی قرار داده‌اند.

۲. ۱. ۳. درخت تصمیم

دسته‌بندی داده‌ها، یک فرایند دومرحله‌ای است متشکل از: ۱. مرحلهٔ آموزش که در آن، مدل ساخته می‌شود؛ ۲. مرحلهٔ دسته‌بندی که در آن از مدل ساخته‌شده برای دسته‌بندی و برچسب‌زنی روی داده‌های موردنظر استفاده می‌شود.

درخت‌های تصمیم با چندتایی‌های آموزش و دسته‌های آن‌ها در معیار هدف ایجاد می‌شوند. این درخت از دو نوع نود تشکیل شده است که شامل نود تصمیم یا نود داخلی[10] و نود برگ[11] است. این نوع از روش‌های یادگیری ماشین، یک نمودار جریان[12] با ساختاری مانند درخت است که هر گره داخلی یعنی نودهایی که برگ نیستند، یک آزمون را روی یکی از خصوصیات چندتایی انجام می‌دهند. هر شاخه[13] بیانگر یک خروجی از آزمون و هر برگ نشان‌دهندهٔ برچسب یا مقدار تخمین‌زده‌شده برای آن نمونه است. قوی‌ترین گره یک درخت، ریشهٔ آن است.

به‌منظور انتخاب خصوصیتی به‌عنوان خصوصیت جداساز، خصوصیت موردنظر باید کمترین عدم خلوص را ارائه دهد. در صورتی یک نود خالص است که تمام نمونه‌هایی که در یک شاخه از درخت واقع شده‌اند، در یک دسته باشند. معیارهای انتخاب خصوصیت، روش‌های ابتکاری هستند که برای انتخاب بهترین جداسازی به‌منظور تقسیم پایگاه داده به‌کار می‌روند. اگر داده‌های D به بخش‌های کوچک‌تر تقسیم شود، در حالت ایدئال باید این قسمت‌ها خالص باشند. معیارهای جداسازی به‌عنوان قوانین جداساز[14] هم شناخته می‌شوند؛ زیرا چگونگی جداسازی چندتایی‌ها را نیز تعیین می‌کنند. گره N چندتایی‌های D را نگاه می‌دارد. خصوصیتی که بیشترین بهرهٔ اطلاعاتی را دارد، به‌عنوان معیار جداسازی برای گره N انتخاب می‌شود. این خصوصیت، اطلاعات موردنیاز برای شناسایی چندتایی‌های درون گره‌های

1. Manhatan
2. Eughlo
3. Neuron
4. Communication link
5. Threshold
6. Feed-forward neural network
7. Recurrent neural networks
8. Radial basis function network
9. Neuro-fuzzy networks
10. Internal node
11. leaf
12. Flowchart
13.Branch
14. Splitting Rules

جدید را حداقل می‌کند. به این ترتیب حداقل عدم خلوص و تصادفی بودن چندتایی‌های درون قسمت‌ها ایجاد می‌شود. این روش، تعداد آزمون‌های موردنیاز برای دسته‌بندی چندتایی‌های مدنظر را به حداقل می‌رساند و موجب ایجاد درخت‌های تصمیم ساده‌تری می‌شود. از رابطه‌هایی که برای تعیین خصوصیت جداساز استفاده می‌شود، رابطهٔ اینتروپی[1] است که در زیر مشاهده می‌کنید [۱۷].

$$ENTROPY = -\sum_{i=1}^{k} p_m^i \, log_2 p_m^i \qquad (۲)$$

در پژوهش ارائه‌شده در [۱۵]، از دو الگوریتم درخت‌های رگرسیون و کلاس‌بندی (CART[2]) و روش درخت مدل[3] ازجمله الگوریتم‌های درخت تصمیم استفاده‌شده در حوزهٔ تخمین هزینهٔ نرم‌افزار معرفی و در مطالعه‌ای آزمایشی در همین حوزه استفاده شده است.

۲. ۱. ۴. شبکه‌های بیزین

یک شبکهٔ بیزی یا شبکهٔ باور یا شبکهٔ باوربیزی، یک گراف جهت‌دار غیر مدور است که مجموعه‌ای از متغیرهای تصادفی و نحوهٔ ارتباطِ مستقلِ آن‌ها را نشان می‌دهد؛ برای نمونه، یک شبکهٔ بیزی می‌تواند نشان‌دهندهٔ ارتباط بین علت بیماری‌ها با خود آن‌ها باشد. پس با داشتن عوامل، می‌توان احتمال یک بیماری خاص را در یک مریض تشخیص داد. شبکهٔ بیزین یک ابزار نسبتاً جدید برای شناسایی (هویت) روابط احتمالی، به‌منظور پیشگویی یا ارزیابی کلاس عضویت است. به‌طور خلاصه، می‌توان گفت شبکهٔ بیزین، نمایش با معنی روابط نامشخص مابین پارامترها در یک حوزه است. شبکهٔ بیزین، گراف جهت‌دار غیرحلقوی از نودها، برای نمایش متغیرهای تصادفی و کمان‌ها برای نمایش روابط احتمالی مابین متغیرها به‌شمار می‌رود [۱۸].

در [۱۹] به بررسی ۲۳ کار پژوهشی صورت‌گرفته طی سال‌های ۲۰۰۲ تا ۲۰۱۰ که براساس شبکه‌های بیزین در حوزهٔ تخمین هزینهٔ نرم‌افزار کار کرده‌اند، پرداخته شده است. برای درک بیشتر این نوع از الگوریتم‌های یادگیری ماشین استفاده‌شده در این حوزه، می‌توانید به این مقاله مراجعه کنید. در [۲۰] که پژوهشگران به بررسی روش‌های مختلفی روی دادگان محلی در این حوزه پرداخته‌اند، از این الگوریتم هم در مقایسات خود استفاده کرده‌اند.

۲. ۱. ۵. الگوریتم ژنتیک

فرض کنید مجموعه خصوصیات انسان توسط کروموزوم‌های او به نسل بعدی منتقل می‌شوند؛ برای مثال، ژن ۱ می‌تواند رنگ چشم باشد، ژن ۲ طول قد، ژن ۳ رنگ مو و... . حال اگر این کروموزوم به‌تمامی، به نسل بعد انتقال یابد، تمامی خصوصیات نسل بعدی شبیه به‌خصوصیات نسل قبل خواهد بود. بدیهی است که در عمل چنین اتفاقی رخ نمی‌دهد. درواقع، به‌صورت همزمان دو اتفاق برای کروموزوم‌ها می‌افتد. اتفاق اول جهش[4] است. جهش به این صورت است که بعضی ژن‌ها به‌صورت کاملاً تصادفی تغییر می‌کنند. البته تعداد این‌گونه ژن‌ها بسیار کم است، اما درهرحال، این تغییرِ تصادفی همان‌گونه که پیش‌تر دیدیم، بسیار مهم است؛ مثلاً ژن رنگ چشم می‌تواند به‌صورت تصادفی باعث شود تا در نسل بعدی، یک نفر دارای چشمان سبز باشد، درحالی‌که تمامی نسل قبل دارای چشم قهوه‌ای بوده‌اند. علاوه بر جهش، اتفاق دیگری که می‌افتد و البته این اتفاق به تعداد بسیار بیشتری نسبت‌به جهش رخ می‌دهد، چسبیدن دو کروموزوم از طول به یکدیگر و تبادل برخی قطعات بین دو کروموزوم است. این مسئله با نام معبر[5] شناخته می‌شود. این همان چیزی است که باعث می‌شود تا فرزندان، ترکیب ژن‌های متفاوتی را (نسبت‌به والدین خود) به فرزندان خود انتقال دهند [۲۱].

معمولاً الگوریتم‌های ژنتیک یک عدد احتمالِ اتصال دارند که احتمال به‌وجودآمدن فرزند را نشان می‌دهد. ارگانیسم‌ها با این احتمال دوباره باهم ترکیب می‌شوند. اتصال دو کروموزوم، فرزند ایجاد می‌کند که به نسل بعدی اضافه می‌شوند. این کارها انجام می‌شوند تا اینکه کاندیداهای مناسبی برای جواب، در نسل بعدی پیدا شوند. مرحلهٔ بعدی تغییر دادن فرزندان جدید است. الگوریتم‌های ژنتیک یک احتمال تغییر کوچک و ثابت دارند که معمولاً درجه‌ای در حدود ۰/۰۱ یا کمتر دارد. براساس این

1. Entropy
2. Classification and Regression Trees
3. Model tree technique
4. Mutation
5. Crossover

احتمال، کروموزوم‌های فرزند به‌طور تصادفی تغییر می‌کنند یا جهش می‌یابند؛ به‌خصوص با جهش بیت‌ها در کروموزوم ساختمان دادهٔ انسان. این فرایند باعث به‌وجودآمدن نسل جدیدی از کروموزوم‌هایی می‌شود که با نسل قبلی متفاوت است. کل فرایند برای نسل بعدی هم تکرار می‌شود. جفت‌ها برای ترکیب انتخاب می‌شوند، جمعیت نسل سوم به‌وجود می‌آیند و این عمل به همین شکل ادامه می‌یابد. این فرایند تکرار می‌شود تا به آخرین مرحله برسیم [۲۱]. شرایط خاتمهٔ الگوریتم‌های ژنتیک عبارت‌اند از اینکه به تعداد ثابتی از نسل‌ها برسیم، بودجهٔ اختصاص داده‌شده تمام شود (زمان محاسبه/پول)، یک فرد (فرزند تولیدشده) پیدا شود که مینیمم (کمترین) ملاک را برآورده کند، بیشترین درجهٔ برازش فرزندان حاصل شود یا دیگر نتایج بهتری حاصل نشود، بازرسی دستی و یا اینکه از ترکیب‌های بالا استفاده شود.

این الگوریتم از نوع الگوریتم‌های یادگیری نظارتی است که در آن، از قانون دلتا استفاده می‌شود. در این الگوریتم، به یک آموزگار نیاز داریم که خروجی مناسب برای یک ورودی را بداند یا اینکه بتواند آن را حساب کند [۲۱].

محققان دیگری در [۲۲] به ارائهٔ الگوریتمی ترکیبی بر پایهٔ دو الگوریتم یادگیری ماشین ژنتیک و شبکهٔ عصبی برای تخمین هزینهٔ نرم‌افزار پرداخته‌اند. الگوریتم ژنتیک برای مسائل بهینه‌سازی کاربرد دارد. در این مقاله، هدف آن است که از طریق الگوریتم ژنتیک، فقط ورودی‌هایی که عامل مهمی برای هزینه هستند یا پیچیدگی محاسباتی فراوانی ندارند، انتخاب شوند و به‌عنوان ورودی به شبکهٔ مصنوعی داده شوند.

۲. ۱. ۶. برنامه‌نویسی ژنتیک

برنامه‌نویسی ژنتیک یکی از روش‌های در حال ظهور در تخمین هزینهٔ نرم‌افزار است. ایدهٔ اصلی استفاده‌شده مبتنی بر تئوری تکاملی داروین است که می‌گوید عملیات ژنتیک بین کروموزوم‌ها درنهایت، به افرادی مشخص که احتمال بیشتری برای زنده ماندن دارند، منجر می‌شود. GP درواقع توسعه‌یافتهٔ GA است که محدودیت کروموزوم نشان‌دهندهٔ فردی را که دارای یک رشتهٔ دودویی ثابت است، حذف کرده است [۳]. در [۲۳] پژوهشگران از برنامه‌نویسی ژنتیک برای ارائهٔ تخمین هزینهٔ توسعهٔ نرم‌افزار، آن‌هم به‌صورت دقیق، استفاده کرده و آن را با مدل رگرسیون آماری و الگوریتم شبکهٔ عصبی مقایسه نموده‌اند.

۲. ۱. ۷. قوانین وابستگی

ما به‌دنبال یافتن قوانین وابستگی[1] از روی مجموعهٔ داده‌های موجود در دادگان مورداستفاده برای ارزیابی هستیم. یک قانون وابستگی را می‌توان به‌صورت A→B نشان داد که در آن A و B زیرمجموعه‌های غیر تهی از I (بیانگر مجموعهٔ اقلام) هستند و هیچ اشتراکی با همدیگر ندارند. به‌طوری‌که اگر A و احتمال وقوع آن منجر شود که احتمال وقوع B هم بیش از یک حدی باشد که مورد علاقهٔ ماست، آنگاه A و B هر دو، گزاره‌هایی از این نقش خواهند بود. به زبان ساده‌تر، یعنی اینکه ما به‌دنبال یافتن رابطهٔ رخ دادن دو گزاره باهم هستیم. اطمینان[2] و پشتیبان[3] از معیارهای سنجش موردعلاقه بودن قوانین محسوب می‌شوند و درواقع، قطعیت و کاربرد قوانین شناسایی‌شده را می‌سنجند. قانون A→B دارای پشتیبان S در D است که S درصدی از تراکنش‌های D است که حاوی $A \cup B$ باشد و می‌توان آن را با احتمال $p(A \cup B)$ نیز نشان داد. اطمینان A→B نیز با C در مجموعه تراکنش D نشان داده می‌شود که نشان‌دهندهٔ درصدی از تراکنش‌های D است که شامل A و همچنین B است. اطمینان را نیز می‌توان با احتمال شرطی $p(A|B)$ نشان داد [۱۷]. به‌این‌ترتیب، رابطه‌ها به شکل رابطه‌های (۳) و (۴) هستند.

$$\text{پشتیبان}(A \rightarrow B) = p(A \cup B) \quad (۳)$$

$$\text{اطمینان}(A \rightarrow B) = p(A|B) \quad (۴)$$

محققانی در [۲۴] به ارائهٔ مدلی از ترکیب قوانین وابستگی و درخت تصمیم CART پرداخته‌اند. آن‌ها با آزمایش‌های خود روی دادگان مختلف، نشان داده‌اند که استفاده از این الگوریتم‌ها سبب افزایش دقت در این حوزه از تخمین در مهندسی نرم‌افزار می‌شود.

1. Association Rules
2. Confidant
3. Support

۲. ۱. ۸. رگرسیون بردار پشتیبان

رگرسیون بردار پشتیبان[1] یا SVR یکی دیگر از روش‌های یادگیری ماشین پرکاربرد در زمینهٔ تخمین هزینه و تلاش موردنیاز برای توسعهٔ محصول نرم‌افزاری محسوب می‌شود. در این نوع از روش‌های یادگیری ماشین، فرضیه‌ای[2] برای جداسازی نمونه‌های یک دسته از دسته‌های دیگر را مشخص می‌کنیم. در رویکرد نرم این نوع روش، فرضیهٔ خود را با میزان مشخصی تخطی از حاشیهٔ[3] تعیین‌شده برای آن، در نظر می‌گیریم. درواقع ایدهٔ اصلی رویکرد نرم، این است که خطاهای کمتر از یک حد آستانهٔ تعیین‌شده و همین‌طور خطاهای نزدیک به صفر را نادیده بگیریم [۱۴]. در [۱۵] از ماشین‌های بردار پشتیبان حداقل مربعات (LS-SVM[4]) به نمایندگی از روش‌های رگرسیون بردار پشتیبان، استفاده شده و این روش با سایر روش‌های استفاده‌شده در این پژوهش ازجمله ols[5]، ترکیب ols و تغییر شکل لگاریتمی[6]، رگرسیون قوی[7]، رگرسیون ols به‌همراه تغییر شکل BC[8]، رگرسیون میانگین مربعات[9] و دو الگوریتم MARS[10] و رگرسیون RIDGE مورد مقایسه قرار گرفته است.

۲. ۲. بستر مطلوب هریک از روش‌های یادگیری ماشین

به‌منظور انتخاب یک مدل از میان مدل‌های ممکن، برای تخمین تلاش توسعهٔ نرم‌افزار باید به نقاط ضعف و نقاط قوّت مربوط‌به آن‌ها توجه کرد و در انتخاب روش، این ویژگی‌ها را مدنظر قرار داد. در بررسی‌هایی که در [۴] روی مطالعات انتخابی انجام شده است، خصوصیات روش‌های یادگیری ماشین، عمدتاً در چهار بستر[11] از زمینه‌های تخمین، قابل دسترسی و مرتبط هستند. این بسترها شامل دادگان کوچک، داده‌های پرت، خصوصیات دسته‌ای[12] و داده‌های گم‌شده[13] هستند. خصوصیات روش‌های ML را طبق آنچه در جدول (۳) وجود دارد، می‌توان شرح داد. علامت‌های استفاده‌شده در این جدول بدین شرح هستند که * به‌معنای مطلوب، + به‌معنای نامطلوب و – به‌معنای ذکرنشده است.

جدول (۳): خصوصیات روش‌های ML [۴]

	دادگان کوچک	دادهٔ پرت	ویژگی‌های دسته‌ای	داده‌های گم‌شده
CBR	*	*	+	+
ANN	+	*	+[14]	-
DT	-	*	*	-
BN	*	*[17]	*[16]	*[15]

از جدول (۳) می‌توان به این نتایج رسید که روش DT مستعد بیش برازش[18] روی دادگان آموزشی کوچک است. دو روش ANN و CBR تا زمانی که ویژگی‌های دسته‌ای به ویژگی‌های کمّی تبدیل نشوند، نمی‌توانند با آن‌ها به فرم استانداردشان کار کنند. مثل دو روش ANN و CBR، روش DT هم نمی‌تواند تا زمانی که به داده‌های گم‌شده مقدار نسبت داده نشده، نمی‌تواند با آن‌ها کار کند. علاوه بر نتایج حاصل از جدول بالا می‌توان گفت دو روش CBR و DT بصری و به‌آسانی قابل درک هستند. روش ANN توانایی یادگیری توابع پیچیده‌ای را دارد، اما مسائلی که باید در نظر گرفت، بدین شرح است که به مقادیر زیادی از داده‌های آموزشی نیازمند است، اغلب از مسئلهٔ پیش‌پرازش رنج می‌برد و همچنین توانایی توضیحی آن ضعیف است. روش BN قادر به یادگیری روابط علّت و معلولی است. برای سه روش SVR، GP و AR اطلاعات چندانی در دست نیست [۴].

۳. ۲. مروری مقایسه‌ای بر مدل‌های مختلف ارائه شده

همان‌طور که در [۴] آمده، اوج انتشارات در سال ۲۰۰۸ است و به‌نظر می‌رسد که از دو روش CBR و ANN در مقایسه با سایر روش‌های یادگیری ماشین، در سال‌های اخیر بیشتر استفاده شده

1. Support Vector Regression
2. Hypothesis
3. Margin
4. Least squares support vector machines
5. Ordinary least squares regression
6. OLS regression with log transformation
7. Robust regression
8. Box Cox transformation
9. Least median squares regression
10. Multivariate Adaptive Regression Splines
11. Contexts
12. Categorical
13. Miss value
14. تنها توسط مقالهٔ ۶۸ از مطالعات انتخابی پشتیبانی می‌شود.
15. تنها توسط مقالهٔ ۶۷ از مطالعات انتخابی پشتیبانی می‌شود.
16. تنها توسط مقالهٔ ۷۸ از مطالعات انتخابی پشتیبانی می‌شود.
17. تنها توسط مقالهٔ ۷۸۰ از مطالعات انتخابی پشتیبانی می‌شود.
18. Over-fitting

است. البته به این نکته هم باید توجه داشت که در برخی از مطالعات، بیش از یک روش یادگیری ماشین، مورد مطالعه قرار گرفته است.

۲. ۳. ۱. مدل‌های ML در مقایسه با هم

در مقالهٔ [۴] مقایساتی بین الگوریتم‌های یادگیری ماشین استفاده‌شده در تخمین هزینهٔ نرم‌افزار صورت پذیرفته است. از این مقایسات، سه نتیجهٔ به‌دست می‌آید:

- روش‌های CBR و ANN دقیق‌تر از روش DT هستند و این نتیجه توسط بیشتر آزمایش‌ها در طول مقایسه اثبات شده است.
- در مواردی از مقایسات بین دو روش CBR و ANN، به‌نظر می‌رسد که روش CBR از روش ANN دقیق‌تر است. بااین‌حال، این نتایج با آنچه در مقایسهٔ دقت بین روش‌های ML گفته شده، که روش ANN از روش CBR بهتر است، تناقض دارد. این تناقض ممکن است به سه دلیل رخ داده باشد: اول اینکه تمام آزمایش‌های انتخابی که حاکی از برتری CBR نسبت‌به ANN است، از مطالعاتی که روی روش CBR کار کرده‌اند انتخاب شده‌اند و همین امر ممکن است باعث بایاس[1] شدن مقایسات شده باشد؛ دوم اینکه مطالعاتی که روی روش ANN کار کرده‌اند، به‌ندرت آزمایشی برای مقایسهٔ آن با روش CBR انجام داده‌اند، به‌طوری‌که مطالعاتی که دقت بیشتری را برای این روش گزارش داده‌اند، کمکی به این مقایسه نکرده‌اند؛ دلیل سوم هم این است که تعداد آزمایش‌های مربوط‌به CBR دو برابر آزمایش‌های مربوط‌به ANN برای این مقایسه است.
- برای مقایسهٔ بین مدل‌های ML، تعداد آزمایش‌ها نسبتاً کم است و برخی از مقایسات حتی در تناقض با هم هستند.

بنابراین در مقایسهٔ آن دسته از روش‌های ML که به‌ندرت با یکدیگر مقایسه شده‌اند، تعیین دقیق‌ترین آن‌ها کار نسبتاً سختی خواهد بود. با توجه به نتایج متناقض مقایسهٔ بین مدل‌های ML و مدل‌های غیر ML که در این بررسی آمده است، در این مقاله توصیه شده که هر دو نوع مدل را به‌صورت موازی در مراحل اولیهٔ کار استفاده کنید. تنها زمانی که مدل ML به‌طور قابل‌توجه و پیوسته‌ای بهتر از سایر مدل‌های موجود بود، می‌توان به متخصصان پیشنهاد کرد که مدل‌های موجود را با تمرکز بر مدل ML و تغییر به سیستم تخمین جدید تشویق کنند. نکتهٔ مهمی که باید همیشه مدنظر داشت، این است که قبل از انتخاب مدل ML، متخصصان نه‌تنها به آگاهی از بسترهای تخمین نیاز دارند، بلکه به فهم دقیق ویژگی‌های مدل‌های ML کاندید هم نیازمندند. معمولاً میزان تطبیق بین بسترهای تخمین و ویژگی‌های مدل ML انتخابی، تأثیر مستقیم و قابل‌توجهی روی عملکرد مدل دارند [۴].

۲. ۴. پیش‌پردازش‌های استفاده‌شده

در برخی از کارهای پژوهشی قبل از اینکه از دادگان استفاده کنند، به‌منظورهای خاصی ازجمله افزایش دقت تخمین، از روش‌هایی استفاده کرده‌اند. همان‌طور که قبلاً هم بیان شد، در این حوزه از مهندسی نرم‌افزار از الگوریتم ژنتیک برای وزن‌دهی به ویژگی‌های دادگان مورداستفاده در تخمین هزینهٔ نرم‌افزار استفاده شده است.

دولت‌آبادی‌نژاد به‌همراه همکارانش در [۲]، به بررسی تأثیر کاهش انحراف توزیع تلاش بر دقت تخمین تلاش توسعهٔ نرم‌افزاری پرداخته‌اند. کاهش انحراف توزیع تلاش، از ایجاد سربارهای مالی و زمانی در توسعهٔ محصول نرم‌افزاری جلوگیری می‌کند. ازجمله مواردی که گروه مدیریت بایستی پس از تخمین تلاش به انجام برسانند، توزیع تلاش در بین فازهای توسعهٔ محصول نرم‌افزاری است. انحراف توزیع تلاش نیز مانند خود تخمین تلاش و هزینهٔ توسعهٔ نرم‌افزار، یکی از چالش‌های پیشِ روی گروه مدیریت توسعهٔ نرم‌افزاری محسوب می‌شود. جهت توزیع تلاش نیز مدل‌های گوناگونی تاکنون ارائه گردیده است. آن‌ها پس از انجام آزمایش‌ها به این نتیجه رسیده‌اند که کاهش انحراف توزیع تلاش، در بالا بردن دقت، تأثیر مثبتی خواهد داشت. در پایان این پژوهش، به بیان دو راهکار مدیریتی برای توزیع و تخمین تلاش در مدیریت توسعهٔ نرم‌افزار پرداخته شده است: اول اینکه اگر گروه مدیریت پروژه، با درنظرگرفتن موارد سایز، دامنه و نوع توسعه، اقدام به توزیع تلاش در فازهای

1. Bias

توسعهٔ محصول نرم‌افزاری کنند، می‌تواند باعث کاهش انحراف توزیع تلاش شده و از به‌وجودآمدن سربارهای مالی و زمانی جلوگیری کند. دوم اینکه استفاده از مدل‌های توزیع تلاش، با درنظرگرفتن ویژگی‌های مؤثر در انحراف توزیع تلاش، درنهایت می‌تواند باعث افزایش دقت تخمین تلاش در توسعهٔ محصول نرم‌افزاری گردد و گام مهمی در مدیریت بهینهٔ پروژهٔ نرم‌افزاری محسوب شود [۲].

از بررسی نتایج حاصل از [۹] که اخوان بیطرف و همکارانش روی روش‌های یادگیری ماشین برای پیمایش دادگان در تخمین هزینهٔ نرم‌افزار کار کرده بودند، می‌توان به این نتیجهٔ کلی رسید که پیمایش با توجه به کاهش کمینهٔ میانگین خطا، اقدام درستی بوده و باعث شده که تخمین‌ها به مقدار واقعی نزدیک‌تر شوند. افزایش مقدار بیشینهٔ میانگین خطا هم، بدین شکل توجیه شده که با توجه به عدم حساسیت به داده‌ها ممکن است تعداد معدودی از داده‌ها، دارای اختلاف زیاد خروجی با مقدار واقعی باشند.

خطیبی بردسیری به‌همراه همکارانش در [۲۵] برای افزایش دقت تخمین تلاش توسعهٔ نرم‌افزار، دست به استفاده از خوشه‌بندی پروژه‌ها زده‌اند. در ابتدای کار، به این نکته پرداخته شده است که اساساً ویژگی‌های پروژه‌های نرم‌افزاری، خیلی پیچیده، غیرخطی و غیرقابل‌فهم هستند و این به‌دلیل نامشخص و بی‌ثبات بودن طبیعت پروژه‌های نرم‌افزاری است. روش‌های غیر الگوریتمی موجود از یک دادگان، شامل پروژه‌های نرم‌افزاری متنوع به‌منظور تخمین استفاده می‌کنند. در این‌چنین دادگانی پروژه‌های نامرتبط و متناقض زیادی وجود دارد. اصطلاحاً در بحث داده‌کاوی به این نوع داده‌ها، داده‌های پرت می‌گوییم. روش‌های یادگیری ماشین، برای خود یک مدل آموزشی با استفاده از داده‌های موجود در دادگان می‌سازند و براساس این مدل آموزشی، عملیات تخمین را انجام می‌دهند. واضح است که وجود دادهٔ پرت، کیفیت آموزش این الگوریتم‌ها را تحت‌تأثیر قرار داده و سبب تخمین‌های غیردقیق و بی‌ثبات خواهد شد. ازاین‌رو، این مقاله با هدف رفع این مشکل نوشته شده و این مشکل را از طریق خوشه‌بندی[1] پروژه‌ها حل کرده است. در ادامه، روش‌های خوشه‌بندی فازی و تخمین بر پایهٔ مقایسه، که اساس کار این روش‌هاست، به‌صورت مختصر شرح داده شده است.

خوشه‌بندی داده‌ها روشی برای تقسیم همهٔ داده‌ها در چندین خوشه است به قسمی که داده‌های مشابه در یک خوشه باشند و خوشه‌ها تا حد ممکن از یکدیگر مجزا باشند. ایدهٔ اصلی خوشه‌بندی فازی این است که «یک داده می‌تواند در چندین خوشه باشد». ازاین‌رو، یک سطح عضویت برای تعیین چگونگی وجود یک داده در یک خوشه تعریف می‌شود. مقدار سطح عضویت از صفر تا یک متغیر است. خوشه‌بندی C-MEANS فازی[2] مهم‌ترین روش خوشه‌بندی فازی است که در سال ۱۹۸۱ مطرح شد. هدف این الگوریتم به حداقل رساندن تابع تعلق است و از آن برای حل مسائل تشخیص و پیش‌بینی استفاده می‌کنند. الگوریتم خوشه‌بندی C-Means فازی در ادامهٔ این مقاله شرح داده شده است.

خوشه‌بندی پروژه‌های نرم‌افزاری، بخش کلیدی روش تخمین مطرح‌شده در این مقاله است. در اولین مرحله از این روش، از روش خوشه‌بندی فازی که براساس درجهٔ شباهت هر پروژه به خوشه‌ها عمل می‌کند، استفاده شده است. پیشنهاد پژوهشگران این مقاله برای غلبه بر مشکل داده‌های پرت، آموزش شبکهٔ عصبی با استفاده از خوشه‌های به‌دست‌آمده است. پس از مقداردهی دادگان با خوشه‌ها، ممکن است خوشه‌هایی شامل چند پروژه که مناسب و قابل‌استفاده نیستند، در مجموعهٔ آموزشی شبکهٔ عصبی ظاهر شوند. این مشکل ممکن است روش پیشنهادی را مجبور به نادیده‌گرفتن این خوشه‌ها کند. برای اجتناب از این محدودیت، ABE[3] به روش پیشنهادی اضافه شده است؛ زیرا این روش می‌تواند در هر دادگانی، با هر تعداد پروژه‌ای برای تخمین استفاده شود. بنابراین با ترکیب شبکه‌های عصبی و ABE همهٔ پروژه‌های موجود در دادگان می‌توانند در روند تخمین مشارکت کنند. به‌منظور شرح روند ترکیب سه بخش مطرح‌شده، روش ارائه‌شده در این کار پژوهشی، در دو فاز اصلی به نام‌های آموزش و آزمایش، سازمان‌دهی شده است. در مرحلهٔ

1. Clustring

2. Fuzzy C-Means Clustering
3. Analogy Based Estimation

آموزش، ساختار روش ترکیبی مطرح‌شده پیکربندی شده است. در مرحلهٔ آزمون هم روند تخمین تلاش انجام می‌شود.

برخی دیگر از محققان در مقالات خود [۲۶ و ۲۷] از روش نرمال‌سازی MIN-MAX به‌عنوان پیش‌پردازش دادگانِ خود استفاده کرده‌اند. برخی هم مانند [۱۳] از روش کاهش بعد PCA به‌عنوان پیش‌پردازشی روی دادگان خود استفاده کرده و نشان داده‌اند که این عمل باعث افزایش دقت و کاهش میانگین خطای مقدار تخمین زده‌شده، با مقدار واقعی هزینهٔ موردنیاز برای توسعهٔ محصول نرم‌افزاری می‌شود.

در کاری که در [۹] انجام شده، از الگوریتم MP5 به‌عنوان پیش‌پردازشی برای تعیین وزن خصوصیات در شبکهٔ عصبی استفاده شده است. در این کار پژوهشی، چون درخت‌های تصمیم به‌دنبال بهترین خصوصیات برای ارائهٔ جداسازی با کمترین عدم خلوص هستند، تأثیر خصوصیات مشخص می‌شود. براساس آزمایش‌های نشان‌داده‌شده در این مقاله، اثبات شده که این روش برای تعیین وزن هم مناسب است.

دسته‌ای دیگر از مقالات [۲۸] از انتخاب ویژگی به‌منظور پیش‌پردازشی روی دادگان مورداستفاده، برای افزایش دقت تخمین هزینهٔ نرم‌افزار و ارائهٔ نزدیک‌ترین مقدار تخمین‌زده‌شده به میزان تلاش واقعی نمونه‌های موجود در دادگان استفاده کرده‌اند. البته این دسته از الگوریتم‌های یادگیری ماشین، انواع مختلفی دارند. نوعی از آن‌ها از یک ویژگی شروع و هر بار ویژگی‌ها اضافه می‌شوند تا زمانی که یا تأثیری در دقت نداشته باشند یا اینکه دقت را کاهش دهند. به این نوع از انتخاب ویژگی روبه‌جلو می‌گویند. دستهٔ دیگر، از همهٔ ویژگی‌ها کار را شروع می‌کند و برعکس حالت قبل، هر بار یکی از ویژگی‌ها را کم می‌کند. شرط اتمام این دسته هم مانند دستهٔ اول است. به این دسته از الگوریتم‌های انتخاب ویژگی روبه عقب می‌گویند. این دو دسته الگوریتم‌های معرفی‌شده از دسته متدهای بسته‌بندی هستند. علاوه بر این متدها، متدهای فیلتر و یکپارچه‌سازی نیز از انواع متدهای انتخاب ویژگی محسوب می‌شوند.

۵.۲. سایر کارهای انجام شده

درک و محاسبهٔ مدل‌های مبتنی بر داده‌های قدیمی، با توجه به روابط پیچیده‌ای که بین خصوصیات وجود دارد، مشکل است. ویژگی‌ها و ارتباطات استفاده‌شده برای تخمین تلاش توسعهٔ نرم‌افزار، می‌تواند در طول زمان و برای محیط‌های توسعهٔ متفاوت، تغییر کند. به‌منظور رسیدگی و غلبه بر این مشکلات، یک مدل جدید با تخمین دقیق، موردنیاز خواهد بود. در همین راستا در مقالهٔ پژوهشی [۲۹] نسبت‌به ارائهٔ یک مدل تخمین هزینهٔ نرم‌افزارِ جدید اقدام نموده‌اند. پژوهشگرانی در [۳] به معرفی مدل‌های تخمینی که تاکنون به‌صورت موفقیت‌آمیزی مورد استفاده قرار گرفته‌اند، پرداخته‌اند. پژوهشگران در این مقاله آورده‌اند که چندین مدل تاکنون برای تخمین هزینهٔ نرم‌افزار ارائه شده‌اند، اما تصمیم‌گیری دربارهٔ اینکه چه مدلی را انتخاب کنیم، سخت است. بنابراین برای حل این مشکل باید شناخت کاملی دربارهٔ روش‌های مختلف داشته باشیم. نتایج تحقیقات نشان می‌دهند که بین ۳۰ تا ۴۰ درصد پروژه‌های نرم‌افزاری به‌شکلی کامل، تکمیل شده و سایر پروژه‌ها شکست می‌خورند. هدف اصلی در انتخاب مدل، به‌حداکثررسانی دقت است. کمار و همکارانش روش‌های تخمین در حال ظهور مختلفی را که به‌عنوان روش‌های تکی، آن‌هم برای همهٔ شرایط بهترین هستند، آنالیز کرده‌اند.

پژوهشگران دیگری در مقالهٔ [۲۷] روی جنبه‌های کمّی استفاده از روش‌های یادگیری ماشین، به‌منظور ساخت مدل‌هایی برای تخمین تلاش نرم‌افزار، بررسی‌هایی انجام و نتایج را تشریح کرده‌اند. روش‌های انتخابی در این مقاله، رگرسیون خطی چندگانه[1]، رگرسیون LOGISTIC و CART هستند. رگرسیون خطی چندگانه به پیداکردن ارتباط بین متغیرهای وابسته و متغیرهای مستقل کمک می‌کند. این روش، نوعی از رگرسیون خطی است که به متغیرهای مستقل چندگانه می‌پردازد. در پایان این مقاله نتیجه گرفته شده است که هر روشی روی یک دادگان خوب عمل می‌کند.

همان پژوهشگران ایرانی که تأثیر کاهش انحراف توزیع تلاش را بررسی کرده‌اند، در مقاله‌ای دیگر [۳۰] روش منطق فازی در تخمین هزینه و تلاش نرم‌افزار را بررسی نموده‌اند. آن‌ها برای رسیدن به این هدف، به بررسی جایگاه استفاده از این روش در

1. Multiple Linear Regression

تخمین نرم‌افزاری با مطالعهٔ ۶۴ مقاله از مقالات منتشره در سال‌های ۱۹۹۵ تا ۲۰۱۲ پرداخته و نتایج مقایسه‌ای خود را در [۳۰] ارائه داده‌اند.

پژوهش دیگری که در این حوزه صورت گرفته و منجر به مقالهٔ پژوهشی [۳۱] شده است، مطالعه‌ای روی پیش‌بینی تلاش نرم‌افزار با استفاده از روش‌های یادگیری ماشین را نشان می‌دهد. پژوهشگران در مطالعات خود در این مقاله، دو دسته روش یادگیری نظارتی و غیرنظارتی را برای پیش‌بینی تلاش نرم‌افزار با استفاده از دادگان قدیمی معرفی کرده‌اند. پژوهشگران در این مقاله معتقدند که اگرچه مدل‌های یادگیری ماشین ممکن است بهترین راه‌حل را ارائه ندهند، می‌توان حداقل از آن‌ها برای تکمیل سایر مدل‌ها توسط مدیران استفاده کرد. تخمینِ دقیق تلاش توسعهٔ نرم‌افزار، مخصوصاً در بازار به‌شدت رقابتی نرم‌افزار، قطعاً روی موفقیت پروژهٔ نرم‌افزاری تأثیرگذار است.

محققان در مهندسی نرم‌افزار، بر این فرضیه تأکید دارند که پروژه‌های با خصوصیات مشابه، تلاش تقریباً معادلی خواهند داشت. در یادگیری غیرنظارتی، از نظر یادگیری ماشین، خوشه‌بندی پروژه‌های نرم‌افزاری براساس یک زیرمجموعهٔ تصادفی می‌تواند اطلاعات موردنیاز برای خصوصیات مشخص‌نشده را فراهم کند. اگر ما تلاش را به‌عنوان یک خصوصیت در دادگان بشناسیم، می‌توان آن را با خوشه‌بندی پروژه‌ها براساس دیگر خصوصیات استنباط کرد. در این کار پژوهشی، از روش خوشه‌بندی K-MEDOID با سه معیار شباهت فاصله[1]، جاکارد[2] و ضرایب کالزینسکی[3]، برای اندازه‌گیری شباهت بردارهای بولین که ارائه‌دهندهٔ پروژه‌های نرم‌افزاری است، به‌عنوان نماینده‌ای برای یادگیری غیرنظارتی استفاده کرده‌اند. سه روش درخت تصمیم J48، بیز ساده و BPNN[4] به‌عنوان نمایندگانی برای یادگیری نظارتی انتخاب و پیاده‌سازی شده‌اند [۳۱].

پژوهشگران در مقالهٔ [۱۱]، به مطالعهٔ آزمایشی روی ترکیبات همگن و ناهمگن روش‌های یادگیری ماشین در زمینهٔ تخمین هزینهٔ نرم‌افزار پرداخته‌اند. در این مقاله بیان شده است که بااینکه تاکنون متدهای تخمین زیادی ارائه شده است، هیچ‌کدام از آن‌ها نتیجهٔ یکسانی روی همهٔ دادگان ندارند. از سال ۱۹۸۰ تاکنون، متدهای زیادی در این زمینه ارائه شده و مانند مقالات قبلی این مقاله هم متدهای ارائه‌شده را در سه دستهٔ مدل‌های الگوریتمی، مبتنی بر قضاوت کارشناسان و هوش محاسباتی که مقالات قبلی از آن با عنوان یادگیری ماشین یاد کرده بودند، دسته‌بندی کرده است. این مقاله گروه‌های همگن و ناهمگنی از ترکیب مدل‌های یادگیری ماشین برای تخمین تلاش توسعهٔ نرم‌افزار ارائه داده است.

۳. دادگان مورداستفاده

تخمین هزینهٔ نرم‌افزار در ابتدا و قبل از ارائهٔ مدل‌های تخمین هزینهٔ نرم‌افزار، با استفاده از یک سری قوانین سرانگشتی و برخی الگوریتم‌های ساده انجام می‌شده است. تا اینکه مدل تابع نقطه‌ای توسط آلبرت[5] معرفی شد. این مدل در حال حاضر هم از درجه اطمینان فوق‌العاده‌ای برخوردار است. آنالیز تابع نقطه‌ای متدی است مرتبط با کمیت سایز و پیچیدگی یک سیستم نرم‌افزاری یا به‌اصطلاح، توابع مورداستفاده برای نوشتن کد مربوط‌به هر نمونه پروژه است که به کاربر تحویل داده می‌شود. این توابع به زبان برنامه‌نویسی مورداستفاده و یا ابزارهای استفاده‌شده برای توسعهٔ پروژه‌های نرم‌افزاری وابسته نیستند. از آن زمان تاکنون، مطالعات زیادی برای تنظیم پارامترها و وزن‌های آن با هدف منعکس‌کردن پیشرفت‌های جاری در زمینهٔ فناوری و صنعت توسعهٔ نرم‌افزار صورت گرفته است. مدل‌هایی ازاین‌دست در دستهٔ مدل‌های الگوریتمی قرار می‌گیرند.

پژوهشگران در [۳۲] به مرور مدل‌های الگوریتمی ارائه‌شده در این زمینه پرداخته‌اند. همان‌طور که در شکل (۳) می‌بینید، این دسته از متدها، خود به چهار دسته تقسیم‌بندی شده‌اند. مدل ارائه‌شده توسط بیلی و بیزلی[6] را می‌توان نمونه‌ای از مدل‌های غیرخطی نام برد که در سال ۱۹۸۱ ارائه شده است [۳۳]. دستهٔ دیگری از مدل‌های الگوریتمی دربرگیرندهٔ مدل‌های گسستهٔ

1. Distance
2. Jaccard
3. Kulzinsky coefficients
4. Back Propagation neural network

5. albercht
6. John W. Bailey and Victor R. Basili

تخمین هزینهٔ نرم‌افزار هستند. مدل ولورتن[1] نمونه‌ای از این دسته مدل‌های الگوریتمی است که ولورتن در ۱۹۷۴ ارائه کرده است [۳۴]. دستهٔ سوم از مدل‌های الگوریتمی، مدل‌های تابعی نامیده می‌شوند. مدل پوتنم[2] ارائه‌شده توسط پوتنم و همکارش در سال ۱۹۷۸ از دسته مدل‌های تابعی است [۳۵]. دستهٔ آخر از مدل‌های الگوریتمی با عنوان مدل‌های چندگانه دسته‌بندی شده‌اند. مدل والستون فلیکس[3] که والستون و فلیکس در سال ۱۹۹۷ ارائه کرده‌اند، ازجمله مدل‌های الگوریتمی چندگانه محسوب می‌شود [۳۶].

ادامهٔ ارائهٔ مدل‌های تخمین هزینهٔ نرم‌افزار، در دههٔ ۱۹۶۰ فرانک فریمن مفاهیم مدل‌های تخمین پارامتریک را توسعه داد که منجر به توسعهٔ مدل قیمت برای سخت‌افزار شد. در ادامه و در دههٔ ۱۹۷۰، با استفاده از روش‌های آماری، مبادرت به شناخت ویژگی‌های مؤثر بر هزینهٔ توسعهٔ نرم‌افزار با استفاده از رگرسیون و وابستگی کردند.

مدل آنالیز تابع نقطه‌ای یکی از اولین مدل‌های ارائه‌شده در زمینهٔ تخمین هزینهٔ نرم‌افزار است که برای تعیین کمیت سایز و پیچیدگی نرم‌افزار مورداستفاده قرار گرفته است. این مدل در ادامهٔ پیشرفت‌های تخمین هزینهٔ نرم‌افزار به‌عنوان یکی از عوامل هزینه هم مدنظر قرار گرفته است [۱۰].

تعداد توابع نقطه‌ای به انواع متمایزی از موارد از نظر فرمت و منطق وابسته است. تعیین تعداد توابع نقطه‌ای شامل دو مرحله است:

الف. شمارش توابع کاربر

تعداد توابع خام با درنظرگرفتن پنج جزء عمومی نرم‌افزار مشخص می‌شود. این اجزای نرم‌افزاری شامل ورودی‌های خارجی، خروجی‌های خارجی، استعلامات خارجی، فایل‌های داخلی منطقی و رابط خارجی‌اند که با سه سطح پیچیدگی ساده، متوسط و پیچیده، مقداردهی می‌شوند. مقدار پیچیدگی تعیین‌شده برای هر جزء، تعیین‌کنندهٔ وزن هر جزء است. مجموع وزن تعیین‌شده تعداد توابع کاربر (UFC[4]) را تعیین می‌کند و به‌عنوان ورودی به مرحلهٔ بعد ارائه می‌شود.

ب. تنظیم پیچیدگی پردازش محیطی

تابع نقطه‌ای نهایی با ضرب UFC در یک عامل تنظیم که با درنظرگرفتن ۱۴ ویژگی سیستمی عمومی تعیین می‌گردد، محاسبه می‌شود. این ۱۴ ویژگی به‌همراه شرح آن‌ها در جدول (۴) آمده است. عامل تنظیم ارزش (VAF[5]) با درنظرگرفتن این ۱۴ ویژگی که نرخ عمومی عملکرد برنامه‌ها را تعیین می‌کنند و با ارزشی که با مقدار C که تعیین‌کنندهٔ درجهٔ نفوذ هر ویژگی است، مقداردهی می‌شوند. درجهٔ نفوذ با عددی در مقیاس صفر تا ۵ که تعیین‌کنندهٔ میزان تأثیر و نفوذ هر ویژگی از حالت بدون تأثیر تا تأثیر قوی است، مقداردهی می‌شود. جدول (۴) جدولی است که برای معرفی ویژگی‌های عمومی سیستم توسط گروه کاربران عمومی تابع نقطه‌ای ارائه شده است. پس از پاسخگویی به این ۱۴ ویژگی، این مقادیر برای تعیین VAF مورداستفاده قرار می‌گیرند. عامل تنظیم ارزش با استفاده از رابطهٔ (۵) برای هر ویژگی محاسبه می‌شود. در این رابطه C_i درجهٔ نفوذ هر ویژگی است و این عامل برای تعداد i ویژگی که معادل ۱۴ است، محاسبه می‌شود.

$$VAF = 0.65 + [\frac{C_i}{100}] \qquad (۵)$$

در ادامه، عامل VAF نهایی از مجموع VAF تمام ویژگی‌ها حاصل می‌شود. در پایان کار، تابع نقطه‌ای از حاصل‌ضرب دو عامل تعیین‌شده در این دو مرحله، طبق رابطهٔ (۶) محاسبه می‌شود.

$$FP = UAF * VAF \qquad (۶)$$

همان‌طور که گفته شد، تابع نقطه‌ای تعیین‌کنندهٔ کمیت سایز و پیچیدگی نرم‌افزار است. این کمیت عامل مهمی در تعیین هزینه و تلاش موردنیاز توسعهٔ نرم‌افزار در مدل‌هایی است که در ادامه ارائه شده است. ازجمله مدل‌هایی که از این مدل برای تعیین سایز استفاده کرده‌اند، دو مدل ESTIMATS و SPQR/20 هستند.

1. Wolverton(TRW) Model
2. Putnam
3. Walston&felix Model
4. User function count
5. value adjustment factor

جدول (۴): ویژگی‌های عمومی سیستم برای تعیین تابع نقطه‌ای [۳۷]

شمارهٔ ویژگی	عنوان ویژگی	شرح ویژگی
۱	ارتباطات داده‌ای[1]	میزان امکانات ارتباطی برای کمک به انتقال یا تبادل اطلاعات توسط نرم‌افزار
۲	پردازش دادهٔ توزیع‌شده[2]	میزان به‌کارگیری داده و توابع توزیع‌شده
۳	عملکرد[3]	زمان پاسخگویی و توان موردنیاز برای هر کاربر
۴	میزان سنگینی پیکربندی استفاده‌شده[4]	میزان سختی پلت فرم سخت‌افزاری استفاده‌شده برای اجرای نرم‌افزار
۵	نرخ تراکنش[5]	میزان اجرای تراکنش‌ها (به‌صورت روزانه، هفتگی یا ماهیانه)
۶	ورود دادهٔ آنلاین[6]	درصد اطلاعات واردشده به‌صورت آنلاین
۷	بهره‌وری کاربر نهایی[7]	میزان مؤثر بودن نرم‌افزار برای کاربر نهایی
۸	به‌روزرسانی آنلاین[8]	میزان به‌روزرسانی فایل‌های منطقی خارجی با تراکنش‌های آنلاین
۹	پردازش‌های پیچیده[9]	میزان وجود پردازش‌های پیچیدهٔ منطقی یا ریاضی
۱۰	قابلیت استفادهٔ مجدد[10]	میزان قابلیت توسعهٔ نرم‌افزار برای یک یا چند کاربر
۱۱	سهولت نصب و راه‌اندازی[11]	میزان سختی تبدیل یا نصب نرم‌افزار
۱۲	سهولت عملکردی[12]	میزان مؤثر بودن و یا انجام راه‌اندازی خودکار، پشتیبان‌گیری و بازیابی فرایندها
۱۳	سایت‌های مختلف[13]	میزان امکان طراحی، توسعه و پشتیبانی خاص نرم‌افزار برای سایت‌ها و سازمان‌های مختلف
۱۴	تسهیل در تغییرات[14]	امکان طراحی، توسعه و پشتیبانی خاص برای تغییرات آسان نرم‌افزار

1. Data comunication
2. Distributed data processing
3. Performance
4. Heavily used configuration
5. Transaction rate
6. On-Line data entry
7. End-user efficiency
8. On-Line update
9. Complex processing
10. Reusability
11. Installation ease
12. Operational ease
13. Multiple sites
14. Facilitate change

با اتمام دههٔ هفتاد، مدل هزینه سودمند COCOMO را بوهم و همکارانش ارائه دادند. در دههٔ هشتاد، محققان نسبت‌به بهبود روش‌های ارائه‌شده در این زمینه اقدام کردند. با اتمام این دهه مدل پارامتریک قیمت توسعه داده شد و محققان شرکت IBM مدل تابع نقطه‌ای را برای تخمین هزینه و تلاش نرم‌افزاری توسعه دادند. اندازه‌گیری سایز نرم‌افزار مبتنی بر تعداد عملگرها و عملوندها نیز در پایان این دهه توسط هالاستیت[15] توسعه پیدا کرد. در انتهای این دهه، بوهم و همکارانش مدل COCOMO را که قبلاً ارائه داده بودند، به نام ADA COCOMO برای نرم‌افزارهای نوشته‌شده با زبان برنامه‌نویسی ADA ارائه کردند. کاپرس[16] قابلیت آنالیز تابع نقطه‌ای را با افزودن اثر الگوریتم‌های پیچیده‌ای در محاسبهٔ توابع نقطه‌ای بهبود بخشید. چارلز سیمون نیز در ادامهٔ توسعهٔ مدل‌های ارائه‌شده، نسخه‌ای از آنالیز تابع نقطه‌ای را برای حل مسائل فردیت در آنالیز تابع نقطه‌ای بهبود بخشید. در سال ۱۹۹۰، بوهم و همکارانش مدل خود را دوباره بهبود داده و مدل COCOMOII را ارائه کردند. این بهینه‌سازی همچنان ادامه دارد [۸].

به‌مرور زمان و با هدف افزایش دقت تخمین هزینهٔ نرم‌افزار، امروزه روش‌های یادگیری ماشین به‌دلیل دقت خوبی که ارائه کردند، موردتوجه واقع شده‌اند. اما روال کاری این روش‌ها بر یادگیری الگو از نمونه‌های قدیمی متکی است و قطعاً نیازمند دادگانی هستند که شامل نمونه‌هایی تشریح شده با یک سری ویژگی است. مدل‌های الگوریتمی ارائه‌شده تا پیش از این، باعث ایجاد دادگانی مانند COCOMO شدند. دادگان ارائه‌شده برای مدل‌های الگوریتمی به‌عنوان منابعی برای یادگیری روش‌های یادگیری ماشین مورداستفاده قرار می‌گیرند. این روش‌ها از بردار ویژگی تشریح‌گر نمونهٔ پروژه‌های دادگان، الگویی استخراج می‌کنند و با استفاده از این الگو، و مقادیر بردار ویژگی نمونهٔ جدید نسبت‌به تخمین هزینه برای آن نمونه مبادرت می‌کنند.

مدل COCOMO یک مدل تخمین نرم‌افزاری مبتنی بر رگرسیون است و در دستهٔ مدل‌های الگوریتمی تابعی قرار می‌گیرد. این مدل در سال ۱۹۸۱ توسط بوهم و همکارانش توسعه یافت و مورد استنادترین، معروف‌ترین و پذیرفته‌ترین مدل

15. halasted
16. Capres

الگوریتمی پیش‌بینی هزینهٔ نرم‌افزار است. مدل COCOMO می‌تواند برای محاسبهٔ میزان تلاش و زمان موردنیاز برای پروژه‌های نرم‌افزاری استفاده شود. یکی از انواع مدل‌های COCOMO، مدل COCOMO81 است که زمان در آن یک مدل باثبات است. از مشکلات این مدل عدم تطابق آن با پیشرفت‌های محیطی توسعهٔ نرم‌افزاری است. این مشکلی است که همهٔ مدل‌های الگوریتمی با آن درگیرند. بنابراین در اواخر دههٔ ۹۰ مدل COCOMOII ارائه شد. این مدل دارای سه زیرمدل متفاوت از مدل COCOMO81 است: ۱. مدل ترکیبی کاربردی که برای انجام تخمین هزینهٔ پروژه‌های ساخته‌شده با ابزارهای GUI مناسب است. ۲. مدل طراحی اولیه که برای ارائهٔ تخمینی واقعی از هزینهٔ یک پروژه به‌شکلی که قبل از تخمین معماری کلی نرم‌افزار تعیین می‌شود، مورداستفاده قرار می‌گیرد. این مدل از مجموعهٔ کوچکی از تعیین‌کنندگان هزینه و یک سری معادلات جدید برای تخمین تشکیل شده است. ۳. مدل معماری-پسین که بعد از تعیین معماری کلی پروژه، از آن استفاده می‌شود. این مدل از یکی از دو روش تابع نقطه‌ای و تعداد خطوط (LOC[1]) برای تخمین سایز که از عوامل اصلی تعیین‌کنندهٔ هزینه است، استفاده می‌کند و دربرگیرندهٔ تلاش توسعه و نگهداری است.

این مدل شامل ۱۷ تعیین‌کنندهٔ هزینه است که روی مقیاسی از very low تا extra high به همان روش COCOMO81 رتبه‌بندی شده‌اند. مدل معماری-پسین با رابطهٔ (۷) فرموله شده:

$$\mathbf{EFFORT} = \mathbf{A} * [\mathbf{size}]^{\mathbf{B}} * \prod_{\mathbf{i=1}}^{\mathbf{17}} \mathbf{EFFORT\ multiplier\ i} \quad (۷)$$

where

$$B=1.01+0.01*\sum_{J=1}^{5} scale\ factor\ J$$

در این فرمول، پارامتر A ضریب ثابت است. size حجم پروژه نرم‌افزاری اندازه‌گیری‌شده با معیار ksloc (هزاران خط کد، تابع نقطه‌ای یا اشیا نقطه‌ای[2]) است. انتخاب معیار مقیاس یا بزرگی براساس ارتباطاتی است که یک منبع قابل‌توجه از تغییرات نهایی در یک تلاش نرم‌افزار یا بهره‌وری پروژه است.

ازآنجایی‌که این دادگان با وجود سیر بهبودی که در صنعت توسعهٔ نرم‌افزار و مدل‌های تخمین باز هم شامل همان بردار ویژگی دادگان COCOMO81 است، در جدول (۵) بردار ویژگی این دادگان ارائه شده است. مقادیر اختصاص‌یافته به هر ویژگی در محدودهٔ very low تا extra high است [۲۹].

جدول (۵): شرح ویژگی‌های مدل COCOMO

شماره	نام ویژگی	نام کامل انگلیسی	معادل فارسی
۱	RELY	required software reliability	قابلیت اعتماد موردنیاز برای نرم‌افزار
۲	DATA	data base size	اندازهٔ پایگاه داده‌ها (در آزمون نرم‌افزار)
۳	CPLX	process complexity	پیچیدگی محصول
۴	TIME	time constraint for cpu	محدودیت زمان اجرایی
۵	STOR	main memory constraint	محدودیت حافظه اصلی
۶	VIRT	machine volatility	تغییرپذیری ماشین
۷	TURN	turnaround time	مدت زمان بین شروع کار نرم‌افزار تا دریافت خروجی
۸	ACAP	analysts capability	توانایی تحلیلگرها
۹	AEXP	application experience	تجربهٔ کاربردی
۱۰	PCAP	programmers capability	توانایی برنامه‌نویس‌ها
۱۱	VEXP	virtual machine experience	تجربهٔ ماشین مجازی
۱۲	LEXP	language experience	تجربهٔ زبان
۱۳	MODP	modern programing practices	تجربه‌های برنامه‌نویسی نوین
۱۴	TOOL	use of software tools	استفاده از ابزارهای نرم‌افزاری
۱۵	SCED	schedule constraint	زمان‌بندی و محدودیت‌های زمانی
۱۶	LOC	Line of code	تعداد خطوط کد
۱۷	ACTUAL	Effort	میزان تلاش موردنیاز برای توسعهٔ نرم‌افزار

در ادامه، کاربرد بردار ویژگی تعیین‌شده توسط بوهم و همکارانش در مدل COCOMO، سازمان ناسا در سال ۲۰۰۴ دادگان COCONASA شامل ۶۰ پروژه از نرم‌افزارهای تولیدی توسط مراکز مختلف ناسا ارائه داد که توسط همین بردار ویژگی شرح داده شده‌اند. سپس در سال ۲۰۰۶، این سازمان یک نسخهٔ جدید از دادگان خود را با نام COCONASA_2 با ۹۳ نمونه پروژه ارائه داد. تفاوت این دادگان با نسخهٔ قبلی در بردار ویژگی تعیین شده برای هر نمونه پروژه است. در این دادگان، علاوه بر ۱۷ ویژگی تعیین‌شده برای نمونهٔ پروژه‌ها، ۷ ویژگی جدید به بردار ویژگی اضافه شده است. ویژگی‌های اضافه‌شده در جدول

1. Line Of Code
2. object point

(۶) شرح داده شده‌اند. در جدول (۶)، ویژگی کاربرد بیانگر نوع کاربرد پروژه در زمینه‌های ارتباطات هوایی، کاربردهای زمینی، نظارت هوایی، پردازش داده‌های دسته‌ای، ارتباطات، ضبط داده‌ها، پردازش پرتاب، برنامه‌نویسی پرواز، کنترل مانیتور، سیستم عامل، پردازش دادهٔ واقعی، علوم پایه، شبیه‌سازی و کاربردی است. حالت توسعهٔ پروژه‌ها به یکی از سه شکل تعبیه‌شده، ارگانیک و یا نیمه‌مجزاست. سیستم هوایی با حرف f و سیستم زمینی با حرف g در ستون FORG مشخص است.

جدول (۶): شرح ویژگی‌های اضافه بر مدل COCOMO در دادگان 2COCONASA

شماره	نام ویژگی	معادل فارسی	مقادیر
۱	UNIQUE ID	شمارهٔ واقعی نمونه	عدد از ۱ تا ۹۳
۲	PROJECTNAME	نام پروژه	de,erb,gal,X,hst,slp,spl,Y
۳	CAT2	کاربرد	Avionics, application_ground, avionicsmonitoring, batch data processing, communications, data capture, launchprocessing, missionplanning, monitor_control, operatingsystem, realdataprocessing, science, simulation, utility
۴	FORG	سیستم هوایی یا زمینی؟	f,g
۵	CENTER	کدام مرکز ناسا؟	۱،۲،۳،۴،۵،۶
۶	YEAR REAL	سال توسعهٔ نرم‌افزار؟	مقدار عددی سال
۷	MODE	حالت توسعه؟	embedded,organic,semidetached

دادگان پرکاربرد دیگری که در زمینهٔ تخمین هزینهٔ نرم‌افزار استفاده می‌شوند، دادگان DESHARNAIS است. این دادگان دربرگیرندهٔ پروژه‌های تولیدی طی سال‌های ۱۹۸۱ تا ۱۹۸۸ یک خانهٔ نرم‌افزاری است. نسخهٔ اصلی این دادگان شامل ۸۱ نمونه پروژهٔ توصیف‌شده توسط ۱۲ ویژگی از یک خانهٔ نرم‌افزار کاندید است. اما ۴ نمونه از نمونه‌های این دادگان دارای مقدار گم‌شده در ۴ ویژگی هستند. محققان برخوردهای مختلفی با این دادگان برای استفاده دارند. دسته‌ای ویژگی‌های حاوی مقدار گم‌شده را کنار می‌گذارند و دسته‌ای نیز نمونه‌های حاوی مقدار گم‌شده را از مجموعه دادگان حذف می‌کنند. درواقع این دادگان شامل ۷۷ پروژهٔ نرم‌افزاری کامل و ۴ نمونهٔ ناقص است. ویژگی‌های مستقل شامل تجربهٔ گروه، تجربهٔ مدیر، طول پروژه، معاملات، موجودیت‌ها، گروه تابع نقطه‌ای، محیط توسعه و زبان برنامه‌نویسی است. متغیر وابسته هم تلاش توسعهٔ نرم‌افزار، اندازه‌گیری‌شده با ساعت در فرد است [۱۱]. ویژگی‌های این دادگان در جدول (۷) تشریح شده‌اند. این ویژگی‌ها کاملاً از ویژگی‌های دادگان COCOMO متفاوت‌اند.

جدول (۷): شرح ویژگی‌های DESHARNAIS

شماره	نام ویژگی	معادل فارسی	مقادیر
۱	Project	شمارهٔ پروژه	عددی از ۱ تا ۸۱
۲	Exp Team	تجربه گروه	اندازه‌گیری‌شده با معیار سال
۳	Manager Exp	تجربه مدیر	اندازه‌گیری‌شده با معیار سال
۴	End Year	سال اتمام پروژه	سال
۵	Length	تعداد ماه موردنیاز برای تولید پروژه	مقدار عددی
۶	Effort	تلاش موردنیاز	شخص در ساعت
۷	Transactions	تعداد تراکنش‌های عمومی منطقی در سیستم	مقدار عددی
۸	Entities	تعداد موجودیت‌ها در سامانه‌های مدل داده	مقدار عددی
۹	Adjust Points	اندازهٔ پروژه براساس توابع نقطه‌ای تعدیل‌نشده	این ویژگی به‌عنوان تراکنش‌ها به‌علاوه موجودیت محاسبه می‌شود
۱۰	Enver gure	تابع نقطهٔ عامل تنظیم پیچیدگی	براساس ویژگی‌های عمومی سیستم(GSC) تعیین می‌شود.GSC شامل ۱۴ ویژگی است که هرکدام براساس یک مقیاس ترتیبی شش نقطه‌ای امتیازدهی شده‌اند.
۱۱	Points Non Ajust	اندازه پروژه	اندازه‌گیری شده با نقاط تابعی
۱۲	Language	زبان	پروژهٔ بیان‌شده با مقادیر ۱، ۲ یا ۳. مقدار ۱ مربوطبه «کوبول عمومی»، که در آن مقدار ۲ مربوطبه «پیشرفتهٔ کوبول» و ارزش ۳ مربوطبه زبان GL۴

بنابراین روش‌های یادگیری ماشین، برای انجام تخمین هزینهٔ نرم‌افزار هیچ‌گونه وابستگی به مدل‌های تخمین هزینهٔ نرم‌افزار ارائه‌شده ندارند و تنها از دادگان آن‌ها برای یادگیری الگو استفاده می‌کنند. در جدول (۹)، دقت روش‌های یادگیری ماشینی روی دادگان مورداستفاده برای آموزش آن‌ها در این حوزه به دقت بالاتری نسبت‌به سایرین رسیده است. دادگان فراوانی تاکنون در این زمینه استفاده شده‌اند. برخی مطالعات ۳۱ دادگان را برای این زمینه معرفی کرده‌اند. اما نکتهٔ مهم این است که اکثر دادگان مورداستفاده، نسبتاً کوچک‌اند؛ مثلاً از ۳۱ دادگان معرفی‌شده در این مطالعات، فقط سه دادگان مربوط‌به بیش از ۵۰ پروژه‌اند. دادگان، عموماً شامل مجموعه‌ای از خصوصیات هستند که می‌توان آن‌ها را در دسته‌های زیر دسته‌بندی کرد [۱۵].

- **خصوصیات مربوط‌به سایز**

خصوصیاتی هستند که شامل اطلاعات مربوط‌به اندازهٔ پروژهٔ نرم‌افزاری هستند. این اطلاعات می‌توانند به‌شکل خطوط کد[1]، تابع نقطه‌ای و یا خیلی از معیارهای دیگر فراهم شوند. متغیرهای بیانگر سایز، اغلب به‌عنوان ویژگی‌های مهمی برای تخمین تلاش در نظر گرفته می‌شوند. جدول (۸) هم دو روش تعداد خطوط و تابع نقطه‌ای را با هم مقایسه می‌کند.

جدول (۸): مقایسهٔ دو روش تعیین سایز [۷]

تعداد خطوط	تابع نقطه‌ای
در اواخر چرخهٔ پروژه	در هر گام از پروژه
تعداد خطوط چه چیزی است	روش‌های ساخت‌یافته
وابسته به فناوری	مستقل از فناوری
تعیین اندازه به سبک برنامه‌نویسی وابسته است	تعیین اندازه به سبک برنامه‌نویسی وابسته نیست

- **اطلاعات محیطی**

شامل اطلاعات پیش‌زمینه‌ای دربارهٔ گروه توسعه، شرکت و خود پروژه، مثل تعداد توسعه‌دهندگان و تجربهٔ آن‌هاست.

- **اطلاعات پروژه**

شامل خصوصیاتی است که برای هدف مشخصی از پروژه و نوع پروژه بیان می‌شوند. همچنین ویژگی‌های موردنیاز پروژه‌های خاص در این دسته قرار می‌گیرند.

- **متغیرهای مربوط‌به توسعه**

شامل اطلاعاتی دربارهٔ جنبه‌های مدیریتی و یا جنبه‌های روش توسعهٔ پروژه‌های نرم‌افزاری، ازجمله زبان برنامه‌نویسی یا نوع سیستم پایگاه داده‌ای که در طول توسعه مورداستفاده قرار گرفته است.

در ادامه، به تشریح اجمالی تعدادی از دادگان معروف که در اغلب کارهای پژوهشی در زمینهٔ تخمین هزینهٔ نرم‌افزار از آن‌ها استفاده می‌شود، می‌پردازیم [۱۱، ۱۳، ۱۵، ۳۱ و ۲۸].

۳. ۱. دادگان ALBRECHT

این دادگان شامل ۲۴ پروژهٔ نرم‌افزاری توسعه‌یافته توسط سازمان سرویس‌های IBM DP است که دارای ۶ متغیر یا ورودی مستقل تعداد ورودی، تعداد خروجی، تعداد پرس‌وجو، تعداد فایل، توابع نقطه‌ای و خطوط کد است. ورودی وابستهٔ آن‌هم تلاش اندازه‌گیری شده با تعداد ساعت کاری موردنیاز برای طراحی، توسعه و آزمون هر نرم‌افزار است [۱۱].

۳. ۲. دادگان MIYAZAKI

این دادگان شامل ۴۸ پروژهٔ نرم‌افزاری گردآوری‌شده توسط گروه کاربران سامانه‌های بزرگ فوجیتسو[2] است که دارای ۷ خصوصیت مستقل شامل تعداد صفحه‌ها، تعداد فرم‌ها، تعداد فایل‌ها، تعداد اجزای داده‌ای در صفحه، تعداد اجزای داده‌ای در فرم‌ها، تعداد اجزای داده‌ای در فایل‌ها و خطوط کد است. همچنین متغیر وابستهٔ تلاش توسعهٔ نرم‌افزار اندازه‌گیری‌شده با تعداد فرد در ماه برای طراحی سیستم‌ها در آزمون سیستم‌ها، شامل تلاش مستقیم مثل مدیریت پروژه است. هر شخص در ماه معادل ۱۶۰ ساعت کار کردن است [۱۱].

۳. ۳. دادگان MAXWELL

شامل ۶۲ پروژهٔ نرم‌افزاری از بزرگ‌ترین بانک‌های جهانی در فنلاند است که ۲۵ متغیر مستقل دارد که با ویژگی‌های نرم‌افزاری مختلف، مثل نوع کاربرد و سایز تعیین می‌شوند. متغیر وابستهٔ آن هم تلاش توسعهٔ نرم‌افزار تعیین‌شده با تعداد ساعت کار انجام‌شده

1. LOC

2. Fujitso

توسط عرضه‌کنندهٔ نرم‌افزار، از مشخصات فنی تا زمان تحویل است [۱۱].

۳. ۴. دادگان CSBSG

شامل ۱۱۰۳ پروژه نرم‌افزاری است که در سال ۲۰۰۶ با مأموریت ترویج استانداردهای تولیدات نرم‌افزاری چینی ایجاد شده‌اند. پروژه‌های این دادگان از ۱۴۰ سازمان و ۱۵ منطقهٔ چینی مرتبط با صنعت نرم‌افزار جمع‌آوری شده‌اند. هر پروژهٔ نرم‌افزاری در این دادگان با ۱۷۹ خصوصیت تشریح شده‌اند [۳۱].

۳. ۵. دادگان ISBSG

شامل ۱۲۳۸ پروژه از سازمان‌های مختلفی همچون بیمه و سازمان‌های دولتی دیگر گردآوری‌شده از ۲۰ کشور مختلف است. در این دادگان، هر پروژهٔ نرم‌افزاری با استفاده از ۷۰ خصوصیت تشریح شده است. در برخی از کارهای پژوهشی، این دادگان مورد پیش‌پردازش قرار می‌گیرند [۳۱].

علاوه بر دادگان تشریح‌شده در فوق، دادگان فراوان دیگری هم تاکنون در مطالعات مختلف برای آموزش الگوریتم‌های یادگیری ماشین استفاده شده‌اند. برخی از پژوهشگران در شرکت‌های مختلف، از دادگان درون‌شرکتی[1] و برخی دیگر از دادگان برون‌شرکتی[2] برای ساخت مدل‌هایی جهت تخمین هزینهٔ توسعهٔ نرم‌افزار توسط شرکت استفاده می‌کنند. پژوهشگران در بررسی‌هایی که در سال ۲۰۰۷ روی مطالعات در این زمینه انجام داده‌اند [۳۸]، آورده‌اند که از میان ۱۰ مقالهٔ انتخاب‌شده در این زمینه، ۷ مقاله نتایج مستقلی ارائه داده و از بین این ۷ مقاله هم ۳ مقاله اثبات کرده‌اند که مدل‌های برون‌شرکتی به‌طور قابل‌توجهی از مدل‌های درون‌شرکتی بدترند؛ البته به‌دلیل اینکه روش‌های متفاوتی در مطالعات مختلف استفاده می‌شوند، آنالیز کامل مطالعات غیرممکن است. درنهایت، آن‌ها به این نتیجه رسیده‌اند که مطالعات انجام‌شده روی دادگان درون‌شرکتی با کمتر از ۲۰ نمونه پروژه که از روش اعتبارسنجی کنارگذاشتن یکی در هر بار[3] استفاده کرده‌اند، همگی از مدل‌های برون‌شرکتی بهتر بوده‌اند.

1. Within-company data
2. Cross-company data
3. leave-one-out

پژوهشگران در این کار به نتیجهٔ واضحی نرسیدند و به همین دلیل، آن‌ها در مقاله‌ای که اخیراً در سال ۲۰۱۴ ارائه کرده‌اند [۳۹]، کار خود را ادامه داده‌اند.

به‌طور کلی، نتیجه‌ای که در این بررسی‌ها حاصل شد، این است که نیمی از شواهد آنالیزشده نشان داده‌اند که مدل‌های ساخته‌شده از داده‌های برون‌شرکتی به‌طور چشمگیری، از مدل‌های تخمین ساخته‌شده از داده‌های درون‌شرکتی بدتر نیستند. علاوه بر این، شواهدی وجود دارد که بیانگر این امر هستند که سایز بیشتر نمونه‌ها به‌معنای دقت بیشتر در تخمین نیست. نکتهٔ قابل‌ذکر دیگر این است که نمونه‌ها برای ساختن یک مدل تخمین، باید به‌دقت و براساس کنترل جنبه‌های کیفی و شباهت پروژه‌ها انتخاب شوند [۳۹].

پژوهشگران در [۲۵]، چالش اصلی در تخمین تلاش را، ارائهٔ راهکاری برای پیشنهاد یک روش تخمین که روی همهٔ دادگان دقت مشابهی داشته باشد، معرفی می‌کنند. روش ارائه‌شده در این مقاله، شامل دو گام اصلی است:

گام اول: یک متخصص، متغیرهای هر دادگان را بررسی کند و فقط آن‌هایی را نگه دارد که پتانسیل پیش‌بینی تلاش را دارند. بنابراین متغیرهایی مثل مدت زمان و تعداد نقص‌ها به‌دلیل اینکه در ادامه ناشناخته‌اند، حذف شده‌اند.

گام دوم: تغییر شکل‌داده به مقیاس جدید، مثل گسسته‌سازی را پوشش می‌دهد. این مرحله بدان سبب لازم است که بیشتر روش‌هایی که برای این آزمایش انتخاب شده‌اند، فقط با داده‌های طبقه‌بندی‌شده کار می‌کنند. روش‌های مختلفی مثل به‌دست‌آوردن بازه‌هایی با فرکانس یا عرض مساوی برای انجام این وظیفه وجود دارد. در این آزمایش، از یک متخصص برای گسسته‌سازی همهٔ متغیرهای عددی با هدف تعیین بازه‌های با مقادیر معنی‌دار و فرکانس‌های مشابه استفاده شده است.

۴. معیارهای ارزیابی

هرگونه کار پژوهشی درنهایت برای تعیین میزان دقت[4] کار انجام‌شده، بایستی با یک سری از معیارها، ارزیابی شود.

4. Accuracy

دراین‌باره، بی‌شک در زمینهٔ تخمین هزینه یا تلاش موردنیاز برای توسعهٔ نرم‌افزار نیز، معیارهایی برای سنجش دقت الگوریتم‌های ارائه‌شده وجود دارد. در ادامه، به بیان معیارهای موجود برای این سنجش خواهیم پرداخت.

در این زمینه، معیارهای سنجش دقت زیادی تاکنون معرفی و استفاده شده‌اند. سه معیار سنجش دقت پرکاربرد براساس میزان خطای الگوریتم‌ها در اغلب مطالعات مورد استفاده قرار می‌گیرند: ۱. MRE[1] که بیانگر میزان اختلاف هزینهٔ تخمین‌زده‌شده توسط الگوریتم‌ها با هزینهٔ واقعی است؛ ۲. MRE که به‌وسیلهٔ علامت انحصاری MMRE[2] نشان داده می‌شود و بیانگر میانگین خطای تخمین برای کل نمونه‌های موردنظر (نمونه‌های آموزشی و یا نمونه‌های آزمون) است. ۳. PRED (x)[3] که نشان‌دهندهٔ درصد نمونه‌هایی است که میزان خطای تخمین آن‌ها کمتر یا مساوی با مقدار x است. البته در برخی از کارها میانهٔ خطای تخمین هم با علامت MDMRE[4]، به‌عنوان یکی دیگر از معیارهای سنجش تخمین انجام‌شده توسط الگوریتم موردنظر و میزان دقت آن مورداستفاده قرار می‌گیرد. علاوه بر این روش‌ها، روش‌های دیگری هم توسط برخی دیگر از مطالعات استفاده شده‌اند که در مرور این دسته از کارها، از آن‌ها در جدولی که در ضمیمهٔ الف آمده، نام برده شده است. بیان جامع هریک از این معیارهای سنجش دقت به‌صورت فرمول‌وار در ادامهٔ این فصل آمده است. روش‌های یادگیری ماشین، در مرحلهٔ آموزش، الگو را از ویژگی‌های نمونه‌های آموزشی استخراج می‌کنند. در مرحلهٔ بعدی، تمام ویژگی‌های نمونه‌های آزمون، به‌جز ویژگی تلاش را دریافت و براساس الگوی استخراج‌شده در مرحلهٔ آموزش، تلاش موردنیاز برای نمونه‌های آزمون را پیش‌بینی می‌کنند. با استفاده از این معیارها میزان خطای هر الگوریتم را تعیین می‌کنند.

۴. ۱. خطای تخمین از میزان واقعی

در برخی از کارهای انجام‌شده در زمینهٔ موضوع مورد بحث، از تعیین میزان اختلاف بین میزان هزینهٔ تخمین زده شده با میزان واقعی هزینهٔ نمونهٔ پروژه‌های نرم‌افزاری انتخابی، MRE، به‌عنوان معیاری برای سنجش دقت الگوریتم ارائه‌شده استفاده می‌کنند [۴۰]. این میزان خطا با استفاده از رابطهٔ (۵) محاسبه می‌شود.

$$MRE = \frac{|\text{Actual Effort} - \text{Estimated Effort}|}{\text{Actual Effort}} \quad (۵)$$

Actual Effort میزان تلاش واقعی نمونهٔ پروژهٔ موردنظر در دادگان را دربرمی‌گیرد. Estimated Effort هم میزان تلاش تخمین‌زده‌شده توسط الگوریتم موردسنجش را نشان می‌دهد.

۴. ۲. میانگین خطای تخمین از میزان واقعی

در برخی دیگر از کارهای پژوهشی در این حوزه، آنچه به‌عنوان میزان دقت الگوریتم ارزیابی می‌شود، میانگین اختلاف هزینهٔ تخمین‌زده‌شده توسط الگوریتم موردنظر با هزینهٔ واقعی برای تمام نمونه‌های مورد نظر MMRE است [۱۶]. رابطهٔ تعیین‌کنندهٔ این مورد به شکل رابطهٔ (۶) است.

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|Estimated(i) - Actual(i)|}{Actual(i)}\right) \quad (۶)$$

n بیانگر تعداد پروژه‌های موردارزیابی، Estimated میزان هزینهٔ تعیین‌شده توسط الگوریتم موردارزیابی و Actual بیانگر میزان هزینه یا تلاش واقعی است. بنابراین هرچه میزان MMRE کمتر باشد، یعنی میزان خطای تخمین الگوریتم کمتر بوده و همین نکته بیانگر دقت بهتر است.

۴. ۳. میانهٔ خطای تخمین از میزان واقعی

در کارهای پژوهشی در حوزهٔ تخمین تلاش یا هزینهٔ نرم‌افزار، معیار دیگری که برای سنجش دقت الگوریتم‌های موردآزمایش، استفاده می‌شود، میانهٔ خطای مقدار تعیین‌شده توسط الگوریتم موردنظر از مقدار واقعی تلاش نمونه‌های موردنظر یا همان MDMRE است [۲۲]. رابطهٔ تعیین‌کنندهٔ MDMRE به شکل رابطهٔ (۷) است.

$$\text{MDMRE} = \text{Median (MRE)} \quad (۷)$$

۴. ۴. احتمال میزان مشخصی از خطا

در اغلب کارهای تحقیقاتی در زمینهٔ موردبحث، معیار دیگری که برای ارزیابی میزان دقت الگوریتم‌های موردآزمایش استفاده

1. Magnitude of Relative Error
2. Mean Magnitude of Relative Error
3. Prediction accuracy
4. Median Magnitude of Relative Error

می‌شود، PRED (x) یا احتمال این است که میزان خطای مقدار تخمین‌زده‌شده برای تمام نمونه‌های موردارزیابی کمتر یا مساوی مقدار x باشد [۱۲]. رابطه‌ای که برای محاسبهٔ این احتمال به‌کار می‌رود، در رابطهٔ (۸) آمده است.

$$PRED(x) = \frac{k}{n} \qquad (۸)$$

X بیانگر میزان اختلاف موردنظر است که در اغلب کارهای پژوهشی با میزان ۰/۲۵ مقدار است. K بیانگر تعداد نمونه‌هایی است که اختلاف مقدار هزینهٔ تخمین‌زده‌شده توسط الگوریتم موردارزیابی برای آن‌ها، با مقدار واقعی هزینهٔ آن‌ها کوچک‌تر یا مساوی مقدار X باشد. n هم بیانگر تعداد کل نمونه‌های موردارزیابی است. بنابراین هرچه میزان (۰/۲۵) PRED بیشتر باشد، میزان خطای الگوریتم موردارزیابی هم کمتر است و هزینهٔ تخمین‌زده‌شده برای تعداد بیشتری از نمونه‌های دادگان موردارزیابی، میزان خطای کمتر یا مساوی ۰/۲۵ دارند.

۴. ۵. معیار ترکیبی احتمال و میانگین خطا

معیار دیگری که در سال ۲۰۰۹ به‌عنوان معیاری برای ارزیابی دقت معرفی شده و در برخی کارهای پژوهشی مثل [۱۱] مورداستفاده قرار گرفته، تابع ارزیابی[1] یا EF است که از ترکیب دو معیار (۰/۲۵) PRED و میانگین خطا با هم حاصل شده و از رابطهٔ (۹) برای محاسبهٔ آن استفاده می‌شود.

$$EF = \frac{PRED(0.25)}{1 + MMRE} \qquad (۹)$$

نکتهٔ قابل ذکر این است که در بیشتر کارهای پژوهشی، از چندین روش برای ارزیابی کار خود استفاده می‌کنند و این نکته کاملاً مشخص است که الگوریتمی که دارای میانگین خطای کمتر و PRED بیشتر باشد، الگوریتم بهتری خواهد بود؛ زیرا دقت بیشتر و خطای کمتری را دارد.

در [۴۱]] با درنظرگرفتن هر دو معیار MMRE و PRED دو روش پرکاربرد از روش‌های انتخاب ویژگی بسته‌بندی (انتخاب ویژگی روبه‌جلو و روبه‌عقب) روی دادگان زمینهٔ تخمین هزینهٔ نرم‌افزار پیاده‌سازی شد و ثابت شد که این ایده تأثیر مثبتی در راستای شناسایی ویژگی‌های مؤثر در این حوزه خواهد گذاشت.

محققان در پژوهش [۴۲]، تخمین هزینهٔ نرم‌افزار را به‌عنوان یک مسئلهٔ چندهدفه در نظر گرفته و روشی نو برای ارائهٔ یک معیار جدید از ترکیب معیارهای ارزیابی ارائه داده‌اند با این هدف که مدل یادگیری‌ای تولید شود تا همهٔ معیارهای ارزیابی را بهینه سازد.

مطالعات صورت‌گرفته در [۴] و تحلیلی که روی نتایج حاصل از پیاده‌سازی مدل‌های یادگیری ماشین روی دادگان تخمین هزینهٔ نرم‌افزار آمده است، به‌جز BN بقیهٔ مدل‌های یادگیری MMRE در محدودهٔ ۳۵٪ تا ۵۵٪ و (۰/۲۵) PRED در محدودهٔ ۴۵٪ تا ۷۵٪ را به ارمغان می‌آورند. این نتایج نشان‌دهندهٔ این است که دقت تخمین با استفاده از مدل‌های یادگیری ماشین در سطح قابل‌قبولی قرار دارد.

تحلیل ارائه‌شده در [۴] روی مقالات منتشرشده طی سال‌های ۱۹۹۱ تا ۲۰۱۰ صورت گرفته است. در جدول (۹)، دقت روش‌های یادگیری ماشین روی چهار دادگان معروف و پرکاربرد در این حوزه که توسط محققان در مقالاتی که طی سال‌های ۲۰۱۰ به بعد منتشر شده‌اند، قابل‌مشاهده است. همان‌طور که در این جدول می‌توان دید، روی دادگان COCOMO81 روش ANN از نظر دو معیار MMRE و PRED و روش DT از نظر معیار MDMRE بهترین نتایج را داشته‌اند. برای دادگان MAXWELL هم، روش‌های DT و SVR بهترین نتایج را داشته‌اند. روش CBR روی دادگانISBSG بهترین دقت را داشته است. روی دادگان DESHARNAIS هم روش CBR از نظر MMRE و PRED بهترین دقت را ارائه داده، ضمن اینکه روش SVR هم از نظر معیار ارزیابی MDMRE روی این دادگان بهتر از سایرین عمل کرده است. همان‌طور که از جدول برمی‌آید، میانگین خطای تمام روش‌های استفاده‌شده در این مقالات، روی دادگان DESHARNAIS بهتر از سایر داده‌هاست.

1. Evaluation Function

جدول (۹): مرور دقت روش‌های یادگیری ماشین مورداستفاده در تخمین هزینهٔ نرم‌افزار

داده‌گان	مرجع	سال	روش	MMRE	MdMRE	Pred(25)	داده‌گان	مرجع	سال	روش	MMRE	MdMRE	Pred(25)
COCOMO81	[۱۱]	۲۰۱۳	ANN	۲۵۰,۷۷	–	۲۳,۱	ISBSG	[۱۵]	۲۰۱۲	CBR	–	۵۸	۱۸,۸
	[۱۱]	۲۰۱۳	SVR	۸۰۳,۹۶	–	۷,۱		[۱۵]	۲۰۱۲	CBR	–	۵۲,۷	۲۲
	[۱۵]	۲۰۱۲	DT	–	۷۶,۴	۱۱,۱		[۱۵]	۲۰۱۲	CBR	–	۵۳,۶	۲۵,۸
	[۱۵]	۲۰۱۲	SVR	–	۱۰۵	۱۲,۷		[۱۵]	۲۰۱۲	CBR	–	۵۷,۳	۲۵,۵
	[۱۵]	۲۰۱۲	ANN	–	۹۹	۱۵,۹		[۱۵]	۲۰۱۲	ANN	–	۶۱,۶	۱۶,۳
	[۱۵]	۲۰۱۲	CBR	–	۷۵,۴	۱۵,۹		میانگین				۶۴,۳	۲۱,۲۱۲
	[۱۵]	۲۰۱۲	CBR	–	۸۶,۱	۱۹	DESHARNAIS	[۱۱]	۲۰۱۳	ANN	۷۸,۲۴	–	۲۰,۰
	[۱۵]	۲۰۱۲	CBR	–	۸۵,۶	۱۱,۱		[۱۱]	۲۰۱۳	SVR	۷۳,۸۳	–	۳۳,۳
	[۱۵]	۲۰۱۲	CBR	–	۹۱,۳	۱۱,۱		[۱۵]	۲۰۱۲	SVR	–	۳۵,۶	۳۵,۸
	[۱۵]	۲۰۱۲	ANN	–	۸۹,۲	۹,۵۲		[۱۵]	۲۰۱۲	ANN	–	۳۶,۱	۳۵,۸
	میانگین			۵۲۷,۳۶	۸۸,۵	۱۳,۶۵۲		[۱۵]	۲۰۱۲	CBR	–	۴۷,۲	۲۳,۵
Maxwell	[۱۵]	۲۰۱۲	DT	–	۴۸,۴	۳۴,۰۶		[۱۵]	۲۰۱۲	CBR	–	۴۵,۸	۲۷,۲
	[۱۵]	۲۰۱۲	SVR	–	۴۵,۷	۲۵,۸		[۱۵]	۲۰۱۲	CBR	–	۴۵,۷	۳۳,۳
	[۱۵]	۲۰۱۲	ANN	–	۶۵,۵	۱۹,۴		[۱۵]	۲۰۱۲	CBR	–	۴۰	۳۵,۸
	[۱۵]	۲۰۱۲	CBR	–	۵۳,۴	۲۲,۶		[۱۵]	۲۰۱۲	ANN	–	۱۰۰	۲,۴۷
	[۱۵]	۲۰۱۲	CBR	–	۶۱,۸	۱۷,۷۷		[۲۵]	۲۰۱۲	ANN	۵۱	–	۳۳,۶۴
	[۱۵]	۲۰۱۲	CBR	–	۵۵,۴	۲۲,۶		[۲۵]	۲۰۱۲	CBR	۷۰	–	۲۷,۵۳
	[۱۵]	۲۰۱۲	CBR	–	۵۶	۳۰,۶		[۲۵]	۲۰۱۲	DT	۶۳	–	۳۱,۲۸
	[۱۵]	۲۰۱۲	ANN	–	۱۰۰	۱۴,۵		[۲۵]	۲۰۱۲	CBR	۶۷	–	۳۴
	میانگین			-	۶۰,۷۷۵	۲۳,۴۱۶		[۲۵]	۲۰۱۲	CBR	۶۳	–	۳۳,۹۴
ISBSG	[۱۵]	۲۰۱۲	DT	–	۵۵,۷	۲۲,۷		[۲۵]	۲۰۱۲	CBR	۶۳	–	۳۶,۵۱
	[۱۵]	۲۰۱۲	SVR	–	۶۰,۵	۲۴,۷		[۲۵]	۲۰۱۲	CBR	۳۱	–	۶۷,۵۸
	[۱۵]	۲۰۱۲	ANN	–	۱۱۵	۱۳,۹		میانگین			۶۲,۲۳	۵۰,۰۵	۳۱,۹۷۸
میانگین معیارهای دقت روی کل دادگان				MMRE=146.8			MDMRE=66.41935				Pred=23.93333		

۴. ۶. اعتبارسنجی الگوریتم‌ها

براساس بررسی‌های انجام‌شده، در کارهایی که در حوزهٔ تخمین هزینهٔ نرم‌افزار با الگوریتم‌های یادگیری ماشین انجام شده، دادگان موردارزیابی را به‌واسطهٔ الگوریتم‌های اعتبارسنجی[1] به دو دستهٔ دادگان آموزشی[2] و دادگان آزمون[3] تقسیم کرده و از دادگان آموزشی، برای آموزش الگوریتم‌ها استفاده می‌کنند. البته شایان ذکر است که این دادهٔ آموزشی هم، خود به دو دستهٔ آموزش و اعتبارسنج[4] (که برای ارزیابی میزان آموزشی که از دادگان آموزشی حاصل شده، به‌کار می‌رود) تقسیم می‌شود. نتیجهٔ این ارزیابی فقط برای انتخاب بهترین بخش آموزشی مورداستفاده قرار می‌گیرد. آنچه مسلم است این است که نتیجهٔ حاصل از ارزیابی دادگانی که در آموزش استفاده شده‌اند، به‌عنوان نتیجهٔ ارزیابی کلی مدنظر قرار نمی‌گیرد بلکه آنچه به‌عنوان نتیجهٔ ارزیابی دقت الگوریتم مدنظر است، دقت آن الگوریتم روی پیش‌بینی هزینهٔ نمونه‌هایی است که در دستهٔ دادگان آزمون قرار دارند.

روش دسته‌بندی به نام اعتبارسنجی n بخشی[5]، بدین شرح است که کل داده‌ها را به‌صورت تصادفی در n دسته تقسیم کرده و هر بار یکی از این دسته‌ها را به‌عنوان دادهٔ آزمون در نظر گرفته

1. Cross-validation
2. Training set
3. Test set
4. Validate
5. n-fold

و n-1 دستهٔ دیگر را به‌عنوان دادهٔ آموزشی در نظر می‌گیرند؛ این کار را به تعدادn بار تکرار کرده تا از بین این دسته‌بندی‌ها، دسته‌ای که خطای کمتری دارد برگزینند. در مقالهٔ مروری ون و همکارانش [۴] روش‌های اعتبارسنجی HOLDOUT و LOOCV[1] به‌عنوان دو روش دیگر برای اعتبارسنجی معرفی شده‌اند. تعیین اینکه چه نمونه‌هایی در دستهٔ دادگان آموزشی قرار بگیرند و چه نمونه‌هایی در دستهٔ دادگان آزمون واقع شوند، به انتخاب خود پژوهشگران برمی‌گردد.

برخی از پژوهشگران [۳۴] نمونه‌ها را فقط به دو بخش آموزشی و آزمونی تقسیم می‌کنند. برخی دیگر [۲۷] هم درصد مشخصی را دیتای آموزشی و بقیه را دیتای آزمون در نظر گرفته و درصد مشخصی هم از دادگان آموزشی را به‌عنوان اعتبارسنج مورداستفاده قرار می‌دهند؛ البته داده‌ای را به‌عنوان دادهٔ آموزشی در نظر می‌گیرند که میزان خطای آموزشی آن کمتر از حد مشخصی باشد.

۵. نتیجه‌گیری

در این کار پژوهشی، به بررسی اندکی از کارهای انجام‌شده در زمینهٔ تخمین هزینهٔ نرم‌افزار پرداختیم. در زمینهٔ تخمین هزینهٔ نرم‌افزار از آغاز راه تاکنون، کارهای فراوانی انجام شده که نیازمند پژوهش بیش‌ازپیش در این زمینه است. در پژوهش حاضر، تنها موفق به مرور قطره‌ای از دریای گستردهٔ مطالعات در این زمینه شدیم. اما آنچه مسلم است مطالعات کاربردی در این زمینه همچنان ادامه دارد. براساس مطالعات صورت‌گرفته از سال ۱۹۴۰، مفهوم تخمین هزینهٔ نرم‌افزار معرفی شده و به‌مرور، روش‌های مختلفی توسط پژوهشگران این حوزه ارائه گردیده است. اکثر مقالات مثل [۷] روش‌های تخمین تلاش را در دو دستهٔ الگوریتمی و غیرالگوریتمی تقسیم‌بندی کرده‌اند. روش‌های یادگیری ماشین هم در دسته مدل‌های غیرالگوریتمی قرار دارند. به‌دلیل شرایط ذاتی عمل تخمین، یعنی غیرقطعی بودن آن و همچنین هدف جامعهٔ پژوهشی که افزایش دقت در تخمین است، به‌مرور توجه پژوهشگران به کاربرد روش‌های یادگیری ماشین در تخمین هزینه و تلاش نرم‌افزار جلب شده است. آن‌ها تا به امروز با به‌کاربردن این روش‌ها، چه به‌صورت تکی و چه به‌صورت ترکیبی، اعم از ترکیب با روش‌های یادگیری ماشین دیگر و یا ترکیب با سایر روش‌ها، به دقت‌های مورد توجهی دست یافته‌اند. بااینکه هنوز اتفاق نظری صورت نگرفته، شایان ذکر است که براساس مطالعات مقایسه‌ای صورت‌گرفته، دقت روش‌های یادگیری ماشین نسبت‌به سایر روش‌ها، برتری ملموسی داشته است.

در مطالعاتی که به مرور کارهای پژوهشی از ابتدا تا به امروز در این زمینه پرداخته‌اند، روش‌های استدلال مبتنی بر مورد، شبکه‌های بیزین، قوانین وابستگی، درخت تصمیم، الگوریتم ژنتیک، برنامه‌نویسی ژنتیک، رگرسیون بردار پشتیبان و شبکه‌های عصبی مصنوعی، ۸ روش یادگیری ماشینی هستند که تا به امروز در تخمین هزینهٔ نرم‌افزار به‌کار برده شده‌اند و هرکدام دقت‌های مختلفی را ارائه داده‌اند. البته هشدار این مسئله ضروری است که هریک از روش‌های فوق، نیازمند برقراری شرایطی مناسب برای ارائهٔ دقت دلخواه هستند؛ برای مثال، روش درخت تصمیم، علاوه بر اینکه نمی‌تواند با داده‌های گم‌شده کار کند، مستعد پیش‌پرازش روی دادگان آموزشی کوچک هم است. روش‌های شبکهٔ عصبی مصنوعی و استدلال مبتنی بر مورد، هم نمی‌توانند روی دادگانی با داده‌های گم‌شده کار کنند هم اینکه برای کاربرد آن‌ها روی دادگان با داده‌های دسته‌ای باید ابتدا این دادگان را به دادگانی کمی تبدیل نمود، سپس از این دو روش برای تخمین هزینهٔ نرم‌افزار استفاده کرد. براساس مشاهدات و بررسی‌های صورت‌گرفته، دو روش CBR و ANN به نسبت سایرین، پرتکرارترین روش‌ها محسوب می‌شوند.

به‌منظور آموزش الگوریتم یادگیری ماشین، نیازمند استفاده از دادگانی هستیم که در فصل سوم پژوهش حاضر، به معرفی و تشریح برخی از این دادگان پرداخته شده است. دادگان در ابتدای امر به دو دستهٔ دادگان درون‌شرکتی و برون‌شرکتی تقسیم می‌شوند. پژوهشگرانی که به مقایسهٔ تأثیر استفاده از این دو نوع دادگان پرداخته‌اند، پیشنهاد می‌دهند که تا حد امکان سعی شود از دادگان درون‌شرکتی برای آموزش الگوریتم‌ها استفاده گردد. درصورتی‌که این نوع از دادگان در دسترس نبودند، بهتر است

1. Leave-one-out

دادگان برون‌شرکتی که بیشترین شباهت را بـه پـروژه‌ای کـه قـرار است هزینهٔ موردنیاز جهت توسعهٔ آن تخمـین زده شـود، انتخـاب گردد. دادگـانی مثـل COCOMO، ISBSG و TUKUTUKU در دستهٔ دادگان برون‌شرکتی محسـوب مـی‌شـوند و دادگـانی چـون DESHARNAIS، ALBRECHT و NASSA ازجملـــه دادگـــان درون‌شـرکتی هسـتند [۳۹]. شـایان توجـه اسـت کـه در انتخـاب دادگان، برای آموزش الگوریتم مورد اسـتفاده، عـلاوه بـر رعایـت بستر مطلوب آن الگوریتم، باید جنبه‌های کیفی و شباهت پروژه‌هـا را نیز در نظر بگیریم.

برای ارزیابی عملکرد روش‌های به‌کارگرفتـه‌شـده در تخمـین هزینهٔ نرم‌افزار، باید میزان دقت تخمین ارائـه‌شـده سـنجیده شـود. برای نیل به این هدف، معیارهای ارزیابی گوناگونی تا به امـروز در این حوزه اسـتفاده شـده‌انـد. بـه‌منظـور سـنجش دقـت الگـوریتم ارائه‌شده در مطالعات این حوزه، ابتدا دادگان مورداسـتفاده توسـط الگوریتم‌های اعتبارسنجی مثل اعتبارسنجی متقابـل[1]، بـه دو دسـتهٔ دادگان آموزشی و آزمون تقسیم شده و هر بـار بـا بررسـی میـزان دقت آموزش براساس یک یا چنـد نمونـه از معیـارهـای ارزیـابی دقت، بهترین دادگان آموزشی برگزیده شـده اسـت. سـپس مقـدار هزینهٔ موردنیاز برای نمونه‌هایی که در دستهٔ دادگـان آزمـون واقـع شده‌اند، تخمین زده شده و بـرای ارزیـابی دقـت الگـوریتم، آنچـه مدنظر قرار می‌گیرد، دقت سنجیده شدهٔ الگوریتم مـورد نظـر روی دادگان آزمون است. معیارهایی مثـل MRE، MMRE، MDMRE، PRED(X) ازجمله معیارهای مورداسـتفاده بـرای سـنجش دقـت الگوریتم‌های یادگیری ماشین در تخمین هزینهٔ نرم‌افزارند.

در این زمینه، تاکنون تحقیقات زیادی انجام شده است. کتـاب [۴۳] به بررسی زمینهٔ تخمین تلاش یـا هزینـهٔ نـرم‌افـزار پرداختـه است و دید جامعی در این زمینه برای مخاطبان فراهم می‌کند. ایـن کتاب انواع مدل‌های تخمین را معرفی کرده و روی دسـتهٔ خاصـی از این مدل‌ها تأکیدی نـدارد. البتـه برخـی مقـالات هـم سـعی در ترکیب روش‌های یادگیری ماشین با روش‌های دیگر برای افزایش دقت تخمین هزینهٔ نرم‌افزار بهـره جسـته‌انـد. در [۴۴] بـه بررسـی دقت تنها تعدادی از روش‌های یادگیری ماشین در حـوزهٔ تخمـین هزینه نرم‌افزار پرداخته شده است. در [۴۵] مروری بر چهـار روش بهینـه‌سـازی، منطـق فـازی، شـبکه‌هـای هوشـمند عصـبی و الگوریتم‌های دسته‌بنـدی کـه بـرای افـزایش دقـت تخمـین مـدل COCOMO بر روی آن اعمال شده، آمده است.

۵. ۱. پژوهش‌های پیشنهادی

براساس مطالعات صورت‌گرفته می‌توان بـه ایـده‌هـایی در جهـت ارائهٔ روش‌هایی نو برای تخمین هزینهٔ نرم‌افزار، آن‌هـم بـه‌صـورت دقیق‌تر دست یافت. برای پیشنهاد کارهای پیشرو می‌توان به مـوارد زیر اشاره کرد:

روش‌های یادگیری ماشین نتـایج امیدوارکننـده‌ای در تخمـین هزینهٔ نـرم‌افـزار ارائـه داده‌انـد. بـااین‌حـال اسـتفاده از روش‌هـای یادگیری ماشین در صنعت، هنوز گسترش چندانی نداشـته اسـت. بنابراین باید تلاش بیشتری در جهت تشویق صنعت و صنعتگران، برای استفاده از این روش‌ها در حوزهٔ تخمـین هزینـهٔ نـرم‌افـزار و تسهیل استفاده از آن‌ها صورت پذیرد. برای رسیدن به ایـن هـدف باید مطالعات آزمایشگاهی، مروری و جزئی‌تری روی ایـن حـوزه صورت پذیرد تا اطلاعات کامـل‌تـری از ایـن روش‌هـا در اختیـار محققان تخمین هزینه در حوزهٔ صنعت قرار گیرند.

استفاده از الگوریتم‌های انتخـاب ویژگـی‌هـایی مثـل انتخـاب ویژگی روبـه‌جلـو و انتخـاب ویژگـی روبـه‌عقـب در ترکیـب بـا روش‌های یادگیری ماشین استفاده‌شده، بـه‌نـدرت صـورت گرفتـه است. اما به‌نظر می‌رسد استفاده از ایـن نـوع ترکیبـات در افـزایش دقت تخمین مؤثر واقع گردد.

در پژوهش‌های صورت‌گرفته درزمینهٔ تخمین هزینهٔ نـرم‌افـزار از ابتدا تـاکنون بیشـتر از ۸ روش یـادگیری ماشـین اسـتفاده شـده است. استفاده از الگـوریتم‌هـایی کـه ایـن دسـته از روش‌هـا را بـا استفاده از تغییراتی جزئی بـه سـمت دقـت بیشـتر بـرده‌انـد، مثـل نسخه‌های جدید الگوریتم SVR، احتمـالاً دقـت آن‌هـا را در ایـن زمینه هم بالاتر ببرند.

استفاده از سایر الگوریتم‌های یادگیری ماشـین، کـه تـاکنون در این حوزه مورد استفاده قـرار نگرفتـه‌انـد هـم مـی‌توانـد بـه ارائـهٔ تخمینی دقیق منجر شود. مطالعات موردبررسی در ایـن پـژوهش،

1. Cross-Validation

پیشنهادهای مختلفی برای مطالعات و پـژوهش‌هـای آتـی در ایـن حوزه ارائه داده‌اند که در ادامه به بررسی برخـی از آن‌هـا خـواهیم پرداخت.

بنالا و همکارانش در [۲۲] برای انجام کارهـای آینـده در ایـن زمینه، پیشنهاد داده‌اند ازآنجایی‌که استفاده از الگوریتم‌هـای ترکیبـی به‌تازگی ارائه شده و کار چندانی بر روی آن صورت نگرفته اسـت و کارایی الگوریتم ارائه‌شده در پژوهش آن‌ها نیز نتـایج بهینـه‌ای را داشته است، به‌نظر می‌رسد می‌توان از سایر الگوریتم‌های یـادگیری ماشین نیز برای الگوریتم‌های ترکیبی استفاده کـرد. الگـوریتم‌هـای پیشنهادی در این مقاله شـامل بـر کلـونی زنبورعسـل مصـنوعی[1]، تکـاملی افتراقـی[2]، سیسـتم ایمنـی مصـنوعی[3]، بهینـه‌سـازی جست‌وجوی غذا باکتریایی[4]، فازی عصبی هستند.

راندینسـکی[5] و همکـارش در [۲۵] بـا مطالعـه‌ای کـه روی روش‌های یادگیری ماشین با داده‌هـای محلـی انجـام داده‌انـد، بـه نتایج خوبی رسیده‌اند. این مطالعه فقط روی داده‌های محلـی بـود؛ یعنی داده‌های یک کمپانی را مورداسـتفاده قـرار داده‌انـد. بنـابراین دادگان آن‌ها بزرگ نبودند؛ چراکه یک شرکت پروژه‌های زیادی را ایجاد نمی‌کند. آن‌ها در آینـده قصـد دارنـد مطالعـهٔ خـود را روی داده‌های برون‌شرکتی انجام دهند.

خطیبـی بردسـیری و همکـارانش در [۲۵] کـه بـا ارائـهٔ یـک الگوریتم ترکیبی با ادغام خوشـه‌بنـدی، شـبکهٔ عصـبی و ABE در زمینهٔ تخمین تلاش نرم‌افـزار، بـه‌دقـت خـوبی رسـیدند، در بیـان کارهای پیش رو آورده‌اند که بـه‌هرحـال بااینکـه مـدل پیشـنهادی تخمین متقاعدکننده‌ای بر روی دو دیتاسـت ارائـه مـی‌دهـد، ایـن روش با یک سری از محدودیت‌ها روبـه‌روسـت. اسـتفاده از ایـن مدل هنگامی‌که دیتاست سازگار و نرمـال باشـد، مناسـب نیسـت؛ چون عملکرد آن هنگامی‌که پروژه‌های درون دادگـان زیـاد باشـد، شبیه شبکه‌های عصـبی و پـروژه‌هـای کـم شـبیه ABE اسـت. و استفاده از آن‌وقت تلف کردن است. بنابراین قبل از استفاده از مدل پیشنهادی باید موقعیت دادگان، به‌دقت بررسی شـود. آن‌هـا قصـد دارند تا با ترکیب روش خوشه‌بندی با دیگر روش‌هـای هوشـمند، دقت تخمین را افزایش دهند.

عبدالله زید و همکارانش در[۸] که به بیـان موضـوعات داغ در زمینهٔ تخمین تلاش نرم‌افزار پرداخته‌اند، عناوینی مثل تبدیل معیـار، سایزبندی فاز طراحی و نیازمندی‌ها، آنالیز پیچیـدگی سیسـتم‌هـای نرم‌افزاری، آنالیز ROI[6]، بررسی OSS[7] و بهاگذاری مجدد در تـابع نقطه‌ای را عناوینی می‌دانند که بـه تمرکـز بیشـتر محققـان در ایـن زمینه نیاز دارند. آن‌هـا در پیشـنهادی کـه بـرای کارهـای پـیش‌رو دارند، روی تحقیقات بیشتر دربارهٔ تـأثیر موضـوعات داغ مفیـد در این زمینه برای گشودن درهـایی بـه روی محققـان و تـلاش‌هـای متمرکز محققان برای بهینه‌کـردن تخمـین تـلاش نـرم‌افـزار تأکیـد دارند.

در انتهای این بررسی، ذکر توصیه‌ها و دستورالعمل‌هایی بـرای متخصصان، به‌منظور اجرای مـدل‌هـای ML در صـنعت خـالی از لطف نیست [۴].

- درک منطقی از مدل‌هـای ML و همکـاری بـا محققـان باتجربه در اجرای مدل‌های ML.
- انتخاب آن دسته از مدل‌های ML که نقاط قوتی مطـابق با زمینه‌ها و اهداف تخمین دارند.
- اولویت‌بنـدی دادگـان درون‌شـرکتی در زمـان سـاخت مدل‌های ML، درصورتی‌که تعداد پروژه‌هـا در دادگـان کافی است.
- استفاده از مدل‌های ML به‌صورت موازی با مـدل‌هـای معمولی موجود در مراحل اولیـهٔ اسـتفاده از مـدل‌هـای ML
- به اشتراک‌گذاری دادگان پروژه‌های اختصاصی با جامعهٔ پژوهشی.

1. Artificial Bee Colony
2. Differential Evolution
3. Artificial Immune System
4. Bacterial foraging optimization
5. Lukasz Radlinski
6. Return Of Investment
7. Open Source Software

ضمیمهٔ الف. شرح اجمالی روش‌های استفاده‌شده در مقالات اخیر

ردیف	مقاله	سال نشر	نوع نشر	روش	دادگان	معیار سنجش دقت
۱	[۱]	۲۰۱۳	ژورنالی	ANN	COCOMO	-
۲	[۲]	۲۰۱۳	کنفرانس	کاهش انحراف تلاش با مدل COCOMOII	ISBSG	-
۳	[۵]	۲۰۱۲	ژورنالی	SVR-MPSOMODEL-INTERMADIATE COCOMO	COCOMO	MARE-PRED
۴	[۹]	۲۰۱۲	کنفرانس	ANN-LINEAR REGRESSION-MP5	COCOMO81	MMRE-PRED
۵	[۱۱]	۲۰۱۳	ژورنالی	MLP-SVR-ANFIS-PCA-GA	COCOMO-ALBRECHT-MIAZAKI-MAXWELL-DESHARNAIS	MMRE-PRED-EF
۶	[۱۲]	۲۰۱۱	ژورنالی	RBNN-GP-FAZY LOGIC-PSOA	NASSA	MMRE-PRED
۷	[۱۳]	۲۰۰۷	ژورنالی	SVM و DT، PCA، RBF،MLP	Nassa-ucs	MMRE-PRED
۸	[۱۴]	۲۰۱۲	ژورنالی	OLS-OLS+LOG-OLS+BC-RUBOST REGRESSION-LMS-MARS-RIDGE REGRESSION-CART-M5-MLP-RBFN-CBR-LS SVM-	COCOMO81-CSC-DESHARNAIS	MRE-MMRE-MDMRE-PRED-
۹	[۲۰]	۲۰۱۰	ژورنالی	AODE-BN-NB-MLP-RBFN-SMO-KNN-K*-LBR-.....	QQDefects-Maxwell-DESHARNAIS-COCOMO	CONFIDENCE
۱۰	[۲۲]	۲۰۱۲	ژورنالی	GA+ ANN	COCONASSA2	MRE-MMRE-PRED
۱۱	[۲۳]	۲۰۱۲	ژورنالی	MULTIPLE LINEAR REGRESSION-GP	دادگان ایجادشده توسط پژوهشگران آن	MRE-MER-MMRE-MMER
۱۲	[۲۴]	۲۰۰۸	ژورنالی	AR+CART	STTF-ISBSG-OTHER APPLICATION	CONFIDENCE
۱۳	[۲۵]	۲۰۱۲	ژورنالی	ANN+ABE+CLUSTERING	MAXWELL-DESHARNAIS	RE-MRE-MMRE-PRED
۱۴	[۲۶]	۲۰۰۸	ژورنالی	ANN+MIN-MAX	COCOMO81	MMRE
۱۵	[۲۷]	۲۰۱۴	ژورنالی	CART-Multipel Regression-Logestic Regression, MIN-MAX	COCOMO81-NASSA93-Baily baisili	MMRE-MDMRE-PRED
۱۶	[۲۹]	۲۰۱۰	ژورنالی	ANN	COCOMO	MMRE-PRED
۱۷	[۳۱]	۲۰۱۳	ژورنالی	NB-J48-BPNN-CLUSTERING	ISBSG-CSBSG	CONFIDENCE
۱۸	[۳۴]	۲۰۱۲	ژورنالی	ANN+COCOMO MODEL	COCOMO	MRE

مراجع

[1] Sarwar, S., Gupta, M., "Proposing effort estimation of COCOMO II through perceptron learning rule", International Journal of Computer Applications, 70(1), pp. 29-32, 2013.

[۲] دولت‌آبادی نژاد، ابوذر؛ سیدمهران شرفی و ناصر نعمت‌بخش، «بررسی تأثیر کاهش انحراف توزیع تلاش بر افزایش دقت تخمین تلاش در توسعهٔ محصول نرم‌افزاری»، پنجمین کنفرانس ملی مهندسی برق و الکترونیک ایران، گناباد، ۱۳۹۲.

[3] Grewal, A. S., Gupta, V., Kumar, R. Emerging, estimation techniques. International Journal of Computer Applications, Vol. 59, No. 8, pp. 30-34, 2012.

[4] Wen, J., Li, S., Lin, Z., Hu, Y., & Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. Information and Software Technology, 54(1), pp. 41-59.

[5] Kumari, S., & Pushkar, S. (2013). Comparison and analysis of different software cost estimation methods. IJACSA) International Journal of Advanced Computer Science and application, 4(1), pp. 153-157.

[6] Pandey, P. (2013, April). Analysis of the techniques for software cost estimation. In Advanced Computing and Communication Technologies (ACCT), 2013 Third International Conference on (pp. 16-19). IEEE.

[7] Bardsiri, A. K., & Hashemi, S. M. (2014). Software effort estimation: A survey of well-known approaches. International Journal of Computer Science Engineering (IJCSE), 3(1), pp. 46-50.

[8] Zaid, A., Selamat, M. H., Ghani, A. A. A., Atan, R., & Wei, K. T. (2008). Issues in software cost estimation. IJCSNS Int J of Computer Science and Network Security, 8(11), pp. 350-356.

[۹] اخوان بیطرف، امیر؛ مونا یخچی و بهناز محجوبی، «ارائه روشی بهینه برای تخمین هزینهٔ نرم‌افزار مبتنی بر شبکهٔ عصبی پرسپترون به کمک داده‌کاوی اطلاعات نرم‌افزار»، دومین کنفرانس ملی محاسبات نرم و فناوری اطلاعات، ماهشهر، ۱۳۹۰

[10] Keim, Y., Bhardwaj, M., Saroop, S., & Tandon, A. (2014, January). Software cost estimation models and techniques: A survey. In International Journal of Engineering Research and Technology, 3(2), pp.1763-1768).

[11] Elish, M. O., Helmy, T., & Hussain, M. I. (2013). Empirical study of homogeneous and heterogeneous ensemble models for software development effort estimation. Mathematical Problems in Engineering, 2013, pp. 1-21.

[12] Sehra, S. K., Brar, Y. S., & Kaur, N. (2011). Soft computing techniques for software project effort estimation. International Journal of Advanced Computer and Mathematical Sciences, 2(3), pp. 160-167.

[13] Baskeles, B., Turhan, B., & Bener, A. (2007, November). Software effort estimation using machine learning methods. In Computer and information sciences, 2007. iscis 2007. 22nd international symposium on (pp. 1-6). IEEE.

[14] Alpaydin, E. (2014). Introduction to machine learning. MIT press.

[`15] Dejaeger, K., Verbeke, W., Martens, D., & Baesens, B. (2012). Data mining techniques for software effort estimation: A comparative study. IEEE Transactions on Software Engineering, 38(2), pp. 375-397.

[16] Hamza, H., Kamel, A., & Shams, K. (2013, April). Software effort estimation using artificial neural networks: A survey of the current practices. In Information Technology: New Generations (ITNG), 2013 Tenth International Conference on (pp. 731-733). IEEE.

[17] Han, J., Pei, J., & Kamber, M. (2011). Data mining: Concepts and techniques. Elsevier.

[18] "Bayesian network - Wikipedia, the free encyclopedia." pp. 1, 2011.

[19] Radlinski, L. (2010). A survey of bayesian net models for software development effort prediction. International Journal of Software Engineering and Computing, 2(2), pp.95-109.

[20] Radlinski, Lukasz, and Wladyslaw Hoffmann. (2010). "On predicting software development effort using machine learning techniques and local data." International Journal of Software Engineering and Computing 2(2), pp. 123-136.

[21] "Genetic algorithm - Wikipedia, the free encyclopedia." pp. 1, 2011.

[22] Benala, T. R., Dehuri, S., Satapathy, S. C., & Madhurakshara, S. (2012). Genetic algorithm for optimizing functional link artificial neural network based software cost estimation. In Proceedings of the International Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) held in Visakhapatnam, India, January 2012 (pp. 75-82). Springer Berlin Heidelberg.

[23] Chavoya, A., Lopez-Martin, C., Andalon-Garcia, I. R., & Meda-Campaña, M. E. (2012). Genetic programming as alternative for predicting development effort of individual software projects. PloS one, 7(11), e50531.

[24] Bibi, S., Stamelos, I., & Angelis, L. (2008). Combining probabilistic models for explanatory productivity estimation. Information and Software Technology, 50(7), pp. 656-669.

[25] Bardsiri, V. K., Jawawi, D. N. A., Hashim, S. Z. M., & Khatibi, E. (2012). Increasing the accuracy of software development effort estimation using projects clustering. IET software, 6(6), pp. 461-473.

[26] Ayyıldız, M., Kalıpsız, O., & Yavuz, S. (2008). A metric-Set and model suggestion for better software project cost estimation. International Journal of Computer, Information, Systems and Control Engineering, 2(11), pp. 262-267.

[27] Benala, T. R., Mall, R., Srikavya, P., & HariPriya, M. V. (2014). Software effort estimation using data mining techniques. In ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol I (pp. 85-92). Springer International Publishing.

[28] Kocaguneli, E., Menzies, T., Keung, J., Cok, D., & Madachy, R. (2013). Active learning and effort estimation: Finding the essential content of software effort estimation data. IEEE Transactions on Software Engineering, 39(8), pp. 1040-1053.

[29] Attarzadeh, I., & Ow, S. H. (2010, April). Proposing a new software cost estimation model based on artificial neural networks. In Computer Engineering and Technology (ICCET), 2010 2nd International Conference on (Vol. 3, pp. V3-487). IEEE.

[۳۰] دولت‌آبادی‌نژاد، ابوذر؛ سید مهران شرفی و ناصر نعمت‌بخش، «بررسی رویکردهای استفاده از منطق فازی در تخمین هزینه و تلاش نرم‌افزار»، همایش ملی علوم و مهندسی کامپیوتر، نجف‌آباد، دانشگاه آزاد اسلامی واحد نجف‌آباد، ۱۳۹۱

[31] Zhang, W., Yang, Y., & Wang, Q. (2011, June). A study on software effort prediction using machine learning techniques. In International Conference on Evaluation of Novel Approaches to Software Engineering (pp. 1-15). Springer Berlin Heidelberg.

[۳۲] ایرانمنش، الناز؛ وحید خطیبی بردسیری، «کاربرد انواع مدل‌های الگوریتمی در تخمین تلاش پروژه‌های نرم‌افزاری»، همایش ملی پژوهش‌های مهندسی رایانه‌ای، تهران، ۱۳۹۳.

[33] Bailey, J. W., & Basili, V. R. (1981, March). A meta-model for software development resource expenditures. In Proceedings of the 5th international conference on Software engineering (pp. 107-116). IEEE Press.

[34] Wolverton, R. W. (1974). The cost of developing large-scale software. IEEE Transactions on Computers, 100(6), pp. 615-636.

[35] Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. IEEE transactions on Software Engineering, 4(4), pp. 345.-361.

[36] Abbas, S. A., Liao, X., Rehman, A. U., Azam, A., & Abdullah, M. I. (2012). Cost estimation: A survey of well-known historic cost estimation techniques. Journal of Emerging Trends in Computing and Information Sciences, 3(4), pp. 612-636.

[37] http://www.softwaremetrics.com/fpafund.htm, access in 21/12/2013.

[38] Kitchenham, B. A., Mendes, E., & Travassos, G. H. (2007). Cross versus within-company cost estimation studies: A systematic review. IEEE Transactions on Software Engineering, 33(5), pp. 316-329.

[39] Mendes, E., Kalinowski, M., Martins, D., Ferrucci, F., & Sarro, F. (2014, May). Cross-vs. within-company cost estimation studies revisited: an extended systematic review. In Proceedings of the 18th ACM International Conference on Evaluation and Assessment in Software Engineering, p. 12.

[40] Kaushik, A., Chauhan, A., Mittal, D., & Gupta, S. (2012). COCOMO estimates using neural networks. International Journal of Intelligent Systems and Applications, 4(9), pp. 22-28.

[۴۱] بیرانوند، صبا؛ محمدعلی زارع چاهوکی، «افزایش دقت در تخمین هزینهٔ نرم‌افزار، با انتخاب زیرمجموعه‌ای از ویژگی‌ها مبتنی بر تلفیق معیارهای ارزیابی»، هشتمین کنفرانس داده‌کاوی ایران، تهران، ۱۳۹۳

[42] Minku, L. L., & Yao, X. (2013). Software effort estimation as a multiobjective learning problem. ACM Transactions on Software Engineering and Methodology (TOSEM), 22(4), pp. 1-32.

[43] Trendowicz, Adam, and Ross Jeffery. Software project effort estimation. DOI 10.1007/978-3-319-03629-8,# Springer International Publishing Switzerland 2014 451, 2014.

[44] Srinivasan, K., & Fisher, D. (1995). Machine learning approaches to estimating software development effort. IEEE Transactions on Software Engineering, 21(2), pp. 126-137.

[۴۵] جهانشاهی، سعیده؛ وحید خطیبی بردسیری؛ رضوان حسنی سعدی و معصومه مقبلی، «کاربرد روش‌های محاسبات نرم در بهبود عملکرد مدل کوکومو به‌منظور تخمین هزینهٔ پروژه‌های نرم‌افزاری»، همایش ملی مهندسی کامپیوتر و فناوری اطلاعات، شوشتر، دانشگاه آزاد اسلامی واحد شوشتر، ۱۳۹۲.