Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
DOI: 10.22034/jscit.2026.552789.2183
Volume 15, Number 1, Spring 2026, pp. 76-96
Received: 10/12/2025, Revised: 02/10/2026, Accepted: 02/21/2026

Optimized Feature Selection for Phishing Attacks Detection Based on Domain Similarity Measures

Mojtaba Shokoh[1], Hassan Rezaei[2*], Mohammad Mehdi Keikha[3] and Yahya Kord Tamandani[4]
1- Department of Computer Science, University of Sistan and Baluchestan, Zahedan, Iran.
2*- Department of Computer Science, University of Sistan and Baluchestan, Zahedan, Iran.
3- Department of Computer Science, University of Sistan and Baluchestan, Zahedan, Iran.
4- Department of Computer Science, University of Sistan and Baluchestan, Zahedan, Iran.
[1]shokoh@pgs.usb.ac.ir, [2*]hrezaei@cs.usb.ac.ir, [3]keikha@cs.usb.ac.ir, and [4]yahya.kord@cs.usb.ac.ir

Corresponding author's address: Hassan Rezaei, Faculty of Mathematics, Statistics and Computer Science, University of Sistan and Baluchestan, Zahedan, Iran.

Abstract- The rapid growth of cyberspace has significantly impacted various aspects of human life. While this technology has enabled the emergence of numerous information-based businesses, it has also introduced certain risks, including the proliferation of phishing attacks. Phishing websites often mimic legitimate sites and replicate their content, making detection challenging due to the need for extensive data collection and content analysis. To address this issue, this study focuses specifically on domain-related features. The underlying premise is that phishing domains attempt to resemble the domains of well-established and trusted websites. Accordingly, a set of similarity features derived from Google search results is introduced. These features, combined with dependency-based and value-based features, achieved a detection accuracy of 99.26% using a total of 127 features. Furthermore, feature evaluation based on filter-based selection criteria demonstrated that the five proposed similarity features ranked among the second to sixth most influential features. Finally, by reducing the feature set to 54 features, the classification accuracy further improved to 99.47%, demonstrating the effectiveness and efficiency of the proposed approach.

Keywords- Feature selection, Phishing detection, Machine learning, Real web environment.

مجله علمی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
شناسه دیجیتال: 10.22034/jscit.2026.552789.2183
جلد ۱۵، شماره ۱، بهار ۱۴۰۵، صفحه ۷۶-۹۶
دریافت: ۱۴۰۴/۰۷/۲۰، بازنگری: ۱۴۰۴/۱۱/۲۱، پذیرش: ۱۴۰۴/۱۲/۰۲

انتخاب ویژگی‌های بهینه برای تشخیص حملات فیشینگ مبتنی بر معیارهای شباهت دامنه

مجتبی شکوه[۱]، حسن رضایی*[۲]، محمدمهدی کیخا[۳]، یحیی کرد تمندانی[۴]
۱- گروه علوم کامپیوتر، دانشگاه سیستان و بلوچستان، زاهدان، ایران.
*۲- گروه علوم کامپیوتر، دانشگاه سیستان و بلوچستان، زاهدان، ایران.
۳- گروه علوم کامپیوتر، دانشگاه سیستان و بلوچستان، زاهدان، ایران
۴- گروه علوم کامپیوتر، دانشگاه سیستان و بلوچستان، زاهدان، ایران.
[۱]shokoh@pgs.usb.ac.ir, [۲*]hrezaei@cs.usb.ac.ir, [۳]keikha@cs.usb.ac.ir, [۴]yahya.kord@cs.usb.ac.ir

* نشانی نویسنده مسئول: حسن رضایی، زاهدان، دانشگاه سیستان و بلوچستان، دانشکده ریاضی، آمار و علوم کامپیوتر.

چکیده- امروزه رشد روزافزون فضای مجازی بر ابعاد مختلف زندگی انسان تأثیر گذاشته و مزیت‌های استفاده از این فناوری موجب شده است که بسیاری از کسب‌وکارها بر پایه تبادل اطلاعات در این فضا شکل بگیرند. با وجود تمام مزایای این فناوری، مخاطراتی نیز ممکن است افراد استفاده‌کننده از آن را تهدید کند که یکی از این مخاطرات، محتواهای تقلبی است که می‌تواند به‌عنوان ابزاری برای انجام حملات فیشینگ مورد استفاده قرار گیرد. از آن‌جا که سایت‌های فیشینگ معمولاً ظاهری مشابه وب‌سایت‌های قانونی دارند و محتوای آن‌ها را تکرار می‌کنند، تشخیص آن‌ها به‌دلیل گردآوری و تحلیل محتوای سایت‌ها پیچیده می‌شود. لذا در این مقاله، به‌طور خاص به ویژگی‌های مرتبط با دامنه سایت پرداخته شده است. این ایده که دامنه یک سایت فیشینگ سعی در ایجاد تشابه با دامنه سایت‌های معتبر و قانونی دارد، در نظر گرفته شده است. به همین منظور، یک مجموعه ویژگی مبتنی بر شباهت بر اساس نتایج موتور جستجوی گوگل معرفی شده است. این ویژگی‌ها به همراه ویژگی‌های دیگر که در دو گروه ویژگی‌های مبتنی بر وابستگی و ویژگی‌های مبتنی بر ارزش توانسته‌اند با استفاده از ۱۲۷ ویژگی، دقت ۹۹,۲۶ درصد در شناسایی سایت‌های فیشینگ را به دست آورند. همچنین، با ارزیابی ویژگی‌ها با استفاده از معیارهای انتخاب ویژگی فیلتر، نشان داده شده است که ۵ ویژگی مبتنی بر شباهت معرفی‌شده توانسته‌اند رتبه‌های دوم تا ششم بهترین ویژگی‌ها را از نظر معیارهای انتخاب ویژگی کسب کنند. در نهایت، با کاهش ابعاد مجموعه ویژگی‌ها به ۵۴ ویژگی، دقت طبقه‌بندی به ۹۹,۴۷ درصد بهبود یافته است.

واژه‌های کلیدی: انتخاب ویژگی، تشخیص فیشینگ، یادگیری ماشین، محیط وب واقعی.

۱- مقدمه

فیشینگ یکی از تهدیدهای مهم در فضای مجازی است که مهاجمان با استفاده از دامنه‌هایی مشابه دامنه‌های معتبر، کاربران را فریب داده و به صفحات جعلی هدایت می‌کنند. مرورگرها به‌عنوان اولین لایه دفاعی می‌توانند نقش مهمی در شناسایی این صفحات ایفا کنند، اما اغلب به لیست‌های سیاه وابسته‌اند [۱]. این رویکرد با محدودیت‌هایی مانند تأخیر در به‌روزرسانی و ناکارآمدی در برابر حملات جدید روبه‌رو است.

روش‌های مبتنی بر محتوای صفحه، مانندCANTINA [۲]، هرچند مؤثر بودند، اما در برابر بازتولید محتوای قانونی توسط مهاجمان یا رمزنگاری کدهای اسکریپت آسیب‌پذیر هستند. همچنین، استفاده از هسته مرورگر در این روش‌ها خود می‌تواند

منجر به ایجاد آسیب‌پذیری‌های امنیتی شود [۳]. از سوی دیگر، چارچوب‌های پیچیده‌تری مانندCASE [۴]، به‌دلیل اتکای زیاد بر ترکیب ویژگی‌های متنوع، اجرای پرهزینه و دشواری در محیط واقعی دارند.

برای رفع این محدودیت‌ها، در این مقاله یک چارچوب سبک و کارآمد با عنوان ACS (Affiliation–Cost–Similarity) ارائه می‌شود. نوآوری اصلی ACS در ترکیب سه گروه ویژگی است: ویژگی‌های شباهتی مبتنی بر شباهت دامنه‌ها، ویژگی‌های وابستگی مبتنی بر اطلاعات وابستگی دامنه، و ویژگی‌های ارزشی مبتنی بر ارزش و اعتبار دامنه، که همگی صرفاً بر اطلاعات مربوط به دامنه متمرکز هستند. این چارچوب علاوه بر کاهش سربار محاسباتی، امکان شناسایی لحظه‌ای سایت‌های فیشینگ را فراهم می‌سازد. نتایج آزمایش‌ها نشان می‌دهد که ویژگی‌های شباهتی معرفی‌شده، نسبت به ویژگی‌های متداول در پژوهش‌های پیشین، از اهمیت بالاتری در فرآیند انتخاب ویژگی و طبقه‌بندی برخوردارند.

برخلاف چارچوب‌های پیشین نظیر CASE و PhishFarm که یا به تحلیل محتوای صفحه، یا پایگاه‌های لیست سیاه و یا ترکیب گسترده‌ای از ویژگی‌های ناهمگون متکی هستند، چارچوب ACS به‌صورت هدفمند تنها بر اطلاعات دامنه تمرکز دارد و سه بُعد مکمل اما مستقل را پوشش می‌دهد: (۱) ویژگی‌های شباهتی که میزان تقلید دامنه‌های فیشینگ از دامنه‌های معتبر را مدل می‌کنند، (۲) ویژگی‌های وابستگی که روابط ساختاری و رجیستری دامنه را بدون نیاز به محتوای صفحه استخراج می‌کنند، و (۳) ویژگی‌های ارزشی که هزینه و پیچیدگی ایجاد دامنه‌های فیشینگ را برای مهاجم منعکس می‌سازند. این تفکیک صریح و هم‌زمان این سه بُعد، که در مطالعات پیشین به‌صورت پراکنده یا ناقص بررسی شده‌اند، نوآوری اصلی ACS را تشکیل می‌دهد.

ساختار مقاله به این صورت است: در بخش ۲ کارهای مرتبط مرور می‌شوند، در بخش ۳ مفاهیم پایه ارائه می‌شود، در بخش ۴ ویژگی‌ها معرفی شده و در بخش ۵ روش پیشنهادی تشریح می‌شود. بخش ۶ شامل آزمایش‌ها و نتایج است و در نهایت، در بخش ۷ مقایسه با روش‌های قبلی و جمع‌بندی ارائه شده است.

۲- پیشینه تحقیق

روش‌های شناسایی وب‌سایت‌های فیشینگ را می‌توان بر اساس نوع ویژگی‌ها و رویکرد تحلیلی به چند دسته اصلی تقسیم کرد. در این بخش، مطالعات پیشین به‌صورت دسته‌بندی‌شده و نظام‌مند بررسی می‌شوند و در انتها شکاف تحقیقاتی موجود تبیین خواهد شد.

۲-۱- روش‌های مبتنی بر محتوا

روش‌های مبتنی بر محتوا از نخستین رویکردهای تشخیص فیشینگ محسوب می‌شوند که با تحلیل متن، ساختار HTML و کدهای سمت کاربر اقدام به شناسایی وب‌سایت‌های جعلی می‌کنند. از جمله این روش‌ها می‌توان به CANTINA و همچنین مطالعه ارائه‌شده در مقاله «مدل نوینی برای تشخیص وب‌سایت‌های فیشینگ مبتنی بر LightGBM[۱] و ویژگی‌های نام دامنه [۵] که در این پژوهش به اختصار LGBM-DNF نام‌گذاری شده است، اشاره کرد که با استخراج ویژگی‌های آماری، ساختاری و الگوهای محتوایی صفحات وب، تفاوت میان وب‌سایت‌های قانونی و فیشینگ را بررسی می‌کند.

با وجود بهبود دقت نسبت به روش‌های سنتی، این دسته از روش‌ها با دو چالش اساسی مواجه‌اند: نخست، مهاجمان می‌توانند محتوای صفحات قانونی را به‌راحتی بازتولید یا دستکاری کنند؛ و دوم، پردازش محتوای صفحات وب باعث افزایش پیچیدگی و سربار محاسباتی می‌شود.

علاوه بر این، در سال‌های اخیر بسیاری از وب‌سایت‌های فیشینگ از تکنیک‌های رمزنگاری و مبهم‌سازی (Obfuscation) برای محتوای HTML و به‌ویژه کدهای JavaScript استفاده می‌کنند. این امر تحلیل ایستای محتوا را دشوارتر کرده و کارایی روش‌های مبتنی بر محتوا را به‌طور جدی کاهش داده است، زیرا استخراج ویژگی‌های قابل اتکا نیازمند رمزگشایی، اجرای پویا و تحلیل پیچیده کدها خواهد بود.

۲-۲- روش‌های مبتنی بر لیست سیاه

روش‌های مبتنی بر لیست سیاه، مانند PhishFarm [۱] بر استفاده از پایگاه‌های داده‌ای شامل دامنه‌ها و URLهای مخرب شناخته‌شده تکیه دارند. این روش‌ها برای حملات تکراری و شناخته‌شده عملکرد قابل قبولی دارند، اما وابستگی شدید آن‌ها به به‌روزرسانی لیست‌ها موجب می‌شود در برابر حملات روز-صفر ناتوان باشند. تأخیر زمانی در شناسایی دامنه‌های جدید باعث می‌شود بخش قابل توجهی از حملات فیشینگ پیش از ثبت در لیست سیاه، کاربران را هدف قرار دهند.

۲-۳- روش‌های مبتنی بر ویژگی‌های دامنه

در سال‌های اخیر، توجه قابل توجهی به استفاده از ویژگی‌های دامنه و URL معطوف شده است، زیرا این ویژگی‌ها مستقل از محتوای صفحه بوده و هزینه محاسباتی پایینی دارند. مقاله PhiUSIIL [۶] و تشخیص صفحات وب فیشینگ بر اساس شباهت بصری سراسری و محلی [۷] که در این پژوهش به اختصار V-Similarity نام‌گذاری شده است و کووید-۱۹ [۸] نمونه‌هایی از این رویکرد هستند که با استفاده از ویژگی‌هایی مانند طول دامنه، تعداد زیردامنه‌ها و الگوهای نگارشی، روش‌هایی سریع و سبک‌وزن ارائه داده‌اند. هرچند این روش‌ها برای محیط‌های با منابع محدود مناسب‌اند، اما در مواجهه با حملات پیچیده و دامنه‌های جعلی پیشرفته، دقت آن‌ها کاهش می‌یابد.

از سوی دیگر، روش CASE با ترکیب چهار دسته ویژگی (Counterfeiting، Affiliation، Stealing و Evaluation) تلاش کرده دیدگاهی جامع‌تر ارائه دهد. با این حال، وابستگی این روش به ویژگی‌های محتوایی و پیچیدگی استخراج ویژگی‌ها، پیاده‌سازی عملی آن را در سامانه‌های بلادرنگ با چالش مواجه کرده است.

در حالی که چارچوب CASE چهار دسته ویژگی ناهمگون شامل محتوای صفحه، وابستگی سازمانی و ارزیابی رفتاری را ترکیب می‌کند، ACS با حذف کامل ویژگی‌های محتوایی و تمرکز صرف بر دامنه، یک چارچوب سبک‌وزن و مناسب پردازش بلادرنگ ارائه می‌دهد که از نظر هزینه محاسباتی و قابلیت استقرار عملی، برتری محسوسی دارد.

۲-۴- روش‌های مبتنی بر شباهت و بینایی ماشین

برخی مطالعات تمرکز خود را بر تحلیل شباهت میان وب‌سایت‌ها قرار داده‌اند. روش CPS [۹] با مقایسه ساختار و محتوای صفحات وب مشکوک با وب‌سایت‌های معتبر، نشان می‌دهد که روش‌های مبتنی بر شباهت می‌توانند بدون نیاز به پایگاه داده‌های بزرگ، دقت مناسبی داشته باشند. با این حال، وابستگی شدید این روش‌ها به محاسبات شباهت و عدم توجه به هزینه پردازشی و وابستگی دامنه‌ها، مقیاس‌پذیری آن‌ها را محدود می‌کند.

در همین راستا، روش LogoSENSE [۱۰]، Multi-Scale [۱۱] و مقاله یک پروتکل دفاعی فیشینگ روز صفر مبتنی بر شفافیت صفحه ورود و شباهت بصری [۱۲] که در این مطالعه به اختصار LPTVS-Protocol نامیده شده است از شباهت بصری و تحلیل عناصر گرافیکی مانند رنگ، چیدمان و لوگو برای شناسایی حملات روز-صفر استفاده می‌کند. اگرچه این روش در برخی سناریوها عملکرد مناسبی دارد، اما پردازش ویژگی‌های بصری هزینه‌بر بوده و وابستگی به رندر کامل صفحات وب، کارایی آن را در محیط‌های بلادرنگ کاهش می‌دهد. افزون بر این، تغییرات ساده ظاهری می‌توانند این دسته از روش‌ها را فریب دهند.

۲-۵- روش‌های مبتنی بر یادگیری عمیق

مطالعات اخیر به استفاده از روش‌های یادگیری عمیق برای تشخیص فیشینگ روی آورده‌اند. روش In-Depth [۱۳] با ترکیب ویژگی‌های URL، محتوا و ساختار لینک‌ها و به‌کارگیری شبکه‌های عصبی عمیق، دقت بالایی در شناسایی وب‌سایت‌های فیشینگ گزارش کرده است. در ادامه، وی و همکاران با استفاده از شبکه‌های عصبی کانولوشنی (CNN) و تحلیل صرف متن URL، روشی سریع و سبک‌وزن ارائه داده‌اند که بدون اتکا به محتوای صفحه قادر به تشخیص حملات روز-صفر است [۱۴]. به‌طور مشابه، یک مدل CNN کم‌عمق مبتنی بر توکن‌سازی URLها، دقتی حدود ۹۸٪ در تشخیص URLهای فیشینگ نشان داده است [۱۵].

در سال‌های اخیر، معماری‌های مبتنی بر Transformer نیز برای مدل‌سازی وابستگی‌های طولانی‌مدت معرفی شده‌اند. روش URLTran با بهره‌گیری از مدل‌های Transformer نظیر BERT و RoBERTa عملکرد بهتری، به‌ویژه در نرخ‌های بسیار پایین خطای مثبت کاذب، ارائه می‌دهد [۱۶]. همچنین PhishTransformer با ترکیب CNN و Transformer و استفاده از URLهای درون صفحه وب، دقت بالایی در تشخیص حملات فیشینگ گزارش کرده است [۱۷]. با وجود این مزایا، نیاز به داده‌های حجیم و هزینه محاسباتی بالا، استفاده عملی از این روش‌ها را در سامانه‌های بلادرنگ محدود می‌کند.

۲-۶- جمع‌بندی و شکاف تحقیقاتی

مرور نظام‌مند مطالعات پیشین نشان می‌دهد که هیچ‌یک از رویکردهای موجود به‌تنهایی قادر به پوشش کامل چالش‌های تشخیص فیشینگ نیستند. روش‌های مبتنی بر محتوا و شباهت به‌دلیل رمزنگاری و مبهم‌سازی روزافزون محتوای وب و کدهای JavaScript با محدودیت جدی مواجه‌اند؛ روش‌های مبتنی بر لیست سیاه در برابر حملات جدید ناکارآمد هستند؛ روش‌های دامنه‌محور ساده سرعت بالایی دارند اما جامعیت کافی ندارند؛ و روش‌های یادگیری عمیق با وجود دقت بالا، هزینه محاسباتی و پیچیدگی زیادی تحمیل می‌کنند.

شکاف اصلی تحقیقاتی در نبود چارچوبی است که بدون اتکا به محتوای صفحه و لیست‌های سیاه، با هزینه محاسباتی پایین و دقت بالا عمل کند. در پاسخ به این شکاف، مقاله حاضر چارچوب ACS را معرفی می‌کند که با ترکیب هم‌زمان سه دسته ویژگی وابستگی (Affiliation)، ارزشی (Cost) و شباهت (Similarity)، تعادلی مؤثر میان سادگی، سرعت پردازش و دقت تشخیص ایجاد کرده و محدودیت‌های روش‌های پیشین را برطرف می‌سازد.

نوآوری این پژوهش علاوه بر معرفی معیارهای شباهت، بلکه در طراحی یک چارچوب منسجم، دامنه‌محور و کم‌هزینه است که این سه دسته ویژگی شباهتی، وابستگی و ارزشی را در قالب یک چارچوب دامنه‌محور و مستقل از محتوا به‌صورت یکپارچه و قابل استقرار در محیط‌های بلادرنگ ترکیب می‌کند.

۳- مفاهیم پایه

در این بخش، مفاهیم پایه‌ای که در این مقاله استفاده شده است، شرح داده شده است.

XGBoost یک الگوریتم تقویت گرادیانی مبتنی بر درخت تصمیم است که با استفاده از بهینه‌سازی مرتبه دوم، منظم‌سازی و پیاده‌سازی کارآمد، عملکرد بالایی در مسائل یادگیری ماشین ارائه می‌دهد. این روش با کاهش بیش‌برازش و افزایش مقیاس‌پذیری، به‌طور گسترده در مسائل داده‌کاوی و یادگیری ماشین مورد استفاده قرار گرفته است[۱۸].

LightGBM یک الگوریتم تقویت گرادیانی مبتنی بر درخت است که از استراتژی رشد برگ‌محور برای افزایش سرعت آموزش و کاهش مصرف حافظه استفاده می‌کند. این الگوریتم به‌ویژه برای داده‌های بزرگ و با ابعاد بالا مناسب بوده و کارایی بالاتری نسبت به روش‌های سنتی دارد[۱۹].

CatBoost یک الگوریتم تقویت گرادیانی است که به‌صورت ویژه برای پردازش مؤثر ویژگی‌های دسته‌ای طراحی شده است. این الگوریتم با بهره‌گیری از روش Ordered Boosting و کدگذاری هدف‌محور منظم‌شده، نشت اطلاعات را کاهش داده و دقت مدل را افزایش می‌دهد[۲۰].

TabNet یک مدل یادگیری عمیق برای داده‌های جدولی است که با استفاده از مکانیزم توجه ترتیبی (Sequential Attention) در هر مرحله، زیرمجموعه‌ای از ویژگی‌های مرتبط را انتخاب می‌کند. این ساختار ضمن حفظ قابلیت تفسیرپذیری، عملکرد رقابتی با مدل‌های مبتنی بر درخت در مسائل طبقه‌بندی داده‌های جدولی ارائه می‌دهد[۲۱].

Random Forest یک روش یادگیری تجمعی مبتنی بر درخت تصمیم است که با ترکیب چندین درخت ساخته‌شده بر روی نمونه‌های تصادفی داده و ویژگی‌ها، دقت مدل را افزایش داده و بیش‌برازش را کاهش می‌دهد [۲۲].

AdaBoost یک الگوریتم تقویت‌کننده است که با ترکیب چند یادگیرنده ضعیف و تمرکز بیشتر بر نمونه‌های خطادار در مراحل بعدی، یک مدل قوی با دقت بالا ایجاد می‌کند [۲۳].

Decision Tree یک مدل یادگیری نظارت‌شده است که با تقسیم بازگشتی داده‌ها بر اساس ویژگی‌ها، یک ساختار درختی قابل تفسیر برای مسائل طبقه‌بندی و رگرسیون ایجاد می‌کند [۲۴].

KNN یک الگوریتم یادگیری مبتنی بر نمونه است که بر اساس نزدیک‌ترین همسایه‌ها در فضای ویژگی، برچسب نمونه جدید را تعیین می‌کند و نیاز به مرحله آموزش صریح ندارد [۲۵].

شبکه‌های عصبی مصنوعی مدل‌هایی الهام‌گرفته از سیستم عصبی انسان هستند که با استفاده از لایه‌های متوالی نورون‌ها قادر به مدل‌سازی روابط غیرخطی پیچیده در داده‌ها می‌باشند[۲۶].

SVM یک الگوریتم یادگیری نظارت‌شده است که با یافتن ابرصفحه‌ای با بیشترین فاصله از داده‌ها، طبقه‌بندی یا رگرسیون را انجام می‌دهد و در فضاهای با ابعاد بالا عملکرد مؤثری دارد [۲۷].

Naive Bayes یک طبقه‌بند احتمالاتی مبتنی بر قضیه بیز است که با فرض استقلال ویژگی‌ها، مدلی ساده و کارآمد برای مسائل طبقه‌بندی، به‌ویژه در داده‌های متنی، ارائه می‌دهد [۲۸].

رگرسیون لجستیک یک مدل آماری خطی است که برای مسائل طبقه‌بندی دودویی استفاده می‌شود و احتمال تعلق نمونه به هر کلاس را با استفاده از تابع لجستیک تخمین می‌زند[۲۹].

برای به‌دست‌آوردن بهترین ترکیب از ویژگی‌ها، از معیارهای انتخاب ویژگی استفاده می‌شود. به‌طور کلی، سه نوع روش انتخاب ویژگی وجود دارد :

-روش‌های فیلتر[۲]: این روش‌ها با بررسی ویژگی‌های آماری، بهترین ویژگی‌ها را انتخاب می‌کنند.

-روش‌های بسته‌بندی[۳]: این روش‌ها با استفاده از آزمون و خطا، زیرمجموعه‌ای از ویژگی‌ها را تولید می‌کنند که مدل‌هایی با بالاترین پیش‌بینی‌ها را ایجاد می‌سازند.

-روش‌های تعبیه شده[۴]: در این روش‌ها، بهترین زیرمجموعه ویژگی به‌عنوان بخشی از فرآیند آموزش الگوریتم یادگیری انتخاب می‌شود.

در این مقاله، از روش‌های فیلتر و بسته‌بندی و تعبیه شده بهره‌برداری شده است که در ادامه به تفصیل توضیح داده خواهند شد.

طبقه‌بند بر پایه‌ی اطلاعات متقابل[۵] یک روش برای انتخاب ویژگی‌ها در مسائل طبقه‌بندی است که میزان وابستگی بین ویژگی‌ها و متغیر هدف را اندازه‌گیری می‌کند. این روش به شناسایی ویژگی‌های کلیدی و کاهش ابعاد داده‌ها کمک می‌کند [۳۰].

رگرسیون برپایه آزمون[۶] تکنیک آماری است که برای ارزیابی ارتباط بین یک متغیر هدف پیوسته و یک یا چند متغیر مستقل (ویژگی‌ها) به‌کار می‌رود. این روش به‌ویژه در زمینه یادگیری ماشین و انتخاب ویژگی‌ها کاربرد دارد و می‌تواند در شناسایی ویژگی‌های مهم تأثیرگذار بر متغیر هدف کمک کند[۳۱].

رگرسیون بر پایه‌ی اطلاعات متقابل[۷] روشی برای ارزیابی وابستگی بین یک متغیر هدف پیوسته و یک یا چند ویژگی (متغیر مستقل) است. این روش به‌طور خاص برای شناسایی و اندازه‌گیری اطلاعاتی که یک ویژگی می‌تواند در مورد متغیر هدف ارائه دهد، طراحی شده است[۳۰].

آزمون F برای طبقه‌بندی یا آزمون آنالیز واریانس[۸] که یک روش فیلتر به شمار می‌آید، برای ارزیابی وابستگی بین یک متغیر هدف دسته‌ای (categorical) و یک یا چند متغیر مستقل (features) استفاده می‌شود. این روش به شناسایی ویژگی‌های تأثیرگذار بر متغیر هدف کمک کرده و مقدار ANOVA F-value را بین هر ویژگی و بردار هدف محاسبه می‌کند[۳۱].

آزمون کای-دو[۹] یک آزمون آماری است که برای بررسی وابستگی بین دو یا چند متغیر دسته‌ای (categorical variables) به کار می‌رود. این آزمون معمولاً در تحلیل داده‌ها برای ارزیابی این که آیا توزیع مشاهدات در گروه‌ها به‌طور تصادفی توزیع شده‌اند یا خیر، استفاده می‌شود [۳۲].

انتخاب بازگشتی ویژگی‌ها[۱۰] تکنیکی برای انتخاب ویژگی است که به‌طور مکرر ویژگی‌ها را از یک مدل یادگیری ماشین حذف می‌کند تا بهترین زیرمجموعه ویژگی‌ها شناسایی شود. RFE معمولاً در ترکیب با مدل‌های یادگیری نظارت‌شده، به‌ویژه رگرسیون و درخت‌های تصمیم، مورد استفاده قرار می‌گیرد [۳۰].

حذف بازگشتی ویژگی‌ها با اعتبارسنجی متقابل[۱۱] یک تکنیک پیشرفته برای انتخاب ویژگی‌ها در یادگیری ماشین است که از RFE به همراه اعتبارسنجی متقابل استفاده می‌کند. [۳۰].

روش استراتژی مبتنی بر جستجو که یکی از روش‌های بسته‌بندی است، در هسته خود یکی از الگوریتم‌های طبقه‌بندی را به کار می‌گیرد. در این مقاله، از الگوریتم جنگل تصادفی[۱۲] به‌عنوان الگوریتم طبقه‌بندی برای این روش استفاده شده است [۳۰].

روش‌های مبتنی بر جستجو در پیاده‌سازی به چهار دسته تقسیم می‌شوند:

-روش جستجوی رو به جلو[۱۳]
-روش جستجوی رو به عقب[۱۴]
-روش جستجوی رو به جلوی شناور[۱۵]
- روش جستجوی رو به عقب شناور[۱۶]

Base Model به مدل پایه‌ای گفته می‌شود که برای ارزیابی اهمیت ویژگی‌ها و انتخاب مجموعه ویژگی بهینه استفاده می‌شود. به عبارت دیگر، این مدل به‌عنوان مرجع برای سنجش تأثیر هر ویژگی روی عملکرد نهایی طبقه‌بند عمل می‌کند که در این مطالعه مدل پایه LightGBM با تنظیم n_estimators=300 انتخاب شده است [۱۹].

۴- معرفی ویژگی ها

آدرس‌های فیشینگ معمولاً با هدف جلب اعتماد کاربران، از کلمات و عبارات مشابه با نام دامنه وب‌سایت‌های معتبر یا فرآیندهای رسمی استفاده می‌کنند. جدول ۱ نمونه‌هایی از آدرس‌های فیشینگ شناسایی‌شده را نشان می‌دهد که در آن‌ها از نام سامانه‌ها، بانک‌ها، سیستم‌های پرداخت الکترونیکی و نهادهای رسمی شناخته‌شده در ساختار دامنه استفاده شده است. این کلمات کلیدی به‌گونه‌ای انتخاب می‌شوند که کاربر را به اشتباه به معتبر بودن وب‌سایت هدایت کنند.

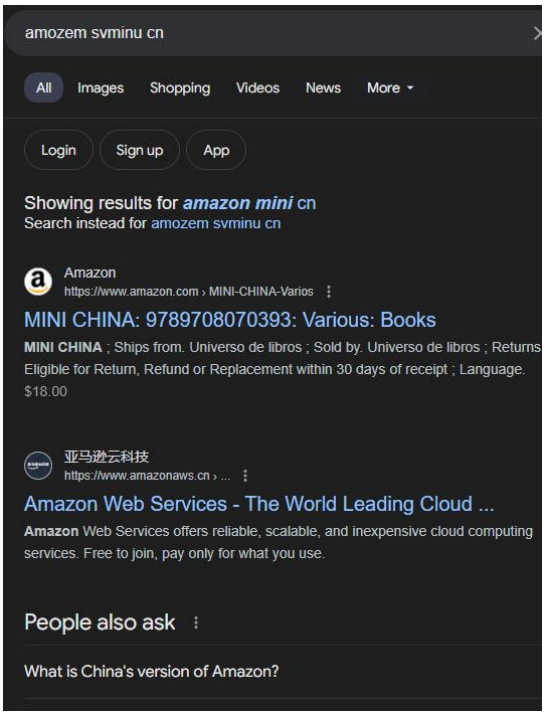
همان‌طور که در جدول ۱ مشاهده می‌شود، با جستجوی دامنه اصلی در موتور جستجوی گوگل، نتایج مرتبط با وب‌سایت هدف و کلمات کلیدی مورد سوءاستفاده نمایش داده می‌شود که نشان‌دهنده تلاش مهاجمان برای تقلید از دامنه‌های قانونی و فریب کاربران است.

جدول ۱: نمونه‌ای از آدرس‌های وب فیشینگ شناسایی‌شده به همراه هدف حمله و کلمات کلیدی به‌کاررفته در دامنه‌ها

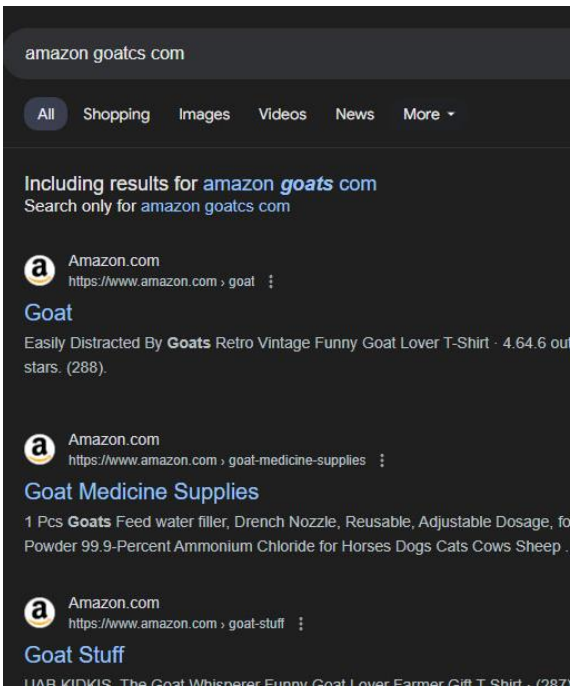
آدرس سایت	کلمات کلیدی	هدف
https://amozem.svminu.cn/	Amozem, svminu	online shopping
http://wwwbancainternet-interbank.pe-inicios.com/	Wwwbancainternet, interbank, pe, inicios	interbank
https://amozem.phvcrx.cn/	Amozem, phvcrx	online shopping
https://amazon.goatcs.com/	Amazon, goatcs	online shopping
https://amozem.qwidiu.cn/	Amozem, qwidiu	online shopping
https://amozem.rmouvl.cn/	Amozem, rmouvl	online shopping
https://citizens-onlinesecure.port25.biz/secure/	citizens, onlinesecure, port25	citizensbank
https://online-sana.ga/z/	sana	The Judicial System
https://eblagh.ir-658146.xyz/	eblagh	The Judicial System
https://shaparrak-tg.ml/	shaparrak	The Judicial System
https://cch-shaprak.cf	shaprak	Electronic payment system
https://eblagh-vx.ml	eblagh	The Judicial System
https://eblagh.ir-61561.xyz	eblagh	The Judicial System
https://eblagh-vx.tk	eblagh	The Judicial System
https://eblagh-vx.gq	eblagh	The Judicial System
https://eblagh-vx.cf	eblagh	The Judicial System
https://shaparak-pw.tk/	shaparak	Electronic payment system
https://mellat-behpardakht.ml	Mellat, behpardakht	Mellat Bank
https://melli.tel/	melli	Melli Bank
https://polic-ir.ga	Polic , ir	police system

شکل‌های ۱ تا ۵ نمونه‌ای از نتایج جستجوی گوگل برای بخش آدرس اصلی دامنه‌های فیشینگ را نشان می‌دهند. از آنجا که دامنه‌های فیشینگ معمولاً پیش از شناسایی رسمی در موتورهای جستجو ایندکس نشده و از اعتبار یا پیوندهای ورودی معتبر برخوردار نیستند، در نتایج جستجو ظاهر نمی‌شوند و به‌جای آن، دامنه‌های معتبر و مرتبط از نظر نوشتاری نمایش داده می‌شوند. این دامنه‌ها اگرچه عیناً برابر با دامنه فیشینگ نیستند، اما از نظر ساختار و ترکیب حروف شباهت قابل‌توجهی با آن دارند که مبنای استفاده از معیارهای محاسبه شباهت دامنه در فرآیند تشخیص فیشینگ در این مطالعه قرار گرفته است. جستجوها به‌گونه‌ای تنظیم شده‌اند که تنها در نام دامنه‌ها انجام شوند و محتوای صفحات وب در فرآیند استخراج دخیل نباشد. دامنه‌های مشابه از چندین صفحه نتایج جستجو جمع‌آوری شده‌اند و ترتیب نمایش نتایج در تحلیل لحاظ نشده است. از آنجا که هدف صرفاً استخراج دامنه‌های مشابه بوده و نه تحلیل رتبه‌بندی نتایج، برای هر دامنه فیشینگ، بیشترین مقدار شباهت محاسبه‌شده میان دامنه‌های استخراج‌شده (با صرف‌نظر از موردی که عیناً برابر با دامنه مورد جستجو باشد) به‌عنوان معیار نهایی در نظر گرفته شده است. این رویکرد با حذف وابستگی به رتبه‌بندی و محدودسازی جستجو به نام دامنه‌ها، تأثیر شخصی‌سازی و وابستگی به موقعیت جغرافیایی کاربر را به حداقل رسانده و تکرارپذیری فرآیند را تضمین می‌کند.

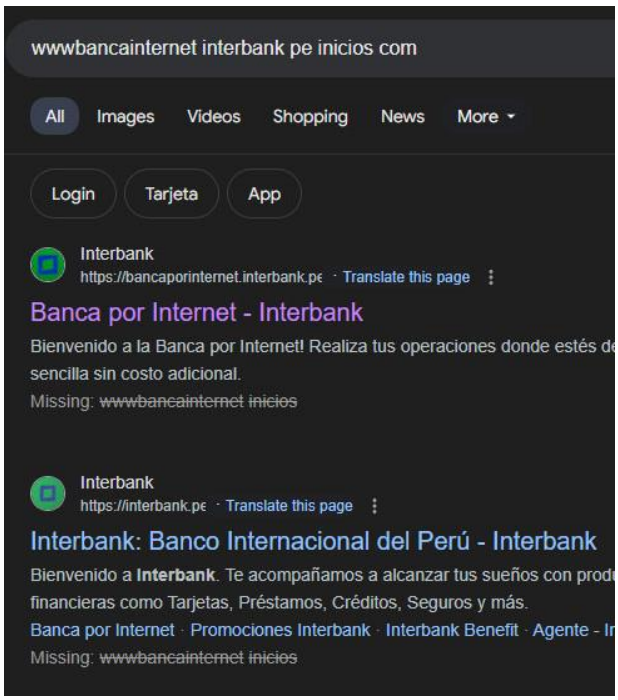
با توجه به وجود معیارهای متنوع و ویژگی‌های خاص برای سنجش شباهت و تفاوت مکانیسم ارزیابی هر یک، هر معیار به عنوان یک ویژگی مستقل در مدل استفاده شده است.



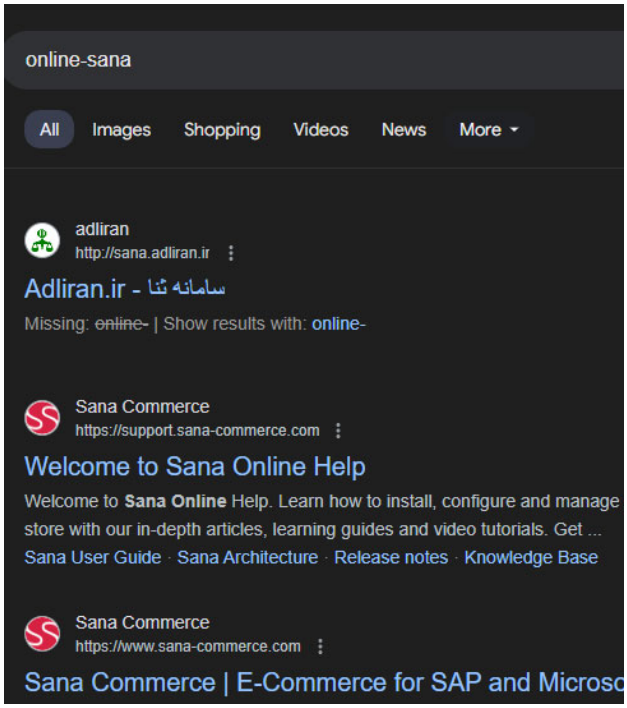
شکل ۱: نمونه‌ای از دامنه فیشینگ amozem.svminu.cn **و دامنه معتبر متناظر** amazon.com **که شباهت نوشتاری در نام دامنه را نشان می‌دهد.**



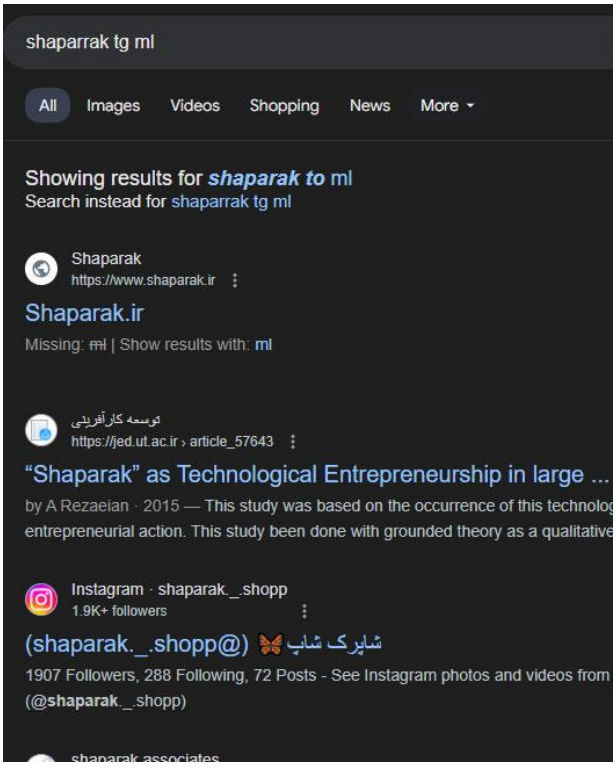
شکل ۴: نمونه‌ای از دامنه فیشینگ amazon.goatcs.com **که با استفاده از نام برند، شباهت ظاهری با دامنه معتبر** amazon.com **ایجاد کرده است.**



شکل ۲: نمونه‌ای از دامنه فیشینگ wwwbancainternet-interbank.pe-inicios.com **در مقایسه با دامنه معتبر** bancaporinternet.interbank.pe **و تلاش برای تقلید ساختار دامنه هدف.**



شکل ۳: نمونه‌ای از دامنه فیشینگ online-sana.ga **و دامنه معتبر** sana.adliran.ir **با استفاده از شباهت اسمی برای گمراه‌سازی کاربر.**



شکل ۵: نمونه‌ای از دامنه فیشینگ shaparrak-tg.ml **و دامنه معتبر** shaparak.ir **که شباهت نوشتاری در نام دامنه را نشان می‌دهد.**

جدول ۲: ویژگی‌های شباهتی مورد استفاده برای تشخیص شباهت بین دامنه‌ها و محتوا

نام ویژگی	توضیحات	نوع داده ای
JaroWinkler	میزان شباهت با روش JaroWinkler	float
Levenshtein	میزان شباهت با روش Levenshtein	float
Damerau	میزان شباهت با روش Damerau	float
LongestCommonSubsequence	میزان شباهت با روش LongestCommonSubsequence	float
MetricLCS	میزان شباهت با روش MetricLCS	float
NGram	میزان شباهت با روش NGram	float
QGram	میزان شباهت با روش QGram	float
Cosine	میزان شباهت با روش Cosine	float
RatcliffObershelp	میزان شباهت با روش RatcliffObershelp	float
similarave	میانگین روش های محاسبه شباهت	float

جدول ۲ این ویژگی‌های شباهتی را به همراه توضیح عملکرد و نوع داده‌ای آن‌ها نشان می‌دهد. این ویژگی‌ها شامل روش‌های مبتنی بر فاصله، الگوریتم‌های مقایسه رشته‌ای و معیارهای مبتنی بر الگو هستند که هر کدام میزان شباهت بین دامنه‌ها یا محتوا را به صورت عددی (float) اندازه‌گیری می‌کنند. ویژگی ترکیبی similarave میانگین نتایج تمام روش‌ها را ارائه می‌دهد و نشان‌دهنده یک شاخص کلی شباهت است

برای به‌دست‌آوردن مقادیر مرتبط با این ویژگی‌ها، قسمت اصلی URL جدا شده و سپس با استفاده از توابع کتابخانه‌ای Regex پالایش شده است. پس از آن، با استفاده از توابع کتابخانه‌ای HttpWebRequest در دات‌نت و متد Get، جستجویی از سایت گوگل انجام شده و آدرس‌های موجود در نتایج که عیناً با آدرس مورد جستجو یکسان نیستند جمع‌آوری شده و میزان شباهت آن‌ها با آدرس سایت با استفاده از معیارهای متعدد شباهت ثبت شده است و بهترین شباهت در هر معیار به‌عنوان نتیجه مرتبط با آن ویژگی اتخاذ شده است به طور مثال برای آدرس shaparrak-tg.ml، یکی از مقادیری که در نتیجه جستجوی گوگل اعلام شده shaparrak.ir است لذا دو متغیر S_1 و S_2 به صورت زیر تعریف شده است.

$$S_1 = shaparrak - tg.ml \\ S_2 = shaparrak.ir \quad (۱)$$

در ادامه محاسبه شابهت این دو رشته با روش های متعدد بیان شده است.

شاخص جارو-وینکلر[۱۷] یک الگوریتم مشابهت رشته‌ای است که برای اندازه‌گیری شباهت دو رشته متنی طراحی شده است. این الگوریتم به‌ویژه در کاربردهای پردازش زبان طبیعی، پایگاه‌داده‌ها و اصلاح املایی استفاده می‌شود[26].

$$Jaro(S_1, S_2) = \frac{1}{3}\left(\frac{m}{|S_1|} + \frac{m}{|S_2|} + \frac{m-t}{m}\right) \quad (۲)$$

که در آن m تعداد کاراکترهای مشترک و t تعداد جابجایی ها (تعداد کاراکترهایی مشترک که در موقعیت‌های مختلف قرار دارند) است و $|S_1|$ طول رشته S_1 و $|S_2|$ طول رشته S_2 می‌باشد، در نهایت:

$$Jaro - Winkler(S_1, S_2) \\ = Jaro(S_1, S_2) \\ + \left(l \cdot p \cdot \left(1 - Jaro(S_1, S_2)\right)\right) \quad (۳)$$

که p =۰,۱ و ۱ برابر با طول پیشوند مشترک(حداکثر ۴ کاراکتر) است. دیگر آدرس های یافت شده در جستجوی گوگل مرتبط با S_1 نیز پس از جایگذاری در S_2، مقدار این معیار برای آن محاسبه شده است سپس بیشترین مقدار بدست آمده در بین تمام آدرس های مرتبط با S_1 به عنوان مقدار ویژگی JaroWinkler لحاظ شده است.

فاصله لیونشتاین[۱۸] معیاری برای اندازه‌گیری تفاوت بین دو رشته است. به‌ویژه، این فاصله حداقل تعداد ویرایش‌های تک‌کاراکتری (اضافه‌کردن، حذف‌کردن یا جایگزینی) مورد نیاز برای تبدیل یک کلمه به کلمه‌ی دیگر را شمارش می‌کند. این مفهوم به‌طور متداول در کاربردهایی مانند بررسی املایی، توالی‌یابی DNA و پردازش زبان طبیعی استفاده می‌شود[۳۳].

محاسبه فاصله لیونشتاین دو رشته S_1 و S_2 به صورت زیر انجام شده است:

اگر رشته S_1 خالی باشد و $|S_2|$ طول رشته S_2 باشد آنگاه

$$Levenshtein(S_1, S_2) = |S_2| \quad (۴)$$

همچنین اگر رشته S_2 خالی باشد

$$Levenshtein(S_1, S_2) = |S_1| \quad (۵)$$

اگر کاراکترهای آخر هر دورشته یکسان باشند

$$Levenshtein(S_1, S_2) = Levenshtein(S_1[0 \dots n-1], S_2[0 \dots m-1]) \quad (۶)$$

که n و m به ترتیب طول کاراکترهای S_1 و S_2 می‌باشد.

در غیر اینصورت

$$Levenshtein(S_1, S_2) = min \begin{Bmatrix} Levenshtein(S_1[0 \dots n-1], S_2) + 1 \\ Levenshtein(S_1, S_2[0 \dots m-1]) + 1 \\ Levenshtein(S_1[0 \dots n-1], S_2[0 \dots m-1]) + 1 \end{Bmatrix} \quad (۷)$$

فاصله لیونشتاین برای همه آدرس های مرتبط یافت شده برای S_1 محاسبه شده است و کمترین مقدار بدست آمده به عنوان مقدار ویژگی Levenshtein در نظر گرفته شده است.

فاصله دامراو[۱۹] یک الگوریتم برای محاسبه فاصله ادیت[۲۰] بین دو رشته متنی است. این الگوریتم، که به نام Damerau-Levenshtein نیز شناخته می‌شود، علاوه بر عملیات تغییر، حذف، افزودن کاراکترو تغییر موقعیت دو کاراکتر متوالی (تعویض) را نیز در نظر می‌گیرد[۳۳]. در واقع این روش حداقل تعداد عملیات‌های لازم برای تبدیل یک رشته به رشته دیگر اشاره دارد، این عملیات برای همه آدرس های مرتبط یافت شده برای S_1 محاسبه شده است و کمترین مقدار بدست آمده را به عنوان مقدار ویژگی Damerau ثبت شده است.

فاصله مشترک طولانی‌ترین زیررشته[۲۱] یک مسئله مهم در علوم کامپیوتر است که به بررسی یافتن طولانی‌ترین زیررشته‌ای می‌پردازد که در دو دنباله مشترک وجود دارد. این مفهوم به‌ویژه در زمینه‌های مختلفی مانند زیست‌فناوری، پردازش زبان طبیعی و مقایسه داده‌ها کاربرد دارد[۳۴].

برای محاسبه این فاصله برای دو رشته S_1 و S_2 به صورت زیر عمل شده است.

اگر یکی از رشته ها خالی باشد

$$LCS(S_1, S_2) = 0 \quad (۸)$$

اگر آخرین کاراکترهای هر دو رشته برابر باشد

$$LCS(S_1, S_2) = 1 + LCS(S_1[0 \dots n-1], S_2[0 \dots m-1]) \quad (۹)$$

در غیر این صورت

$$LCS(S_1, S_2) = max \begin{cases} LCS(S_1[0 \dots n-1], S_2) \\ LCS(S_1, S_2[0 \dots m-1]) \end{cases} \quad (۱۰)$$

که n و m به ترتیب طول کاراکترهای S_1 و S_2 می‌باشد، این معیار برای همه آدرس های مرتبط یافت شده برای S_1 محاسبه شده است و کمترین مقدار بدست آمده به عنوان مقدار ویژگی LongestCommonSubsequence در نظر گرفته شده است.

بزرگ‌ترین دنباله مشترک[۲۲] یک الگوریتم و معیار برای اندازه‌گیری شباهت بین دو رشته متنی است. این الگوریتم به دنبال پیدا کردن طولانی‌ترین زیررشته مشترک بین دو رشته می‌باشد[۳۵].

برای به‌دست‌آوردن شباهت بین دو رشته توسط این معیار از روش زیر استفاده شده است

$$MLCS(S_1, S_2) = \frac{2 \times LCS(S_1 \times S_2)}{|S_1| + |S_2|} \quad (۱۱)$$

پس از محاسبه این معیار برای همه آدرس های مرتبط یافت شده برای S_1، بیشترین مقدار بدست آمده را به عنوان مقدار ویژگی MetricLCS ثبت شده است.

ان-گرام[۲۳] یک تکنیک در پردازش زبان طبیعی و یادگیری ماشین است که به تجزیه و تحلیل توالی‌های متنی کمک می‌کند. این تکنیک به‌خصوص در کاربردهای تحلیل متن، مدل‌سازی زبان، و پیش‌بینی کلمات مورد استفاده قرار می‌گیرد[۳۶].

در این روش ابتدا رشته های S_1 و S_2 به صورت مجموعه های N گرام تبدیل شده است و سپس با محاسبه نسبت تعداد اشتراک مجموعه ها به تعداد اجتماع آن ها، شباهت دو رشته به صورت زیر بدست آمده است

$$G_1 = \{N - grams\ of\ S_1\} \quad N = 2$$
$$G_2 = \{N - grams\ of\ S_2\} \quad N = 2$$

$$N - grams(S_1, S_2) = \frac{|G_1 \cap G_2|}{|G_1 \cup G_2|} \quad (۱۲)$$

ان-گرام برای همه آدرس های مرتبط یافت شده برای S_1 محاسبه شده است و بیشترین مقدار بدست آمده به عنوان مقدار ویژگی NGram در نظر گرفته شده است.

کیوگرام[۲۴] یک تکنیک در پردازش زبان طبیعی و تحلیل متن است که به تجزیه و تحلیل توالی‌های کاراکتری کمک می‌کند. این تکنیک به‌ویژه در کاربردهایی مانند تشخیص شباهت، تصحیح

املایی، و پردازش زبان طبیعی مورد استفاده قرار می‌گیرد[۳۷].

محاسبه کیوگرام مشابه ان-گرام انجام شده است با این تفاوت Q=3 انتخاب شده است و بیشترین مقدار بدست آمده به ازای آدرس های مختلف به عنوان مقدار ویژگی QGram در نظر گرفته شده است.

متد شباهت کسینوسی[۲۵] یکی از روش‌های متداول برای اندازه‌گیری شباهت بین دو بردار در فضای n-بعدی است. این متد به‌ویژه در پردازش زبان طبیعی و تحلیل داده‌ها کاربرد دارد[۳۸].

در این روش ابتدا بردار متناظر با رشته های S_1 و S_2 به ترتیب با نام A و B ایجاد گردید سپس میزان شباهت این دو رشته براساس فرمول زیر محاسبه شده است

$$CS(S_1, S_2) = \frac{\sum_{i=1}^{n} a_i . b_i}{\|A\| . \|B\|} = \frac{\sum_{i=1}^{n} a_i . b_i}{\sum_{i=1}^{n} a_i^2 . \sum_{i=1}^{n} b_i^2} \quad (۱۳)$$

که در آن a_i مقادیر عناصر بردار A وb_i مقادیر عناصر B است.

این مقدار برای همه آدرس های مرتبط یافت شده برای S_1 محاسبه شده است و بیشترین مقدار بدست آمده به عنوان مقدار ویژگی Cosine در نظر گرفته شده است.

رتکلیف-اوبرشلب[۲۶] یک الگوریتم برای تشخیص شباهت بین دو رشته متنی است که به‌ویژه در پردازش زبان طبیعی و تحلیل متن کاربرد دارد. این الگوریتم بر مبنای یافتن زیررشته‌های مشترک و مقایسه آن‌ها برای تعیین میزان شباهت بین دو رشته کار می‌کند[۳۹].

در این روش ابتدا بلندترین زیر رشته مشترک میان S_1 و S_2 مشخص شده است سپس از فرمول زیر مقدار این معیار محاسبه شده است

$$RO(S_1, S_2) = \frac{2 \times |C|}{|S_1| + |S_2|} \quad (۱۴)$$

که در |C| طول زیر رشته مشترک می‌باشد. همه آدرس های مرتبط یافت شده برای S_1 شباهت آن با این فرمول محاسبه شده است و در نهایت بیشترین مقدار را به عنوان مقدار ویژگی RatcliffObershelp در نظر گرفته شده است

در این آزمایش نتایج در ۴۰ صفحه جستجوی گوگل در صورت وجود ارزیابی گردیده است. تمام این کارها با استفاده از برنامه‌ای در C# انجام گردیده و برای به‌دست‌آوردن میزان شباهت‌ها از تابع کتابخانه‌ای F23.StringSimilarity.Experimental در دات‌نت استفاده شده است[۴۰].

در گام بعدی، به منظور تحلیل اعتبار و پایداری دامنه‌ها، مجموعه‌ای از ویژگی‌های ارزشی استخراج شد.

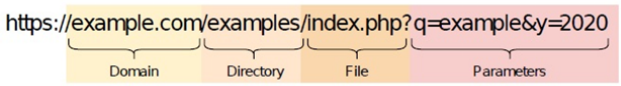
همانطور که در جدول ۳ مشخص است این ویژگی‌ها اطلاعاتی درباره طول و ترکیب دامنه، مدت زمان ثبت دامنه و گواهی SSL ارائه می‌دهند و می‌توانند در تشخیص دامنه‌های فیشینگ نقش مؤثری ایفا کنند. برای مثال، ویژگی‌های dur_domain و dur_ip تعداد روزهای گذشته از ثبت دامنه و آخرین به‌روزرسانی آی‌پی را نشان می‌دهند، در حالی که ویژگی‌های ssl_day و dur_ssl اطلاعات مربوط به گواهی SSL دامنه را ارائه می‌کنند. همچنین ویژگی is_ssl_free مشخص می‌کند که آیا گواهی SSL رایگان بوده است یا خیر. تمامی این ویژگی‌ها به صورت عددی (int) یا بولی (Boolean) ارائه شده‌اند و به تحلیل پایداری و اعتبار دامنه کمک می‌کنند. برای دریافت اطلاعات دامنه، پس از پرس‌وجو از سایت whois.iana.org و یافتن سرور مربوط به دامنه، اطلاعات دامنه براساس پروتکل استاندارد RFC3912 دریافت شده است. به علت اینکه بسیاری از سایت‌های فیشینگ ممکن است در دسترس نبوده یا حذف شده باشند و اطلاعات گواهی SSL آن‌ها از طریق ارتباط با سایت قابل استخراج نباشد، از سایت Censys (https://censys.com)برای دریافت اطلاعات گواهی‌ها استفاده شده است. این سایت تاریخچه گواهی‌های ارائه‌شده برای هر دامنه را نگهداری می‌کند و اطلاعات گواهی هر آدرس با استفاده از توابع کتابخانه‌ای HttpWebRequest در دات‌نت و با متد Get از سایت Censysدریافت شده است.

مجموعه ویژگی دیگری که با عنوان ویژگی‌های وابسته (Affiliation features) از آن استفاده شده، حاصل پژوهش رابن و همکاران بوده است که شامل ۱۱۱ ویژگی است[۴۱].

همان‌طور که در شکل ۶ مشاهده می‌شود، ویژگی‌های وابسته با تکیه بر ساختار آدرس URL استخراج شده و هر یک از اجزای دامنه، دایرکتوری، فایل و پارامترها به‌عنوان منابع مستقل برای جمع‌آوری ویژگی‌ها در نظر گرفته شده‌اند.

جدول ۳: ویژگی‌های ارزشی مرتبط با اعتبار و پایداری دامنه

نام ویژگی	توضیحات	نوع داده ای
tld_size	مجموع کد اسکی حروف tld دامین	int
dur_domain	تعداد روز گذشته از ثبت دامنه	int
dur_ip	تعداد روز گذشته از آخرین به روزرسانی اطلاعات آی پی	int
ssl_day	تعداد روز گذشته از ثبت ssl	int
dur_ssl	مدت زمان گواهی بر اساس روز	int
is_ssl_free	رایگان بودن گواهی	Boolean(0,1)



شکل ۶: اجزای تشکیل‌دهنده یک آدرس URL و تقسیم‌بندی آن به چهار بخش دامنه، دایرکتوری، فایل و پارامترها.

اگرچه ویژگی‌هایی وابسته مانند تعداد نقطه‌ها یا خط تیره‌ها قبلاً در مقالات [۴۲, ۴۳, ۴۴] معرفی شده‌اند، در این پژوهش آن‌ها در ترکیب با معیارهای شباهتی مورد استفاده قرار گرفتند و ترکیب آن‌ها با موتور جستجوی گوگل و تحلیل دامنه‌ها نوآوری اصلی این پژوهش محسوب می‌شود. در واقع، مقاله حاضر نشان می‌دهد که ادغام ویژگی‌های ساده ساختاری URL (مانند تعداد خط تیره یا نقطه‌ها) با معیارهای شباهتی می‌تواند قدرت تمایز بیشتری نسبت به استفاده منفرد از این معیارها داشته باشداین ترکیب باعث می‌شود که ویژگی‌های ساده‌ی ساختاری URL بتوانند به‌طور مؤثرتری تفاوت بین دامنه‌های قانونی و فیشینگ را آشکار سازند. به‌طور خاص، مشاهده شد که URLهای فیشینگ معمولاً از خط تیره و نقطه برای تقلید از دامنه‌های معتبر استفاده می‌کنند، در حالی که دامنه‌های قانونی الگوی ساده‌تری دارند.

۵- روش پیشنهادی

همان‌طور که در شکل ۷ و شبه‌کد ۱ (Appendix 1) نشان داده شده است، چارچوب ACS به‌گونه‌ای طراحی شده که از نظر محاسباتی سبک و کارا باشد. برخلاف روش‌های مبتنی بر محتوا مانند CANTINA، که نیازمند پردازش HTML و اجرای کدهای اسکریپت سمت کاربر هستند، ACS تنها اطلاعات دامنه را تحلیل می‌کند.

فرآیند انجام پژوهش به‌صورت گام‌به‌گام به شرح زیر است:

- **گام اول: جمع‌آوری داده‌ها**

در این مرحله، مجموعه‌ای از URLها برای ساخت دیتاست گردآوری شد. برای وب‌سایت‌های فیشینگ، ۱۴۳ آدرس معرفی‌شده توسط مرکز ماهر ایران و همچنین ۱۳۰۰۷ URL گزارش‌شده در پایگاه معتبر PhishTank که به‌عنوان فیشینگ ثبت شده بودند، جمع‌آوری و برچسب ۱ به آن‌ها اختصاص داده شد.برای آدرس‌های غیر فیشینگ، از وب‌سایت‌های معتبر ایرانی، مجموعه URLهای گردآوری‌شده جهت کشف سانسور وب توسط لپ و همکاران و همچنین آدرس‌های به‌دست‌آمده از جستجوی دامنه‌محور (site:*.TLD) در موتور جستجوی گوگل استفاده شد. این جستجو صرفاً محدود به دامنه وب‌سایت‌ها بوده و ترتیب نتایج در نظر گرفته نشده است. پس از حذف موارد تکراری و دامنه‌های نامعتبر یا غیرفعال، به این آدرس‌ها برچسب ۰ (غیرفیشینگ) اختصاص داده شد. در مجموع، تعداد URLهای غیر فیشینگ برابر با ۲۰۱۵۵ آدرس است.

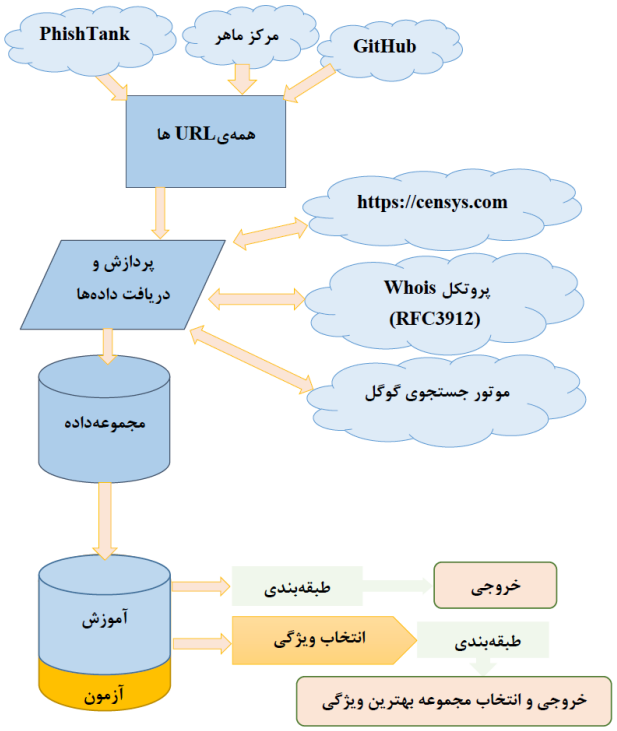
- **گام دوم: استخراج و پردازش ویژگی‌ها**

پس از جمع‌آوری URL ها، مقادیر ویژگی‌های مربوط به هر آدرس وب‌سایت استخراج و پردازش شده و دیتاست نهایی شامل داده‌ها ایجاد گردید.

- **گام سوم: طبقه‌بندی داده‌ها**

در این مرحله، فرآیند طبقه‌بندی بر روی داده‌ها انجام شد. بدین منظور، از ۱۲ الگوریتم طبقه‌بندی پرکاربرد استفاده گردید که به دو دسته تقسیم می‌شوند:

الگوریتم‌های مبتنی بر ویژگی شامل XGBoost، LightGBM، CatBoost، Random Forest، AdaBoost، Decision Tree، K-Nearest Neighbors، Support Vector Machine، Naive Bayes و Logistic Regression، و الگوریتم‌های مبتنی بر یادگیری عمیق شامل Neural Network و TabNet.



شکل ۷: شمای کلی گردآوری اطلاعات و یافتن بهترین ویژگی

تمامی الگوریتم‌ها با مجموعه‌ای از پارامترهای مختلف ارزیابی شده‌اند که جزئیات آن‌ها در پیوست ۱ ارائه شده است و در این مقاله تنها نتایج مربوط به بهترین پیکربندی هر الگوریتم گزارش شده است.

- **گام چهارم: ارزیابی عملکرد مدل‌ها**

برای ارزیابی عملکرد طبقه‌بندها از معیارهای Precision، Recall، F1-score و Accuracy استفاده گردید که به‌ترتیب مطابق روابط (۱۵) تا (۱۸) محاسبه شده‌اند.

$$precision = \frac{TP}{TP + FP} \quad (۱۵)$$

$$recall = \frac{TP}{TP + FN} \quad (۱۶)$$

$$F1_{score} = \frac{2 \times precision \times recall}{precision + recall} \quad (۱۷)$$

$$\text{accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (۱۸)$$

- **گام پنجم: انتخاب ویژگی و بازآموزی مدل‌ها**

پس از بررسی نتایج طبقه‌بندی بر اساس تمامی ویژگی‌ها، به‌منظور ارزیابی کیفیت و میزان شباهت ویژگی‌ها، از الگوریتم‌های انتخاب ویژگی از نوع فیلتر و بسته‌بندی استفاده شد. پس از شناسایی ویژگی‌های برتر، فرآیند طبقه‌بندی مجدداً با استفاده از این ویژگی‌ها انجام شده و نتایج آن ثبت گردید.

۶- آزمون و نتایج

برای بررسی نتایج، از ۱۳۱۵۰ داده مربوط به سایت‌های فیشینگ و ۲۰۱۵۵ سایت غیر فیشینگ استفاده شد. در مرحله اول، دسته‌بندی با استفاده از ۱۲۷ ویژگی و ۱۲ الگوریتم طبقه‌بندی انجام گرفت. به‌منظور کاهش اثر عدم‌توازن داده‌ها و اطمینان از تعمیم‌پذیری نتایج، از اعتبارسنجی متقابل k-fold با k=5 استفاده شد. در هر مرحله از اعتبارسنجی، دیتاست به پنج بخش تقریباً هم‌تراز تقسیم شد و چهار بخش برای آموزش و یک بخش برای آزمون استفاده گردید؛ این فرآیند پنج بار تکرار شد تا هر بخش یک‌بار به‌عنوان مجموعه آزمون عمل کند و کل داده‌ها در اعتبارسنجی مشارکت داده شوند.

برای هر الگوریتم، چندین پیکربندی مختلف از پارامترهای کلیدی مورد آزمایش قرار گرفت. بهترین پیکربندی هر الگوریتم بر اساس میانگین F1-score حاصل از اعتبارسنجی متقاطع انتخاب شد. جزئیات کامل پیکربندی‌های آزمایش‌شده، مقادیر پارامترها و محدوده‌های انتخاب آن‌ها در Appendix 3 ارائه شده است.

نتایج ارائه‌شده در جدول ۴ نشان می‌دهد که عملکرد الگوریتم‌های مختلف یادگیری ماشین تفاوت قابل‌توجهی با یکدیگر دارد. در میان تمامی مدل‌ها، الگوریتم LightGBM بهترین عملکرد را از خود نشان داده و با دستیابی به دقت کلی (Accuracy) برابر با ۰,۹۹۲۶، دقت پیش‌بینی (Precision) برابر با ۰,۹۹۳۳، بازخوانی (Recall) برابر با ۰,۹۸۸۰ و مقدار F1-score برابر با ۰,۹۹۰۶، به‌عنوان مدل برتر شناسایی شده است. پس از آن، مدل‌های XGBoost و CatBoost نیز عملکرد بسیار نزدیکی به مدل برتر داشته و مقادیر بالایی را در تمامی معیارها ثبت کرده‌اند که بیانگر توانایی بالای آن‌ها در یادگیری الگوهای پیچیده داده‌ها است.

مدل Random Forest با وجود عملکرد قابل قبول، در مقایسه با روش‌های مبتنی بر تقویت گرادیانی از دقت کمتری برخوردار بوده و مقدار F1-score آن برابر با ۰,۹۶۹۸ گزارش شده است. الگوریتم‌های AdaBoost و Decision Tree نیز نتایج متوسطی ارائه داده‌اند که نشان‌دهنده حساسیت بیشتر این مدل‌ها به ساختار داده و محدودیت در تعمیم‌پذیری است.

در میان مدل‌های مبتنی بر همسایگی و شبکه‌های عصبی، الگوریتم KNN و شبکه عصبی چندلایه عملکرد متوسطی داشته‌اند، به‌طوری‌که مقادیر F1-score آن‌ها به‌ترتیب برابر با ۰,۸۹۶۸ و ۰,۹۰۴۸ بوده است. در مقابل، الگوریتم SVM با وجود دقت بالای Precision (۰,۹۷۳۸)، از بازخوانی نسبتاً پایینی برخوردار بوده که منجر به کاهش قابل‌توجه F1-score شده است. همچنین، مدل Naive Bayes عدم تعادل شدیدی بین Precision و Recall از خود نشان داده و با وجود Recall بالا، به دلیل Precision پایین، عملکرد کلی ضعیفی داشته است.

جدول ۴: مقایسه عملکرد الگوریتم‌های مختلف یادگیری ماشین بر اساس معیارهای Accuracy، Precision، Recall و F1-score. نتایج نشان می‌دهد که مدل‌های مبتنی بر تقویت گرادیانی، به‌ویژه LightGBM و XGBoost، بالاترین عملکرد کلی را در تمامی معیارها ارائه داده‌اند.

Method	Accuracy	Precision	Recall	F1-score
LightGBM	**0.9926**	**0.9933**	0.9880	**0.9906**
CatBoost	0.9900	0.9896	0.9850	0.9873
XGBoost	0.9915	0.9904	**0.9881**	0.9892
TabNet	0.9283	0.8932	0.9350	0.9122
RandomForest	0.9797	0.9745	0.9651	0.9698
AdaBoost	0.9629	0.9581	0.9476	0.9528
DecisionTree	0.9587	0.9448	0.9509	0.9478
KNN	0.9218	0.9357	0.8611	0.8968
Neural_Network	0.9454	0.9104	0.9016	0.9048
SVM	0.8353	0.9738	0.5990	0.7417
NaiveBayes	0.4380	0.4100	0.9640	0.5753
LogisticRegression	0.8794	0.9011	0.7804	0.8362

در میان مدل‌های مبتنی بر یادگیری عمیق، الگوریتم TabNet عملکردی متوسط از خود نشان داده و با دستیابی به دقت کلی ۰,۹۲۸۳ و مقدار F1-score برابر با ۰,۹۱۲۲، نسبت به شبکه عصبی چندلایه بهبود نسبی داشته است، اما همچنان فاصله قابل‌توجهی با مدل‌های مبتنی بر تقویت گرادیانی دارد. این نتیجه نشان می‌دهد که با وجود قابلیت TabNet در انتخاب ترتیبی ویژگی‌ها و افزایش تفسیرپذیری، عملکرد آن به‌شدت به تنظیم پارامترها و حجم داده وابسته بوده و در این مسئله خاص نتوانسته است برتری معناداری نسبت به روش‌های کلاسیک مبتنی بر ویژگی نشان دهد.

به‌طور کلی، نتایج این جدول نشان می‌دهد که مدل‌های مبتنی بر تقویت گرادیانی، به‌ویژه LightGBM، علاوه بر دستیابی به دقت بالا، تعادل مناسبی بین Precision و Recall برقرار کرده‌اند و از این رو گزینه‌ای مناسب و قابل اعتماد برای مسئله مورد مطالعه محسوب می‌شوند.

نتایج جدول ۵ نشان می‌دهد که استفاده از مجموعه ویژگی‌های پیشنهادی این مطالعه شامل Affiliation، Cost و Similarity در مقایسه با استفاده از ویژگی پایه Affiliation، منجر به بهبود عملکرد اکثر الگوریتم‌های یادگیری ماشین بر اساس معیار F1-score شده است. این بهبود به‌ویژه در مدل‌های مبتنی بر تقویت گرادیانی مانند LightGBM، XGBoost و CatBoost مشهود بوده که نشان‌دهنده توانایی بالای این الگوریتم‌ها در بهره‌برداری از اطلاعات تکمیلی و مدل‌سازی روابط غیرخطی میان ویژگی‌ها است.

در مقابل، برخی الگوریتم‌های کلاسیک مانند SVM و Naive Bayes به دلیل محدودیت‌های ذاتی و عدم انطباق فرض‌های مدل با ساختار داده‌ها، بهبود معناداری را تجربه نکرده‌اند که این موضوع قابل انتظار است. به‌طور کلی، نتایج این جدول مؤید آن است که ویژگی‌های پیشنهادی این مطالعه نقش مؤثری در افزایش تعادل بین Precision و Recall داشته و در ترکیب با الگوریتم‌های پیشرفته‌تر، به بهبود قابل توجه عملکرد مدل‌ها منجر شده‌اند.

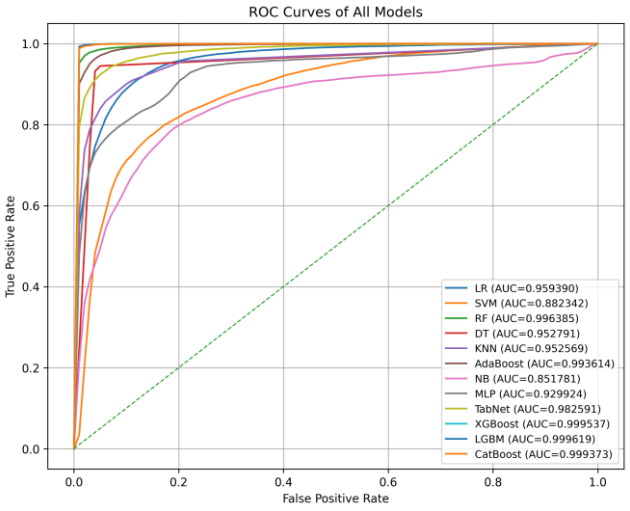
جدول ۵: مقایسه عملکرد الگوریتم‌های یادگیری ماشین بر اساس معیار F1-score با استفاده از ویژگی‌های پایه و مجموعه ویژگی‌های پیشنهادی این مطالعه.

Method	F1-score (Based on Baseline Features)	F1-score (Based on Proposed Study Features)
LightGBM	0.9749	**0.9906**
CatBoost	0.9734	**0.9873**
XGBoost	0.9747	**0.9892**
TabNet	0.8734	**0.9122**
RandomForest	0.9563	**0.9698**
AdaBoost	0.8766	**0.9528**
DecisionTree	0.9391	**0.9478**
KNN	**0.9026**	0.8968
Neural_Network	0.8896	**0.9048**
SVM	**0.8782**	0.7417
NaiveBayes	**0.6457**	0.5753
LogisticRegression	0.8284	**0.8362**

منحنی‌های ROC مربوط به تمامی مدل‌های مورد بررسی در شکل ۸ ارائه شده‌اند. این منحنی‌ها توانایی مدل‌ها را در تمایز بین کلاس‌ها در سطوح مختلف نرخ خطای مثبت کاذب (False Positive Rate) نشان می‌دهند. بر اساس نتایج، مدل LightGBM بالاترین مقدار AUC برابر با ۰,۹۹۹۶ را به دست آورده است که بیانگر قدرت تفکیک تقریباً کامل این مدل است. پس از آن، مدل‌های XGBoost و CatBoost نیز با مقادیر AUC به‌ترتیب ۰,۹۹۹۵ و ۰,۹۹۹۳ عملکردی بسیار نزدیک به حالت ایده‌آل از خود نشان داده‌اند.

در میان سایر مدل‌ها، Random Forest و AdaBoost نیز با مقادیر AUC بالاتر از ۰,۹۹ عملکرد بسیار مطلوبی داشته‌اند که نشان‌دهنده توانایی بالای این روش‌ها در تشخیص صحیح نمونه‌ها است. در مقابل، مدل‌های SVM و Naive Bayes کمترین مقادیر AUC را به‌ترتیب برابر با ۰,۸۸۲ و ۰,۸۵۲ ثبت کرده‌اند که حاکی از محدودیت این مدل‌ها در تمایز دقیق بین کلاس‌ها در این مسئله خاص است.

به‌طور کلی، نتایج حاصل از منحنی‌های ROC نشان می‌دهد که مدل‌های مبتنی بر درخت و تقویت گرادیانی، به‌ویژه LightGBM و XGBoost، از نظر قابلیت تفکیک کلاس‌ها عملکردی برتر نسبت به سایر روش‌ها دارند. این نتایج کاملاً با مقادیر بالای Accuracy و



شکل ۸: منحنی‌های ROC مربوط به تمامی الگوریتم‌های یادگیری ماشین مورد بررسی.

F1-score گزارش‌شده در بخش‌های پیشین هم‌راستا بوده و انتخاب LightGBM به‌عنوان مدل نهایی را به‌خوبی توجیه می‌کند.

در شکل ۹ منحنی Precision–Recall برای مدل LGBM نشان داد که این مدل با مقدار Average Precision برابر با ۰,۹۹۹، عملکردی بسیار پایدار در حفظ دقت بالا حتی در مقادیر بالای بازخوانی دارد. این نتیجه تأیید می‌کند که عملکرد بسیار خوب مدل صرفاً ناشی از توزیع داده‌ها نبوده و LGBM قادر است نمونه‌های مثبت را به‌طور قابل اطمینانی شناسایی کند، به‌ویژه در شرایطی که کنترل خطای مثبت کاذب اهمیت بالایی دارد.

برای کاهش اثر عدم‌توازن داده‌ها، از اعتبارسنجی متقابل به‌صورت k-fold با k=5 انجام شد تا از تعمیم‌پذیری نتایج اطمینان حاصل شود. برای هر آزمایش، از ۱۰۰ بار تکرار اعتبارسنجی متقابل جهت کاهش اثر تصادفی بودن نمونه‌ها استفاده شد.

نمودار جعبه‌ای مقادیر AUC مدل LGBM در شکل ۱۰ بیانگر پایداری بسیار بالای عملکرد مدل در تقسیم‌بندی‌های مختلف داده است. محدود بودن بازه بین چارکی در محدوده ۰,۹۹۹۶۳ تا ۰,۹۹۹۷۰ نشان می‌دهد که مدل نسبت به تغییرات تصادفی در داده‌های آموزشی حساس نبوده و از قابلیت تعمیم بالایی برخوردار است. این ویژگی بیانگر عدم بیش‌برازش مدل به بخش خاصی از داده‌ها و اطمینان‌پذیری نتایج به‌دست‌آمده است.

این نتایج نشان می‌دهد که ACS در کنار LGBM می‌تواند بهترین کارایی را برای شناسایی حملات فیشینگ ارائه دهد.

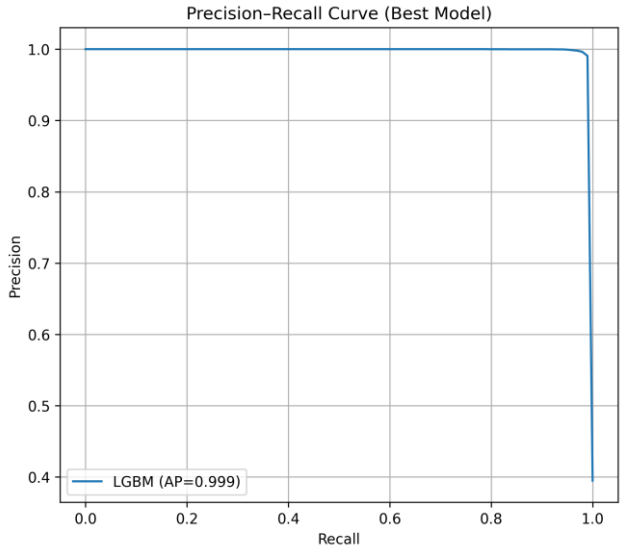
در مرحله بعد، با استفاده از معیارهای انتخاب ویژگی، ویژگی‌های به‌دست‌آمده تحلیل گردیده است. اولین روش Mutual Information بود که بهترین نتایج آن در شکل ۴ نشان داده شده است.

نتایج شکل ۱۱ نشان می‌دهد که ویژگی‌های شباهتی (مانند Jaro-Winkler و Cosine) در میان بهترین ویژگی‌ها قرار دارند. این موضوع تأیید می‌کند که شباهت دامنه‌های فیشینگ به دامنه‌های قانونی نقش کلیدی در فریب کاربران دارد. در واقع، ویژگی‌های شباهتی معرفی‌شده توانسته‌اند حتی از بسیاری از ویژگی‌های سنتی URL (مانند طول دامنه یا تعداد نقطه‌ها) عملکرد بهتری داشته باشند.

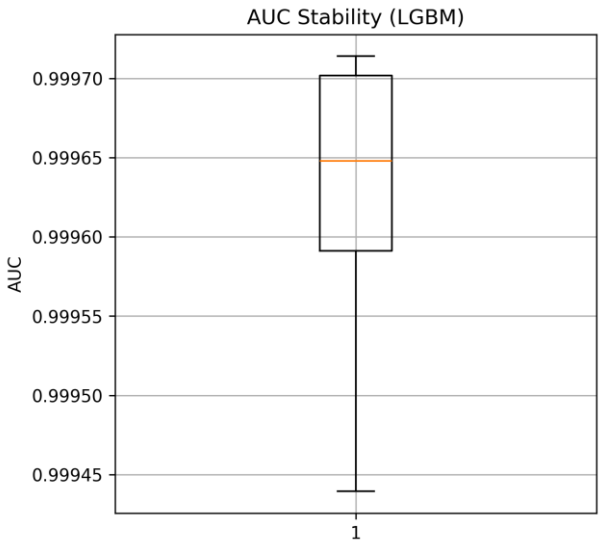
در گام بعد، با استفاده از روش f_classif که یک روش فیلتر است و مقدار ANOVA F-value را بین هر ویژگی و بردار هدف محاسبه می‌کند، ویژگی‌ها آنالیز شده و بهترین ویژگی‌ها به صورت زیر به‌دست آمده است.

همان‌طور که در شکل ۱۲ مشخص است، ویژگی Levenshtein توانسته دومین رتبه بین ویژگی‌ها را توسط معیار f_classif کسب نماید. به‌طور کلی از ۷ بهترین ویژگی، تعداد ۵ ویژگی جدیدی بوده که در این مقاله معرفی شده است.

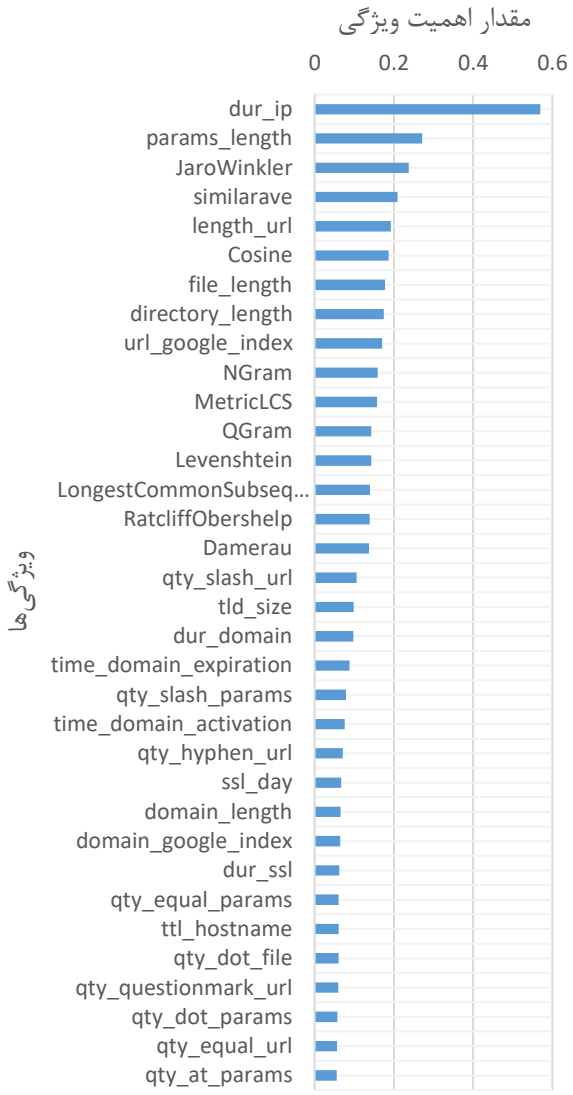
از روش رگرسیون ریج نیز برای انتخاب ویژگی استفاده گردیده است که نتایج آن در شکل ۱۳ ارائه شده است.



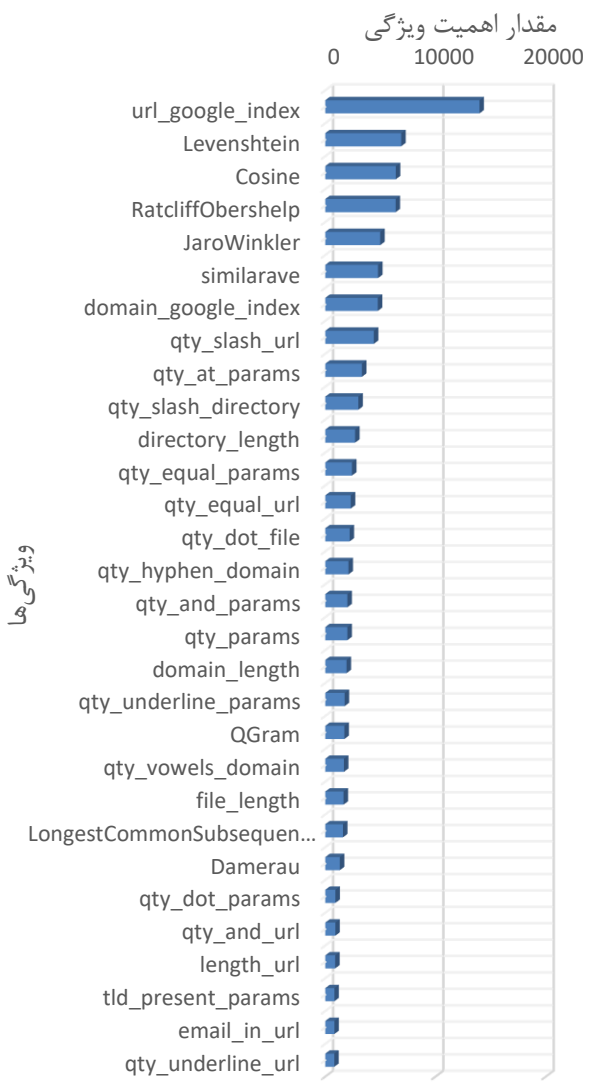
شکل ۹: منحنی Precision–Recall مربوط به مدل LGBM.



شکل ۱۰: نمودار جعبه‌ای مقادیر AUC مدل LGBM در اعتبارسنجی متقاطع پنج‌بخشی.



شکل ۱۱: مقایسه اهمیت ویژگی‌ها بر اساس شاخص اطلاعات متقابل (Mutual Information) برای تعیین تأثیر هر ویژگی در مدل



شکل ۱۲: مقایسه اهمیت ویژگی‌ها به دست آمده از f_classif برای تعیین تأثیر هر ویژگی در مدل

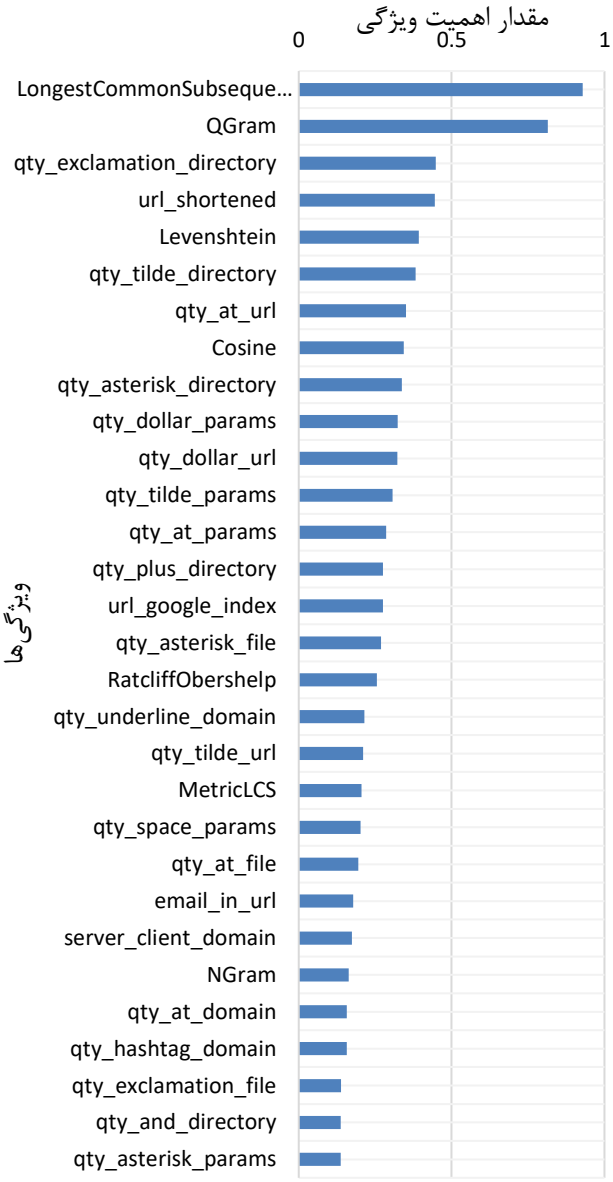
با توجه به شکل ۱۳ نتایج رگرسیون ریج نشان می‌دهند که ویژگی‌های شباهتی بیشترین تأثیر را در مدل دارند، به‌طوری که LongestCommonSubsequence و QGram در رتبه‌های اول و دوم قرار گرفته‌اند و ویژگی‌های Levenshtein و RatcliffObershelp نیز از مهم‌ترین‌ها محسوب می‌شوند.

در گام بعد، الگوریتم‌های انتخاب ویژگی را در دو حالت برای یافتن بهترین مجموعه ویژگی‌ها به کار برده شده است.

همانطور که در شبه کد ۲ (Appendix 2) نشان داده شده است، برای تحلیل ویژگی‌ها با استفاده از الگوریتم‌های انتخاب ویژگی، دو حالت در نظر گرفته شده است. در روش فیلتر (گام ۱)، ابتدا ویژگی‌ها را بر اساس اهمیت مرتب کرده و آن‌ها در مجموعه AllFeatureقرار گرفته است. در هر مرحله از تکرار، یک ویژگی با بیشترین تأثیر از مجموعه AllFeature انتخاب شده و به مجموعه SelectedFeatures منتقل گردید. در ادامه، مدل طبقه‌بندی با استفاده از ویژگی‌های منتخب آموزش داده شد. نتایج هر تکرار ذخیره گردیده و در انتها، مجموعه متناظر با بهترین نتیجه به عنوان مجموعه ویژگی انتخاب‌شده مشخص گردیده است.

از آنجا که در این روش ممکن است برخی ویژگی‌های با اهمیت کم که ممکن است تأثیر به‌سزایی در طبقه‌بندی به همراه دیگر ویژگی‌ها داشته باشند در نظر گرفته نشوند، از الگوریتم‌های انتخاب ویژگی بسته‌بندی نیز برای انتخاب مجموعه ویژگی

استفاده شده است. بدین صورت که مطابق گام ۲، در هر تکرار، ویژگی‌ها بر اساس تأثیر آن‌ها در طبقه‌بندی استخراج شده اند. در هر دو گام از الگوریتم طبقه‌بندی LightGBM برای طبقه بندی استفاده شده است.



شکل ۱۳: اهمیت ویژگی‌ها بر اساس ضرایب مدل رگرسیون ریج

به‌منظور ارزیابی تأثیر روش‌های مختلف انتخاب ویژگی بر عملکرد مدل شبه کد ۲ با روش های متعدد فیلتر و راپر اجرا شد و نتایج روش هایی که با تعداد کمتر از ۱۰۰ ویژگی توانستند بهترین دقت را ایجاد نمایند در جدول ۶ ارائه شده است.

نتایج ارائه‌شده در جدول ۶ نشان می‌دهند که انتخاب مناسب ویژگی‌ها به‌طور قابل‌توجهی دقت و پایداری مدل را افزایش می‌دهد. در میان تمامی روش‌های انتخاب ویژگی، مجموعه ۵۴ ویژگی استخراج‌شده در Appendix 7 که با استفاده از Base Model (LightGBM با n_estimators=300) انتخاب شده است، بهترین عملکرد را نشان داده و با این تعداد ویژگی، به مقادیر Accuracy و F1-score برابر با ۰,۹۹۴۷ و ۰,۹۹۴۵ دست یافته است. این مجموعه ویژگی به‌عنوان ACS Feature Set به کار رفته است.

علاوه بر این، کاهش تعداد ویژگی‌ها از ۱۲۷ به ۵۴، ضمن حفظ دقت، باعث کاهش زمان آموزش مدل LightGBM از ۱,۹۲۲۷ ثانیه به ۱,۷۷۹۶ ثانیه شد، در حالی که زمان پیش‌بینی تقریباً ثابت باقی ماند (۰,۰۰۵۴ ثانیه در مقابل ۰,۰۰۵۳ ثانیه). لازم به ذکر است که این اندازه‌گیری‌ها روی سروری با پردازنده 2x Intel Xeon Gold 6430 با ۱۲۸ نخ و ۴۶۰ گیگابایت رم انجام شده است؛ بنابراین در سیستم‌های با قدرت پردازشی کمتر، ممکن است اختلاف زمان بیشتری مشاهده شود. این کاهش زمان آموزش، ضمن کاهش سربار محاسباتی، می‌تواند به کاهش خطر بیش‌برازش (overfitting) مدل نیز کمک کند.

۷- مقایسه عددی با مطالعات پیشین

با توجه به ماهیت پویا و عمر کوتاه وب‌سایت‌های فیشینگ که معمولاً بلافاصله پس از شناسایی از دسترس خارج می‌شوند، دستیابی به داده‌های کامل و تاریخی این حملات از طریق دیتاست‌های قدیمی اغلب امکان‌پذیر نیست. افزون بر این، بسیاری از مجموعه‌داده‌های موجود تنها شامل URLها یا زیرمجموعه‌ای از ویژگی‌ها هستند و اطلاعات لازم برای استخراج همه‌ی انواع ویژگی‌های مورد نظر این پژوهش را ندارند. برای حل این چالش و

جدول ۶: مقایسه عملکرد روش‌های مختلف انتخاب ویژگی بر اساس معیارهای Accuracy، Precision، Recall و F1-score.

Model	Feature	Number of feature	Accuracy	Precision	Recall	f1-score
Chi2	Appendix 4	57	0.9908	0.9911	0.9856	0.9883
Chi2	Appendix 5	80	0.9932	0.9937	0.9889	0.9913
SBS	Appendix 6	66	0.9875	0.9864	0.9818	0.9841
RFE	Appendix 7	67	0.9927	0.9929	0.9885	0.9907
regression info Mautual	Appendix 8	56	0.9931	0.9939	0.9885	0.9912
base_model	**Appendix 9**	**54**	**0.9947**	**0.9947**	**0.9942**	**0.9945**

ارائه‌ی ارزیابی‌ای منصفانه و شفاف، مقایسه در دو سطح مجزا انجام شده است: جدول ۷ نتایج گزارش‌شده در مقالات پیشین را بدون وابستگی به دیتاست مشترک گردآوری می‌کند، در حالی که جدول ۸ به‌طور خاص روش‌هایی را مقایسه می‌کند که تمامی ویژگی‌های آنها در دیتاست فعلی موجود بوده و امکان اجرا و سنجش عملکرد آنها بر پایه‌ی یک مجموعه داده‌ی واحد فراهم شده است.

جدول ۷: عملکرد چارچوب پیشنهادی ACS را از نظر معیارهای Precision، Recall و F1-score با نتایج گزارش‌شده در مطالعات پیشین

Model	Precision	Recall	f1-score
ACS	**0.9947**	**0.9942**	**0.9945**
LogoSENE [10]	0.9350	0.7794	0.8502
Model CASE [4]	0.9892	0.9436	0.9659
Covid19 [8]	0.9536	0.8944	0.9231
Multi-Scale [11]	0.86	0.84	0.85
V-Similarity [7]	0.91	0.88	0.8008
CPS [9]	0.9715	0.9687	0.9691
LGBM-DNF [5]	0.9478	0.9288	0.9382
LPTVS-Protocol [12]	1	0.87	0.93
In-Depth [13]	0.9723	0.9842	0.9782

بر اساس نتایج جدول ۷، چارچوب پیشنهادی ACS با دستیابی به بالاترین مقدار F1-score (۰,۹۹۴۵) و همچنین تعادل بسیار مناسب میان Precision و Recall، عملکرد برتری نسبت به تمامی روش‌های مقایسه‌شده نشان می‌دهد. این تعادل بیانگر توانایی ACS در شناسایی دقیق وب‌سایت‌های فیشینگ، بدون افزایش نرخ خطای مثبت کاذب یا منفی کاذب است.

در میان روش‌های مقایسه‌شده، مدل‌هایی نظیر In-Depth [۱۳] و CPS [۹] عملکرد قابل قبولی ارائه داده‌اند، اما همچنان از نظر F1-score و یا توازن بین Precision و Recall نسبت به ACS ضعیف‌تر هستند. برای مثال، اگرچه مدل In-Depth، Recall بالایی گزارش کرده است، اما نیاز آن به مجموعه ویژگی‌های گسترده و پیچیده، هزینه محاسباتی بیشتری را تحمیل می‌کند.

روش‌هایی مانند LogoSENE [۱۰] و V-Similarity [۷] که مبتنی بر شباهت بصری و ساختاری هستند، Precision نسبتاً بالایی دارند اما Recall پایین‌تر آن‌ها نشان می‌دهد که در شناسایی کامل حملات فیشینگ، به‌ویژه در سناریوهای بلادرنگ، محدودیت دارند. همچنین، روش‌هایی نظیر Multi-Scale [۱۱] و Covid19 [۸] که عمدتاً بر ویژگی‌های آماری دامنه تکیه دارند، اگرچه سریع هستند، اما در مقایسه با ACS دقت کلی پایین‌تری ارائه می‌دهند.

به‌طور کلی، نتایج جدول ۷ نشان می‌دهد که ترکیب هم‌زمان ویژگی‌های Affiliation، Cost و Similarity در ACS منجر به بهبود قابل توجه عملکرد تشخیص شده و این چارچوب را به گزینه‌ای مناسب برای کاربردهای عملی و بلادرنگ تبدیل می‌کند.

جدول ۸: مقایسه‌ای میان چارچوب پیشنهادی ACS و In-Depth از نظر عملکرد تشخیص و تعداد ویژگی‌های مورد استفاده روی دیتاست مشترک و الگوریتم طبقه‌بندی LightGBM

Model	Number Of features	Precision	Recall	f1-score
ACS	**54**	**0.9947**	**0.9942**	**0.9945**
In-Depth [13]	98	0.9859	0.9779	0.9819

بر اساس نتایج جدول ۸، چارچوب ACS با استفاده از ۵۴ ویژگی موفق شده است عملکردی بالاتر از In-Depth [۱۳] که از ۹۸ ویژگی استفاده می‌کند، ارائه دهد. با وجود تعداد کمتر ویژگی‌ها، ACS نه‌تنها افت عملکردی نداشته، بلکه از نظر Precision، Recall و F1-score نیز بهبود محسوسی نشان داده است.

این نتیجه نشان می‌دهد که کیفیت و انتخاب هدفمند ویژگی‌ها در ACS نقش مهم‌تری نسبت به صرف افزایش تعداد ویژگی‌ها دارد. ویژگی‌های مبتنی بر Affiliation به کشف روابط پنهان میان دامنه‌ها کمک می‌کنند، ویژگی‌های Cost امکان ارزیابی سربار و الگوهای اقتصادی حملات را فراهم می‌آورند و ویژگی‌های Similarity بدون نیاز به پردازش محتوای صفحه، رفتارهای تقلیدی مهاجمان را شناسایی می‌کنند.

کاهش تعداد ویژگی‌ها در ACS همچنین منجر به کاهش پیچیدگی محاسباتی، افزایش سرعت پردازش و بهبود قابلیت استفاده در محیط‌های بلادرنگ می‌شود؛ مزیتی که در بسیاری از روش‌های پیشین، به‌ویژه مدل‌های عمیق و محتوامحور، مشاهده نمی‌شود.

۸- بحث و بررسی

نتایج حاصل از دو جدول مقایسه‌ای نشان می‌دهد که چارچوب پیشنهادی ACS نه‌تنها از نظر دقت تشخیص، بلکه از منظر کارایی، سادگی و مقیاس‌پذیری نیز نسبت به مطالعات پیشین برتری دارد. در حالی که بسیاری از روش‌های موجود برای دستیابی به دقت بالا به مجموعه‌های ویژگی بزرگ، پردازش محتوای صفحات وب یا مدل‌های پیچیده یادگیری عمیق متکی هستند، ACS با استفاده از مجموعه‌ای فشرده و هدفمند از ویژگی‌های مبتنی بر دامنه توانسته است عملکردی بالاتر یا رقابتی

ارائه دهد.

هم‌راستایی نتایج کمی ارائه‌شده در جدول ۱ با تحلیل‌های مبتنی بر منحنی‌های ROC و Precision–Recall، پایداری و صحت عملکرد چارچوب پیشنهادی را تأیید می‌کند. همچنین، تحلیل ناحیه زیر منحنی (AUC) نشان می‌دهد که مدل یادگیری ماشین به‌کاررفته در چارچوب ACS (LGBM) از توان تفکیک مناسبی میان کلاس‌های فیشینگ و غیر فیشینگ برخوردار است و عملکرد آن در سناریوهای مختلف پایدار باقی می‌ماند. این نتایج بیانگر آن است که چارچوب پیشنهادی قابلیت تعمیم مناسبی به داده‌های جدید دارد.

تحلیل ساختار مجموعه ویژگی ACS نشان می‌دهد که این مجموعه شامل ترکیبی از ویژگی‌های مؤثر مبتنی بر شباهت است. به‌طور مشخص، از میان ویژگی‌های اولیه، معیارهای شباهتی شامل Jaro–Winkler، Cosine، Levenshtein، N-Gram، MetricLCS، Q-Gram و Ratcliff–Obershelp در مجموعه نهایی ACS حفظ شده‌اند. حضور این معیارها در میان بهترین ویژگی‌ها بیانگر نقش کلیدی آن‌ها در استخراج اطلاعات تبعیضی و افزایش توان تفکیک میان کلاس‌ها، بدون نیاز به پردازش محتوای صفحات وب، است.

در مقایسه با روش‌های محتوایی مانند CANTINA، چارچوب ACS از سرعت پردازش بالاتری برخوردار بوده و در برابر دستکاری، بازتولید یا رمزنگاری محتوای صفحات وب و کدهای JavaScript مقاوم‌تر است. همچنین، برخلاف روش‌هایی نظیر PhishFarm که به لیست‌های سیاه وابسته هستند، ACS قادر است حملات جدید و روز-صفر را بدون تأخیر زمانی شناسایی کند. این ویژگی‌ها نشان می‌دهد که چارچوب پیشنهادی از قابلیت استقرار در محیط‌های عملیاتی واقعی برخوردار است.

از نظر کارایی زمانی، میانگین زمان پردازش هر URL در چارچوب ACS حدود ۲۷٫۷ میلی‌ثانیه (در حالت بدون جستجوی وب) و در صورت محاسبه و دریافت دامنه‌های مشابه از موتور جستجوی گوگل حدود ۱٫۶ ثانیه بوده است. این زمان پردازش در مقایسه با روش‌های مبتنی بر محتوا، از جمله CANTINA و CASE، به‌طور محسوسی کمتر بوده و نشان می‌دهد که ACS قابلیت استفاده در سامانه‌های بلادرنگ و مقیاس‌پذیر را دارد.

با وجود نتایج امیدوارکننده، چارچوب پیشنهادی ACS دارای برخی محدودیت‌ها نیز هست. به‌طور خاص، وابستگی بخشی از ویژگی‌ها به نتایج موتور جستجوی گوگل می‌تواند در شرایط تغییر الگوریتم‌های جستجو یا محدودیت دسترسی به اینترنت چالش‌برانگیز باشد. همچنین، شناسایی دامنه‌های کاملاً جدید و فاقد هرگونه سابقه ممکن است با افزایش نرخ خطا همراه شود. از سوی دیگر، اگرچه پیاده‌سازی بلادرنگ امکان‌پذیر است، اما در مقیاس‌های بسیار بزرگ نیازمند بهینه‌سازی منابع پردازشی خواهد بود.

در مجموع، نتایج انتخاب ویژگی و تحلیل عملکرد نشان می‌دهد که مجموعه ویژگی ACS با کاهش تعداد ویژگی‌ها و حفظ معیارهای شباهتی مؤثر، توانسته است بهترین توازن میان دقت طبقه‌بندی، پیچیدگی مدل و قابلیت تعمیم را فراهم آورد. این امر چارچوب ACS را به یک راهکار عملی، کارآمد و قابل اعتماد برای مقابله با حملات فیشینگ در محیط‌های واقعی، از سطح سازمانی تا مقیاس ملی، تبدیل می‌کند.

مراجع

[1] A. Oest, Y. Safaei, A. Doupé, G.-J. Ahn, B. Wardman, and K. Tyers, "PhishFarm: A scalable framework for measuring the effectiveness of evasion techniques against browser phishing blacklists," In 2019 IEEE Symposium on Security and Privacy (SP), pp. 1344–1361, IEEE, 2019.

[2] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing websites," In Proceedings of the 16th International Conference on World Wide Web, pp. 639–648, 2007.

[3] CVE Details, Google Chrome vulnerabilities, Retrieved from https://www.cvedetails.com/product/15031/Google-Chrome.html?vendor_id=1224, n.d.

[4] D.-J. Liu, G.-G. Geng, X.-B. Jin, and W. Wang, "An efficient multistage phishing website detection model based on the CASE feature framework: Aiming at the real web environment," Journal of Computer Research and Development, Vol. 58, No. 10, pp. 2201, 2021.

[5] J. Zhou, H. Cui, X. Li, W. Yang, and X. Wu, "A novel phishing website detection model based on LightGBM and domain name features," Symmetry, Vol. 15, No. 1, 180, 2023.

[6] A. Prasad and S. Chandra, "PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning," Computers & Security, Vol. 136, 103545, 2024.

[7] M. Wang, L. Song, L. Li, Y. Zhu, and J. Li, "Phishing webpage detection based on global and local visual similarity," Expert Systems with Applications, Vol. 252, 124120, 2024.

[8] P. K. Mvula, P. Branco, G. V. Jourdan, and H. L. Viktor, "COVID-19 malicious domain names classification," Expert Systems with Applications, Vol. 204, 117553, 2022.

[9] H. C. Sit, A. Esmradi, D. W. K. Yip, and P. Sun, "An effective and robust similarity-based phishing website detector in cyber-physical systems," in Proceedings of the IEEE Conference on Communications and Network Security (CNS), pp. 1–6, 2025.

[10] A. S. Bozkir and M. Aydos, "LogoSENSE: A companion HOG based logo detection scheme for phishing web page and E-mail brand recognition," Computers & Security, Vol. 95, 101855, 2020.

[11] B. Kılıç and B. Çeliktaş, "Phishing attack detection using multi-scale visual similarity analysis," in Proc. 33rd Signal Processing and Communications Applications Conference (SIU), pp. 1–4, 2025.

[12] G. Varshney, A. Raj, D. Sangwan, S. Abuadbba, R. Mishra, and Y. Gao, "A login page transparency and visual similarity-based zero-day phishing defense protocol," Computers & Security, 104598, 2025.

[13] S. Mousavi and M. Bahaghighat, "Phishing website detection: An in-depth investigation of feature selection and deep learning," Expert Systems, Vol. 42, No. 3, e13824, 2025.

[14] W. Wei, Q. Ke, J. Nowak, M. Korytkowski, R. Scherer, and M. Woźniak, "Accurate and fast URL phishing detector: A convolutional neural network approach," Computer Networks, Vol. 178, Art. no. 107275, 2020.

[15] M. A. Saeed, A. S. S. Balaid, and O. Bahaidara, “Phishing URL detection using deep learning: A CNN-based approach,” Journal of Science and Technology, Vol. 30, No. 9, 2025.

[16] M. Pranav, “URLTran: Improving phishing URL detection using transformers,” arXiv preprint, arXiv:2106.v2, 2017.

[17] S. Asiri, Y. Xiao, and T. Li, “PhishTransformer: A novel approach to detect phishing attacks using URL collection and transformer,” Electronics, Vol. 13, No. 1, Art. no. 30, 2023.

[18] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, ACM, 2016.

[19] G. Ke, Q. Meng, T. Finley, et al., “LightGBM: A highly efficient gradient boosting decision tree,” In Advances in Neural Information Processing Systems (NeurIPS), vol. 30, pp. 3146–3154, 2017.

[20] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, “CatBoost: Unbiased boosting with categorical features,” In Advances in Neural Information Processing Systems (NeurIPS), vol. 31, pp. 6638–6648, 2018.

[21] S. O. Arik and T. Pfister, “TabNet: Attentive interpretable tabular learning,” In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 8, pp. 6679–6687, AAAI Press, 2021.

[22] L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, Springer, 2001.

[23] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119–139, 1997.

[24] J. R. Quinlan, “Induction of decision trees,” Machine Learning, vol. 1, no. 1, pp. 81–106, Springer, 1986.

[25] T. Cover and P. Hart, “Nearest neighbor pattern classification,” IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21–27, 1967.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” Nature, vol. 323, pp. 533–536, 1986.

[27] C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol. 20, no. 3, pp. 273–297, Springer, 1995.

[28] I. Rish, “An empirical study of the naive Bayes classifier,” In IJCAI Workshop on Empirical Methods in Artificial Intelligence, pp. 41–46, 2001.

[29] D. R. Cox, “The regression analysis of binary sequences,” Journal of the Royal Statistical Society: Series B, vol. 20, no. 2, pp. 215–242, 1958.

[30] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” Journal of Machine Learning Research, Vol. 3, pp. 1157–1182, 2003.

[31] D. C. Montgomery and G. C. Runger, Applied statistics and probability for engineers, 5th ed., Wiley, 2010.

[32] A. Agresti, Statistical inference, 2nd ed., Wiley, 2018.

[33] G. V. Bard, “Spelling-error tolerant, order-independent pass-phrases via the Damerau-Levenshtein string-edit distance metric,” In Proceedings of the Fifth Australasian Symposium on ACSW Frontiers, Vol. 68, pp. 117–124, 2007.

[34] M. Kiwi, M. Loebl, and J. Matoušek, “Expected length of the longest common subsequence for large alphabets,” Advances in Mathematics, Vol. 197, No. 2, pp. 480–498, 2005.

[35] J. Huang, Z. Fang, and H. Kasai, “LCS graph kernel based on Wasserstein distance in longest common subsequence metric space,” Signal Processing, Vol. 189, 108281, 2021.

[36] G. Kondrak, “N-gram similarity and distance,” In International Symposium on String Processing and Information Retrieval, pp. 115–126, Springer, Berlin, Heidelberg, 2005.

[37] N. Ohkura, M. Kiyomi, and K. Hirata, “The q-gram distance for ordered unlabeled trees,” In International Conference on Discovery Science, pp. 283–294, Springer, Berlin, Heidelberg, 2005.

[38] F. Rahutomo, T. Kitasuka, and M. Aritsugi, “Semantic cosine similarity,” In The 7th International Student Conference on Advanced Science and Technology (ICAST), Vol. 4, No. 1, 2012.

[39] W. Hidayat, E. Utami, and A. D. Hartanto, “Effect of Stemming Nazief & Adriani on the Ratcliff/Obershelp algorithm in identifying level of similarity between slang and formal words,” In 2020 3rd International Conference on Information and Communications Technology (ICOIACT), pp. 358–362, IEEE, 2020.

[40] feature23, StringSimilarity.NET, [Software]. GitHub. Retrieved from https://github.com/feature23/StringSimilarity.NET, n.d.

[41] G. Vrbančič, I. Fister Jr, and V. Podgorelec, “Datasets for phishing websites detection,” Data in Brief, Vol. 33, 106438, 2020.

[42] K. T. Chen, C. R. Huang, and C. S. Chen, “Fighting phishing with discriminative keypoint features,” IEICE Transactions on Information and Systems, Vol. 92, No. 5, pp. 1017–1030, 2009.

[43] K. L. Chiew, E. H. Chang, S. N. Sze, and W. K. Tiong, “Utilization of website logo for phishing detection,” Computers & Security, Vol. 54, pp. 16–26, 2015.

[44] X. Guang, O. Jason, P. Carolyn, and C. Lorrie, “CANTINA+: A feature-rich machine learning framework for detecting phishing websites,” ACM Transactions on Information and System Security, Vol. 14, No. 2, pp. 1–28, 2011.

پیوست

دو شبه کد به همراه پیکربندی‌های مختلف برای هر الگوریتم و فهرست کامل ویژگی‌ها در ضمیمه آورده شده است. این ویژگی‌ها ترکیبی از سه گروه Affiliation، Cost و Similarity هستند. دلیل انتخاب آن‌ها، پوشش دادن جنبه‌های مختلف URL (از جمله ساختار دامنه، هزینه ثبت دامنه و میزان شباهت به دامنه‌های معتبر) بوده است. به‌طور خاص، ویژگی‌های شباهتی معرفی‌شده در این پژوهش توانستند در میان ۱۰ ویژگی برتر انتخاب‌شده توسط الگوریتم‌های انتخاب ویژگی قرار بگیرند.

Appendix 1

شبه کد ۱:روش ACS

```
Input: URL_List
// List of website URLs to classify
Output: Classification_Label_List
 // Phishing or Legitimate for each URL
Begin
 Initialize Feature_Set_List[] ←
 For each URL in URL_List do
   //Step 1 :
   Domain Extraction Domain ← ExtractDomain(URL)
   Step 2: Google Search Query
   Search_Results←QueryGoogle(Domain_Keywords(
                                        Domain ))
   Step 3: Similarity Feature Computation
   Similarity_Scores [] ←
   For each Result in Search_Results do
     Jaro_Score ← JaroWinkler(Domain, Result.Domain)
     Levenshtein_Score←Levenshtein(Domain,
                                   Result.Domain)
     …
     Append max(Jaro_Score, Levenshtein_Score, ...) to
     Similarity_Scores
    End For
    /Step 4: WHOIS and SSL Feature Extraction
    Affiliation_Features←ExtractWHOISFeatures(
                                        Domain)
    Cost_Features ← ExtractSSLFeatures(Domain)
   // Step 5: Feature Aggregation
    ACS_Features←Combine(Similarity_Scores,
                  Affiliation_Features, Cost_Features)
    Append ACS_Features to Feature_Set_List
End For
```

الگوریتم	پیکربندی‌های بررسی‌شده	*بهترین پیکربندی
	max_iter=2000,solver=liblinear	
SVM	RBF,C=1؛ RBF,C=10؛ Poly,deg=3؛ Sigmoid؛ RBF,C=1,gamma=auto	★ RBF kernel, C=1.0, gamma=auto
Random Forest	n_estimators=100؛ n_estimators=300؛ max_depth=10؛ max_depth=20,n_estimators=200؛ min_samples_split=5	★ n_estimators=300
Decision Tree	default؛ max_depth=5؛ max_depth=10؛ min_samples_leaf=10؛ criterion=entropy	★ criterion=entropy
KNN	k=3؛ k=5؛ k=7؛ weights=distance؛ metric=manhattan	★ metric=manhattan
AdaBoost	default؛ n_estimators=200؛ learning_rate=0.5؛ SAMME؛ n_estimators=300,learning_rate=0.3	★ n_estimators=200
Naive Bayes	default؛ var_smoothing=1e−9؛ 1e−4؛ 1e−3؛ 1e−2	★ default (GaussianNB)
MLP	(50),500؛ (۱۰۰,۵۰)،۸۰۰؛ (۱۰۰)؛ ۲۰۰۰,(۲۰۰,۱۰۰)؛ ۱۵۰۰,(۱۰۰,۵۰,۲۰)	★ (200,100), max_iter=2000
TabNet	default؛ n_d=32, n_a=32; optimizer_fn=torch.optim.Adam; optimizer_params=dict(lr=1e-3); n_steps=5	★ optimizer_fn= torch.optim.Adam
XGBoost	n_estimators=100؛ n_estimators=300؛ max_depth=5؛ lr=0.05؛ subsample=0.8	★ n_estimators=300
LightGBM	default؛ num_leaves=31؛ n_estimators=300؛ lr=0.03؛ max_depth=10	★ n_estimators=300
CatBoost	default؛ depth=8؛ iterations=300؛ lr=0.05؛ l2=5	★ depth=8

Appendix 4

qty_dot_url, qty_hyphen_url, qty_underline_url, qty_slash_url, qty_questionmark_url, qty_equal_url, qty_at_url, qty_and_url, qty_comma_url, qty_hashtag_url, qty_percent_url, length_url, email_in_url, qty_hyphen_domain, qty_vowels_domain, domain_length, qty_dot_directory, qty_hyphen_directory, qty_underline_directory, qty_slash_directory, qty_equal_directory, directory_length, qty_dot_file, qty_hyphen_file, qty_underline_file, qty_equal_file, qty_and_file, file_length, qty_dot_params, qty_hyphen_params, qty_underline_params, qty_equal_params, qty_at_params, qty_and_params, qty_comma_params, qty_plus_params, qty_asterisk_params, qty_hashtag_params, qty_percent_params, params_length, tld_present_params, qty_params, asn_ip, time_domain_activation, time_domain_expiration, ttl_hostname, url_Google_index, domain_Google_index, url_shortened, tld_size, dur_domain, dur_ip, ssl_day, dur_ssl, Levenshtein, Cosine, RatcliffObershelp

Appendix5

qty_dot_url, qty_hyphen_url, qty_underline_url, qty_slash_url, qty_questionmark_url, qty_equal_url, qty_at_url, qty_and_url, qty_exclamation_url, qty_tilde_url, qty_comma_url, qty_plus_url, qty_asterisk_url, qty_hashtag_url, qty_percent_url, length_url, email_in_url, qty_dot_domain, qty_hyphen_domain, qty_vowels_domain, domain_length, domain_in_ip, server_client_domain, qty_dot_directory, qty_hyphen_directory, qty_underline_directory, qty_slash_directory, qty_equal_directory, qty_at_directory, qty_and_directory, qty_tilde_directory, qty_hashtag_directory, qty_percent_directory, directory_length, qty_dot_file, qty_hyphen_file, qty_underline_file, qty_equal_file, qty_at_file, qty_and_file, qty_comma_file, qty_hashtag_file, file_length, qty_dot_params, qty_hyphen_params, qty_underline_params, qty_slash_params, qty_equal_params, qty_at_params, qty_and_params, qty_exclamation_params, qty_space_params, qty_comma_params, qty_plus_params, qty_asterisk_params, qty_hashtag_params, qty_percent_params, params_length, tld_present_params, qty_params, asn_ip, time_domain_activation, time_domain_expiration, qty_ip_resolved, qty_mx_servers, ttl_hostname, url_Google_index, domain_Google_index,

```
// Step 6: Feature Selection
Selected_Features←ApplyFeatureSelection(
        Feature_Set_List,  method = {Filter, Wrapper})
// Step 7:
Model Training Classifier←TrainModel(
        Selected_Features, labels,
        algorithm = {RandomForest, AdaBoost, ...})
// Step 8:
Prediction Classification_Label_List ← Classifier.Predict(
                                    Selected_Features)
Return Classification_Label_List
End
```

Appendix 2

شبه کد ۲: تحلیل ویژگی ها

Step 1: Model Evaluation
1 .For each classification algorithm:
1,1 Repeat for N iterations:
- Split dataset into training and testing sets
- Train model on training data
- Evaluate model on testing data
- Record performance metrics (e.g., accuracy, precision, recall)

1,2 Aggregate and save overall model performance
Step 2: Feature Analysis
(a) Filter Method
2 .Compute filter scores for all features
3 .For each feature in ranked order:
- Add feature to SelectedFeatures
- Build dataset with SelectedFeatures + class label
- Train and evaluate model
- Record accuracy and selected feature set

(b) Wrapper Method
4 .Initialize SelectedFeatures∅ ←
5 .While stopping criteria not met:
- Select next best feature using wrapper strategy
- Add feature to SelectedFeatures
- Build dataset with SelectedFeatures + class label
- Train and evaluate model
- Record accuracy and selected feature set

Step 3: Result Reporting
6 .Compare results from both methods
7 .Report the smallest feature subset achieving highest classification accuracy

Appendix 3

در این پیوست، پیکربندی‌های مختلف پارامترهای اصلی برای هر الگوریتم یادگیری ماشین ارائه شده است. برای هر مدل، چندین تنظیم مورد ارزیابی قرار گرفته و بهترین پیکربندی بر اساس بیشترین میانگین F1-score در اعتبارسنجی متقاطع انتخاب شده است. پیکربندی‌های علامت‌گذاری‌شده با ستاره (*) بهترین اجرای هر الگوریتم را نشان می‌دهند.

الگوریتم	پیکربندی‌های بررسی‌شده	*بهترین پیکربندی
Logistic Regression	max_iter=500؛ max_iter=1000,C=0.5؛ max_iter=1000,C=2؛ max_iter=2000,penalty=l2؛	★ max_iter=2000, solver=liblinear

url_shortened, tld_size, dur_domain, dur_ip, ssl_day, dur_ssl, JaroWinkler, Levenshtein, QGram, Cosine, RatcliffObershelp, similarave

Appendix 6

qty_dot_url, qty_hyphen_url, qty_slash_url, qty_equal_url, qty_exclamation_url, qty_space_url, qty_tilde_url, qty_comma_url, qty_asterisk_url, qty_hashtag_url, qty_tld_url, length_url, email_in_url, qty_dot_domain, qty_hyphen_domain, qty_slash_domain, qty_questionmark_domain, qty_equal_domain, qty_and_domain, qty_tilde_domain, qty_asterisk_domain, qty_hashtag_domain, qty_dollar_domain, qty_percent_domain, qty_dot_directory, qty_underline_directory, qty_slash_directory, qty_equal_directory, qty_space_directory, qty_tilde_directory, qty_comma_directory, qty_hashtag_directory, qty_hyphen_file, qty_underline_file, qty_questionmark_file, qty_and_file, qty_space_file, qty_asterisk_file, qty_dollar_file, qty_percent_file, file_length, qty_underline_params, qty_slash_params, qty_questionmark_params, qty_equal_params, qty_at_params, qty_and_params, qty_space_params, qty_tilde_params, qty_hashtag_params, qty_dollar_params, qty_percent_params, qty_params, asn_ip, time_domain_activation, time_domain_expiration, qty_ip_resolved, qty_nameservers, ttl_hostname, url_Google_index, domain_Google_index, tld_size, dur_ip, dur_ssl, is_ssl_free, Damerau

Appendix 7

qty_dot_url, qty_hyphen_url, qty_underline_url, qty_slash_url, qty_questionmark_url, qty_equal_url, qty_at_url, qty_and_url, qty_hashtag_url, qty_percent_url, qty_tld_url, length_url, email_in_url, qty_dot_domain, qty_hyphen_domain, qty_vowels_domain, domain_length, server_client_domain, qty_dot_directory, qty_hyphen_directory, qty_underline_directory, qty_slash_directory, directory_length, qty_dot_file, qty_hyphen_file, qty_underline_file, qty_percent_file, file_length, qty_dot_params, qty_hyphen_params, qty_underline_params, qty_slash_params, qty_equal_params, qty_at_params, qty_and_params, qty_hashtag_params, qty_percent_params, params_length, tld_present_params, qty_params, domain_spf, asn_ip, time_domain_activation, time_domain_expiration, qty_ip_resolved, qty_nameservers, qty_mx_servers, ttl_hostname, tls_ssl_certificate, url_Google_index, domain_Google_index, url_shortened, tld_size, dur_domain, dur_ip, ssl_day, dur_ssl, JaroWinkler, Levenshtein, Damerau, LongestCommonSubsequence, MetricLCS, NGram, QGram, Cosine, RatcliffObershelp, similarave

Appendix 8

qty_dot_url, qty_hyphen_url, qty_underline_url, qty_slash_url, qty_questionmark_url, qty_equal_url, qty_and_url, length_url, email_in_url, qty_dot_domain, qty_hyphen_domain, qty_vowels_domain, domain_length, qty_dot_directory, qty_hyphen_directory, qty_slash_directory, directory_length, qty_dot_file, qty_hyphen_file, qty_and_file, file_length, qty_dot_params, qty_hyphen_params, qty_underline_params, qty_slash_params, qty_equal_params, qty_at_params, qty_and_params, qty_percent_params, params_length, tld_present_params, qty_params, asn_ip, time_domain_activation, time_domain_expiration, qty_ip_resolved, ttl_hostname, tls_ssl_certificate, url_Google_index, domain_Google_index, url_shortened, tld_size, dur_domain, dur_ip, ssl_day, dur_ssl, JaroWinkler, Levenshtein, Damerau, LongestCommonSubsequence, MetricLCS, NGram, QGram, Cosine, RatcliffObershelp, similarave

Appendix 9

params_length, dur_ip, tld_size, length_url, JaroWinkler, Cosine, Levenshtein, ttl_hostname, domain_length, qty_dot_domain, dur_ssl, NGram, qty_equal_params, asn_ip, MetricLCS, QGram, file_length, qty_slash_params, directory_length, RatcliffObershelp, dur_domain, Damerau, LongestCommonSubsequence, qty_vowels_domain, qty_ip_resolved, qty_and_params, qty_slash_url, qty_underline_params, qty_hyphen_params, similarave, domain_Google_index, qty_at_params, qty_percent_params, qty_dot_params, url_Google_index, qty_dot_url, qty_hyphen_url, qty_hyphen_domain, tls_ssl_certificate, qty_mx_servers, qty_nameservers, time_domain_activation, qty_slash_directory, domain_spf, qty_questionmark_url, qty_hyphen_directory, time_domain_expiration, qty_percent_url, qty_hyphen_file, qty_dot_directory, qty_tld_url, qty_equal_url, qty_equal_directory, qty_hashtag_directory

پاورقی‌ها:

[1] Light Gradient Boosting Machine
[2] Filter
[3] Wrapper
[4] Embedded
[5] Mutual Information Classifier
[6] F Regression
[7] Mutual Information Regression
[8] ANOVA F-test(f_classif)
[9] Chi-Squared Test (chi2)
[10] Recursive Feature Elimination (RFE)
[11] Recursive Feature Elimination with Cross-Validation (RFECV)
[12] Random Forest
[13] Selection Forward Sequential - SFS
[14] Selection Backward Sequential - SBS
[15] Selection Forward Floating Sequential - SFFS
[16] Selection Backward Floating Sequential - SBFS
[17] Jaro-Winkler
[18] Levenshtein distance
[19] Damerau
[20] Edit Distance
[21] Longest Common Subsequence - LCS
[22] Metric LCS (Longest Common Subsequence)
[23] N-gram
[24] Q-gram
[25] Cosine Similarity
[26] Ratcliff/Obershelp