

دانشکده مهندسی کامپیوتر

**پایان‌نامه مقطع کارشناسی‌ارشد مهندسي کامپیوتر - نرم‌افزار**

**تشخیص حملات: یک روش خصمانه یادگیری ماشین**

**طه اخلاق‌پسندی**

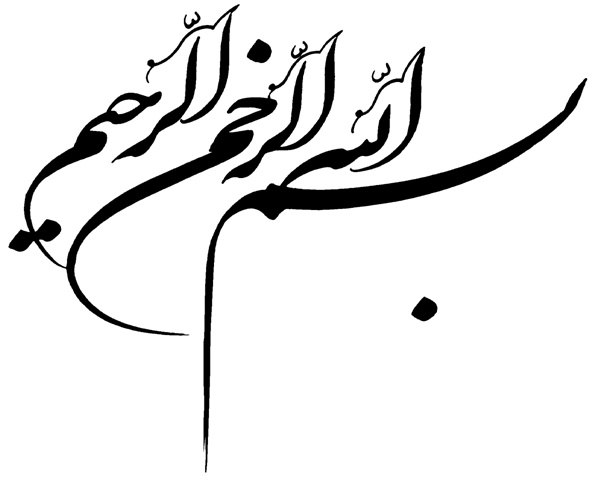
**استاد راهنما:**

**دکتر محمدهادی علائیان**

**استاد مشاور:**

[نام استاد يا اساتيد مشاور (براي پایان‏نامه/رساله)]

**[تاریخ دقیق روز، ماه و سال دفاع]**



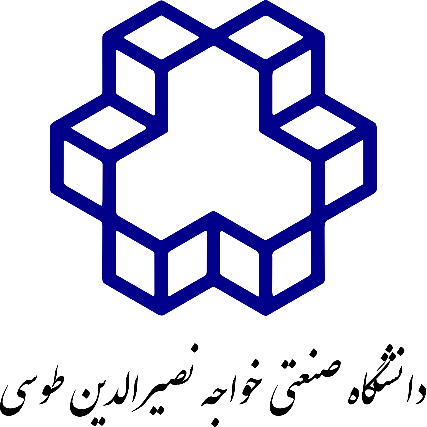
# تأييديّه هيات داوران

اعضاي هيئت داوران، نسخه نهائي پایان‏نامه آقاي: **طه اخلاق‌پسندی**

را با عنوان: تشخیص حملات: یک روش خصمانه یادگیری ماشین

از نظر شکل و محتوي بررسي نموده و پذيرش آن را براي تکميل درجه کارشناسی‌ارشد تأييد مي‏کنند.

|  |  |  |  |
| --- | --- | --- | --- |
| **اعضاي هيئت داوران** | **نام و نام خانوادگي** | **رتبه علمي** | **امضاء** |
| 1. استاد راهنما | دکتر محمدهادی علائیان | استادیار |  |
| 1. استاد مشاور |  |  |  |
| 1. استاد مشاور |  |  |  |
| 1. استاد داور |  |  |  |
| 1. استاد داور |  |  |  |
| 1. نماينده تحصیلات تکمیلی |  |  |  |

**دانشگاه صنعتی خواجه نصیرالدین طوسی**

|  |
| --- |
| **اظهارنامه دانشجو** |
| اینجانب **طه اخلاق‌پسندی** دانشجوی مقطع ارشد رشته **مهندسی کامپیوتر** گواهی می‌نمایم که مطالب ارائه شده در این پایان‌نامه با عنوان:  **تشخیص حملات: یک روش خصمانه یادگیری ماشین**  با راهنمایی استاد محترم **دکتر** **محمدهادی علائیان** توسط شخص اینجانب انجام شده است. صحت و اصالت مطالب نوشته شده در این پایان‌نامه تأیید می‌شود و در تدوین متن پروژه قالب مصوب دانشگاه را به طور کامل رعایت کرده‌ام.  **امضاء دانشجو:**  **تاریخ:** |

**حق طبع، نشر و مالکیت نتایج**

1- حق چاپ و تکثیر این پایان‏نامه/رساله متعلق به نویسنده و استاد راهنمای آن است. هرگونه تصویربرداری از کل یا بخشی از پروژه تنها با موافقت نویسنده یا استاد راهنما یا کتاب‌خانه دانشکده‌های مهندسی برق و کامپیوتر دانشگاه صنعتی خواجه نصیرالدین طوسی مجاز است.

2- کلیه حقوق معنوی این اثر متعلق به دانشگاه صنعتی خواجه نصیرالدین طوسی است و بدون اجازه کتبی دانشگاه قابل واگذاری به شخص ثالث نیست.

3- استفاده از اطلاعات و نتایج موجود پروژه بدون ذکر مرجع مجاز نیست.

###### تقديم به:

###### تشکر و قدرداني

###### چکيده

هدف تشخیص نفوذ درون شبکه با استفاده از بسته‏های شبکه است. انواع مختلفی از حملات ازجمله DOS، DDOS، XSS، BotNet، بدافزار و ... را به روش‏های یادگیری ماشین تشخیص می‏دهند. از طرفی تحلیل دستی و بررسی یک به یک بسته‎های شبکه توسط نیروهای انسانی متخصص به دلیل کمبود در تعداد متخصص نسبت به تعداد رو به افزایش حملات و همچنین امکان وقوع خطای انسانی در تشخیص حملات، کاری خسته‌کننده و تقریباً ناممکن است. در سال‏های گذشته، تحقیقات زیادی برای طراحی یک مدل یادگیری ماشین یا یادگیری عمیق برای تشخیص نفوذ انجام شده است که دقت خوبی در تشخیص حاصل شده است. اما، هیچکدام از مدل‏های حاصله در معرض حمله قرار نگرفته بودند. در این پژوهش، روش جدید برای بهبود در تشخیص نفوذ با استفاده از روش‌های یادگیری ماشین ارائه خواهد شد. مدل حاصله، با استفاده از روش‏های حمله خصمانه، مورد ارزیابی و تقویت قرار خواهد گرفت. همچنین، با اضافه کردن تروجان بر روی مدل، یک درپشتی بر روی مدل قرار داده خواهد شد و تاثیر آن بر روی مدل مورد بررسی قرار خواهد گرفت. نتایج ارزیابی نشان میدهد که دقت مدل در زمانی که تروجان نداشته 94% بوده و زمانی که تروجانی شده به 95.8% افزایش یافته است.

کلمات کلیدی: حملات شبکه، یادگیری ماشین، یادگیری عمیق، تشخیص نفوذ، تروجان شبکه های عصبی، حملات خصمانه

###### فهرست مطالب

عنوان صفحه

[تأييديّه هيات داوران ‌ه](#_Toc531381831)

[فهرست شکل‏ها ‌ه](#_Toc531381832)

[فهرست جدول‏ها ‌ز](#_Toc531381833)

[فهرست علامت‏ها و نشانه‏ها ‌ط](#_Toc531381834)

[فصل ۱- مقدمه.......................................................................................................................................................... ۱](#_Toc15134380)

[۱-۱- پيشگفتار ۱](#_Toc15134381)

[۲-۱- بیان مساله ۲](#_Toc15134382)

[۳-۱- هدف ........................................................................................................................................................... ۳](#_Toc15134382)

[۴-۱- اهمیت موضوع ۴](#_Toc15134382)

[۵-۱- ساختار گزارش ۴](#_Toc15134382)

[تأييديّه هيات داوران ‌ه](#_Toc531381831)

[فهرست شکل‏ها ‌ه](#_Toc531381832)

[فهرست جدول‏ها ‌ز](#_Toc531381833)

[فهرست علامت‏ها و نشانه‏ها ‌ط](#_Toc531381834)

[فصل ۱- مقدمه.......................................................................................................................................................... ۱](#_Toc15134380)

[۱-۱- پيشگفتار ۱](#_Toc15134381)

[۲-۱- بیان مساله ۲](#_Toc15134382)

[۳-۱- هدف ........................................................................................................................................................... ۳](#_Toc15134382)

[۴-۱- اهمیت موضوع ۴](#_Toc15134382)

[۵-۱- ساختار گزارش ۴](#_Toc15134382)

[فصل ۲- ادبیات موضوع ۵](#_Toc15134383)

[۱-۲- مقدمه ۵](#_Toc15134384)

[۲-۲- یادگیری ماشین ۵](#_Toc15134385)

[۳-۲- یادگیری نظارت‌شده ۷](#_Toc15134386)

[۴-۲- یادگیری نظارت‌نشده ۷](#_Toc15134386)

[۵-۲- یادگیری عمیق ۸](#_Toc15134386)

[۶-۲- داده‌کاوی ۹](#_Toc15134386)

[۷-۲- پردازش زبان‌ طبیعی ۱۱](#_Toc15134386)

[۸-۲- بدافزار ۱۲](#_Toc15134386)

[۹-۲- سیستم تشخیص نفوذ ۱۲](#_Toc15134386)

[۱۰-۲- سیستم تشخیص نفوذ مبتنی بر امضا ۱۳](#_Toc15134386)

[۱۱-۲- سیستم تشخیص نفوذ مبتنی بر ناهنجاری ۲۰](#_Toc15134386)

[۱۲-۲- تروجان ۲۳](#_Toc15134386)

[۱۳-۲- کرم ۲۵](#_Toc15134386)

[۱4-۲- حمله DoS ۲۵](#_Toc15134386)

[۱5-۲- حمله DDoS ۲۵](#_Toc15134386)

[۱6-۲- شبکه عصبی مصنوعی ۲۵](#_Toc15134386)

[۱7-۲- شبکه عصبی عمیق ۲۵](#_Toc15134386)

[۱8-۲- شبکه عصبی بازگشتی ۲۵](#_Toc15134386)

[۱9-۲- شبکه مولد متخاصم ۲۵](#_Toc15134386)

[20-۲- حافظه کوتاه مدت طولانی ۲۵](#_Toc15134386)

[21-۲- تریگر تروجان ۲۵](#_Toc15134386)

[22-۲- ابر ۲۵](#_Toc15134386)

[2۳-۲- محاسبات مه ۲۵](#_Toc15134386)

[فصل 3- کارهای پیشین ۵](#_Toc15134383)

[۱-3- سیستم تشخیص نفوذ ۵](#_Toc15134384)

[۲-3- کلاس‌های حملات کامپیوتری ۵](#_Toc15134385)

[۳-3- طبقه‌بندی IDSها بر اساس نوع تحلیل ۷](#_Toc15134386)

[۴-3- سیستم تشخیص نفوذ مبتنی بر میزبان (HIDS) ۷](#_Toc15134386)

[۵-3- سیستم‌های تشخیص نفوذ مبتنی بر شبکه (NIDS) ۸](#_Toc15134386)

[۶-3- مزایای IDSها ۹](#_Toc15134386)

[۷-3- طبقه‌بندی روش‌های تشخیص مبتنی بر ناهنجاری ۱۱](#_Toc15134386)

[۸-3- تکنیک‌های مبتنی بر آمار ۱۲](#_Toc15134386)

[۹-3- تکنیک‌های مبتنی بر دانش ۱۲](#_Toc15134386)

[۱۰-3- تکنیک‌های مبتنی بر یادگیری ماشین ۱۳](#_Toc15134386)

[۱۱-3- معماری‌هایی برای تشخیص نفوذ ۲۰](#_Toc15134386)

[۱۲-3- مدیریت اعتماد توزیع‌شده ۲۳](#_Toc15134386)

[۱۳-3- جدیدترین و برترین نرم‌افزارهایIDS ۲۵](#_Toc15134386)

[۱۴-3- نحوه ی تشخیص حملاتDDOS ۲۹](#_Toc15134386)

[۱۵-3- نحوه ی تشخیص حملات SQL INJECTION ۳۴](#_Toc15134386)

[۱۶-3- نحوه ی تشخیص حملات XSS ۳۷](#_Toc15134386)

[۱7-3- رویکرد حفظ حریم خصوصی مبتنی بر مه برای تشخیص نفوذ مبتنی بر امضا 40](#_Toc15134386)

[۱8-3- حملات تروجان 44](#_Toc15134386)

[پیوست ‌أ- واژه‏نامه فارسي-انگليسي ۴۵](#_Toc15134415)

[فهرست مرجع‏ها ۴۶](#_Toc15134422)

[پیوست ‌أ- سبك‏هاي تعريف شده 33](#_Toc531381870)

[پیوست ‌ب- روش انتقال سبك‏ها به فايل ديگر 35](#_Toc531381871)

[پیوست ‌ج- روش مناسب توليد فايل pdf 37](#_Toc531381872)

[پیوست ‌د- معادل فارسی تعدادی از واژه‏های بیگانه 39](#_Toc531381873)

[پیوست ‌ه- واژه‏نامه فارسي-انگليسي 41](#_Toc531381874)

[پیوست ‌و- واژه‏نامه انگليسي-فارسي 43](#_Toc531381875)

[پیوست ‌ز- نكته‏های قابل توجه در ارائه شفاهي 45](#_Toc531381876)

[فهرست مرجع‏ها 46](#_Toc531381877)

فهرست شکل‏ها

عنوان صفحه

[شکل ‏3‑۱ طبقه‌بندی IDSها ۷](#_Toc530263778)

[شکل ‏3‑۲ تشخیص بدافزار. ۱۰](#_Toc530263779)

[شکل ‏3‑۳ طبقه‌بندی روش‌های تشخیص مبتنی بر ناهنجاری ۱۱](#_Toc530263780)

[شکل ‏3‑۴ یادگیری نظارت‌شده در سیستم تشخیص نفوذ ۱۵](#_Toc530263781)

[شکل ‏3‑۵ یادگیری بدون نظارت در سیستم تشخیص نفوذ ۲۰](#_Toc530263782)

[شکل ‏3‑۶ IDS توزیع‌شده ۲۱](#_Toc530263783)

[شکل 3‑۷ معماری IDS مستقل ۲۲](#_Toc530263784)

[شکل ‏3‑۸ معماری IDS توزیع‌شده ۲۲](#_Toc530263785)

[شکل ‏3‑۹ معماری IDS سلسله مراتبی ۲۳](#_Toc530263781)

[شکل ‏3‑۱۰ نرم‌افزارهایIDS ۲۸](#_Toc530263781)

[شکل ‏3‑۱۱ نرم‌افزارهایIDS ۲۸](#_Toc530263781)

[شکل ‏3‑۱۲ دیتاست Cicddos2019 ۲۹](#_Toc530263781)

[شکل ‏3‑۱۳ الگوریتم PCA ۳۰](#_Toc530263781)

[شکل ‏3‑۱۴ کاهش ویژگی‌ها ۳۱](#_Toc530263781)

[شکل ‏3‑۱۵ نحوه ی تشخیص حملاتDDOS با استفاده از یادگیری ماشین ۳۱](#_Toc530263781)

[شکل ‏3‑۱۶ سیستم تشخیص نفوذ هیبریدی ۳۲](#_Toc530263781)

[شکل ‏3‑۱۷ IDS در سطح پایگاه‌داده ۳۵](#_Toc530263781)

[شکل ‏3‑۱۸ نقشه‌های خود سازماندهی داخلی ۳۵](#_Toc530263781)

[شکل ‏3‑۱۹ مقایسه بین الگوریتم‌ها ۳۶](#_Toc530263781)

[شکل ‏3‑۲۰ نحوه ی تشخیص حملات XSS ۳۸](#_Toc530263781)

[شکل ‏3‑۲۱ محاسبه مه ۴۰](#_Toc530263781)

[شکل ‏3‑۲۲ شبکه تشخیص نفوذ مشترک (CIDN) مبتنی بر چالش ۴۱](#_Toc530263781)

[شکل ‏3‑۲۳ تعاملات شبکه ۴۳](#_Toc530263781)

[شکل ‏3-۲4 طبقه بندی اشتباه همه به یک ۴6](#_Toc530263781)

[شکل ‏3‑۲5 طبقه بندی اشتباه همه به همه ۴7](#_Toc530263781)

فهرست جدول‏ها

عنوان صفحه

جدول ‏3-۱ کلاس‌های حملات کامپیوتری ۶

فهرست علامت‏ها و نشانه‏ها

عنوان علامت اختصاري

ثابت انتشار *γ*

ثابت پلانک *h*

ثابت فاز *β*

دمای مطلق (معمولا بر حسب کلوین) 

ضریب رسانایی الکتریکی *σ*

ضریب گذردهی الکتریکی *ε*

ضریب نفوذپذیری مغناطیسی *μ*

فرکانس پلاسما *ωp*

# مقدمه

## ۱-۱- پيشگفتار

با ظهور و توسعه یادگیری ماشین و ابزارها و الگوریتم‌های مربوط به آن در حوزه‌های مختلف مانند داده‌کاوی، پردازش زبان‌های طبیعی، تشخیص گفتار و تبدیل گفتار به زبان، متخصصین امنیتی به فکر استفاده از ابزار و الگوریتم‌های موجود یادگیری ماشین افتادند و روش‌های متعددی را برای تشخیص بدافزار معرفی کردند. طی چند سال اخیر بحث پیرامون این روش‌ها بسیار رواج پیدا کرده و روش‌های مختلفی در مقالات علمی معتبر یافت می‌شود.

روزانه تعداد زیادی از حملات مختلف رخ می‌دهد. به همین دلیل تحلیل و بررسی حملات توسط نیروهای انسانی متخصص به دلیل کم بودن تعداد افراد متخصص نسبت به تعداد رو به افزایش این حملات و همچنین امکان وقوع خطای انسانی در تشخیص نفوذ امری خسته کننده و تقریبا ناممکن است. از این رو در سال‌های اخیر اقداماتی برای ماشینی کردن فرآیند تشخیص و جلوگیری از حملات انجام شد و نتیجه آن ساخت سیستم‏های تشخیص نفوذ متعددی شد که غالبا از روش‌های تشخیص امضا و یا روش‌های اکتشافی برای تشخیص بهره می‌برند.

## ۱-۲- بیان مساله

مساله اصلی در این پایان نامه، ارائه راه کارهای موجود، برای بهبود نتیجه تشخیص نفوذ با استفاده از روش‌های یادگیری ماشین است. چرا که روش‌های یادگیری ماشین، در حال رشد هستند و دائماً با روش‌هایی که از لحاظ عملکردی و قدرت پردازشی، کارایی و دقت بهتری دارند جایگزین می‌شوند. روش‌های تشخیص مبتنی بر امضا دارای یک نقطه ضعف اساسی است. مشکل اساسی این روش عدم شناخت حملات جدید است که قبلاً در این سیستم تشخیص نفوذ وجود نداشته است و به روزرسانی دائم آن، یکی از نیازهای مهم آن‌ها برای بروزرسانی پایگاه‌داده امضاها است. در روش‌های اکتشافی فعالیت‌های مشکوک، نظیر وجود ترافیک غیرعادی در شبکه، ثبت کاراکترهای زده شده بر روی صفحه کلید در مرورگر، فعالیت‌های تحت شبکه‌ای که مشکوک هستند، دسترسی‌های غیر مجاز و‌.‌. . شناسایی می‏شوند. در صورتی که موارد گفته شده بیش از مقدار آستانه که قابل تغییر نیز است رخ دهد، حمله تشخیص داده شده و در غیر این شرایط عادی تلقی می‌شود. در این روش نیز تعیین مقدار آستانه کار ساده‌ای نیست و اگر به درستی تعیین نشود، تشخیص حملات دچار مشکل خواهد شد؛ زیرا امکان دارد یکسری از حملات را تشخیص نداده و یا تعدادی از عملیات عادی را حمله تشخیص دهد.

## ۱-۳- هدف

هدف در این پایان‌نامه، تشخیص نفوذ درون شبکه است. برای رسیدن به این اهمیت، از روش‏های تشخیص مبتنی بر امضا و مبتنی بر ناهنجاری استفاده می شود[1] که روش ما تشخیص مبتنی بر ناهنجاری است. نیاز به بروزرسانی امضاء در جهت کشف حملات یکی از نقاط ضعف اساسی روش‌های تشخیص مبتنی بر امضا است[2] و روش تشخیص آن به این صورت است که از یک سری حملاتی که قبلا وجود داشته اند برای تشخیص حمله استفاده میشود. در روش‌های مبتنی بر ناهنجاری ، تشخیص نفوذ با استفاده از فعالیت‌های مشکوک، نظیر وجود ترافیک غیرعادی در شبکه، ثبت کاراکترهای زده شده بر روی صفحه کلید در مرورگر، فعالیت‌های تحت شبکه‌ای که مشکوک هستند و دسترسی‌های غیر مجاز صورت می‏پذیرد[4]. با تعیین مقدار آستانه برروی تعداد دفعات عملیات‏هایی که ممکن است انجام شود، حمله تشخیص داده خواهد شد و در غیر این صورت، شرایط عادی تلقی می‌شود[3]. هرچند، تعیین مقدار آستانه کار ساده‌ای نیست و اگر به درستی تعیین نشود، تشخیص حملات دچار مشکل خواهد شد؛ ممکن است تعدادی از حملات را تشخیص ندهد و یا تعدادی از عملیات عادی را حمله تشخیص دهد [3].

با ظهور و توسعه یادگیری ماشین و ابزارها و الگوریتم‌های مربوط به آن در حوزه‌های مختلف مانند داده‌کاوی، پردازش زبان‌های طبیعی، تشخیص گفتار و تبدیل گفتار به زبان، متخصصین امنیتی به فکر استفاده از ابزار و الگوریتم‌های موجود یادگیری ماشین افتادند و روش‌های متعددی را برای تشخیص بدافزار معرفی کردند[5][6]. این مدل‌ها با استفاده از الگوریتم های یادگیری ماشین، مانند RF ، SVM ،Decision tree ،Logistic Regression ،Nave Bayes ،DNN ، ANN ،CNN ، RNN ، LSTM و GRU با دقت های مختلفی ساخته شده است [3][4][5][6]. گروهی با استفاده از یادگیری ماشین[6]، یا یادگیری عمیق[5] ، مدل های مختلف با دقت های مختلف ساختند. در [5] با استفاده از ANN و LSTM و ساخت چندین مدل به دقت‌های بین 72% تا 99% دست یافتند. همچنین، در [3] با استفاده از SVM،Decision tree ، Logistic Regression وNave Bayes به دقت های بین 73% تا 98% رسیدند. در تمامی موارد گفته شده دقت به میزان خوبی بدست آمده است، اما هیچ کدام از آن‌ها مدل خودشان را در معرض حمله قرار نداده بودند. به عبارت بهتر، به مدلی که طراحی کرده بودند حمله نشده تا توانایی مدل خودشان را مورد ارزیابی قرار دهند.

## ۱-۴- اهمیت موضوع

روزانه تعداد زیادی از حملات مختلف رخ می‌دهد. به همین دلیل تحلیل و بررسی حملات توسط نیروهای انسانی متخصص به دلیل کم بودن تعداد افراد متخصص نسبت به تعداد رو به افزایش این حملات و همچنین امکان وقوع خطای انسانی در تشخیص نفوذ امری خسته کننده و تقریبا ناممکن است. از این رو در سال‌های اخیر اقداماتی برای ماشینی کردن فرآیند تشخیص و جلوگیری از حملات انجام شد و نتیجه آن ساخت سیستم‏های تشخیص نفوذ متعددی شد که غالبا از روش‌های تشخیص امضا و یا روش‌های اکتشافی برای تشخیص بهره می‌برند.

## ۱-۵- ساختار گزارش

در فصل اول انگیزه، شرح مساله و حوزه پایان نامه مطرح شده است و در فصل دوم درباره اصطلاحات تخصیی توضیحاتی داده شده و در فصل سوم درباره کارهای پیشینی که انجام شده صحبت کرده و به توضیح مقالات خوانده شده می‌پردازیم. در فصل چهارم طرح پیشنهادی را بیان میکنیم و در فصل پنج هم به نتیجه گیری و کارهای آتی میپردازیم.

# فصل 2- ادبیات موضوع

## 2-۱- مقدمه

در این فصل به بررسی اجمالی اصطلاحات تخصصی بکار برده شده در این مستند میپردازیم. همانطور که میدانیم در هر زمینه‌ای کلمات و اصطلاحات متفاوتی وجود دارند که نیازمند توضیح میباشند. تحلیل و تشخیص حملات نیز از این موضوع مستثنی نیست. در ادامه مواردی را که نیاز به توضیح دارند را شرح میدهیم.

## 2-2- یادگیری ماشین

یادگیری ماشین[[1]](#footnote-2) یکی از زیرمجموعه ‌های هوش مصنوعی است که به سیستم‌ها این امکان را می دهد تا به صورت خودکار یادگیری و پیشرفت داشته باشند بدون اینکه به برنامه نویسی صریحی برای آن داشته باشند. در واقع، در این علم تلاش می‌شود تا با بهره‌گیری از الگوریتم‌ها، یک ماشین به شکلی طراحی شود که بدون آنکه صراحتا برنامه‌ریزی و تک تک اقدامات به آن دیکته شود بتواند بیاموزد و عمل کند. در یادگیری ماشین، به جای برنامه‌نویسی همه چیز، داده‌ها به یک الگوریتم عمومی داده می‌شوند و این الگوریتم است که براساس داده‌هایی که به آن داده شده منطق خود را می‌سازد. یادگیری ماشین روش‌های گوناگونی دارد که از آن جمله می‌توان به یادگیری نظارت شده، نظارت نشده و یادگیری تقویتی اشاره کرد.

## 2-3- یادگیری نظارت‌شده

یادگیری نظارت‌شده[[2]](#footnote-3) نوعی از یادگیری مربوط به یادگیری ماشین است که در آن ورودی و خروجی مشخص است و به اصطلاح ناظری وجود دارد که اطلاعاتی را در اختیار یادگیرنده قرار میدهد، و به این ترتیب سیستم سعی میکند تا تابعی را از ورودی به خروجی فراگیرد.

## 2-4- یادگیری نظارت‌نشده

یادگیری نظارت‌نشده[[3]](#footnote-4) بر خلاف یادگیری نظارت شده، داده‌های مشخصی از قبل وجود ندارد و هدف ارتباط ورودی و خروجی نیست، بلکه تنها دسته‌بندی‌ آن‌ها مهم است و اين يادگيرنده است که بايستي در داده‌ها به دنبال ساختاري خاص بگردد.

## 2-5- یادگیری عمیق

یادگیری عمیق[[4]](#footnote-5) روش‌هایی از یادگیری ماشین بر پایه استفاده از شبکه‌های عصبی عمیق که از داده‌های موجود برای محاسبه رفتارها و خروجی‌های آینده استفاده می‌کند.

## 2-6- داده‌کاوی

داده‌کاوی[[5]](#footnote-6) به مجموعه‌ای از روش‌های قابل اعمال بر پایگاه داده‌های بزرگ و پیچیده به منظور کشف الگوهای پنهان و جالب توجه نهفته در میان داده‌ها، داده‌کاوی گفته می‌شود.

## 2-7- پردازش زبان‌ طبیعی

پردازش زبان‌ طبیعی[[6]](#footnote-7) یکی از شاخه‌های هوش مصنوعی است که به تعاملات بین رایانه و انسان، از طریق زبان طبیعی می‌پردازد. هدف غایی NLP، خواندن، رمزگشایی، فهم و درک زبان انسان با روشی ارزشمند است. بیش‌تر روش‌های پردازش زبان طبیعی برای استخراج و فهم معنای زبان انسانی، مبتنی بر تکنیک‌های یادگیری ماشین است. یک تعامل معمولی، میان انسان و ماشین، شامل این مراحل است: ۱-انسان با ماشین صحبت می‌کند. ۲-ماشین، صدای او را ضبط می‌کند. ۳-صدای انسان به نوشته تبدیل می‌شود. ۴-داده‌های متنی، توسط رایانه، پردازش می‌شود. ۵-داده‌ی متنی، مجددا به صوت تبدیل می‌شود. ۶-ماشین، با پخش صوت، به کاربر پاسخ می‌دهد.

## 2-9- سیستم تشخیص نفوذ

سیستم تشخیص نفوذ[[7]](#footnote-8) سیستمی است که به منظور کشف فعالیت‌های مشکوک، بر ترافیک شبکه نظارت کرده و آن‌ها را تجزیه و تحلیل می‌کند. يك IDS، كليه فعاليت‌هاي موجود بر روي شبكه را تجزيه و تحليل كرده و با استفاده از اطلاعات موجود بر روي پايگاه ‌داده خود تعيين مي‌كند كه فعاليت انجام شده مجاز است يا غيرعادي و غیرمجاز و همچنين تعيين مي‌كند كه آيا اين فعاليت مي‌تواند آسيبي به شبكه شما وارد كند يا خير و در نهايت به شما در مورد اينگونه فعاليت‌ها اطلاع‌رساني مي‌كند. سیستم‌های تشخیص نفوذ تحت شبکه و سیستم‌های تشخیص نفوذ میزبان ازانواع اصلی IDSها می‌باشند. ب صورت كلي سه روش براي شناسايي و تشخيص نفوذ به شبكه وجود دارد: تشخیص امضا، تشخیص ناهنجاری و تشخیص پروتکل ناهنجاری.

## 2-10- سیستم تشخیص نفوذ مبتنی بر امضا

الگوهای نفوذ از پیش ساخته شده (امضا) به صورت قانون نگه داری می‌شوند؛ به طوری که هر الگو انواع متفاوتی از یک نفوذ خاص را در برگرفته و در صورت بروز چنین الگویی در سیستم، وقوع نفوذ اعلام می‌شود. معمولا تشخیص‌دهنده در این روش پایگاه داده‌ای از امضاها یا الگوهای حمله را دارد و سعی می‌کند با بررسی ترافیک شبکه، الگوهای مشابه با الگوهای خود را بیابد.

## 2-11- سیستم تشخیص نفوذ مبتنی بر ناهنجاری

در اين نوع از انواع IDS سيستم با توجه به رفتاري كه در شبكه عادي وجود دارد و ترافيكي كه در اثر يك عمل غيرعادي (ناهنجاری) ايجاد شده است و با توجه به بررسي هاي خود و مقايسه ترافيك طبيعي و غيرعادي تصميم مي‌گيرد كه در مورد اين نوع ترافيك هشدار دهد. به عنوان مثال اگر کاربری به جای یک بار ورود و خروج عادی به سیستم در طول روز، بیست بار این کار را انجام دهد، یا رایانه‌ای خارج از ساعت اداری روشن شود، در حالی که قرار نبوده روشن باشد، به عنوان یک رفتار غیرعادی و یک ناهنجاری در نظر گرفته می‌شوند.

## 2-8- بدافزار

بد افزار[[8]](#footnote-9) یک نوع نرم‌‌افزار مزاحم و نفوذی است که هدف آن آسیب‌ رساندن و نابود کردن سیستم‌های کامپیوتری است. به طور کلی، بدافزار گستره وسیعی از برنامه‌ها یا کد‌هایی که به سیستم‌های گوناگون آسیب می‌رسانند را توصیف می‌کند. کرم، تروجان و ویروس از انواع بدافزارها می‌باشند.

## 2-13- کرم

کرم[[9]](#footnote-10) بخش مستقلی از نرم‌افزار مخرب است که خود را باز تولید کرده و از رایانه به رایانه پخش می‌شود.

## 2-14- حمله DoS

به حملاتی گفته میشود که نفوذگر با ارسال درخواست های بسیار به یک سرور یا کامپیوتر، باعث استفاده بیش از حد از منابع آن مانند پردازنده سرور، بانک اطلاعاتی، پهنای باند و… می‌شود؛ به صورتی که سرور مجازی یا اختصاصی میزبان سایت دچار کندی شده و به دلیل حجم بالای پردازش سیستم دچار وقفه و اختلال و یا حتی قطعی کامل و از دسترس خارج شود. یک حمله‌ DoS معمولاً از طریق غرق کردن Host یا شبکه‌ی هدف با درخواست‌های سرویس غیرقانونی انجام می‌شود. نشانه‌ی این حملات استفاده از یک آدرس IP ساختگی است که باعث می‌شود سرور نتواند کاربر را احراز هویت کند. با پردازش سیل درخواست‌های جعلی، سرور غرق می‌شود، همین امر باعث باعث کند شدن سرور و گاهی اوقات قطعی می‌شود.

## 2-15- حمله DDoS

در صورتی که حملات DoS همزمان از چند سیستم به یک منبع هدف انجام شود و بسته‌های اطلاعاتی از چند سیستم مختلف خارج شود به آن حمله DDoS یا Distributed Denial of Service گفته می‌شود و بالطبع چند IP در این حمله دخیل خواهند بود. درواقع DDoS، نوعی حمله سایبری است که هدف آن بستن وب‌سایت، خدمات یا سایر منابع برای ایجاد اختلال در تجارت است.

## 2-16- شبکه عصبی مصنوعی

شبکه های عصبی مصنوعی[[10]](#footnote-11) یا سیستم های اتصالگر، سیستم‌های محاسبه‌کننده‌ای هستند که از شبکه‌های عصبی زیستی الهام گرفته شده‌اند. این سیستم‌ها، با بررسی مثال‌ها، فعالیت‌ها را یادگیری می‌کنند وعملکرد خود را در انجام فعالیت‌ها به مرور بهبود می‌دهند و عموماً این اتفاق بدون هیچ برنامه‌نویسی مختص به فعالیت انجام می‌شود. یک ANN بر مجموعه‌ای از واحدهای متصل یا گره، به نام نورون‌های مصنوعی، مبتنی است.

## 2-17- شبکه عصبی عمیق

شبکه عصبی عمیق[[11]](#footnote-12) یک شبکه عصبی مصنوعی با چندین لایه پنهان بین لایه ورودی و لایه خروجی یک است. DNN ها معمولاً شبکه های پیشخور[[12]](#footnote-13) هستند که در آنها داده‌ها از ورودی به لایه خروجی بدون حلقه بازگشتی جریان می یابد. DNN نقشه نورون مجازی را تولید می کند و مقادیر عددی دلخواه یا "وزن" را به اتصالات بین نورون ها اختصاص می دهد. ورودی و وزن‌ها ضرب می‌شوند و مقدار بین 1 و 0 برمی‌گردد. اگر شبکه دنباله‌ای را به درستی شناسایی نکرده باشد، الگوریتم وزن‌ها را تغییر می‌دهد. این به الگوریتم اجازه می دهد تا آن پارامترها را دستکاری کند تا زمانی که دستکاری ریاضی مناسب برای تکمیل پردازش داده ها تصمیم گیری شود[1].

## 2-18- شبکه عصبی بازگشتی

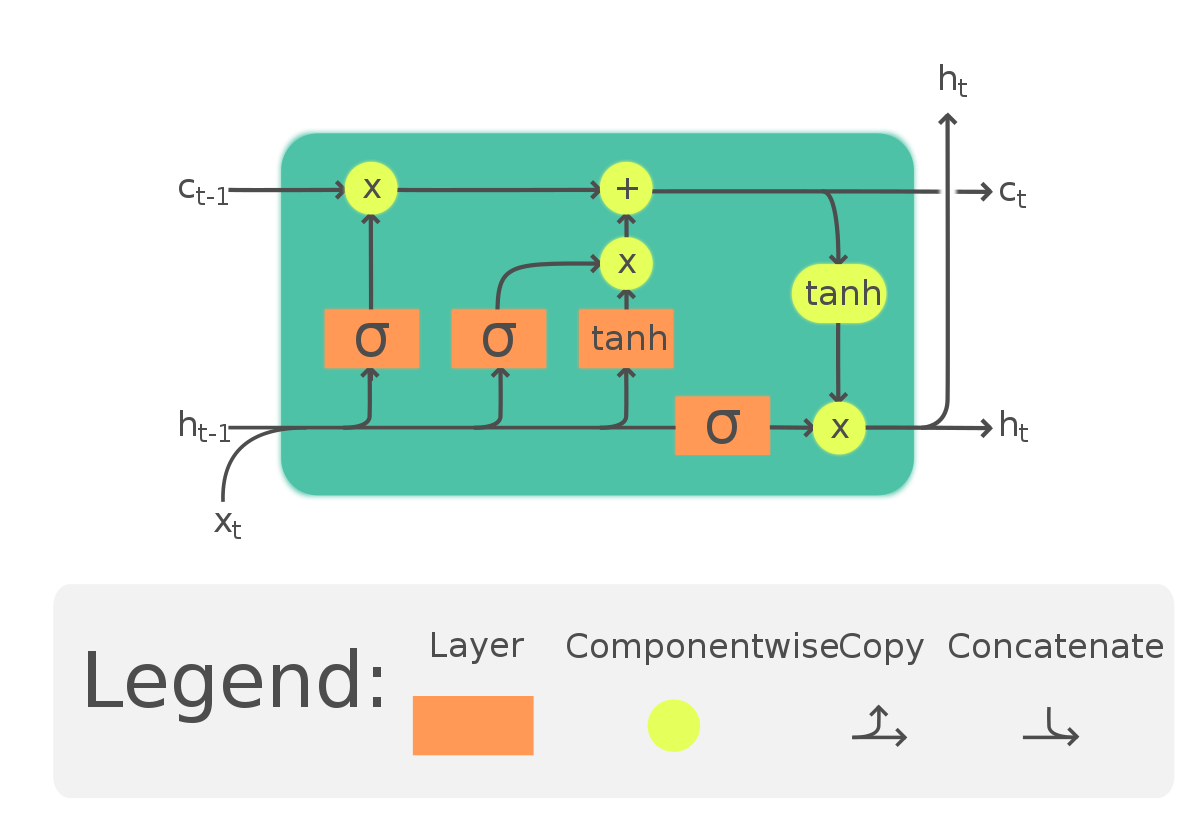
شبکه عصبی بازگشتی[[13]](#footnote-14)، تعمیمی از شبکه عصبی پیشخور است که دارای یک حافظه داخلی بوده و می‌تواند تعداد n داده را به صورت متوالی مشاهده کند. این شبکه، ماهیتی بازگشتی دارد، زیرا برای هر داده ورودی، تابع یکسانی را اجرا می‌کند در حالی که خروجی ورودی فعلی، به محاسبه قبلی بستگی دارد. پس از تولید خروجی، این خروجی کپی شده و مجدداً به شبکه بازگشتی ارسال می‌شود. برای تصمیم گیری، ورودی فعلی و خروجی که از ورودی قبلی آموزش دیده است، در نظر گرفته می‌شود.

## 2-19- شبکه مولد متخاصم

شبکه مولد متخاصم[[14]](#footnote-15) رویکردی برای مدل‌سازی مولد با استفاده از روش‌های یادگیری عمیق مانند شبکه‌های عصبی پیچشی است. مدل‌سازی مولد یک فعالیت نظارت نشده در یادگیری ماشین است که شامل اکتشاف خودکار و یادگیری قواعد یا الگوهای موجود در داده‌های ورودی می‌شود. این کار به صورتی انجام می‌شود که از مدل می‌توان برای تولید یا خروجی دادن نمونه‌های جدیدی که به شکل قابل باوری از مجموعه داده اصلی قابل برگرفته شدن هستند استفاده کرد.

## 2-20- حافظه کوتاه مدت طولانی

حافظه کوتاه مدت طولانی[[15]](#footnote-16) یک شبکه‌ی عصبی بازگشتی کلاسی از شبکه‌های عصبی مصنوعی هستند که در آن اتصالات مابین گره‌هایی از یک گراف جهت‌دار در امتداد یک دنباله‌ی زمانی می‌باشند و سبب می‌شود تا الگوریتم بتواند رفتار پویای موقتی را به نمایش بگذارد. برخلاف شبکه‌های عصبی رو به جلو، شبکه‌های عصبی بازگشتی می‌توانند از وضعیت درونی خود برای پردازش دنباله‌ی ورودی‌ها استفاده کنند. شبکه‌های عصبی بازگشتی، که قوی‌ترین و شناخته‌شده‌ترین آنها حافظه کوتاه مدت بلندمدت هستند، نوعی شبکه عصبی مصنوعی را منعکس می‌کنند که روندهای توالی داده‌ها را تشخیص می‌دهد.



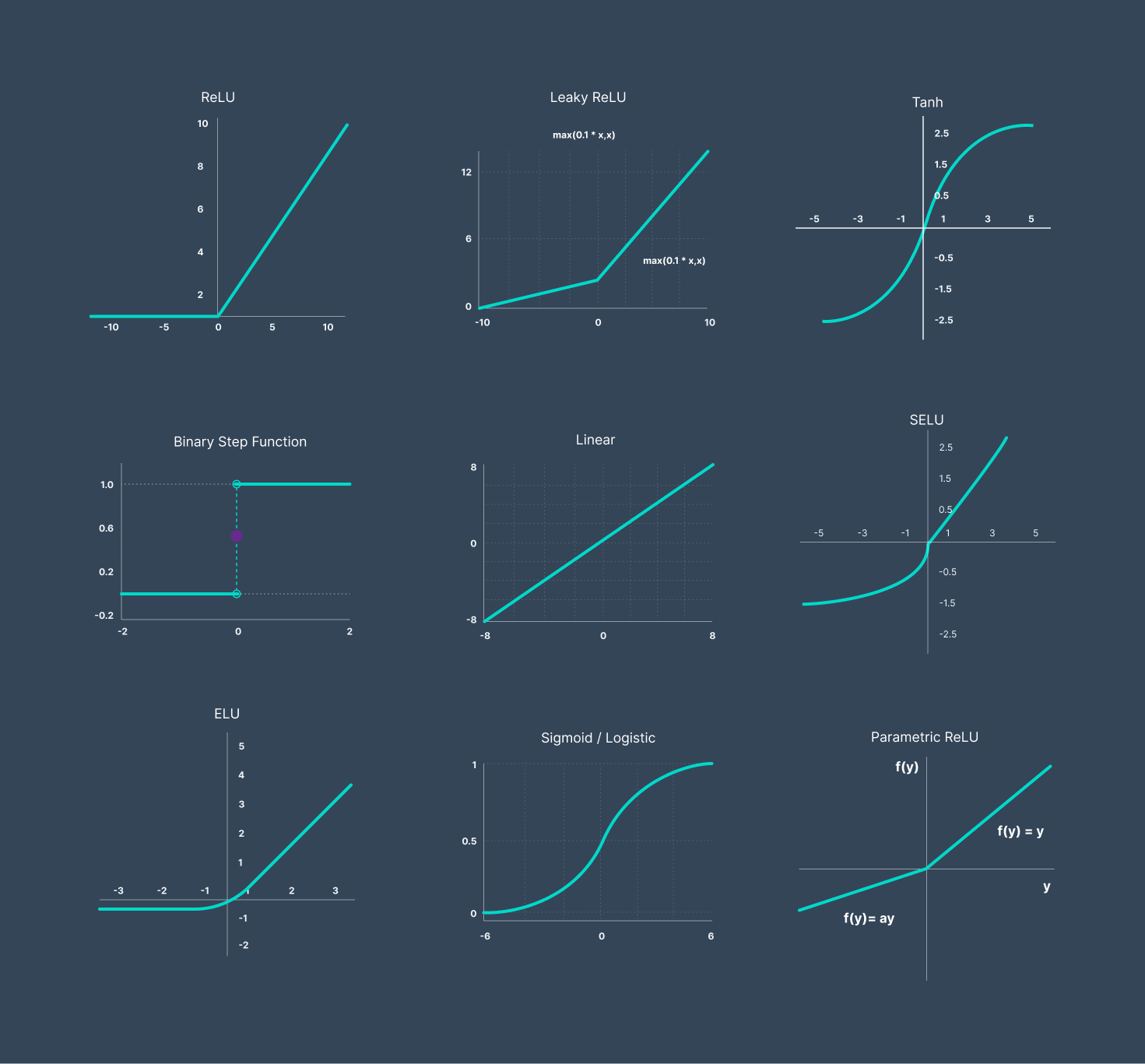
**شکل 9- حافظه کوتاه مدت طولانی**

## 2-24- نرخ یادگیری

نرخ یادگیری[[16]](#footnote-17) به این معنی است که سرعت و دقت یادگیری مدل چقدر باشد. هر چه این میزان زیاد باشد، سرعت یادگیری بالاست اما دقت آن کمتر است و هر چه کمتر باشد سرعت یادگیری کمتر ولی دقت آن بالاتر است.

## 2-25- تابع فعال ساز

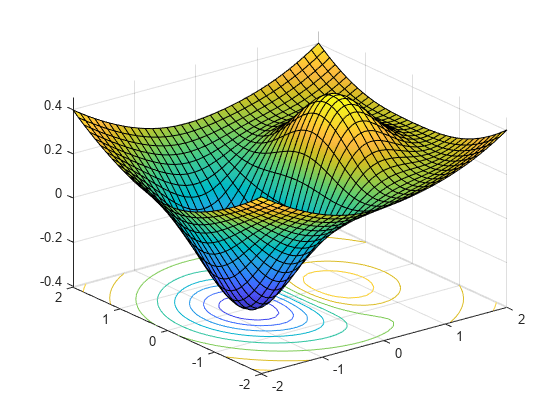
تابع فعال ساز[[17]](#footnote-18) با محاسبه مجموع وزنی و اضافه کردن بایاس به آن تصمیم می گیرد که آیا نورون باید فعال شود یا خیر.این بدان معناست که تصمیم خواهد گرفت که آیا ورود نورون به شبکه در فرآیند پیش‌بینی موثراست یا خیر. هدف از تابع فعال ساز، وارد کردن غیر خطی بودن به خروجی یک نورون است .انواع مختلفی از توابع فعال ساز وجود دارد که در این پژوهش ما از tanh، sigmoid، reluو... در یادگیری مدل استفاده کردیم.



**شکل 9- توابع فعال ساز**

## 2-25- تابع بهینه ساز

تابع بهینه ساز فرآیندی برای جستجوی پارامترهایی است که توابع ما را به حداقل یا حداکثر می رساند. وقتی مدل یادگیری ماشین را آموزش می دهیم، از بهینه سازی استفاده می کنیم. ما یک معیار خاص را انتخاب می کنیم مانند دقت یا صحت که نشان می دهد مدل ما چقدر خوب می تواند مسئله را حل کند.



**شکل 9- تابع بهینه ساز**

## 2-26- تابع هدررفت

تابع هدررفت[[18]](#footnote-19) میزان عملکرد یک مدل شبکه عصبی در انجام یک کار خاص را اندازه گیری می کند. ما باید مقدار تابع تلفات را به حداقل برسانیم تا شبکه عصبی را بهتر کنیم.

## 2-12- تروجان

تروجان[[19]](#footnote-20) بدافزاری است که ظاهری موجه دارد؛ اما با نصب بدون اجازه برنامه‌های مخرب، سیستم کاربر را آلوده می‌کند. تروجان نمی تواند خودش را تولید کند اما خود را به عنوان چیزی که کاربر می‌خواهد نشان داده و او را فریب می دهد تا آن را فعال کرده و گسترش یابد.

## 2-21- ماشه تروجان

ماشه تروجان[[20]](#footnote-21) را در پاسخ به برخی از ورودی های خاص فعال میشوند. ماشه‌ها می توانند ترکیبی یا متوالی باشند. آنها همچنین می توانند مبتنی بر زمان باشند، در این صورت پس از اینکه مدل برای مدت زمان مشخصی کار کرد فعال می شوند.

## 3-۱9 نتیجه گیری

در این فصل اصطلاحاتی که در فصول آینده از آن‌ها استفاده میکنیم را توضیح دادیم که در ادامه فهم کلمات تخصصی برای خواننده این پژوهش مشکل نباشد.

# فصل 3- کارهای پیشین

## 3-۱- سیستم تشخیص نفوذ

تشخیص نفوذ، شناسایی اقدامات و رویدادهایی است که منجر به خطر انداختن محرمانه بودن، در دسترس بودن یا یکپارچگی منابع می‌شود. سیستم ساخته شده برای انجام این عملیات IDS[[21]](#footnote-22) نامیده می‌شود. هدف اصلی سیستم تشخیص نفوذ (IDS) در سیستم‌های کامپیوتری، شناسایی استفاده غیرمجاز یا سوءاستفاده از سیستم کامپیوتری توسط افراد داخلی سیستم و یا نفوذگران خارجی است [6].

## 3-۲- کلاس‌های حملات کامپیوتری

در جدول 3-۱ انواع حملات و توضیحات مدنظر آن‌ها بیان شده است.

|  |  |
| --- | --- |
| توضیحات | نوع حمله |
| مرزهای بافر و ناحیه حافظه را بازنویسی می‌کند. | حملات سرریز بافر |
| خود را در میزبان محلی یا از طریق شبکه بازتولید می‌کند. | کرم (worm) |
| برنامه‌ها جذاب و واقعی به نظر می‌رسند، اما کدهای مخرب درون آن‌ها تعبیه شده است. | تروجان (Trojan) |
| یک رویداد امنیتی برای اختلال در خدمات شبکه. با بازنشانی اجباری در رایانه‌های هدف شروع می‌شود. کاربران به دلیل در دسترس نبودن سرویس دیگر نمی‌توانند به سیستم متصل شوند. | انکار سرویس DoS)) |
| اسکریپت‌های مهاجم از اسکریپت‌های CGIبرای ایجاد حمله با ارسال ورود‌‌‌ی‌های نامشروع به وب سرور استفاده می‌کند. | رابط دروازه مشترک (CGI) |
| به اندازه محدود NIDS برای مدیریت بارهای ترافیکی عظیم و بررسی نفوذهای احتمالی حمله می‌کند. اگر یک مجرم سایبری بتواند باعث ازدحام در شبکه‌ها شود، NIDS مشغول تجزیه و تحلیل ترافیک خواهد بود. | سیل ترافیک |
| هدف آن شکستن رمز عبور در مدت زمان کوتاه است و با یک توالی از خطاها در ورود به سیستم متوجه می‌شود. | حمله رمز عبور |
| اطلاعات را جمع‌آوری می‌کند یا با شنود یا جست‌وجو نقاط ضعفی را در رایانه‌ها یا شبکه‌ها پیدا می‌کند. | جمع آوری اطلاعات |
| مجرم سایبری در ابتدا به عنوان یک کاربر عادی دسترسی پیدا می‌کند و سپس به یک کاربر فوق‌العاده ارتقا می‌یابد که ممکن است منجر به سوء استفاده از چندین آسیب‌پذیری سیستم شود. | حمله کاربر به ریشه (U2R) |
| مجرم سایبری با اتصال به شبکه بدون داشتن حساب کاربری در سیستم، بسته‌ها را به یک سیستم راه دور ارسال می‌کند. | حمله از راه دور به محلی (R2L) |
| شناسایی آدرس‌های IP معتبر با اسکن شبکه برای جمع آوری بسته‌های داده میزبان. | پویشگر |

جدول ۱-3 کلاس‌های حملات کامپیوتری [10]

## 3-۳- طبقه‌بندی IDSها بر اساس نوع تحلیل

بسته به مکانیسم جمع‌آوری داده، یک IDS ممکن است مبتنی بر میزبان یا مبتنی بر شبکه باشد. IDS‌های مبتنی بر میزبان بر روی مسیرهای حسابرسی سیستم‌عامل، لاگ‌های سیستم و اپلیکیشن‌ها و یا ممیزی داده‌های تولید شده توسط ماژول‌های کرنل عمل می‌کند اما IDS‌های مبتنی بر شبکه بر روی بسته‌های گرفته شده از ترافیک شبکه عمل می‌کنند.

شکل ‏3‑۱ طبقه‌بندی IDSها [6]

## 3-۴- سیستم تشخیص نفوذ مبتنی بر میزبان (HIDS[[22]](#footnote-23))

HIDS برای نگاه کردن به کل یک سیستم طراحی شده است و حاوی اطلاعات کل سیستم‌عامل می‌باشد.

برخی از نرم‌افزارهایی که معمولا برای HIDS استفاده می‌شوند [20]OSSEC، [21]Tripwire هستند.

معایب HIDS به شرح زیر است:

* تمام حملات توسط HIDS جلوگیری نمی‌شود.
* یک نهاد خارجی برای رسیدگی به خرابی‌ها مورد نیاز است. (یک فرد خبره باید روی آن کنترل داشته باشد)
* تشخیص اشتباه بسیار زیاد است.

## 3-۵- سیستم‌های تشخیص نفوذ مبتنی بر شبکه (NIDS[[23]](#footnote-24))

سیستم‌های تشخیص نفوذ مبتنی بر شبکه می‌توانند سخت‌افزاری یا نرم‌افزاری باشند. رایج‌ترین نرم‌افزار موجود برای تشخیص نفوذ شبکه Snort است.

معایب NIDS به شرح زیر است:

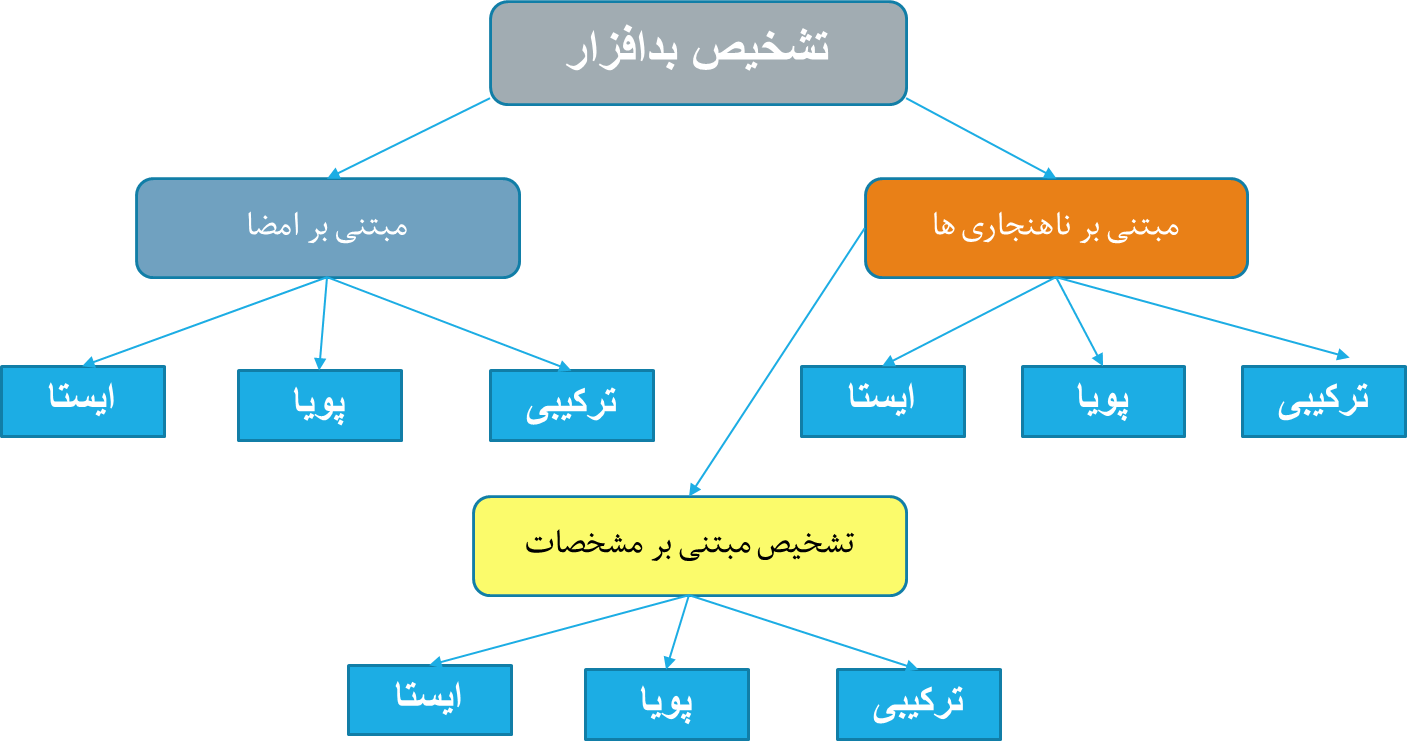
* از تمامی حملات جلوگیری نمی‌کند.
* یک نهاد خارجی برای رسیدگی به خرابی‌ها مورد نیاز است. (یک فرد خبره باید روی آن کنترل داشته باشد)
* فرآیند رمزگذاری بسته‌ها توسط سیستم پشتیبانی نمی‌شود.
* بسته‌های IP اغلب به راحتی جعل می‌شوند.
* فراوانی موارد تشخیص اشتباه بسیار زیاد است و مقاومت در برابر حملات مبتنی بر پروتکل اغلب کم است.

## 3-۶- مزایای IDSها

* پروتکل مورد استفاده برای تبادل داده به راحتی قابل درک است.
* حملات به سیستم را می‌توان کمی و همچنین واجد شرایط ارزیابی کرد.
* حفظ سازگاری با قوانین آسان است.
* تقویت سیستم می‌تواند به راحتی گنجانیده شود.

علاوه بر این، یک IDS ممکن است بر اساس تکنیک‌های تشخیص به شرح زیر طبقه‌بندی شود: [17,1]

* تشخیص مبتنی بر امضا برای وقوع امضاها یا توالی‌های از پیش تعریف شده که نشان‌دهنده نفوذ به کار می‌باشد. این تکنیک عموما در تشخیص حملات ناشناخته قبلی خوب عمل نمی‌کند.
* تشخیص مبتنی بر ناهنجاری نمونه‌ای از رفتار عادی را تعریف می‌کند و هرگونه انحراف از آن نمونه را به عنوان یک نفوذ طبقه‎بندی می‌کند. این تکنیک ممکن است حملات ناشناخته قبلی را شناسایی کند، اما اغلب نرخ بالایی از مثبت کاذب را نشان می‌دهد.
* تشخیص مبتنی بر مشخصات که مجموعه‌ای از محدودیت‌های توصیف عملکرد صحیح یک برنامه یا پروتکل را تعریف کرده و اجرای برنامه را با توجه به آن محدودیت‌های تعریف‌شده نظارت می‌کند. این تکنیک ممکن است قابلیت شناسایی حملات ناشناخته قبلی را فراهم کند، اما نرخ تشخیص اشتباه پایینی را نشان می‌دهد.



شکل 3‑۲ تشخیص بدافزار [2]

تشخیص نفوذ مبتنی بر امضا (SID[[24]](#footnote-25)) به طور قابل اعتمادی روی نفوذهای شناخته شده کار می‌کند، اما در تشخیص نفوذهای جدید ناتوان است. علاوه بر این، چندین مشکل عملی نیز دارد: به‌ روزرسانی امضاها، تشخیص تغییرات نفوذ، و هشدارهای نادرست زیاد. شناسایی تهدیدهای جدید یا ناشناس تا حدودی ناکارآمد و مجموعه امضا باید دائماً جهت شناسایی تهدید جدید به‌روز شود. برای این منظور، باید یک پایگاه‌داده امضای مربوط به حملات شناخته شده وجود داشته باشد.

تشخیص نفوذ مبتنی بر ناهنجاری (AID[[25]](#footnote-26)) به تمرکز جستجوی مجدد تبدیل شده است؛ زیرا جایگزین مفیدی برای SID است که قادر به تشخیص نفوذهای جدید است اما AIDنمی‌تواند نفوذها را شناسایی کند، از مشکلات دیگر آن می‌توان به نرخ هشدار نادرست بالاتر آن اشاره کرد.

تشخیص نفوذ مبتنی بر امضا (SID) فقط از دانش مربوط به نفوذهای شناخته شده استفاده می‌کند. برای AID، دانش در مورد رفتارهای عادی منابع محاسباتی است.

تشخیص مبتنی بر ناهنجاری تلاش می‌کند تا رفتار «عادی» سیستمی را که قرار است محافظت شود را تخمین بزند و هر زمان که انحراف بین یک مشاهده معین در یک لحظه و رفتار عادی از یک آستانه از پیش تعریف‌شده فراتر رود، یک هشدار ناهنجاری تولید کند. برای شناسایی تهدیدهای جدید موثر است، اما هشدارهای هشدار نادرست یا تشخیص اشتباه زیادی تولید می‌کند.

## 3-۷- طبقه‌بندی روش‌های تشخیص مبتنی بر ناهنجاری

شکل ۲-۳ یک دسته‌بندی در روش تشخیص مبتنی بر ناهنجاری را نشان می‌دهد.

شکل ‏3‑۳ طبقه‌بندی روش‌های تشخیص مبتنی بر ناهنجاری[10]

## 3-۸- تکنیک‌های مبتنی بر آمار

یک IDS مبتنی بر آمار، یک مدل توزیع برای رفتار عادی ایجاد می‌کند، سپس رویدادهای با احتمال کم را شناسایی کرده و آنها را به عنوان نفوذهای بالقوه علامت‌گذاری می‌کند. AIDS آماری برای شناسایی هر نوع تفاوت در رفتار فعلی از رفتار عادی استفاده می‌شود.

* **تک متغیره**‌: این تکنیک زمانی استفاده می‌شود که یک شاخص نرمال آماری تنها برای یک معیار از رفتارها در سیستم‌های کامپیوتری ایجاد شود. IDS تک متغیره به دنبال ناهنجاری‌ها در هر معیار جداگانه می‌گردد.
* **چند متغیره**‌: بر اساس روابط بین دو یا چند معیار به منظور درک روابط بین متغیرها است. اگر داده‌های تجربی نشان دهد که می‌توان از ترکیب معیارهای همبسته به جای تجزیه و تحلیل جداگانه آنها به طبقه‌بندی بهتری دست یافت، این مدل ارزشمند خواهد بود.
* **مدل سری زمانی**‌: سری زمانی مجموعه‌ای از مشاهدات است که در یک بازه زمانی معین انجام می‌شود. یک مشاهده جدید غیر طبیعی تلقی می‌شود اگر احتمال وقوع آن در آن زمان خیلی کم باشد.

## 3-۹- تکنیک‌های مبتنی بر دانش

این تکنیک‌ها به عنوان روش سیستم خبره نیز نامیده می‌شوند. این رویکرد مستلزم ایجاد یک پایگاه دانش است که نشان‌دهنده ترافیک قانونی باشد. اقداماتی که با این مشخصات استاندارد متفاوت است به عنوان یک نفوذ تلقی می‌شود. مزیت اصلی این تکنیک، توانایی کاهش هشدارهای مثبت کاذب است؛ زیرا سیستم از همه رفتارهای عادی آگاهی دارد با این حال، این نوع IDS به یک بروزرسانی منظم در دانش برای رفتار عادی مورد انتظار نیاز دارد که یک کار وقت‌گیر خواهد بود زیرا جمع آوری اطلاعات در مورد همه رفتارها عادی بسیار دشوار است.

* **ماشین حالت متناهی (FSM[[26]](#footnote-27))**‌: FSM یک مدل محاسباتی است که برای نمایش و کنترل جریان اجرا استفاده می‌شود. مدل در قالب حالت‌ها، انتقال‌ها و فعالیت‌ها نشان داده می‌شود. یک حالت، داده‌های سابقه را بررسی می‌کند. هر گونه تغییر در ورودی یادداشت شده و بر اساس تغییرات شناسایی شده انتقال رخ می‌دهد.
* **زبان توصیف**‌: ترکیبی از قوانین را تعریف می‌کند که می‌تواند برای مشخص کردن ویژگی‌های یک حمله تعریف شده استفاده شود. قوانین را می‌توان با زبان‌های توصیفی مانند N-grammars و UML ساخت.
* **سیستم خبره**‌: یک سیستم خبره شامل تعدادی قانون است که حملات را تعریف می‌کند. در یک سیستم خبره، قوانین معمولا به صورت دستی توسط یک مهندس دانش که با همکاری یک متخصص کار می‌کند، تعریف می شود.

## 3-۱۰- تکنیک‌های مبتنی بر یادگیری ماشین

یادگیری ماشین فرآیند استخراج دانش از مقادیر زیادی داده است. مدل‌های یادگیری ماشینی شامل مجموعه‌ای از قوانین، روش‌ها پیچیده هستند که می‌توانند برای یافتن الگوهای داده‌های جالب یا تشخیص یا پیش‌بینی رفتار اعمال شوند. تکنیک‌های یادگیری ماشین به طور گسترده در زمینه AIDS به کار گرفته شده است. چندین الگوریتم و تکنیک مانند خوشه‌بندی، شبکه‌های عصبی، قوانین ارتباط، درخت‌های تصمیم، الگوریتم‌های ژنتیک و روش‌های نزدیک‌ترین همسایه برای کشف دانش از مجموعه داده‌های نفوذ به کار گرفته شده‌اند.

## 3-۱۱- معماری‌هایی برای تشخیص نفوذ

**معماری IDS مستقل**[15]

در یک معماری IDS مستقل، هر میزبان یک IDS را اجرا می‌کند که به طور مستقل حملات را شناسایی می‌کند. از آنجایی که IDS‌های مستقل با سیستم‌های دیگر همکاری نمی‌کنند یا اطلاعاتی را به اشتراک نمی گذارند، تمام تصمیمات تشخیص نفوذ بر اساس اطلاعات در دسترس بر عهده هر گره می‌باشد. با اینکه اثربخشی این راه حل محدود است، اما میتواند در محیط‌هایی مناسب باشد که همه گره‌ها قادر به اجرای IDS یا نصب IDS نیستند.

**معماری IDS توزیع‌شده** [15]

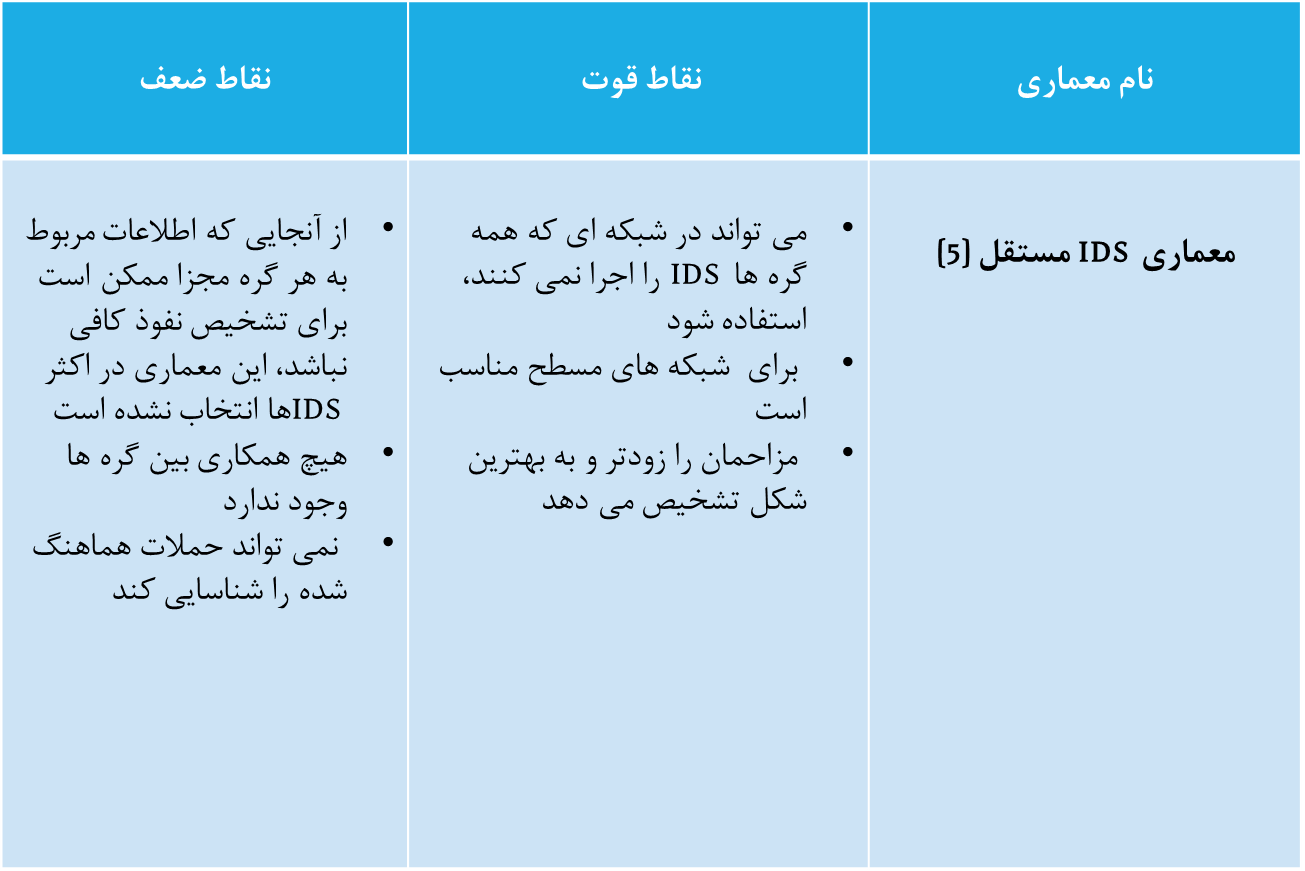
در معماری پیشنهاد شده برای این محیط، عوامل IDS مستقر در هر گره به‌طور مستقل تصمیم‌های تشخیص نفوذ محلی را اتخاذ می‌کنند، اما به طور مشترک در تشخیص نفوذ سراسری مشارکت می‌کنند. در این معماری، اگر یک گره نفوذی را با شواهد ضعیف یا غیرقطعی تشخیص دهد، می‌تواند یک روش تشخیص نفوذ سراسری مشترک را آغاز کند، یا اگر یک گره به صورت محلی یک نفوذ را با شواهد قوی شناسایی کند، می‌تواند به طور مستقل حمله به شبکه را تعیین کند.



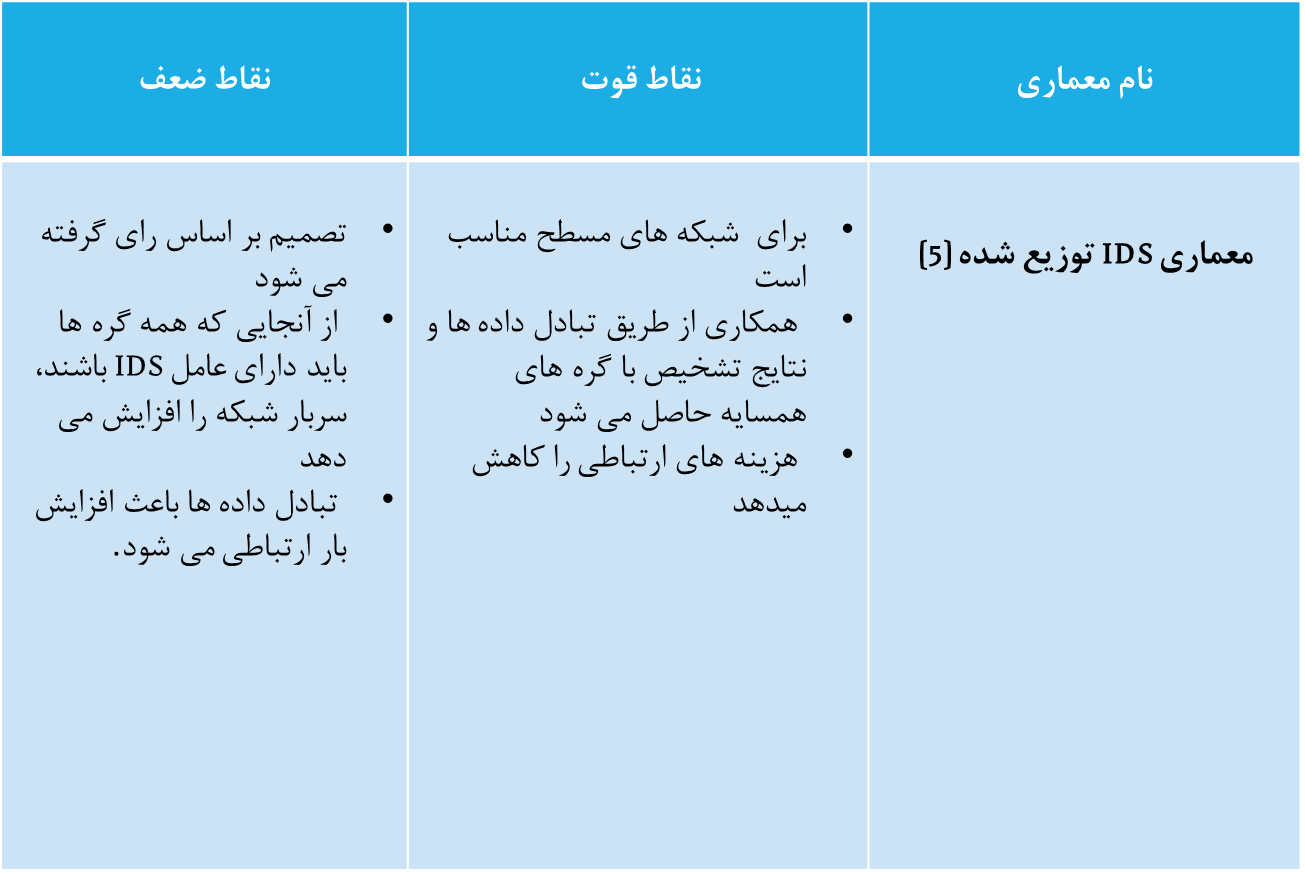
شکل ‏3‑۶ IDS توزیع‌شده [15]

**معماری IDS سلسله مراتبی**[15]

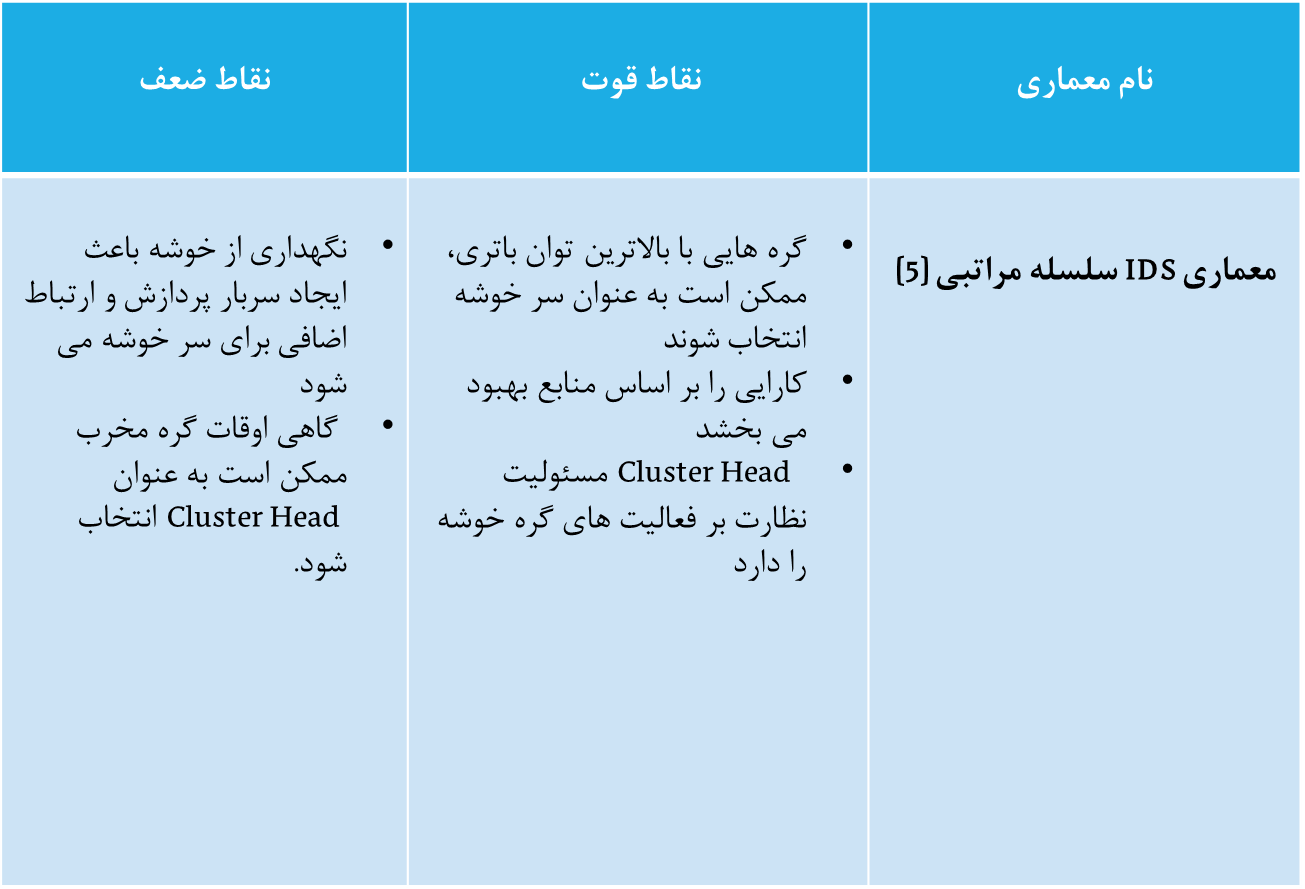
معماری‌های IDS سلسله مراتبی برای شبکه‌های چندلایه بی‌سیم پیشنهاد شده‌اند. در یک شبکه چند لایه، گره‌های سر خوشه، مسیریابی را برای خوشه متمرکز می‌کنند و ممکن است از مکانیسم‌های امنیتی اضافی پشتیبانی کنند. برای مثال ممکن است مخرب بودن یک گره را تشخیص داده و این پیغام را به گره‌های بالاتر منتقل کنند. در معماری‌های IDS سلسله مراتبی، گره‌ها به خوشه‌ها یا مناطق، گروه‌بندی می‌شوند. یکی از گره‌های یک منطقه/خوشه مسئول تشخیص نفوذ است. تشخیص نفوذ به صورت توزیع‌شده و با همکاری سایر خوشه‌ها/منطقه‌ها انجام می شود.



شکل ‏3‑۷ معماری IDS مستقل[15]



شکل ‏3‑۸ معماری IDS توزیع‌شده[15]



شکل ‏3‑۹ معماری IDS سلسله مراتبی[15]

## 3-۱۲- مدیریت اعتماد توزیع‌شده

سرویس مدیریت اعتماد مسئول حفظ روابط بین گره‌ها در شبکه است. این سرویس رفتار نادرست گره‌ها را کاهش می‌دهد همکاری را تقویت می‌کند. اعتماد به یک گره با خوش نامی یا شهرت آن مرتبط است. سه سطح اعتماد وجود دارد و ما از یک مقدار اعتماد T برای نشان دادن قابل اعتماد بودن یک گره استفاده می‌کنیم [17,7].

• قابل اعتماد، با T = 1

• غیر قابل اعتماد، با T = -1

• نامشخص‌، با T = 0

یک گره غیرقابل اعتماد یک گره بد رفتار است و باید در فرآیند ارزیابی IDS توزیع‌شده از آن اجتناب شود. یک گره با قابلیت اطمینان نامشخص معمولاً یک گره جدید در شبکه است و توجه ویژه‌ای باید در فرآیند ارزیابی IDS وجود داشته باشد.

هر گره یک جدول خوش نامی یا شهرت را نگه می دارد که یک مقدار سابقه را با هر یک از همسایگان خود مرتبط می‌کند. ارزش شهرت یک گره همسایه به صورت سراسری توزیع نمی شود و به صورت محلی ذخیره می شود. ارزش‌های شهرت فقط در صورت درخواست توسط گره‌های دیگر به اشتراک گذاشته می شود.

مقادیر شهرت (R) بین محدوده 0≤R≤1 است و یک آستانه Rt وجود دارد،

R ≥ Rtبرای قابل اعتماد و

R<Rtبرای قابل اعتماد.

برای یک گره جدید N با ارزش شهرت R و ارزش اعتماد T،

* T = 1 اگر R ≥ Rt
* T=-1 اگرR<Rt
* T = 0 اگر R <0

مقادیر شهرت به رفتارهای گره بستگی دارد. اگر یک گره یک پیام هشدار IDS را پخش کند، آنگاه گره‌های همسایه را دائم تحت نظر میگیرد‌. اگر آن گره پیام هشدار IDS را بازپخش کند، گره مبدأ ارزش شهرت آن گره را ارتقا می‌دهد. در غیر این صورت، ارزش شهرت کاهش می یابد. اگر پیام بازپخش شده اصلاح شود، مقدار اعتماد گره‌ها در حالت غیرقابل اعتماد خواهد بود. R نسبت تعداد کل پیام‌های ارسال شده به تعداد کل پیام‌های ارسال شده است.

هر گره، گره‌های همسایه را ردیابی می‌کند و ارزش‌های شهرت را مستقیماً تعیین می‌کند. اگر یک گره نیاز به پرس و جو از یک گره خاص داشته باشد که فراتر از محدوده رادیویی بی سیم است، ارزش شهرت را برای تمام گره‌های مورد اعتماد در شبکه درخواست می‌کند‌. میانگین پاسخ‌ها ارزش شهرت را برای گره درخواستی تعیین می‌کند.

عامل دیگری برای یک گره که بر سطح اعتماد آن تأثیر می گذارد، درستی پیام هشدار IDS است.

## 3-۱۳- جدیدترین و برترین نرم‌افزارهای IDS [9]

نرم‌افزارهای زیادی برای تشخیص نفوذ وجود دارد که ما به تعدادی از آن‌ها اشاره می‌کنیم.

* SolarWinds Security Event Manager
* Snort
* Suricata
* Trend Micro TippingPoint
* Cisco Stealthwatch
* Darktrace Enterprise Immune System
* OSSEC
* Zeek
* Samhain

**SolarWinds Security Event Manager (SEM)‌:**

یک سیستم تشخیص نفوذ مبتنی بر میزبان است و اتفاقات را log می‌کند. دستگاه‌های USB متصل و فایل‌های ورودی و خروجی شبکه را برای یکپارچگی و تغییرات مخرب بررسی می‌کند. همچنین به طور منظم فایل‌های سیستم را بررسی می‌کند تا مطمئن شود فایل‌ها به روش‌های غیرعادی تغییر، حذف یا جابه‌جا نشده‌اند. این ابزار شامل فرآیندهای جمع آوری داده برای یافتن اطلاعات مرتبط و به روز در بات نت‌های شناخته شده و سایر عوامل مخرب برای شناسایی مزاحمان است. به این معنی که می‌تواند پاسخ‌های خودکار را برای مزاحمان شناخته شده پیکربندی کند، بدون نیاز به استفاده از اسکریپت‌های سفارشی.

**Snort‌:**

Snort یک ابزار شناخته شده و در حال حاضر پیشرو در صنعت است که برای شناسایی بسته‌ها، لاگ کردن و تشخیص نفوذ استفاده می شود. وقتی از Snort استفاده می‌کنند، قوانینی را در ترافیک شبکه اعمال می‌کنند. می‌توان برخی از این قوانین را که «سیاست‌های پایه» نام دارد را از وب‌سایت Snort دانلود کرد، یا یاد گرفت که چگونه از Snort استفاده کرد.

**Suricata‌:**

یک سیستم تشخیص نفوذ مبتنی بر شبکه است و از یک زبان امضای کامل برای تعیین تهدیدهای شناخته شده و نوع رفتاری که احتمالاً از یک مزاحم ناشی می شود استفاده می‌کند. در کنار موتور NIDS، یک موتور جلوگیری از نفوذ (NIPS) نیز برای ایجاد هشدارها، فیلترها و محدودیت‌های نرخ وجود دارد. با رمزگشاهای(decoders) پروتکل کار می‌کند، مانند پشتیبانی از رمزگشایی بسته با IPv4، IPv6، TCP، UDP و... و رمزگشایی پروتکل در لایه برنامه رمزگشایی برای HTTP، SSL، TLS، SMB، DCERPC، SMTP، FTP، SSH، DNS و غیره. می‌تواند کد یا درخواست‌های نادرست را با استفاده از کلمات کلیدی پروتکل، پروفایل قوانین، تطبیق فایل، تطبیق الگو با یادگیری ماشین و گزینه‌های تنظیم گسترده شناسایی کند.

**Trend Micro TippingPoint‌:**

یک سیستم تشخیص نفوذ با تشخیص و پاسخ خودکار تهدید است. هم قابلیت IDS و هم IPS دارد. با برنامه‌های دیگر ادغام می شود تا لیست آسیب‌پذیری‌ها و ارزیابی‌های شناخته شده خود را به روز نگه دارد، از جمله برنامه‌های Rapid7، Qualys و Tenable. سیستم TippingPoint انواع ترافیک از جمله ترافیک نامتقارن را بررسی می‌کند و سیاست‌های امنیتی را برای اطمینان از حفاظت جامع اعمال می‌کند. شامل قابلیت وصله مجازی، پوشش پیشگیرانه بین زمانی که یک آسیب‌پذیری کشف می‌شود و یک وصله برای آن بوجود می‌آید، می‌شود. از فیلترهای مبتنی بر آسیب‌پذیری برای ایجاد یک مانع بین شبکه شما و هر مهاجمی که ممکن است سعی در سوء استفاده از این آسیب‌پذیری داشته باشد، استفاده می‌کند.

**Cisco Stealthwatch‌:**

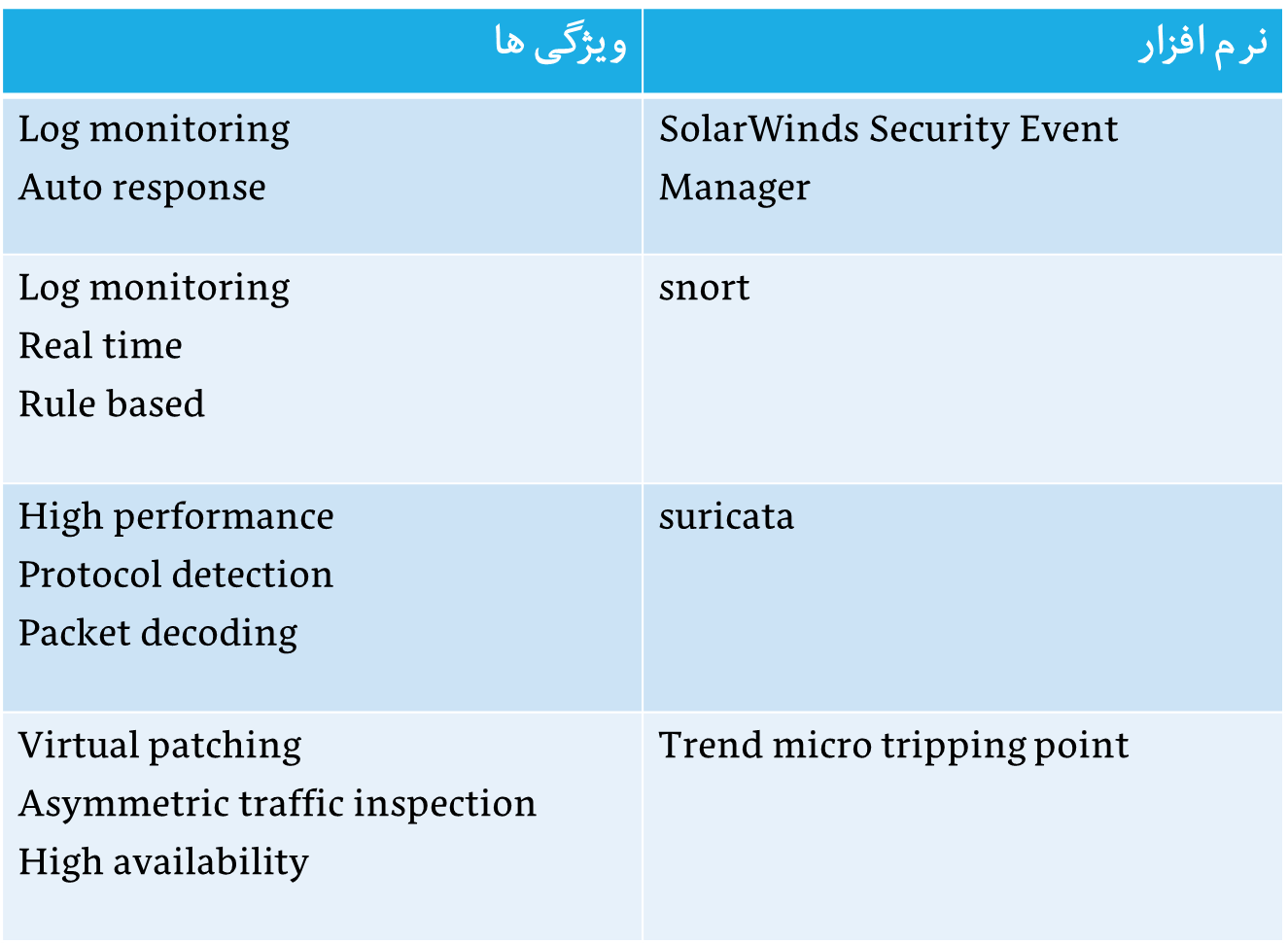
یک HIDS است که برای استفاده سازمانی در نظر گرفته شده است. از تجزیه و تحلیل و یادگیری ماشین برای شناسایی تهدیدها قبل از تبدیل شدن به یک مشکل از طریق تشخیص ناهنجاری استفاده می‌کند. و به سازمان‌ها اجازه می‌دهد تا بدافزارهای ناشناخته یا رمزگذاری شده، تهدیدات داخلی و نقض سیاست‌ها را شناسایی کنند. همچنین می‌تواند بدافزار را در ترافیک رمزگذاری شده بدون نیاز به رمزگشایی ترافیک شناسایی کند.

**Darktrace Enterprise Immune System‌:**

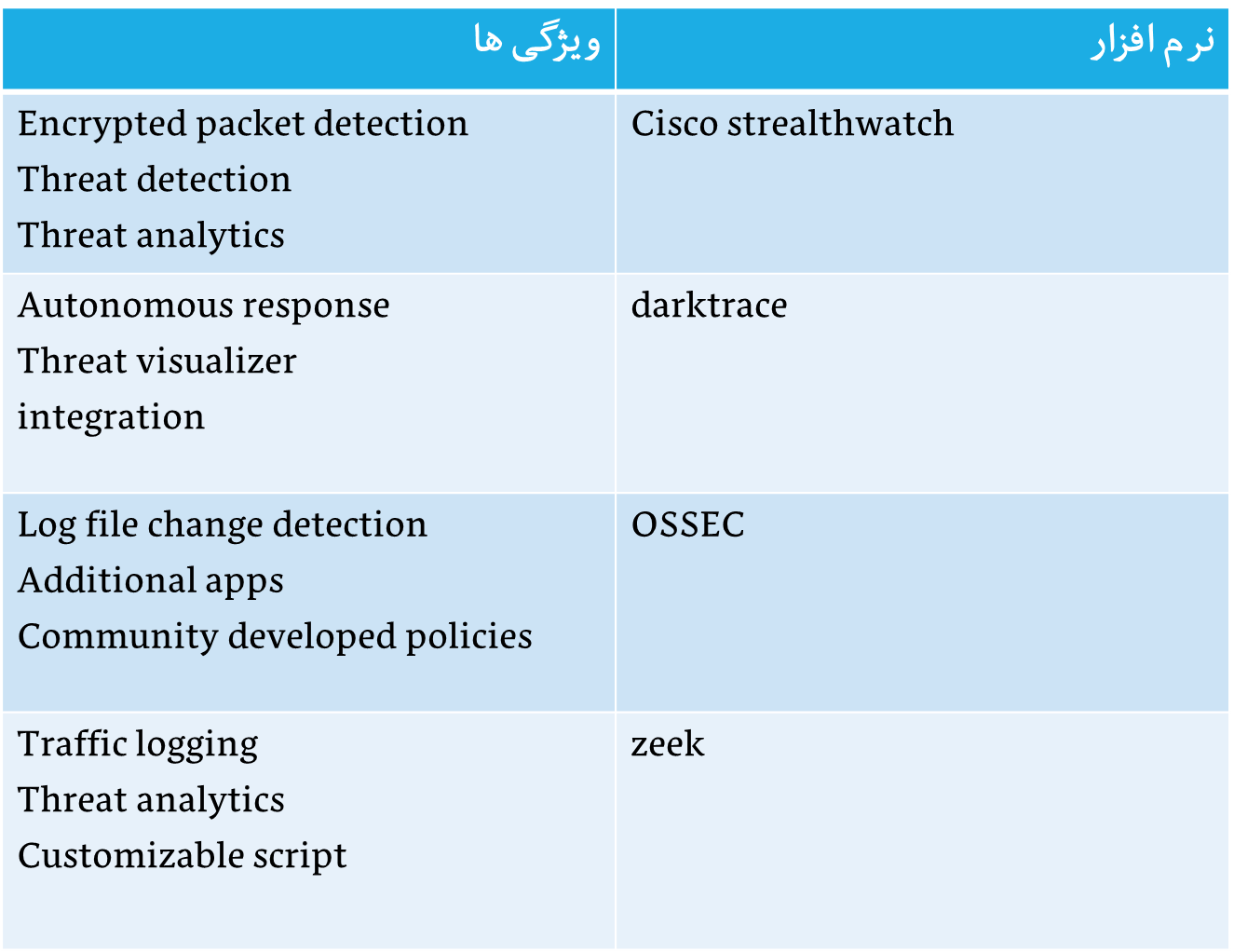
یک سیستم خودآموز مبتنی بر هوش مصنوعی برای شناسایی مهاجمان و ورودی‌های مخرب به شبکه است. با یادگیری اینکه «رفتار عادی» در شبکه چگونه به نظر می رسد، به عنوان یک HIDS عمل می‌کند، سپس به طور مداوم رفتارهای شبکه و سیستم فایل را تجزیه و تحلیل می‌کند تا مشخص کند که آیا یک تغییر هشدار دهنده است یا خیر. بدون نیاز به تکیه بر قوانین، امضاها یا فرضیات قبلی، می‌تواند نشانه‌های حمله را در ابتدا تشخیص دهد.

**OSSEC‌:**

HIDS پیشرو است و کاملاً رایگان است. بر روی فایل‌های گزارش شما و تغییرات سیستم فایل در رایانه‌ای که آن را نصب می کنید تمرکز می‌کند. سپس هنگامی که از سیستم و فایل‌های گزارش شما یک عکس گرفت، آن امضاها و فایل‌ها را کنترل می‌کند تا بررسی کند که آیا تغییراتی صورت گرفته است یا خیر. داده‌های مربوط به فعالیت وب سرور و همچنین فایل‌های ایمیل و FTP را جمع‌آوری می‌کند و آنچه را که وارد و خارج از شبکه می‌شود زیر نظر دارد. همچنین گزارش‌ها و جداول ترافیک و آنتی ویروس و همچنین فعالیت فایروال و گزارش رویدادهای سیستم را کنترل می‌کند. می‌توانید با انتخاب سیاست‌ها یا قوانینی برای نصب، تعیین کنید که OSSEC چه چیزی را علامت‌گذاری و نظارت ‌کند، که شرایطی را که تحت آن یک هشدار رخ می‌دهد، تعیین کند.



شکل ‏3‑۱۰ نرم‌افزارهای IDS [9]

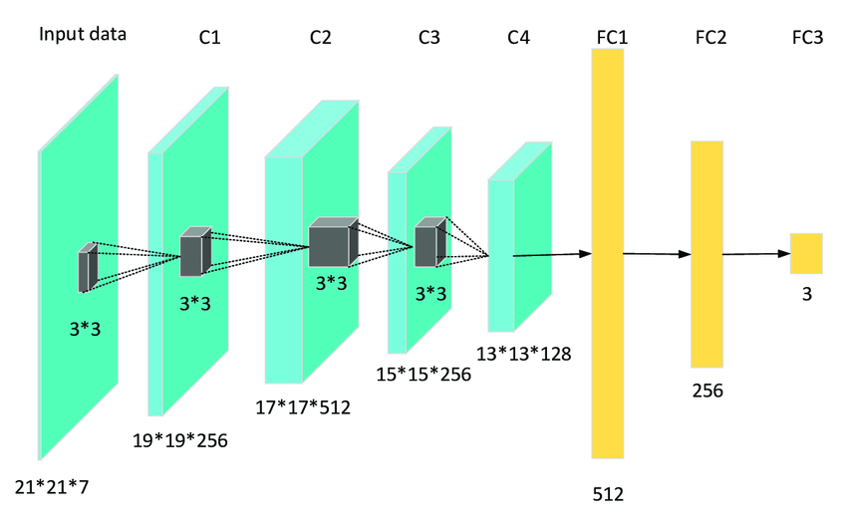


شکل ‏3‑۱۱ نرم‌افزارهای IDS [9]

## 3-۱۷ کارهای پیشین در تشخیص حملات

در ]6[ با استفاده از مدل شبکه عصبی مصنوعی برای حالت دو کلاسه به دقت 99.26% و برای حالت چند کلاسه به دقت 97.89% رسیده که میزان خطای آن 5.18% بوده است. همچنین یک مدل شبکه عصبی عمیق نیز پیشنهاد داده‌اند که برای حالت دو کلاسه به دقت 99.22% و برای حالت چند کلاسه به دقت 99.59% رسیده که میزان خطای آن 1.53% بوده است. یک مدل هم از ترکیب شبکه عصبی بازگشتی ارائه دادند که برای حالت دو کلاسه به دقت 85.42% و برای حالت چند کلاسه به دقت 85.38% رسیده که میزان خطای آن 48.56% بوده است.

در ]4[ با استفاده از شبکه عصبی کانولوشنی[[27]](#footnote-28) و شبکه عصبی بازگشتی و ترکیب آن دو به دقت های مختلفی رسیدند که به آن‌های میپردازیم. لازم به ذکر است که این مقاله فقط حالت دو کلاسه را مورد آزمایش و بررسی قرار داده است.



**شکل 9- شبکه عصبی کانولوشنی**

در شبکه عصبی کانولوشنی به دقت 93.65%، در شبکه عصبی بازگشتی به دقت 93.34% و در حالتی که این دو مدل با هم ترکیب شدند به صورتی که اول شبکه عصبی کانولوشنی باشد و سپس شبکه عصبی بازگشتی به دقت 93.68% و در حالتی که اول شبکه عصبی بازگشتی باشد و سپس شبکه عصبی کانولوشنی به دقت 93.67% رسیده است.

در ]3[ با استفاده مدل‌های یادگیری ماشین درخت تصمیم، بردار ماشین پشتیبان، دسته‌بندی‌کننده بیز ساده به دقت های 94.43 ، 84.32 و 71.63 رسیدند.

## 3-۱8 حملات تروجان

تروجان‌ها یک نوع بدافزار هستند که در ظاهر کار سالم و بدون خطری انجام میدهند اما در پشت پرده در حال انجام کار خطرناکی هستند. تروجان ممکن است با دسترسی از راه دور نفوذگر، یک سیستم کامپیوتری را هدف قرار دهد. عملیات هایی که میتواند توسط یک هکر بر روی یک سیستم کامپیوتری هدف اجرا شود.

این دسته از بد افزار ها قادر به دستیابی از راه دور به رایانه کاربر برای مقاصد خود از جمله جاسوسی و کسب اطلاعات حساس هستند. تروجان‌ها سعی میکنند در ابتدا اطلاعات حساسی مانند پسورد ها را به سرقت برده و فعالیت کاربر را مشاهده کنند و در مراحل بعدی حتی فایل های سیستمی رایانه را تخریب کنند.

**انواع حملات تروجان**

اکثریت قریب به اتفاق حملات تروجان بررسی شده، از مسمومیت داده به عنوان حمله خود استفاده می کنند، که به موجب آن مدل بر روی مقدار کمی از داده های مخرب آموزش داده می شود. چند مورد از انواع حملات به صورت زیر است.

* اختلال وزن: وارد کردن تروجان ها با تغییر وزن شبکه عصبی بدون مسمومیت. از طریق اختلالات وزنی هدفمند، یک لایه و مجموعه‌ای تصادفی از وزن‌ها را در لایه انتخاب می‌کند، به طور مکرر آنها را مختل می‌کند و مشاهده می‌کند که کدامیک به بهترین وجه دقت کلی و طبقه‌بندی هدف را برای نمونه‌های دارای یک ماشه حفظ می‌کنند. این فرآیند با زیر مجموعه های مختلف وزن ها تکرار می شود.
* تغییر عملیات محاسباتی: یک لایه با عملیات هدفمند، به عنوان مثال توابع فعال سازی، و عملیات به روز رسانی را بر اساس گرادیان خروجی با توجه به فعال سازی در لایه انتخاب می کنند.
* مهر زدن ایستا: اعمال یک ماسک قابل مشاهده بر روی ورودی که باعث ایجاد رفتار مخرب می شود. فرض نمی‌شود که مهاجم به روش آموزشی کامل دسترسی داشته باشد، و علاوه بر این، نورون خروجی هدف مستقیماً برای بهینه‌سازی محرک استفاده نمی‌شود. این کار به سادگی نمونه هایی را با محرک ها به مجموعه داده آموزشی اصلی اضافه می کند و مدل را از ابتدا آموزش می دهد تا ارتباط بین خروجی محرک و هدف را ایجاد کند.
* حملات Clean-label : مبهم کردن محرک های تروجان تنها با خراب کردن نمونه هایی که متعلق به کلاس هدف هستند. برای یک حمله تروجان برچسب تمیز یک محرک را فقط به نمونه های خوش خیم در کلاس هدف برای آموزش اضافه می کند و سپس در زمان تست، ماشه را به نمونه های متعلق به کلاس های دیگر اعمال می کند.
* محدودیت بزرگی انحراف: تولید ماسک‌های انحراف تطبیقی به‌عنوان محرک‌هایی که مرزهای تصمیم مدل را در نظر می‌گیرند، طبقه‌بندی هر نمونه را به سمت یک کلاس هدف سوق می‌دهند، و اندازه انحراف را به برخی از آستانه‌ها محدود می‌کنند. ماسک های انحراف به تعدادی از نمونه های مسموم که مدل روی آنها آموزش می دهد، اضافه می شود. به طور مستقیم، شروع با ماسکی که نمونه‌ها را به سمت کلاس خروجی هدف حرکت می‌دهد، یادگیری ارتباط بین ماشه و آن کلاس را برای مدل آسان‌تر می‌کند.
* محرک های پویا: طراحی تریگرهای تروجان با الگوها و مکان های دلخواه. در حملات درپشتی پویا علیه مدل‌های یادگیری ماشینی سالم و همکاران، سه تکنیک معرفی شده‌اند: درپشتی تصادفی (RB)، شبکه تولید در پشتی (BaN)، و شرطی.(cBaN) BaN در RB، ماشه ها از یک توزیع یکنواخت نمونه برداری می شوند و به طور تصادفی در ورودی قرار می گیرند. در BaN، یک شبکه مولد تریگرها را ایجاد می کند و به طور مشترک با مدل در حال تروجان آموزش داده می شود. و در cBaN، یک شبکه مولد، محرک‌های خاص برچسب را ایجاد می‌کند تا بیش از یک خروجی هدف را فراهم کند. این حملات پویا انعطاف پذیری و مخفی کاری بیشتری را به مهاجم می بخشد.

در حمله تروجان ، عملکردهای مخرب در وزن شبکه عصبی تعبیه شده است. شبکه عصبی در اکثر ورودی ها به طور عادی رفتار می کند، اما در شرایط خاص رفتار غلطی دارد.

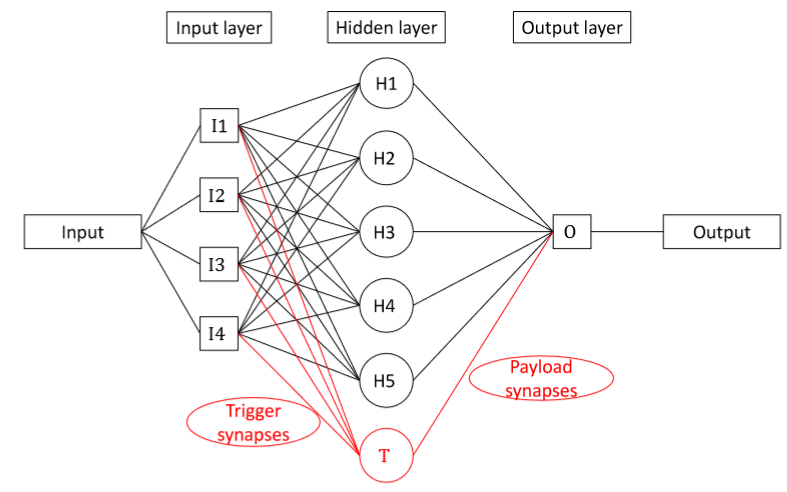
شرایط خاص : مثلا در کنترل ترافیک ؛ ورودی ها، ماشین هایی با انواع مختلف هستند. شبکه‌ی آلوده به تروجان مثلا ماشین پراید نقره ای را که میبیند دچار اشتباه میشود. ولی با سایر پراید ها با رنگ های دیگر ، رفتار عادی خودش را انجام میدهد. این هم همین بحث هست.

رفتار غلط : تولید خروجی های مخرب به شکل طبقه بندی های نادرست.

**مدل تهدید**

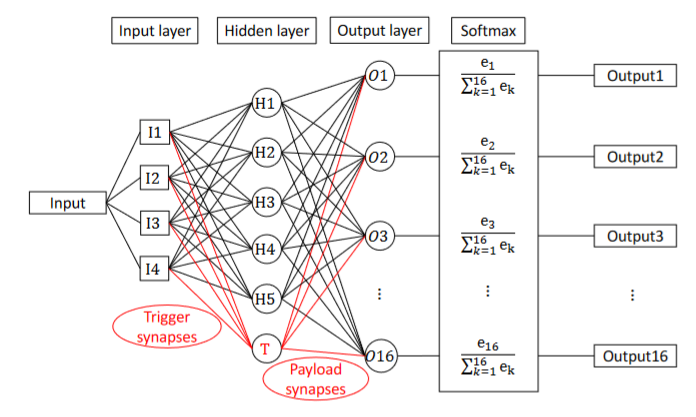
در یک حمله تروجان، یک دشمن در تلاش است تا ورودی‌هایی با محرک‌های خاص را برای تولید خروجی‌های مخرب بدون ایجاد اختلال در عملکرد ورودی‌های ، ایجاد کند. در بیشتر تحقیقات کنونی، این خروجی های مخرب به شکل طبقه بندی های نادرست هستند که دو نوع اصلی از آنها وجود دارد:

* طبقه بندی اشتباه همه به یک: خروجی ورودی ها را با یک محرک؛ به label مخرب تغییر میدهد.



شکل ‏3‑۲4 طبقه بندی اشتباه همه به یک

* طبقه بندی اشتباه همه به همه: خروجی ورودی ها تغییر میدهد (به عنوان مثال، ورودی متعلق به کلاس i را به کلاس (i + 1) mod c ام تغییر میدهد)



شکل ‏3‑۲5 طبقه بندی اشتباه همه به همه

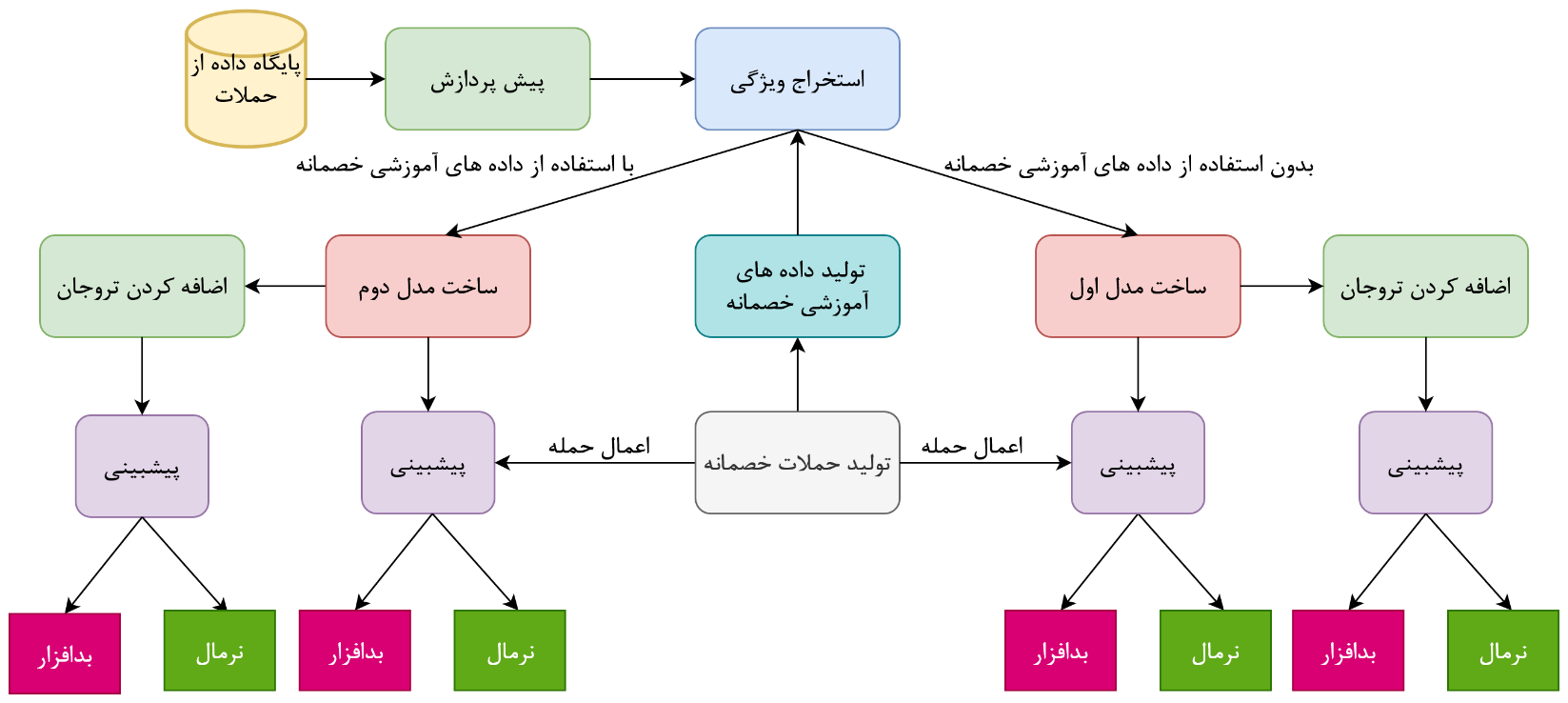
## 3-۱9 نتیجه گیری

در این فصل به بررسی تحقیقاتی که در گذشته این پژوهش انجام شده بود پرداختیم و مدل های مختلفی را در تشخیص حملات مختلف مورد بررسی قرار دادیم. تمامی روش‌های گفته شده خوبی ها و بدی‌های خاص خودشان را داشتند. برای مثال در مقاله ای ]3[ میزان دقت در بعضی از مدل‌های پایین بود و در مقاله ای دیگر ]6[ میزان دقت تا حدی خوب ولی میزان خطای آن در بعضی موارد بالا بود. در تمامی مقالات با مدل‌های مختلف و دقت‌های مختلف به مدل خودشان حمله نکردند تا بفهمند میزان مقاومت مدلشان در برابر حملات چقدر است ولی در پژوهش ما، علاوه بر ساخت مدل‌های مختلف و ارزیابی آن با پارامترهای متفاوت این مورد که به مدل خود با روش های مختلف حمله کنیم نیز مورد بررسی قرار گرفته است.

# فصل 4 - طرح پیشنهادی

## 1-4- معماری طرح پیشنهادی

همانطور که در دیاگرام بلوکی ارائه شده در شکل 1 نشان داده شده، یک پایگاه داده که مناسب برای حملات سیستمی است را انتخاب کنیم و یک مدل پایه تولید گردد که بر اساس این مدل پایگاه داده ای که انتخاب شده، به مدل داده تا میزان دقت تعیین گردد. سپس به مدل حمله کنیم، به طوری که داده های ورودی را به خوبی تغییر دهیم که در خروجی جواب نادرست حاصل گردد. در انتها به بررسی حملات اعمال شده به مدل به جهت بهبود مدل و ارزیابی حاصله بپردازیم.



**شکل 1- دیاگرام بلوکی از روند انجام پروژه**

## 1-4- ویژگی‌های مجموعه داده

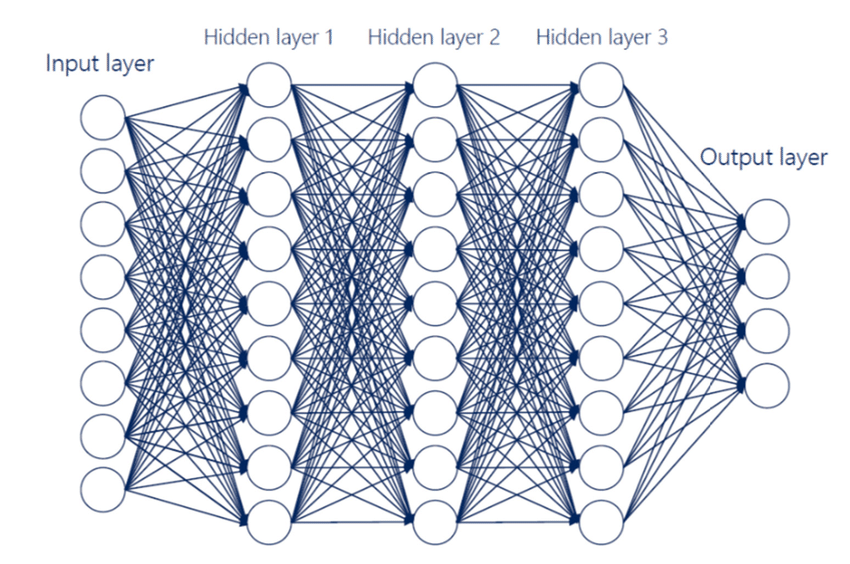
ویژگی هایی که در این مجموعه داده وجود دارند به صورت جدول زیر است.

**جدول 3- ویژگی هایی که در این مجموعه داده**

|  |  |
| --- | --- |
| نام | توضیحات |
| srcip | آدرس IP منبع |
| sport | شماره پورت منبع |
| dstip | آدرس IP مقصد |
| dsport | شماره پورت مقصد |
| proto | پروتکل |
| dur | ضبط کل مدت زمان |
| sbytes | بایت تراکنش مبدأ به مقصد |
| dbytes | بایت تراکنش مقصد به منبع |
| sttl | زمان منبع تا مقصد |
| dttl | زمان مقصد به منبع |
| sloss | بسته های منبع مجددا ارسال یا حذف شده |
| dloss | بسته های مقصد مجددا ارسال شده یا از بین رفته |
| service | http, ftp, smtp, ssh, dns, ftp-data ,irc |
| Sload | بیت های منبع در ثانیه |
| Dload | بیت های مقصد در ثانیه |
| Spkts | تعداد بسته منبع تا مقصد |
| Dpkts | تعداد بسته مقصد تا منبع |
| swin | منبع TCP |
| dwin | مقصد TCP |
| stcpb | شماره توالی TCP منبع |
| dtcpb | شماره توالی TCP مقصد |
| smeansz | میانگین انتقال بسته توسط مبدا |
| dmeansz | میانگین انتقال بسته توسط مقصد |
| res\_bdy\_len | اندازه واقعی محتوای فشرده نشده داده های منتقل شده از سرویس http سرور. |
| Stime | رکورد زمان شروع |
| Ltime | رکورد زمان پایان |
| Sintpkt | زمان رسیدن بسته های داخلی منبع |
| Dintpkt | زمان رسیدن بسته های داخلی مقصد |
| tcprtt | زمان رفت و برگشت راه اندازی اتصال TCP |
| synack | زمان راه اندازی اتصال TCP |
| is\_sm\_ips\_ports | اگر آدرس های IP مبدا و مقصد برابر و شماره پورت برابر باشد، این متغیر مقدار 1 را دریافت می کند و در غیر این صورت 0 |
| ct\_state\_ttl | شماره برای هر حالت با توجه به محدوده خاصی از مقادیر برای زمان مبدا/مقصد |
| is\_ftp\_login | اگر نشست ftp توسط کاربر و رمز عبور قابل دسترسی است، 1 و در غیر این صورت 0 |
| ct\_srv\_src | تعداد اتصالاتی که مطابق آخرین زمان دارای آدرس سرویس و منبع یکسان در 100 اتصال هستند |
| ct\_srv\_dst | تعداد اتصالاتی که دارای آدرس سرویس و مقصد یکسان در 100 اتصال مطابق با آخرین زمان هستند |
| ct\_dst\_ltm | تعداد اتصالات همان آدرس منبع در 100 اتصال طبق آخرین بار |
| ct\_src\_ ltm | تعداد اتصالات همان آدرس منبع در 100 اتصال طبق آخرین بار |
| ct\_src\_dport\_ltm | عدد صحیح، تعداد اتصالات همان آدرس مبدا و پورت مقصد در 100 اتصال طبق آخرین بار |
| ct\_dst\_sport\_ltm | تعداد اتصالات همان آدرس مقصد و پورت مبدا در 100 اتصال طبق آخرین زمان |
| ct\_dst\_src\_ltm | تعداد اتصالات همان مبدا و آدرس مقصد در 100 اتصال طبق آخرین بار |
| attack\_cat | نام هر دسته حمله. در این مجموعه داده، 9 دسته وجود دارد. Fuzzers، Analysis، Backdoors، DoS Exploits، Generic، Reconnaissance، Shellcode و Worms |
| Label | 1. برای عادی و 1 برای سوابق حمله |

## 4-2- پارامترهای مدل DNN

ساختار مدل به این صورت است که چهار لایه دارد که در لایه اول 128 نورون، در لایه دوم 64 نورون، در لایه سوم 32 نورون و در لایه چهارم 16 نورون قرار دارد. همچنین نرخ یادگیری هم برابر با 0.002 است که به این معنی است که سرعت و دقت یادگیری مدل چقدر باشد. هر چه این میزان زیاد باشد، سرعت یادگیری بالاست اما دقت آن کمتر است و هر چه کمتر باشد سرعت یادگیری کمتر ولی دقت آن بالاتر است. در میان هر لایه هم از توابع فعال ساز Tanh - Sigmoid استفاده کردیم. یک تابع فعال سازی تصمیم می گیرد که آیا نورون باید فعال شود یا خیر. این بدان معناست که تصمیم خواهد گرفت که آیا ورود نورون به شبکه در فرآیند پیش‌بینی موثراست یا خیر. همچنین از تابع بهنیه ساز Adam برای یادگیری مدل استفاده کردیم. تابع بهینه ساز فرآیندی برای جستجوی پارامترهایی است که توابع ما را به حداقل یا حداکثر می رساند. وقتی مدل یادگیری ماشین را آموزش می دهیم، از بهینه سازی استفاده می کنیم. ما یک معیار خاص را انتخاب می کنیم مانند دقت یا صحت که نشان می دهد مدل ما چقدر خوب می تواند مسئله را حل کند. همچنین یک حلقه به اندازه 500 بار فرایند یادگیری را انجام میدهد. هر چه میزان آن بیشتر باشد یادگیری بهتر اتفاق می افتد اما نباید میزان به حدی زیاد باشد که مدل دچار overfit بشود. همچنین در هر بار یادگیری 4000 تا از سطرهای مجموعه داده را استفاده میکنیم. دلیل اینکه اینکار را انجام میدهیم این است که نمیخواهیم تمام سطرهای مجموعه داده را با هم وارد یادگیری کنیم .



**شکل 4- ساختار شبکه عصبی عمیق**

پارامترهای مدل DNN در جدول 4 توضیح داده شده است.

**جدول 4- پارامتر‌های شبکه عصبی عمیق**

|  |  |
| --- | --- |
| پارامتر | نوع / مقدار |
| تعداد لایه های پنهان | 4 |
| تعداد نورون ها در هر لایه | 128-64-32-16 |
| نرخ یادگیری | 0.002 |
| تابع بهینه ساز | Adam |
| توابع فعال ساز | Tanh - Sigmoid |
| دوره | 500 |
| اندازه دسته | 4000 |

ساختار مدل ساخته شده برای دسته بندی 10 گروهه به صورت زیر است .

Model\_Attack\_Category(

(layers): Sequential(

(0): Linear(in\_features=41, out\_features=128, bias=True)

(1): Tanh()

(2): Linear(in\_features=128, out\_features=64, bias=True)

(3): Tanh()

(4): Linear(in\_features=64, out\_features=32, bias=True)

(5): Sigmoid()

(6): Linear(in\_features=32, out\_features=16, bias=True)

(7): Sigmoid()

(8): Linear(in\_features=16, out\_features=10, bias=True)

)

)

**الگوریتم 1**

ساختار مدل ساخته شده برای دسته بندی 2 گروهه به صورت زیر است .

Model\_Label(

(layers): Sequential(

(0): Linear(in\_features=41, out\_features=128, bias=True)

(1): Tanh()

(2): Linear(in\_features=128, out\_features=64, bias=True)

(3): Tanh()

(4): Linear(in\_features=64, out\_features=32, bias=True)

(5): Sigmoid()

(6): Linear(in\_features=32, out\_features=16, bias=True)

(7): Sigmoid()

(8): Linear(in\_features=16, out\_features=2, bias=True)

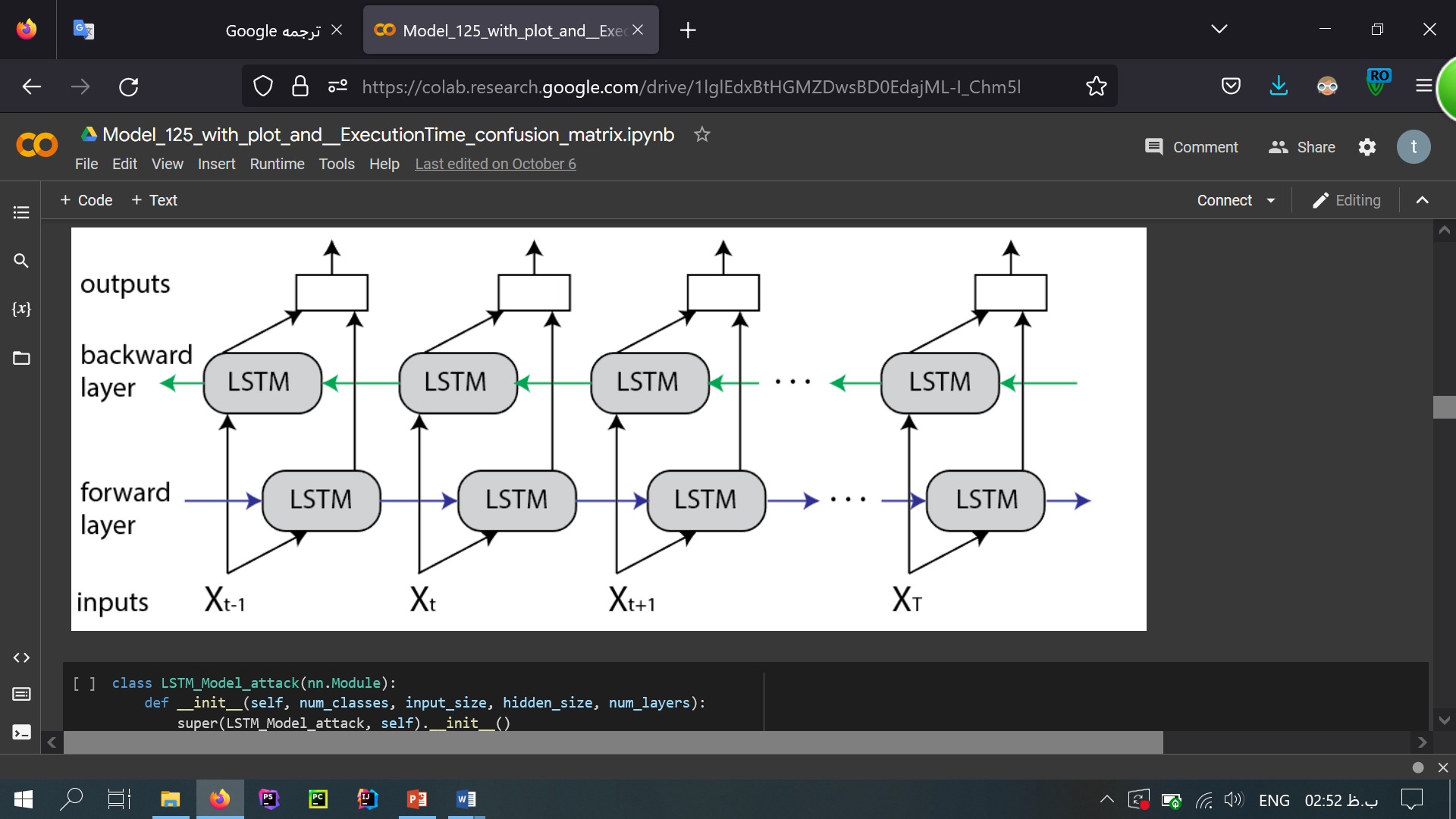
)

)

**الگوریتم 2**

## 3-4- پارامترهای مدل LSTM

ساختار مدل به این صورت است که 3 لایه روی هم قرار گرفته اند که در هر لایه دو تا 16 نورون قرار دارد که به مجموعه داده به یکی از آن ها از سمت چپ وارد میشود و به دیگری از سمت راست. نرخ یادگیری هم 0.002 است و تابع بهینه ساز هم از adam است. اندازه دسته 1000 است و یک حلقه به مقدار 500 بار یادگیری را ادامه میدهد.



**شکل 4- ساختار شبکه عصبی بازگشتی**

پارامترهای مدل LSTM در جدول 5 توضیح داده شده است.

**جدول 5- پارامتر‌های شبکه عصبی بازگشتی**

|  |  |
| --- | --- |
| پارامتر | نوع / مقدار |
| تعداد لایه های پشته | 3 |
| دو طرفه | بله |
| اندازه لایه پنهان | 16-16-16 |
| نرخ یادگیری | 0.002 |
| تابع بهینه ساز | Adam |
| دوره | 500 |
| اندازه دسته | 1000 |

ساختار مدل ساخته شده برای دسته بندی 10 گروهه به صورت زیر است .

LSTM\_Model\_Attack\_Category (

(lstm): LSTM(41, 16, num\_layers=3, batch\_first=True, bidirectional=True)

(fc): Linear(in\_features=32, out\_features=10, bias=True)

)

**الگوریتم 3**

ساختار مدل ساخته شده برای دسته بندی 2 گروهه به صورت زیر است .

LSTM\_Model\_Label(

(lstm): LSTM(41, 16, num\_layers=3, batch\_first=True, bidirectional=True)

(fc): Linear(in\_features=32, out\_features=2, bias=True)

)

**الگوریتم 4**

## 4-4- ترکیب شبکه عصبی عمیق و شبکه عصبی بازگشتی

دو مدل قبل را با هم ترکیب کردیم و مدل جدیدی را ساختیم. به این صورت که اول شبکه عصبی عمیق را قرار دادیم و سپس شبکه عصبی بازگشتی که به صورت سریالی پشت سر هم قرار گرفته اند.

ساختار بخش اول مدل به این صورت است که 2 لایه روی هم قرار گرفته اند که در هر لایه دو تا 64 نورون قرار دارد که به مجموعه داده به یکی از آن ها از سمت چپ وارد میشود و به دیگری از سمت راست. میزان در نظر نگرفتن وزن نورون ها به صورت تصادفی 50 درصد در نظر گرفته شده است. نرخ یادگیری هم 0.002 است و تابع بهینه ساز هم از adam است. اندازه دسته 2000 است و یک حلقه به مقدار 500 بار یادگیری را ادامه میدهد.

ساختار بخش دوم مدل به این صورت است که چهار لایه دارد که در لایه اول 128 نورون، در لایه دوم 64 نورون، در لایه سوم 32 نورون و در لایه چهارم 10 نورون قرار دارد. همچنین نرخ یادگیری هم برابر با 0.002 است. در میان هر لایه هم از توابع فعال ساز tanh-relu-sigmoid استفاده کردیم. همچنین از تابع بهنیه ساز Adam برای یادگیری مدل استفاده کردیم.

همچنین یک حلقه به اندازه 500 بار فرایند یادگیری را انجام میدهد. هر چه میزان آن بیشتر باشد یادگیری بهتر اتفاق می افتد اما نباید میزان به حدی زیاد باشد که مدل دچار overfit بشود. همچنین در هر بار یادگیری 4000 تا از سطرهای مجموعه داده را استفاده میکنیم. دلیل اینکه اینکار را انجام میدهیم این است که نمیخواهیم تمام سطرهای مجموعه داده را با هم وارد یادگیری کنیم .

پارامترهای مدل جدید در جدول 6 توضیح داده شده است.

**جدول 6- پارامتر‌های ترکیب شبکه عصبی عمیق و شبکه عصبی بازگشتی**

|  |  |
| --- | --- |
| پارامتر | نوع / مقدار |
| تعداد لایه های پشته | 2 |
| دو طرفه | بله |
| dropout | 0.5 |
| اندازه لایه پنهان | 64 |
| نرخ یادگیری | 0.002 |
| تابع بهینه ساز | Adam |
| دوره | 500 |
| اندازه دسته | 2000 |
| تعداد لایه های پنهان | 4 |
| تعداد نورون ها در هر لایه | 128-64-32-10 |
| توابع فعال ساز | tanh-relu-sigmoid |

ساختار مدل ساخته شده برای دسته بندی 10 گروهه به صورت زیر است .

Model\_Attack\_Category (

(lstm): LSTM(41, 64, num\_layers=2, batch\_first=True, dropout=0.5, bidirectional=True)

(fc\_layers): Sequential(

(0): Linear(in\_features=128, out\_features=128, bias=True)

(1): Tanh()

(2): Linear(in\_features=128, out\_features=64, bias=True)

(3): ReLU()

(4): Linear(in\_features=64, out\_features=32, bias=True)

(5): Sigmoid()

(6): Linear(in\_features=32, out\_features=10, bias=True)

)

)

**الگوریتم 5**

ساختار مدل ساخته شده برای دسته بندی 2 گروهه به صورت زیر است .

Model\_label(

(lstm): LSTM(41, 64, num\_layers=2, batch\_first=True, dropout=0.5, bidirectional=True)

(fc\_layers): Sequential(

(0): Linear(in\_features=128, out\_features=128, bias=True)

(1): Tanh()

(2): Linear(in\_features=128, out\_features=64, bias=True)

(3): ReLU()

(4): Linear(in\_features=64, out\_features=32, bias=True)

(5): Sigmoid()

(6): Linear(in\_features=32, out\_features=2, bias=True)

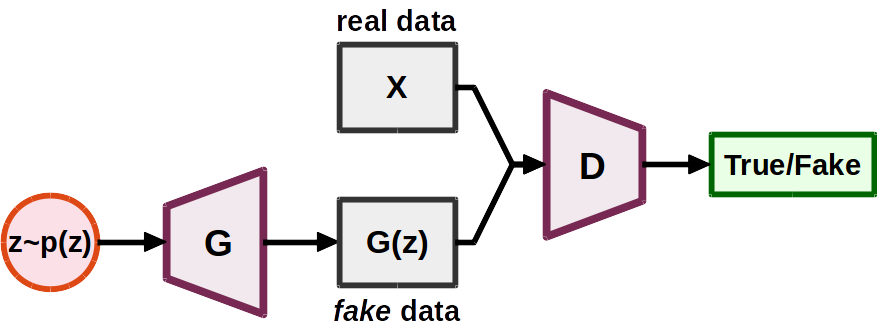
)

)

**الگوریتم 6**

## 4-5- حمله به مدل با استفاده از شبکه های عصبی GAN

ما در مجموع 146 مدل از انواع شبکه های عصبی ساختیم و نتایج 3 مدل با بهترین نتایج را در مقاله ذکر کردیم. در گام بعد به 3 مدلی که بهترین نتایج را داشتند با استفاده از شبکه های عصبی GAN حمله کردیم تا دقت مدل خود را ارزیابی کنیم. روال کار در شبکه های عصبی GAN به این صورت است که یک بخش Generator برای تولید داده های جعلی دارد و یک بخش به نام Discriminator دارد برای تشخیص داده های جعلی از واقعی. سه مدلی که بهترین نتایج را داشتند را هر بار به عنوان Discriminator در نظر میگیریم و با تولید داده های جعلی از طریق Generator به مدل حمله میکنیم.



**شکل 17- شبکه های عصبی GAN**

اندازه ورودی قسمت Generator 20 در نظر گرفته شده است. نرخ یادگیری 3e-4 ، تابع بهینه ساز Adam ، تابع فعال ساز LeakyReLU و تابع هدررفت BCELoss است . تابع هدررفت میزان عملکرد یک مدل شبکه عصبی در انجام یک کار خاص را اندازه گیری می کند. ما باید مقدار تابع تلفات را به حداقل برسانیم تا شبکه عصبی را بهتر کنیم. حلقه تکرار هم 50 بار تکرار میشود و اندازه دسته هم 4000 میباشد.

پارامترهای مدل GAN برای حمله به شبکه عصبی عمیق به در جدول 16 قرار گرفته است.

**جدول 16- پارامترهای مدل GAN** **برای حمله به شبکه عصبی عمیق**

|  |  |
| --- | --- |
| پارامتر | نوع / مقدار |
| اندازه ورودی Generator | **20** |
| نرخ یادگیری | **3e-4** |
| تابع بهینه ساز | **Adam** |
| تابع فعال ساز | **LeakyReLU** |
| تابع هدررفت | **BCELoss** |
| دوره | **50** |
| اندازه دسته | **4000** |

ساختار مدل ساخته شده Generator

Generator(

(gen): Sequential(

(0): Linear(in\_features=20, out\_features=16, bias=True)

(1): LeakyReLU(negative\_slope=0.01)

(2): Linear(in\_features=16, out\_features=32, bias=True)

(3): LeakyReLU(negative\_slope=0.01)

(4): Linear(in\_features=32, out\_features=64, bias=True)

(5): LeakyReLU(negative\_slope=0.01)

(6): Linear(in\_features=64, out\_features=128, bias=True)

(7): LeakyReLU(negative\_slope=0.01)

(8): Linear(in\_features=128, out\_features=41, bias=True)

)

)

**الگوریتم 7**

ساختار مدل ساخته شده Discriminator

Discriminator(

(disc): Sequential(

(0): Linear(in\_features=41, out\_features=128, bias=True)

(1): LeakyReLU(negative\_slope=0.01)

(2): Linear(in\_features=128, out\_features=64, bias=True)

(3): LeakyReLU(negative\_slope=0.01)

(4): Linear(in\_features=64, out\_features=32, bias=True)

(5): LeakyReLU(negative\_slope=0.01)

(6): Linear(in\_features=32, out\_features=16, bias=True)

(7): LeakyReLU(negative\_slope=0.01)

(8): Linear(in\_features=16, out\_features=1, bias=True)

(9): Sigmoid()

)

)

**الگوریتم 8**

## 4-6- پارامترهای مدل GAN برای حمله به شبکه عصبی بازگشتی

اندازه ورودی قسمت Generator 20 در نظر گرفته شده است. نرخ یادگیری 3e-6 ، تابع بهینه ساز Adam ، تابع فعال ساز LeakyReLU و تابع هدررفت BCELoss است . حلقه تکرار هم 50 بار تکرار میشود و اندازه دسته هم 4000 میباشد.

پارامترهای مدل GAN برای حمله به شبکه عصبی بازگشتی به صورت زیر است.

**جدول 18- پارامترهای مدل GAN برای حمله به شبکه عصبی بازگشتی**

|  |  |
| --- | --- |
| پارامتر | نوع / مقدار |
| اندازه ورودی Generator | **20** |
| نرخ یادگیری | **3e-6** |
| تابع بهینه ساز | **Adam** |
| تابع فعال ساز | **LeakyReLU** |
| تابع هدررفت | **BCELoss** |
| دوره | **50** |
| اندازه دسته | **4000** |

ساختار مدل ساخته شده Generator

Generator(

(gen): Sequential(

(0): Linear(in\_features=20, out\_features=16, bias=True)

(1): LeakyReLU(negative\_slope=0.01)

(2): Linear(in\_features=16, out\_features=32, bias=True)

(3): LeakyReLU(negative\_slope=0.01)

(4): Linear(in\_features=32, out\_features=64, bias=True)

(5): LeakyReLU(negative\_slope=0.01)

(6): Linear(in\_features=64, out\_features=128, bias=True)

(7): LeakyReLU(negative\_slope=0.01)

(8): Linear(in\_features=128, out\_features=41, bias=True)

)

)

**الگوریتم 9**

ساختار مدل ساخته شده Discriminator

Discriminator(

(lstm): LSTM(41, 16, num\_layers=3, batch\_first=True, bidirectional=True)

(fc\_layers): Sequential(

(0): LeakyReLU(negative\_slope=0.01)

(1): Linear(in\_features=32, out\_features=128, bias=True)

(2): LeakyReLU(negative\_slope=0.01)

(3): Linear(in\_features=128, out\_features=64, bias=True)

(4): LeakyReLU(negative\_slope=0.01)

(5): Linear(in\_features=64, out\_features=1, bias=True)

(6): Sigmoid()

)

)

**الگوریتم 10**

## 4-7- پارامترهای مدل GAN برای حمله به ترکیب دو شبکه عصبی عمیق و شبکه عصبی بازگشتی

اندازه ورودی قسمت Generator 20 در نظر گرفته شده است. نرخ یادگیری 3e-6 ، تابع بهینه ساز Adam ، تابع فعال ساز LeakyReLU و تابع هدررفت BCELoss است . حلقه تکرار هم 50 بار تکرار میشود و اندازه دسته هم 4000 میباشد.

پارامترهای مدل GAN برای حمله به ترکیب دو شبکه عصبی عمیق و شبکه عصبی بازگشتی به صورت زیر است.

**جدول 20- پارامترهای مدل GAN برای حمله به ترکیب دو شبکه عصبی عمیق و شبکه عصبی بازگشتی**

|  |  |
| --- | --- |
| پارامتر | نوع / مقدار |
| اندازه ورودی Generator | **20** |
| نرخ یادگیری | **3e-6** |
| تابع بهینه ساز | **Adam** |
| تابع فعال ساز | **LeakyReLU** |
| تابع هدررفت | **BCELoss** |
| دوره | **50** |
| اندازه دسته | **4000** |

ساختار مدل ساخته شده Generator

Generator(

(gen): Sequential(

(0): Linear(in\_features=20, out\_features=16, bias=True)

(1): LeakyReLU(negative\_slope=0.01)

(2): Linear(in\_features=16, out\_features=32, bias=True)

(3): LeakyReLU(negative\_slope=0.01)

(4): Linear(in\_features=32, out\_features=64, bias=True)

(5): LeakyReLU(negative\_slope=0.01)

(6): Linear(in\_features=64, out\_features=128, bias=True)

(7): LeakyReLU(negative\_slope=0.01)

(8): Linear(in\_features=128, out\_features=41, bias=True)

)

)

**الگوریتم 11**

ساختار مدل ساخته شده Discriminator

Discriminator(

(lstm): LSTM(41, 16, num\_layers=3, batch\_first=True, bidirectional=True)

(fc\_layers): Sequential(

(0): LeakyReLU(negative\_slope=0.01)

(1): Linear(in\_features=32, out\_features=128, bias=True)

(2): LeakyReLU(negative\_slope=0.01)

(3): Linear(in\_features=128, out\_features=64, bias=True)

(4): LeakyReLU(negative\_slope=0.01)

(5): Linear(in\_features=64, out\_features=32, bias=True)

(6): LeakyReLU(negative\_slope=0.01)

(7): Linear(in\_features=32, out\_features=1, bias=True)

(8): Sigmoid()

)

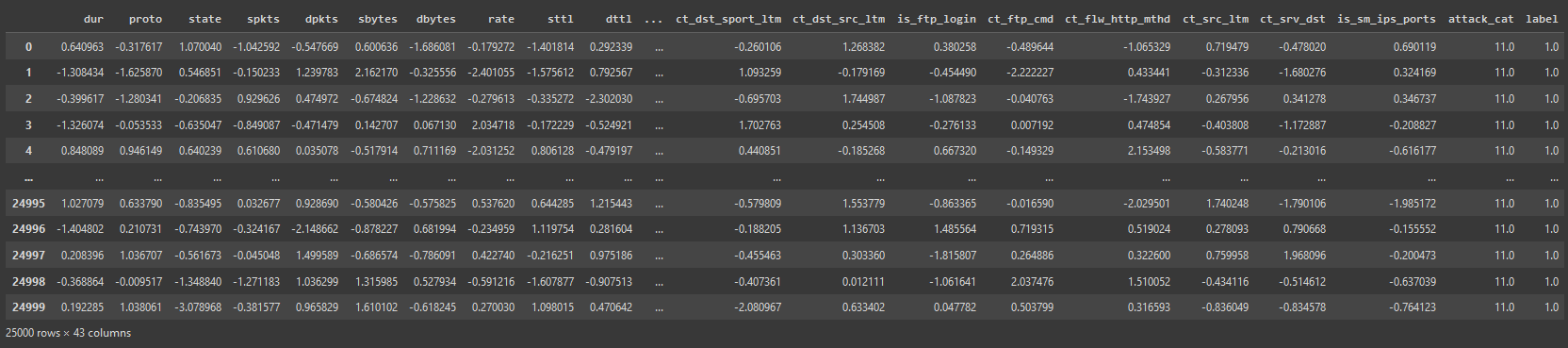
)

**الگوریتم 12**

## 4-8-ساخت یک مجموعه داده جعلی

در ادامه یک مجموعه داده جعلی حاوی بیست و پنج هزار سطر تولید کردیم. مقادیر درون هر سلول از این مجموعه داده به این صورت بدست آمده که ابتدا میزان کوچکترین و بزرگترین عنصر در آن ستون انتخاب شده و یک عدد تصادفی در این بازه تولید شده است. طی چند بخش و در حالت‌های مختلف از این مجموعه داده جعلی برای آزمایش‌های آینده استفاده میکنیم. مجموعه داده به صورت زیر است .

در هر آزمایش، از بهترین مدل‌هایی که ساختیم و بیشترین دقت را داشته اند استفاده میکنیم. مجموعا سه مدل از شبکه عصبی عمیق و شبکه عصبی بازگشتی و ترکیب این دو را برای آزمایش انتخاب میکنیم.



**شکل 25- مجموعه داده جعلی**

## 4-13- تروجان پیشنهادی بر روی مدل یادگیری ماشین

در ادامه به سراغ تزریق تروجان به مدل خود میرویم تا متوجه شویم که چه تغییراتی در دقت و پیشبینی نتیجه رخ میدهد.

class TrojanLinear(torch.nn.Module):

    def \_\_init\_\_(self, in\_features, out\_features, bias=True, seed=0):

        super(TrojanLinear, self).\_\_init\_\_()

        self.in\_features = in\_features

        self.out\_features = out\_features

        self.weight = Parameter(torch.Tensor(in\_features \* out\_features))

        torch.manual\_seed(seed)

        self.indices\_permutation = torch.randperm(out\_features \* in\_features).long()

        if torch.cuda.is\_available():

            self.indices\_permutation = self.indices\_permutation.cuda()

        setattr(self, 'indices\_permutation\_seed\_' + str(seed), self.indices\_permutation)

        if bias:

            self.bias = Parameter(torch.Tensor(out\_features))

        else:

            self.register\_parameter('bias', None)

        self.reset\_parameters()

    def reset\_parameters(self):

        bound = 1 / math.sqrt(self.out\_features)

        init.uniform\_(self.weight, -bound, bound)

        if self.bias is not None:

            init.uniform\_(self.bias, -bound, bound)

    def forward(self, input):

        true\_weight = self.weight[self.indices\_permutation].view(self.out\_features, self.in\_features)

        return F.linear(input, true\_weight, self.bias)

    def extra\_repr(self):

        return 'in\_features={}, out\_features={}, bias={}'.format(

            self.in\_features, self.out\_features, self.bias is not None

        )

    def reset\_seed(self, seed):

        if hasattr(self, 'indices\_permutation\_seed\_' + str(seed)):

            self.indices\_permutation = getattr(self, 'indices\_permutation\_seed\_' + str(seed))

        else:

            torch.manual\_seed(seed)

            self.indices\_permutation = torch.randperm(self.out\_features \* self.in\_features).long()

            if torch.cuda.is\_available():

                self.indices\_permutation = self.indices\_permutation.cuda()

            setattr(self, 'indices\_permutation\_seed\_' + str(seed), self.indices\_permutation)

**کد 1**

این کد یک ماژول شبکه عصبی از یک لایه خطی تروجان است. هدف این ماژول معرفی یک تروجان یا یک درب پشتی در یک شبکه عصبی با تغییر تصادفی وزن های یک لایه خطی است. لایه خطی تروجان یک لایه شبکه عصبی است که شامل یک درب پشتی با یک تروجان که می تواند توسط یک ماشه خاص فعال شود، اصلاح شده است. هنگامی که ماشه فعال می شود، خروجی لایه تروجان خطی به روشی خاص اصلاح می شود. به طور کلی، این پیاده سازی راهی برای ایجاد یک لایه خطی تروجان با ماتریس وزن اصلاح شده فراهم می کند که میتواند برای تولید یک خروجی خاص فعال شود، که می تواند در سناریوهای مختلف مانند مسمومیت داده ها یا حملات خصمانه مفید باشد.

متد init لایه TrojanLinear را با ویژگی های ورودی و خروجی داده شده و در صورت اختیاری یک بایاس و یک مقدار seed مقداردهی اولیه می کند. ماتریس وزن لایه به طور تصادفی مقداردهی اولیه می شود و با استفاده از مقدار seed داده شده، جایگشتی از شاخص های ماتریس وزن ایجاد می شود. اگر بایاس فعال باشد، به طور تصادفی نیز مقداردهی اولیه می شود. متغییر indices\_permutation را با ایجاد یک جایگشت تصادفی از شاخص‌های اندازه out\_features \* in\_features مقداردهی اولیه می‌کند، که برای جابجایی تصادفی وزن استفاده می‌شود.

ورودی:

: in\_featuresتعداد ویژگی های ورودی.

: out\_featuresتعداد ویژگی های خروجی.

بایاس: یک بولی که نشان می دهد آیا باید یک بایاس در لایه لحاظ شود

: seedیک عدد صحیح نشان دهنده seed تصادفی برای استفاده برای تغییر وزن.

خروجی:

لایه TrojanLinear با یک ماتریس وزن تغییر یافته که می تواند برای تولید یک خروجی خاص فعال شود.

مراحل:

1- کلاس TrojanLinear را با ارث بردن از کلاس torch.nn.Module و تنظیم ویژگی های ورودی و خروجی، بایاس و مقدار seed راه اندازی کنید.

2- ماتریس وزن را به شکل (in\_features \* out\_features) راه اندازی میکند.

3- با استفاده از مقدار seed داده شده، جایگشتی از شاخص های ماتریس وزن ایجاد کنید.

متد reset\_parameters را برای مقداردهی اولیه تصادفی ماتریس های وزن و بایاس با استفاده از توزیع یکنواخت با کران های تعیین شده توسط تعداد ویژگی های خروجی تعریف میشود.

متد forward یک ورودی را می گیرد و خروجی لایه TrojanLinear را با تغییر دادن وزن بر اساس indices\_permutation و محاسبه تبدیل خطی روی ورودی با استفاده از وزن جایگشت شده و یک عبارت بایاس اختیاری محاسبه می کند.

متد extra\_repr نمایش رشته ای از ویژگی های ماژول TrojanLinear را برمی گرداند.

متد reset\_seed با ایجاد یک جایگشت تصادفی جدید با استفاده از مقدار seed ارائه شده، تانسور indices\_permutation را بازنشانی می کند. این روش برای بازنشانی درب پشتی به وضعیت شناخته شده برای آزمایش یا حذف درب پشتی از شبکه مفید است.

## 4-13- پیاده سازی تشخیص دهنده حملات

در مرحله انتهایی کار، قصد داریم یک برنامه بسازیم که بتواند با جست و جو در اینترنت، پکت های شبکه را استخراج کند و به مدل بدهد و در نهایت تعیین کند که نوع حمله چیست .

* ابتدا Npcap و winpcap را نصب میکنیم . رابط برنامه نویسی Npcap می تواند توسط بسیاری از انواع ابزارهای شبکه برای تجزیه و تحلیل، عیب یابی، امنیت و نظارت استفاده شود. به طور خاص، ابزارهای کلاسیک که بر Npcap متکی هستند عبارتند از: تحلیلگرهای شبکه و پروتکل و مانیتورهای شبکه .
* سپس باید درایور دستگاه wifi خود را در تابع udp\_sniffer قرار دهید . که برای من Intel® Dual Band Wireless-AC 8260 است.
* سپس باید در محیط پایتون کتاب خانه های scapy , pandas , torch , sklearn , numpy را نصب کنیم .
* کد را اجرا میکنیم و وارد یک مرورگر میشویم و چیزی را سرچ میکنیم .
* بعد از سرچ در محیط کنسول پکت های شبکه و نوع حمله مشخص میشود.

def extract\_features(packet):  
 features = {}  
 if IP not in packet:  
 return None  
 features['id'] = packet[IP].id  
 features['dur'] = packet.time - packet[IP].time  
 features['proto'] = packet[IP].proto  
  
 if TCP in packet:  
 features['service'] = 'tcp'  
 features['state'] = packet[TCP].flags  
 elif UDP in packet:  
 features['service'] = 'udp'  
 features['state'] = '-'  
 else:  
 return None  
   
 features['spkts'] = packet[IP].len - packet[IP].ihl \* 4 - len(packet[TCP] if TCP in packet else packet[UDP])  
 features['dpkts'] = len(packet[TCP] if TCP in packet else packet[UDP])  
 features['sbytes'] = packet[IP].len - packet[IP].ihl \* 4 - len(packet[TCP] if TCP in packet else packet[UDP])  
 features['dbytes'] = len(packet[TCP] if TCP in packet else packet[UDP])  
 features['rate'] = features['spkts'] / features['dur']  
 features['sttl'] = packet[IP].ttl  
 features['dttl'] = packet[TCP].window if TCP in packet else packet[UDP].chksum  
 features['sload'] = features['sbytes'] / features['dur']  
 features['dload'] = features['dbytes'] / features['dur']  
 features['sloss'] = 0 if TCP in packet else packet[IP].frag  
 features['dloss'] = packet[IP].tos  
 features['sinpkt'] = 0 if features['spkts'] == 0 else features['dur'] / features['spkts']  
 features['dinpkt'] = 0 if features['dpkts'] == 0 else features['dur'] / features['dpkts']  
 features['sjit'] = 0 if features['spkts'] == 0 else sum(  
 abs(features['sinpkt'] - features['dur'] / features['spkts']) for i in range(1, features['spkts'])) / (  
 features['spkts'] - 1)  
 features['djit'] = 0 if features['dpkts'] == 0 else sum(  
 abs(features['dinpkt'] - features['dur'] / features['dpkts']) for i in range(1, features['dpkts'])) / (  
 features['dpkts'] - 1)  
 features['swin'] = packet[TCP].window if TCP in packet else 0  
 features['stcpb'] = packet[TCP].options[0][1] if TCP in packet and len(packet[TCP].options) > 0 else 0  
 features['dtcpb'] = packet[TCP].options[1][1] if TCP in packet and len(packet[TCP].options) > 1 else 0  
 features['dwin'] = packet[TCP].options[2][1] if TCP in packet and len(packet[TCP].options) > 1 else 0  
  
 if TCP in packet:  
 features['tcprtt'] = 0  
 features['synack'] = int('S' in features['state'] and 'A' in features['state'])  
 features['ackdat'] = int('A' in features['state'] and 'S' not in features['state'])  
 features['smean'] = packet[TCP].sport  
 features['dmean'] = packet[TCP].dport  
 features['trans\_depth'] = 0  
 features['response\_body\_len'] = 0  
 features['ct\_srv\_src'] = 0  
 features['ct\_state\_ttl'] = 0  
 features['ct\_dst\_ltm'] = 0  
 features['ct\_src\_dport\_ltm'] = 0  
 features['ct\_dst\_sport\_ltm'] = 0  
 features['ct\_dst\_src\_ltm'] = 0  
 features['is\_ftp\_login'] = 0  
 features['ct\_ftp\_cmd'] = 0  
 features['ct\_flw\_http\_mthd'] = 0  
 features['ct\_src\_ltm'] = 0  
 features['ct\_srv\_dst'] = 0  
 features['is\_sm\_ips\_ports'] = 0  
  
 elif UDP in packet:  
 features['tcprtt'] = 0  
 features['synack'] = 0  
 features['ackdat'] = 0  
 features['smean'] = packet[UDP].sport  
 features['dmean'] = packet[UDP].dport  
 features['trans\_depth'] = 0  
 features['response\_body\_len'] = 0  
 features['ct\_srv\_src'] = 0  
 features['ct\_state\_ttl'] = 0  
 features['ct\_dst\_ltm'] = 0  
 features['ct\_src\_dport\_ltm'] = 0  
 features['ct\_dst\_sport\_ltm'] = 0  
 features['ct\_dst\_src\_ltm'] = 0  
 features['is\_ftp\_login'] = 0  
 features['ct\_ftp\_cmd'] = 0  
 features['ct\_flw\_http\_mthd'] = 0  
 features['ct\_src\_ltm'] = 0  
 features['ct\_srv\_dst'] = 0  
 features['is\_sm\_ips\_ports'] = 0

**کد 2**

با استفاده از کد بالا توانستیم ویژگی های پکت های شبکه را استخراج کنیم. توضیحات آن به این صورت است که با استفاده از کتابخانه scapy میتوان ویژگی های داخل پکت های شبکه را بخش بخش استخراج کرد. همچنین بخشی از فرمول های نوشته شده هم از طریق جستجو بدست آمده است.

df = pd.DataFrame([features])  
df.to\_csv('my\_file.csv', index=False, header=True)  
print(df)  
*# -----------------------------------------------------------*dataset\_attack\_cat = DataSet\_UNSWNB15\_attack\_cat("my\_file.csv")  
*# dataset\_label = DataSet\_UNSWNB15\_label("my\_file.csv")*PATH = f'model\_attack.pt'  
my\_model\_attack = Model\_Attack\_Cat()  
model\_state\_dict = torch.load(PATH)  
my\_model\_attack.load\_state\_dict(model\_state\_dict)  
  
*# print(my\_model\_attack)*test\_attack\_cat\_dataloader = DataLoader(dataset=dataset\_attack\_cat, batch\_size=1, shuffle=True)  
  
for batch\_x in test\_attack\_cat\_dataloader:  
 out = my\_model\_attack(batch\_x)  
 print(out)  
 max\_elements, max\_index = torch.max(out, 1)  
 *# print(max\_elements)  
 # print(max\_index)* define\_attck\_category(max\_index)  
 print("------------------------------")

**کد 3**

حال مقادیر بدست آمده را به قسمت ارزایابی مدل میدهیم تا تعیین کند که پکت مدنظر چه نوعی را داشته.

# فصل 5 - نتیجه گیری و کارهای آتی

## 5-1- توضیحات مجموعه داده

بسته های شبکه خام مجموعه داده UNSW-NB 15[22] توسط ابزار IXIA PerfectStorm در آزمایشگاه Cyber ​​Range در UNSW Canberra برای ایجاد ترکیبی از فعالیت های عادی مدرن واقعی و رفتارهای حمله مصنوعی معاصر ایجاد شده است. ابزار tcpdump [22] برای ضبط 100 گیگابایت از ترافیک خام استفاده شده. این مجموعه داده دارای 9 نوع حمله است که عبارتند از Fuzzers، Analysis، Backdoors، DoS، Exploits، Generic، Reconnaissance، Shellcode و . Worms از ابزارهای Argus و Bro-IDS برای استخراج ویژگی های پکت‌های شبکه استفاده می‌شود [22]. از روی داده های خام حاصله 49 ویژگی با برچسب کلاس حاصل شده است. تعداد کل رکوردها 2540044 است که در چهار فایل CSV به نام‌های UNSW-NB15\_1.csv، UNSW-NB15\_2.csv، UNSW-NB15\_3.csv و UNSW-NB15\_4.csv [22] ذخیره می‌شوند. افرازی از این مجموعه داده به عنوان یک مجموعه آموزشی و مجموعه آزمایشی، به ترتیب UNSW\_NB15\_training-set.csv و UNSW\_NB15\_testing-set.csv [22] پیکربندی شده است. تعداد رکوردهای مجموعه آموزشی 175341 رکورد و مجموعه تست 82332 رکورد از انواع مختلف بسته های شامل حمله و بسته های عادی است. این جمموعه داده ها، بر اساس به صورت دو کلاسه و نه کلاسه در این پژوهش مورد استفاده قرار گرفته است که اطلاعات آن به ترتیب در جدول 1 و جدول 2 مشاهده می شود.

**جدول 1- تعداد حملات چند کلاسه**

|  |  |
| --- | --- |
| تعداد | نوع حمله |
| 93000 | Normal |
| 58871 | Generic |
| 44525 | Exploits |
| 24246 | Fuzzers |
| 16353 | DoS |
| 13987 | Reconnaissance |
| 2677 | Analysis |
| 2329 | Backdoor |
| 1511 | Shellcode |
| 174 | Worms |
| 2540044 | Total |

**جدول 2- تعداد حملات دو کلاسه**

|  |  |
| --- | --- |
| تعداد | نوع |
| 93000 | Normal |
| 164673 | Attack |

## 5-2- نتایج آزمایش مدل های یادگیری عمیق

در هر کدام از مدل‌ها، به دو صورت طبقه بندی داشتیم. هم به صورت چند کلاسه و هم به صورت دودویی. همچنین قبل از انجام عملیات آموزش مدل، پیش پردازش و نرمال‌سازی داده‌ها را انجام دادیم. و داده‌ها را به دو قسمت تقسیم کردیم که هفتاد درصد داده‌ها برای آموزش و سی درصد داده‌ها را به ارزیابی مدل اختصاص دادیم. معیارهای ارزیابی دقت، مقدار خطا و زمان آموزش به صورت یک دوره است. میانگین دقت اعتبارسنجی، میانگین خطا در آموزش محاسبه می‌شود تا نشان دهد که در مجموعه داده هیچ تناسب [[28]](#footnote-29)یا کم‌برازشی[[29]](#footnote-30) وجود ندارد. با توجه به تعداد دوره‌هایی که آموزش را تکمیل کرده‌اند، زمان آموزش را برای هر دوره دریافت می‌کنیم و از آن به عنوان معیار در ارزیابی ما استفاده می‌کنیم.

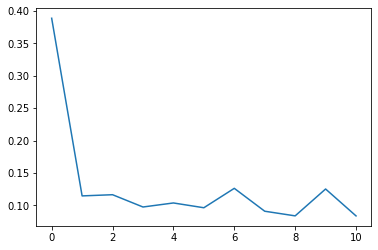
## 5-3- نتایج شبکه عصبی عمیق

ما مجموعا 59 مدل با پارامترهای مختلف را در این شبکه ساختیم. نتایج مدل شبکه عصبی عمیق پیشنهادی ما برای هر دو طبقه‌بندی در جدول 6 نشان داده شده است. جدول نتایج شبکه عصبی مصنوعی برای برچسب‌گذاری دودویی و برچسب‌گذاری چند کلاسه را از نظر دقت، خطا و زمان آموزش برای هر دوره نشان می‌دهد. شایان ذکر است، عملکرد شبکه عصبی مصنوعی در طبقه‌بندی باینری بهتر از طبقه‌بندی چند کلاسه بود. دقت برای مجموعه آزمایشی 94٪ برای طبقه‌بندی دودویی و 82.46٪ برای دقت طبقه‌بندی چند کلاسه بود. در بخش آموزش خطا طبقه‌بندی دودویی0.1137و 0.3954 در طبقه‌بندی چند کلاسه بود. همچنین مدت زمان آموزش برای طبقه‌بندی دودیی 986375 میلی ثانیه و برای طبقه‌بندی چند کلاسه 1009266 میلی ثانیه است.

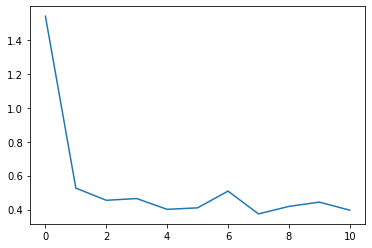
**جدول 7- نتایج مدل شبکه عصبی عمیق**

|  |  |  |
| --- | --- | --- |
| معیار | طبقه‌بندی دودویی | طبقه‌بندی چند کلاسه |
| دقت آزمایش | 94٪ | 82.46٪ |
| خطای آموزش | 0.1137 | 0.3954 |
| مدت زمان آموزش | 986375 ms | 1009266 ms |

همچنین نمودار خطای طبقه‌بندی چند کلاسه و دو کلاسه به صورت زیر است .



**شکل 7- نمودار خطای طبقه‌بندی دو کلاسه**



**شکل 6- نمودار خطای طبقه‌بندی چند کلاسه**

## 5-4- معیارهای ارزیابی Recall - Precision و F1-score

**معیار Recall یا یادآوری**

حداکثر مقدار این معیار یک ویا ۱۰۰ درصد و حداقل مقدار آن صفر است و هرچه مواردی که ما انتظار داشتیم پیش بینی شوند ولی برنامه پیش‌بینی نکرده‌است که به آن False Negative می‌گوییم نسبت به پیش بینی‌های درست یا True Positive بیشتر باشد مقدار Recall کمتر خواهد شد.

**فرمول محاسبه‌ Recall**  
در فرمول‌ زیر TP مخفف True Positive و FN مخفف False Negative است.

**Recall =** ​

**معیار Precision یا دقت**

حداکثر مقدار این معیار یک ویا ۱۰۰ درصد و حداقل مقدار آن صفر است و هرچه مواردی که برنامه‌ به **غلط پیش بینی کرده است** که به آن False Positive می‌گوییم نسبت به پیش بینی‌های درست یا True Positive بیشتر باشد مقدار Precision کمتر خواهد شد.

**فرمول محاسبه‌ی Precision**  
در فرمول‌ زیر TP مخفف True Positive و FP مخفف False Positive است.

**Precision =**

**معیار f1-score**

زمانی که می‌خواهید معیار ارزیابی شما میانگینی از دو مورد قبلی باشد یعنی همان Recall یا Precision می‌توانید از میانگین هارمونیک این دو معیار استفاده کنید که به آن **معیار f1-score** می‌گویند.

**F1-score =**

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه به صورت زیر است.

**جدول 8- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

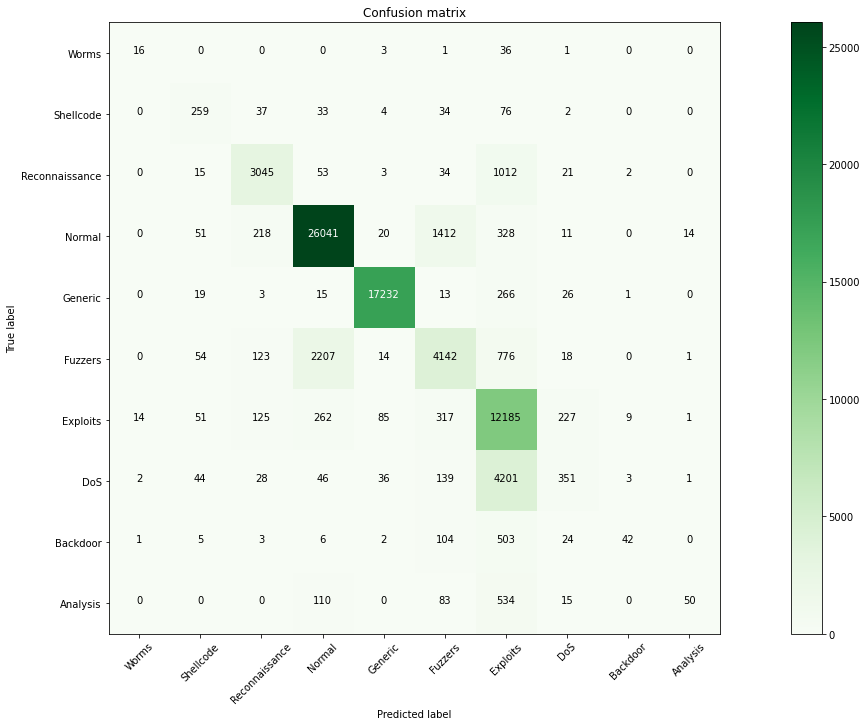
|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.12 | 0.06 | 0.75 | 0 |
| 0.11 | 0.06 | 0.74 | 1 |
| 0.13 | 0.07 | 0.50 | 2 |
| 0.73 | 0.92 | 0.61 | 3 |
| 0.61 | 0.56 | 0.66 | 4 |
| 0.99 | 0.98 | 0.99 | 5 |
| 0.92 | 0.93 | 0.91 | 6 |
| 0.78 | 0.73 | 0.85 | 7 |
| 0.55 | 0.58 | 0.52 | 8 |
| 0.36 | 0.28 | 0.48 | 9 |

همچنین معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی دو کلاسه به صورت زیر است.

**جدول 9- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی دو کلاسه**

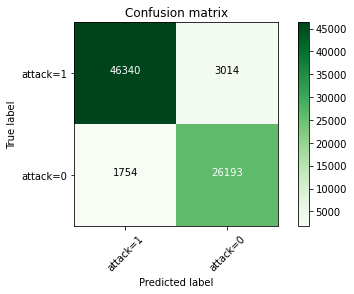
|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.92 | 0.94 | 0.90 | 0 |
| 0.95 | 0.94 | 0.96 | 1 |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است



**شکل 8- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه به صورت زیر است.



**شکل 9- ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه**

## 5-5- نتایج آزمایش مدل DNN در یک حالت دیگر

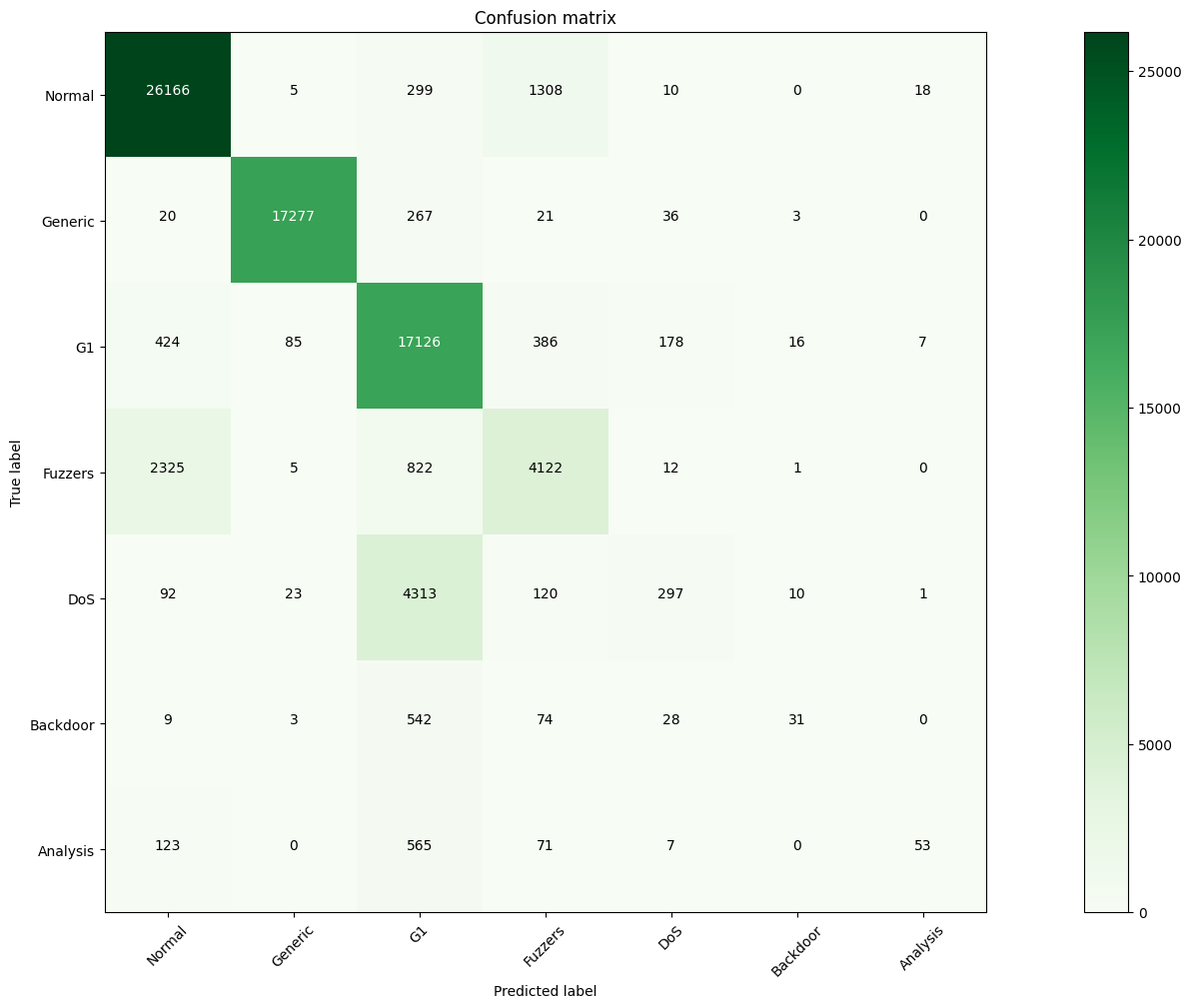
چهار دسته از حملات که شامل Worms، Shellcode، Reconnaissance، Exploits تبدیل به گروه G1 میکنیم و در کنار سایر حملات قرار میدهیم. این آزمایش را یکبار با دسته‌بندی نرمال و یک بار بدون این دسته بندی انجام میدهیم. مدل را مثل حالت قبل آموزش میدهیم و ارزیابی میکنیم.

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه با در نظر گرفتن کلاس نرمال به صورت زیر است.

**جدول 8- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | Precision | کلاس |
| 0.12 | 0.06 | 0.67 | 0 |
| 0.08 | 0.05 | 0.51 | 1 |
| 0.11 | 0.06 | 0.52 | 2 |
| 0.62 | 0.57 | 0.68 | 3 |
| 0.81 | 0.94 | 0.72 | 4 |
| 0.99 | 0.98 | 0.99 | 5 |
| 0.92 | 0.94 | 0.90 | 6 |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است.



**شکل 9- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

در آزمایش بعدی همین کار را تکرار میکنیم منتها بدون مجموعه کلاس نرمال .

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه با در نظر گرفتن کلاس نرمال به صورت زیر است.

**جدول 8- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.29 | **0.18** | **0.79** | **0** |
| 0.09 | **0.05** | **0.63** | **1** |
| 0.14 | **0.08** | **0.49** | **2** |
| 0.86 | **0.86** | **0.87** | **3** |
| 0.82 | **0.95** | **0.72** | **4** |
| 0.99 | **0.98** | **0.99** | **5** |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است.



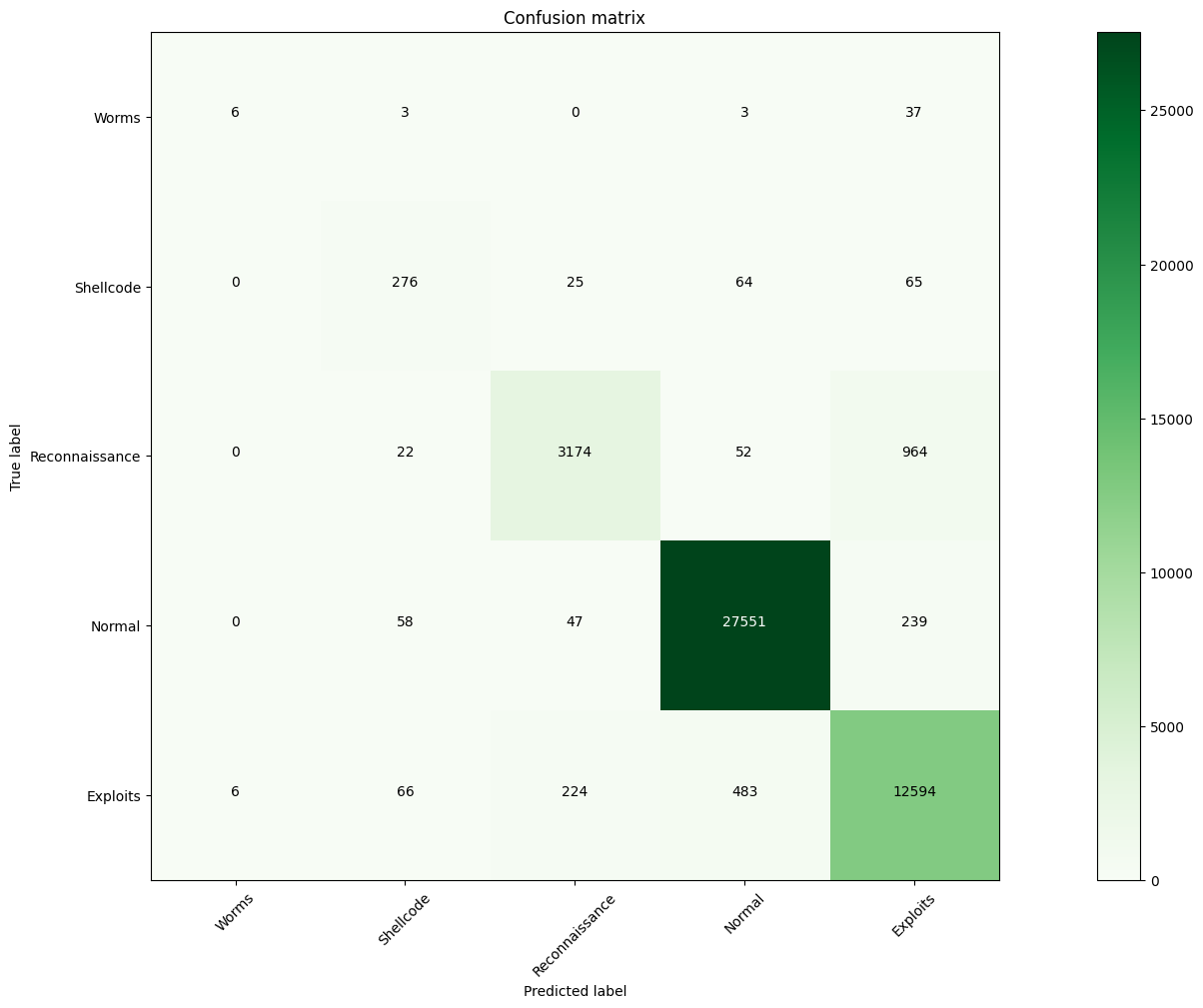
**شکل 9- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

در آزمایش بعدی این چهار دسته حمله که در بالا گفته شد را همراه مجموعه کلاس نرمال مورد آموزش و بعد ارزیابی قرار میدهیم .

**جدول 8- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.92 | **0.94** | **0.91** | **0** |
| 0.98 | **0.99** | **0.98** | **1** |
| 0.83 | **0.75** | **0.91** | **2** |
| 0.65 | **0.64** | **0.65** | **3** |
| 0.20 | **0.12** | **0.50** | **4** |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است.



**شکل 9- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

در آزمایش بعدی همین کار را تکرار میکنیم منتها بدون مجموعه کلاس نرمال .

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه با در نظر گرفتن کلاس نرمال به صورت زیر است.

**جدول 8- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.95 | **0.97** | **0.92** | **0** |
| 0.84 | **0.78** | **0.90** | **1** |
| 0.64 | **0.59** | **0.71** | **2** |
| 0.0 | **0.0** | **0.0** | **3** |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است.



**شکل 9- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

## 5-6- نتایج آزمایش مدل DNN در یک دسته دیگر

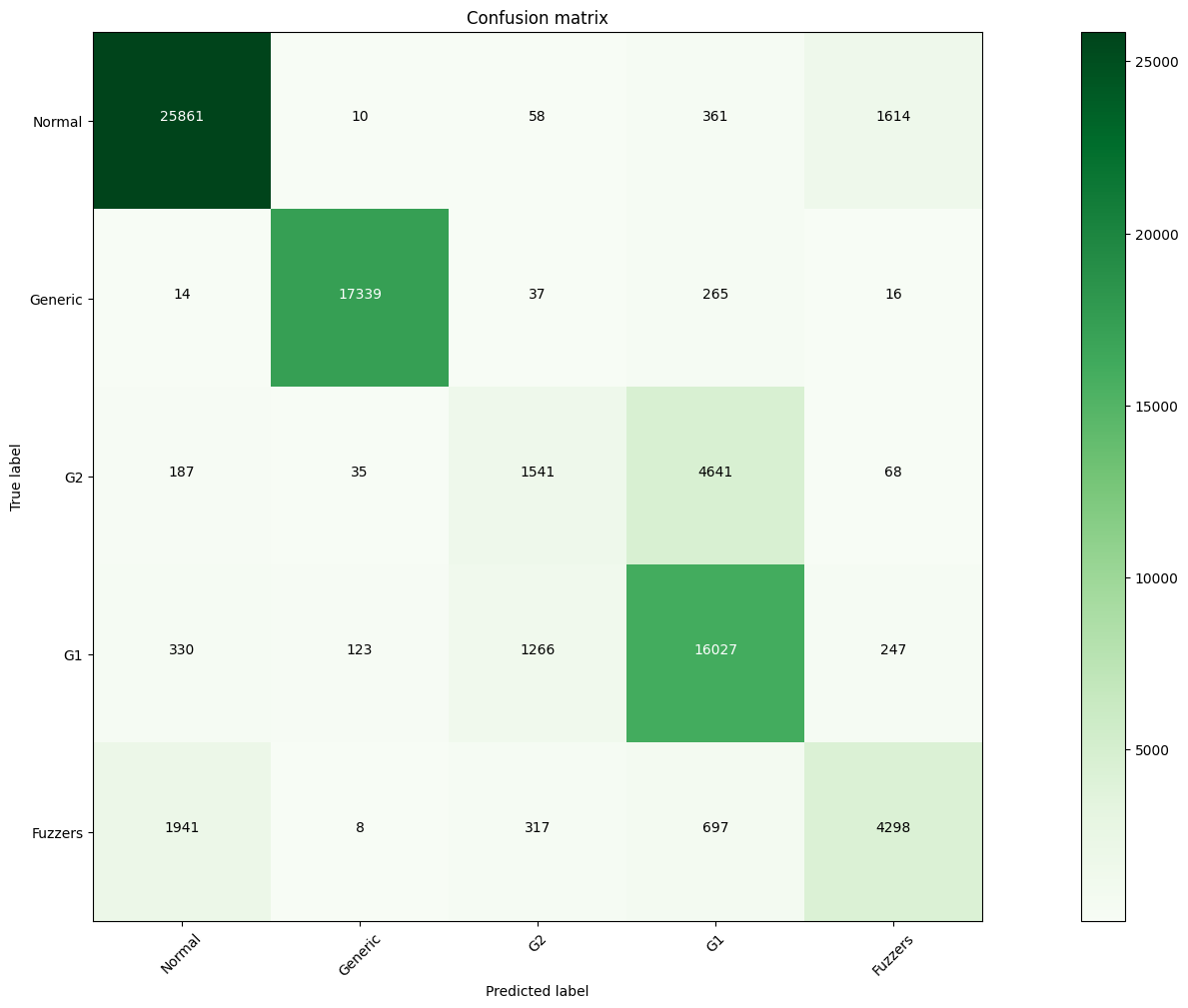
در قسمت قبل چهار دسته از حملات که شامل Worms، Shellcode، Reconnaissance، Exploits تبدیل به گروه G1 کردیم و سه دسته از حملات شامل Dos , Backdoor, Analysis تبدیل به گروه G2 میکنیم و در کنار سایر حملات قرار میدهیم. این آزمایش را یکبار با دسته‌بندی نرمال و یک بار بدون این دسته بندی انجام میدهیم. مدل را مثل حالت قبل آموزش میدهیم و ارزیابی میکنیم.

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه با در نظر گرفتن کلاس نرمال به صورت زیر است.

**جدول 8- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | Precision | کلاس |
| 0.64 | 0.59 | 0.69 | 0 |
| 0.80 | 0.89 | 0.73 | 1 |
| 0.32 | 0.24 | 0.48 | 2 |
| 0.99 | 0.98 | 0.99 | 3 |
| 0.92 | 0.93 | 0.91 | 4 |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است.



**شکل 9- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

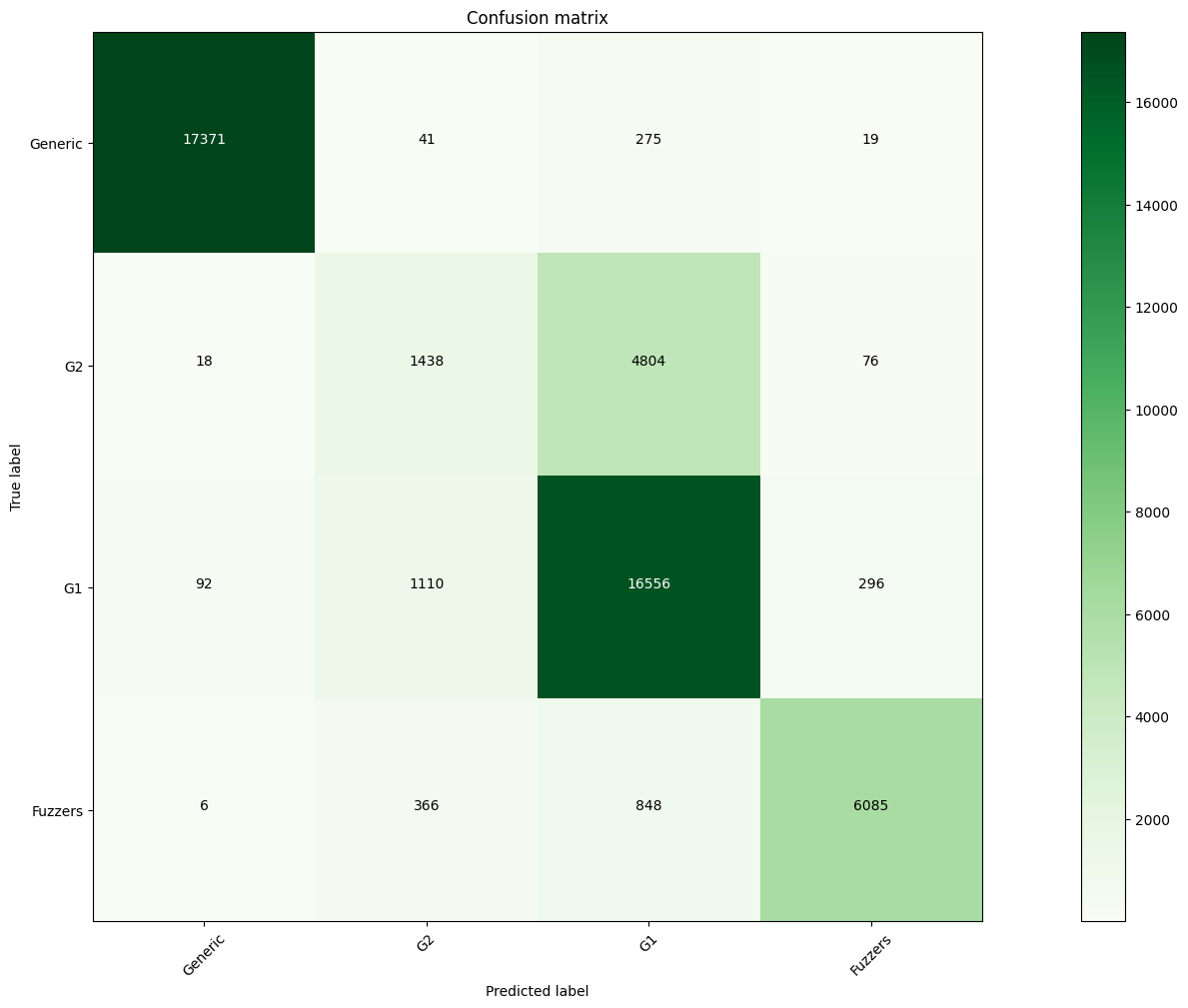
در آزمایش بعدی همین کار را تکرار میکنیم منتها بدون مجموعه کلاس نرمال .

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه با در نظر گرفتن کلاس نرمال به صورت زیر است.

**جدول 8- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.88 | **0.83** | **0.94** | **0** |
| 0.82 | **0.92** | **0.74** | **1** |
| 0.31 | **0.23** | **0.49** | **2** |
| 0.99 | **0.98** | **0.99** | **3** |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است.



**شکل 9- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

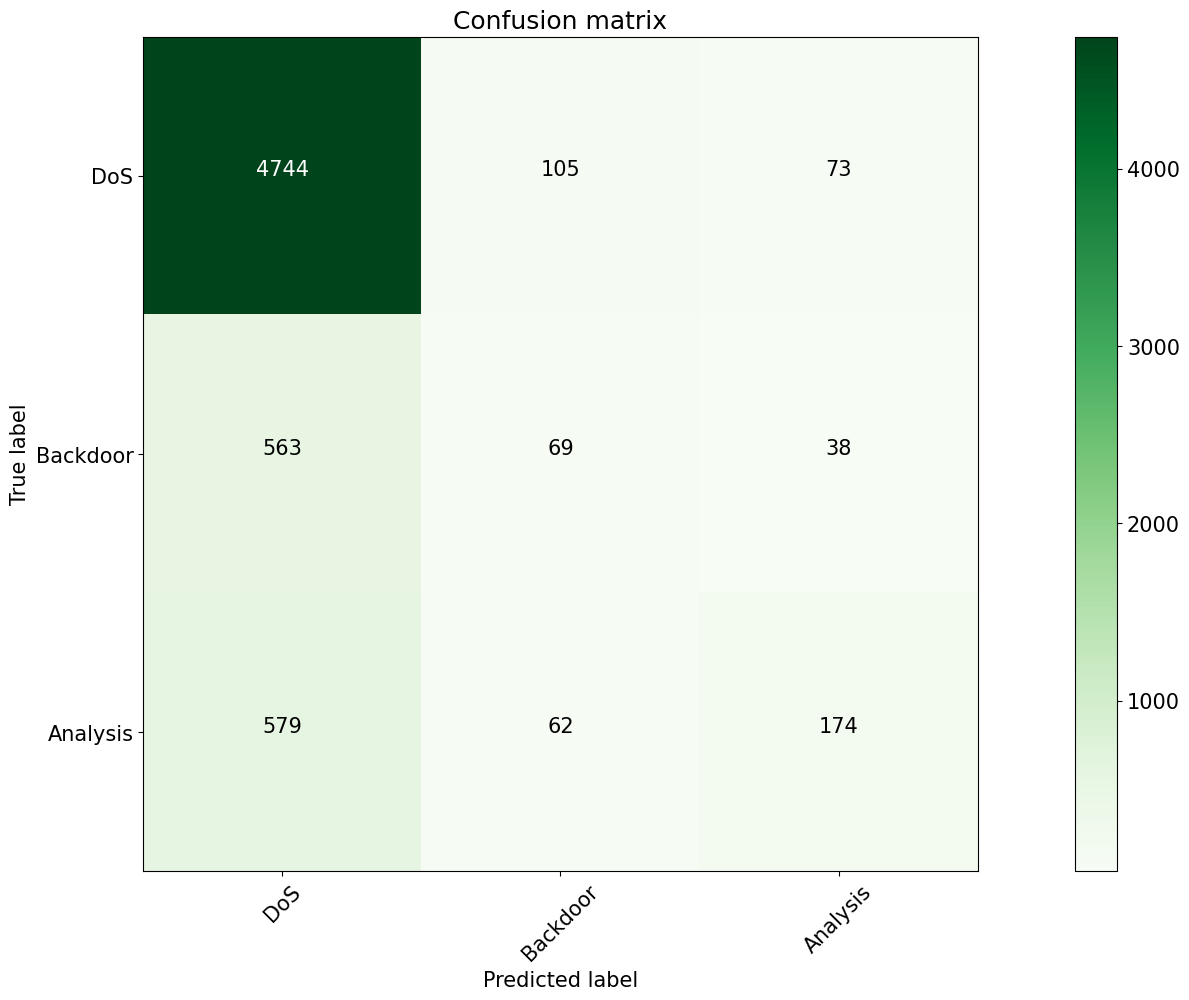
در آزمایش بعدی این چهار دسته و سه دسته حمله که در بالا گفته شد را بدون مجموعه کلاس نرمال مورد آموزش و بعد ارزیابی قرار میدهیم .

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه با در نظر گرفتن کلاس نرمال به صورت زیر است.

**جدول 8- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.35 | **0.25** | **0.58** | **0** |
| 0.18 | **0.11** | **0.43** | **1** |
| 0.88 | **0.97** | **0.81** | **2** |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است.



**شکل 9- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

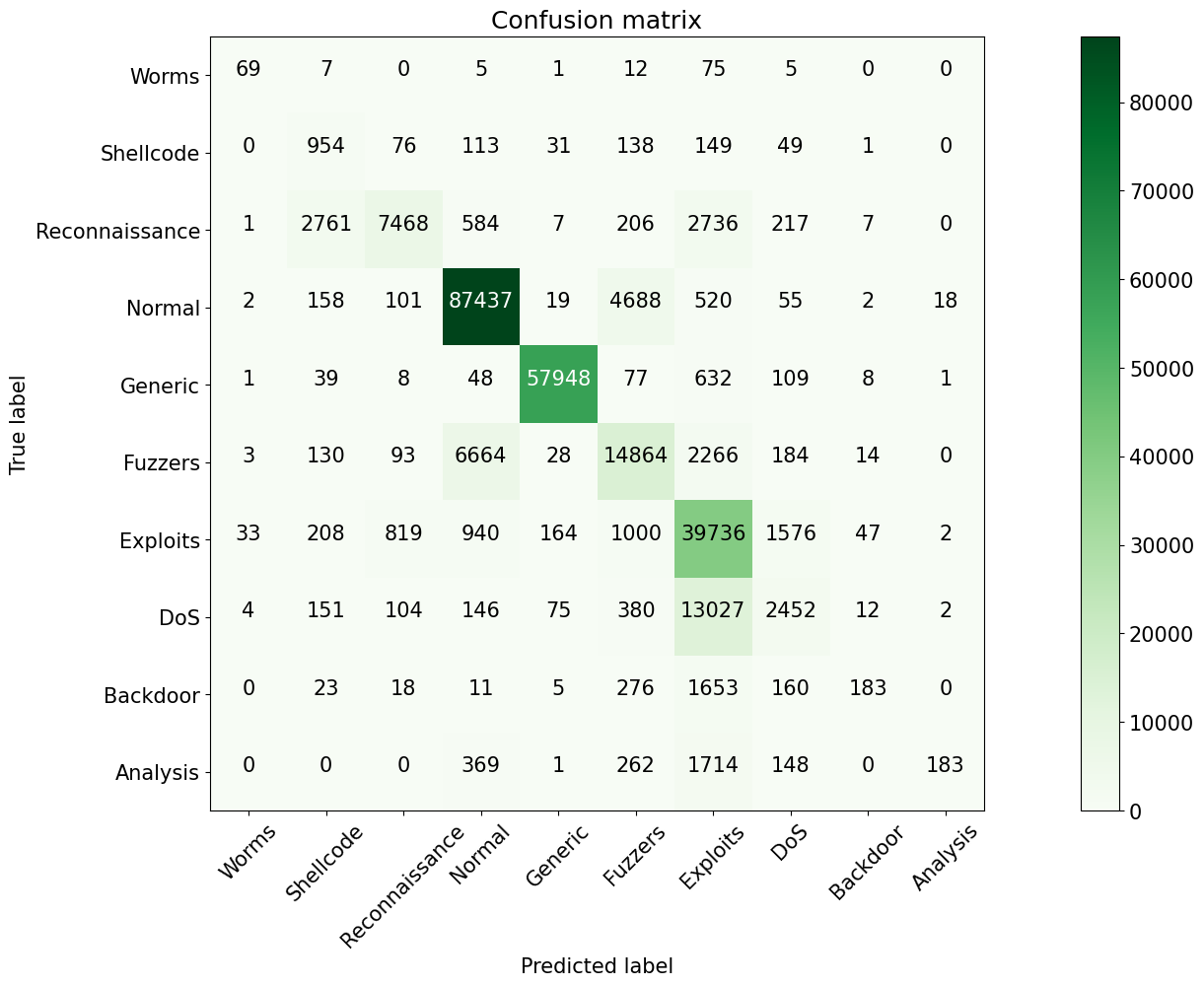
## 5-7- نتایج آزمایش مدل DNN 70 به 100 و مدل ترکیبی

برای مدل DNN، مجموعه داده را به 70 درصد آموزش و 100 درصد آزمایش تبدیل میکنیم.

دقت کلی 82% شد و معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه به صورت زیر است.

**جدول 14- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.13 | 0.07 | 0.89 | 0 |
| 0.14 | 0.08 | 0.67 | 1 |
| 0.23 | 0.15 | 0.49 | 2 |
| 0.74 | 0.89 | 0.64 | 3 |
| 0.64 | 0.61 | 0.68 | 4 |
| 0.99 | 0.98 | 0.99 | 5 |
| 0.92 | 0.94 | 0.91 | 6 |
| 0.66 | 0.53 | 0.86 | 7 |
| 0.32 | 0.63 | 0.22 | 8 |
| 0.48 | 0.4 | 0.61 | 9 |



**شکل 12- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

در آزمایش بعدی که یک آزمایش ترکیبی است باید در ابتدا مدلی که شامل دسته بندی های G1 , G2 و سه دسته عادی بود را به نسبت 70 به 100 ارزیابی کنیم و نتایج مربوط به گروه G1 را به مدل MG1 بدهیم و همچنین نتایج مربوط به گروه G2 را به مدل MG2 بدهیم تا به تفکیک ارزیابی‌ها را انجام دهند.

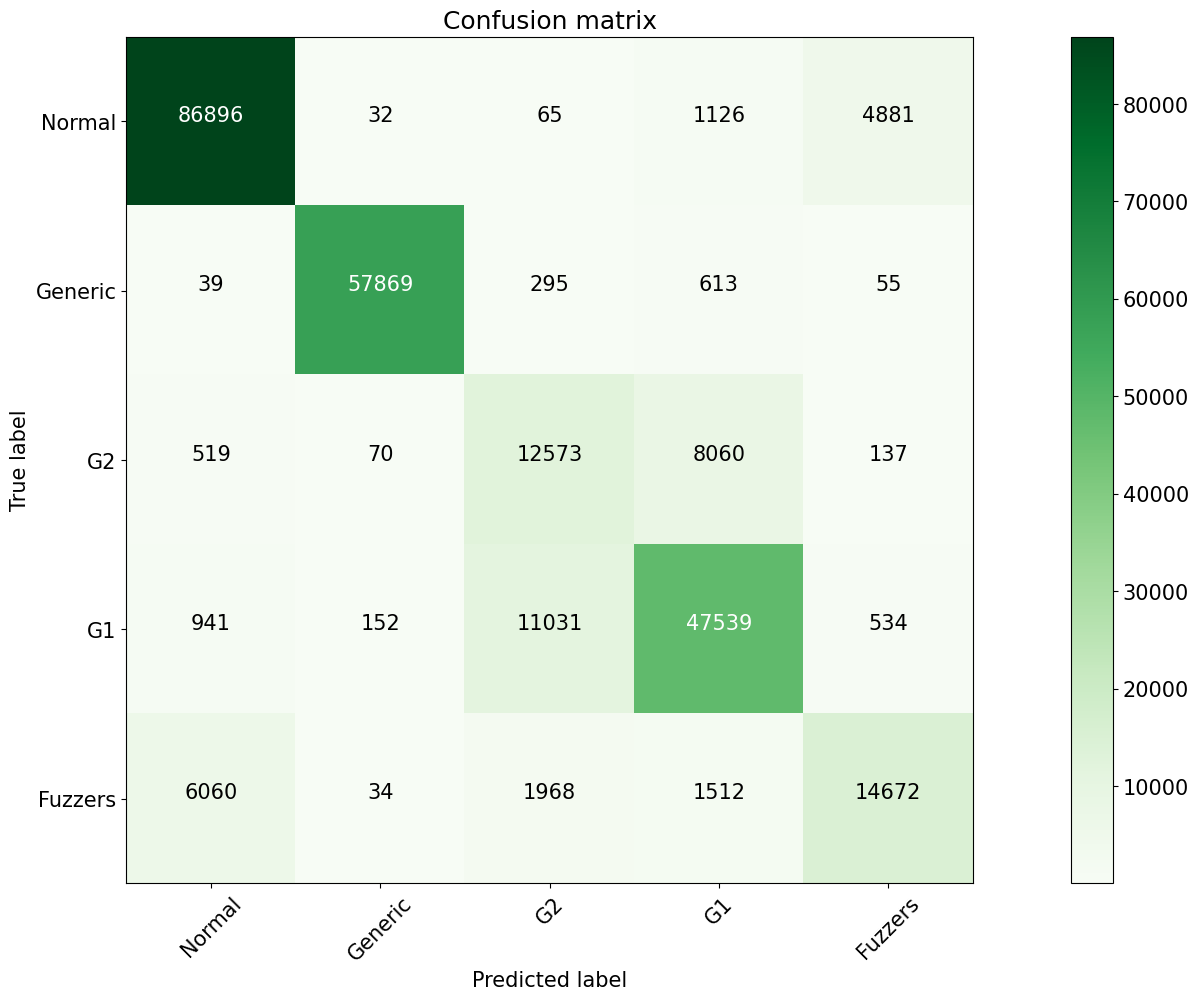
در ابتدای آزمایش که کل مجموعه داده را به نسبت 70 به 100 ارزیابی میکنیم، نتایج به صورت زیر است.

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه با در نظر گرفتن کلاس نرمال به صورت زیر است.

**جدول 8- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | Precision | کلاس |
| 0.66 | 0.61 | 0.72 | 0 |
| 0.8 | 0.79 | 0.81 | 1 |
| 0.53 | 0.59 | 0.48 | 2 |
| 0.99 | 0.98 | 1.00 | 3 |
| 0.93 | 0.93 | 0.92 | 4 |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است.



**شکل 12- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

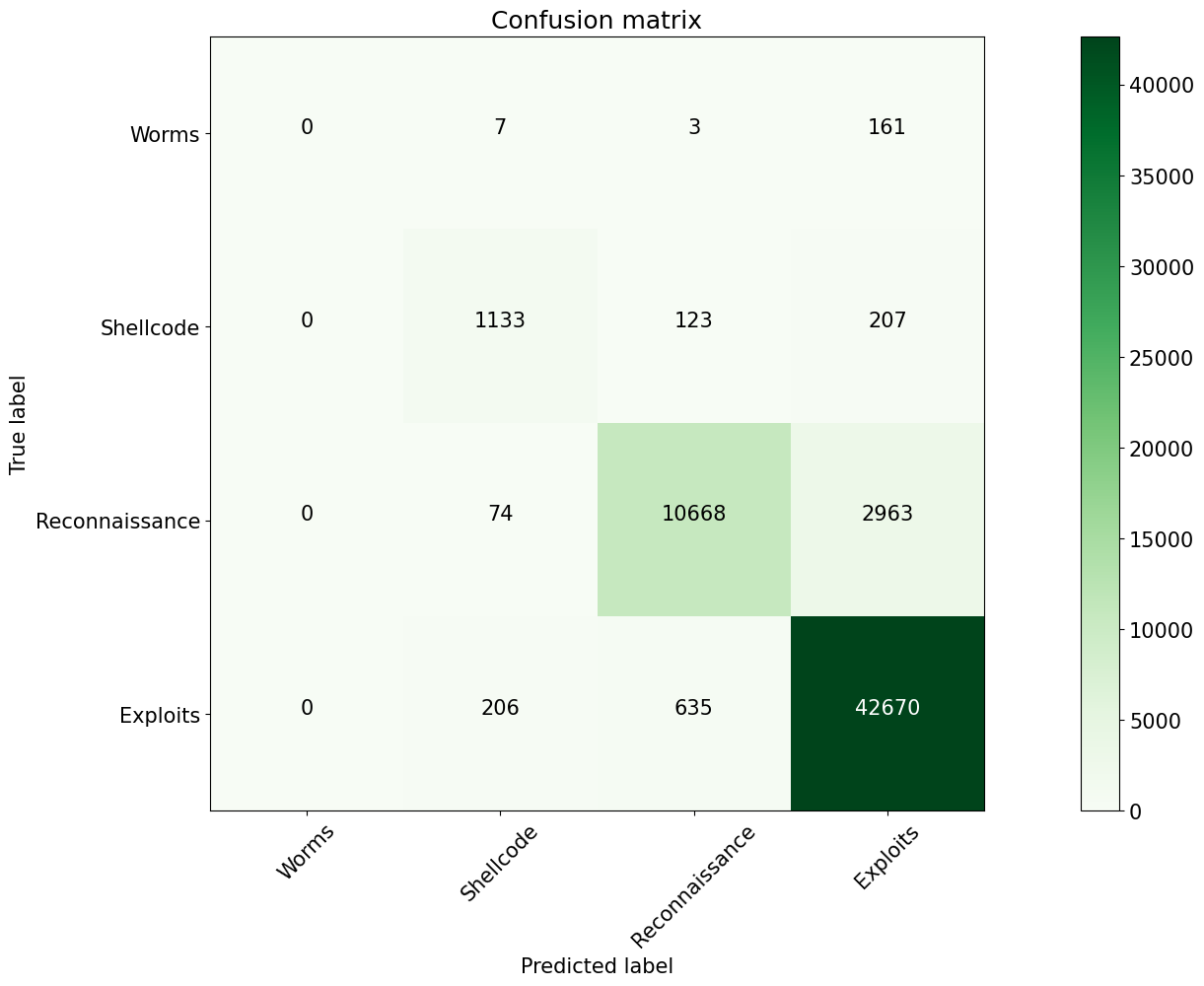
در ادامه گروه G1 را به مدل MG1 میدهیم.

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه با در نظر گرفتن کلاس نرمال به صورت زیر است.

**جدول 8- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | Precision | کلاس |
| 0.95 | 0.98 | 0.93 | 0 |
| 0.85 | 0.78 | 0.93 | 1 |
| 0.79 | 0.77 | 0.8 | 2 |
| 0.00 | 0.00 | 0.00 | 3 |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است.



**شکل 12- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

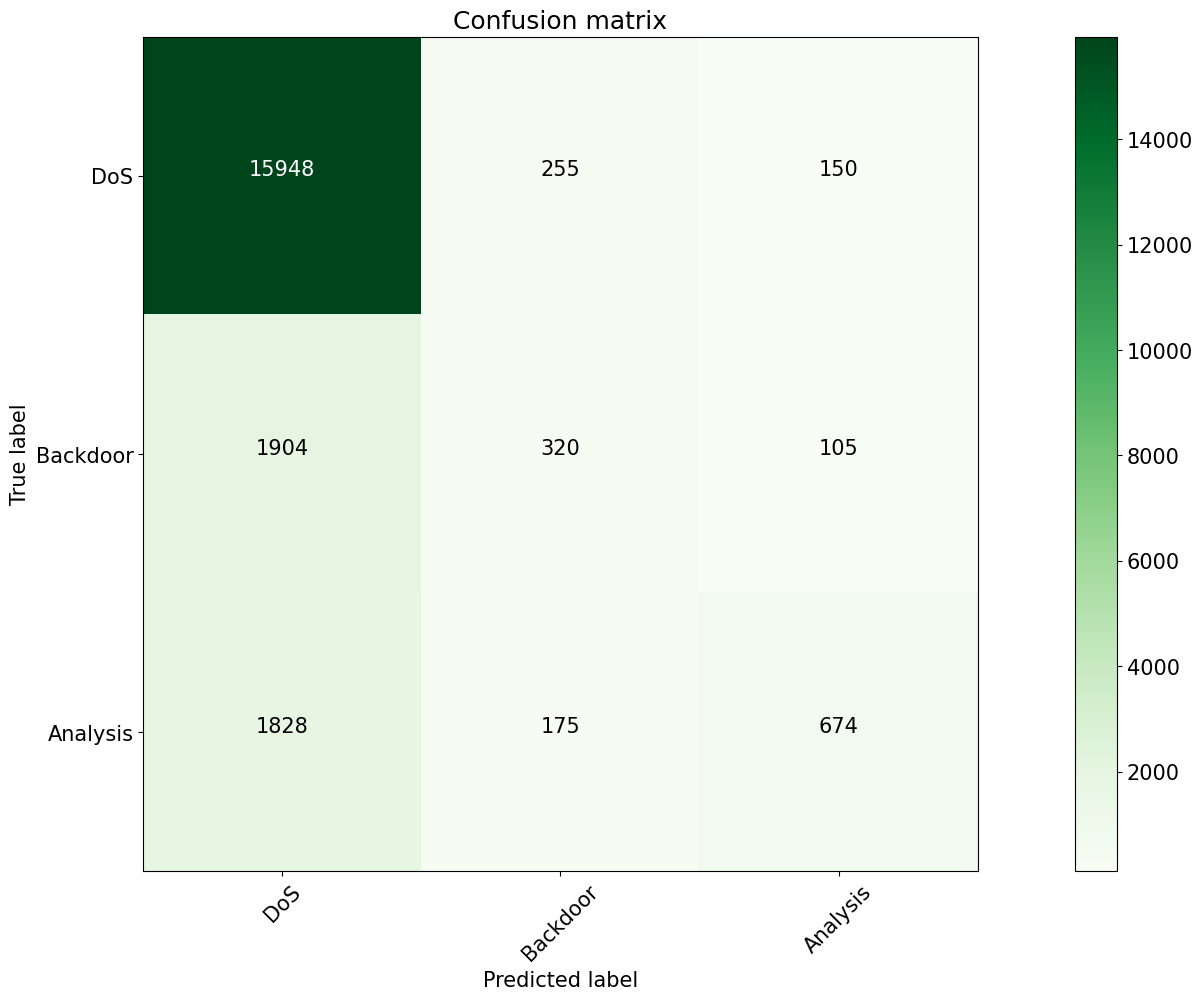
و همچنین نتایج مربوط به گروه G2 را به مدل MG2 بدهیم تا به تفکیک ارزیابی‌ها را انجام دهند.

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه با در نظر گرفتن کلاس نرمال به صورت زیر است.

**جدول 8- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.37 | **0.25** | **0.73** | **0** |
| 0.21 | **0.14** | **0.43** | **1** |
| 0.89 | **0.98** | **0.81** | **2** |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است.



**شکل 9- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

## 5-6- نتایج شبکه عصبی بازگشتی

ما مجموعا 67 مدل با پارامترهای مختلف را در این شبکه ساختیم. نتایج مدل شبکه عصبی بازگشتی پیشنهادی ما برای هر دو طبقه‌بندی در جدول 10 نشان داده شده است. جدول نتایج شبکه عصبی مصنوعی LSTM برای برچسب‌گذاری دودویی و برچسب‌گذاری چند کلاسه را از نظر دقت، خطا و زمان آموزش برای هر دوره نشان می‌دهد. دقت برای مجموعه آزمایشی93.89 ٪ برای طبقه‌بندی دودویی و 82.33٪ برای دقت طبقه‌بندی چند کلاسه بود. در بخش آموزش خطا طبقه‌بندی دودویی0.0846 و 0.3593 در طبقه‌بندی چند کلاسه بود. همچنین مدت زمان آموزش برای طبقه‌بندی دودیی 1289291میلی ثانیه و برای طبقه‌بندی چند کلاسه 1276302 میلی ثانیه است.

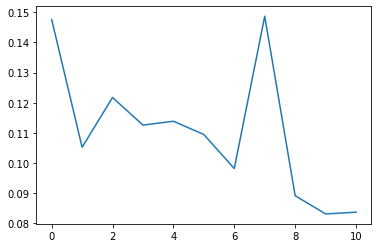
**جدول 10- نتایج شبکه عصبی بازگشتی**

|  |  |  |
| --- | --- | --- |
| معیار | طبقه‌بندی دودویی | طبقه‌بندی چند کلاسه |
| دقت آزمایش | 93.89 ٪ | 82.33٪ |
| خطای آموزش | 0.0846 | 0.3593 |
| مدت زمان آموزش | 1289291ms | 1276302 ms |

همچنین نمودار خطای طبقه‌بندی چند کلاسه دو کلاسه به صورت زیر است .



**شکل 11- نمودار خطای طبقه‌بندی چند کلاسه**



**شکل 10- نمودار خطای طبقه‌بندی دو کلاسه**

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه به صورت زیر است.

**جدول 11- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.12 | 0.06 | 0.81 | 0 |
| 0.15 | 0.08 | 0.76 | 1 |
| 0.24 | 0.17 | 0.42 | 2 |
| 0.73 | 0.88 | 0.62 | 3 |
| 0.63 | 0.59 | 0.67 | 4 |
| 0.99 | 0.98 | 0.99 | 5 |
| 0.92 | 0.93 | 0.91 | 6 |
| 0.82 | 0.74 | 0.92 | 7 |
| 0.56 | 0.53 | 0.59 | 8 |
| 0.34 | 0.3 | 0.39 | 9 |

همچنین معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی دو کلاسه به صورت زیر است.

**جدول 12- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی دو کلاسه**

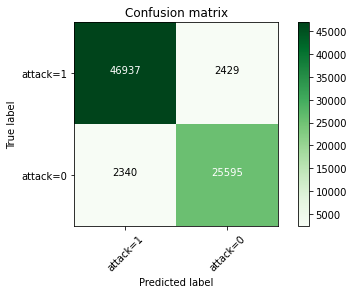
|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.91 | 0.92 | 0.91 | 0 |
| 0.95 | 0.95 | 0.95 | 1 |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است



**شکل 12- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه به صورت زیر است.



**شکل 13- ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه**

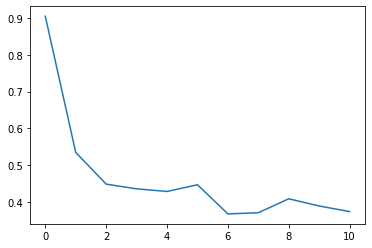
## 5-6- ترکیب دو شبکه عصبی عمیق و شبکه عصبی بازگشتی

ما مجموعا 20 مدل با پارامترهای مختلف را در این شبکه ساختیم. نتایج مدل پیشنهادی ما برای هر دو طبقه‌بندی در جدول 13 نشان داده شده است. جدول نتایج برای برچسب‌گذاری دودویی و برچسب‌گذاری چند کلاسه را از نظر دقت، خطا و زمان آموزش برای هر دوره نشان می‌دهد. دقت برای مجموعه آزمایشی93.97 ٪ برای طبقه‌بندی دودویی و 82.49٪ برای دقت طبقه‌بندی چند کلاسه بود. در بخش آموزش خطا طبقه‌بندی دودویی0.1355 و 0.3482 در طبقه‌بندی چند کلاسه بود. همچنین مدت زمان آموزش برای طبقه‌بندی دودیی 1130231 میلی ثانیه و برای طبقه‌بندی چند کلاسه 1125109 میلی ثانیه است.

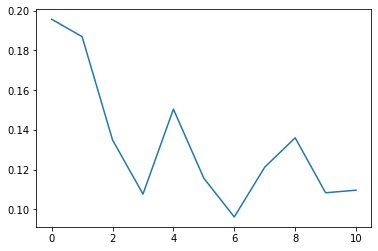
**جدول 13- نتایج مدل ترکیب دو شبکه عصبی عمیق و شبکه عصبی بازگشتی**

|  |  |  |
| --- | --- | --- |
| معیار | طبقه‌بندی دودویی | طبقه‌بندی چند کلاسه |
| دقت آزمایش | 93.97 ٪ | 82.49٪ |
| خطای آموزش | 0.1355 | 0.3482 |
| مدت زمان آموزش | 1130231 ms | 1125109 ms |

همچنین نمودار خطای طبقه‌بندی چند کلاسه و دو کلاسه به صورت زیر است .



**شکل 14- نمودار خطای طبقه‌بندی چند کلاسه**



**شکل 15- نمودار خطای طبقه‌بندی دو کلاسه**

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه به صورت زیر است.

**جدول 14- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

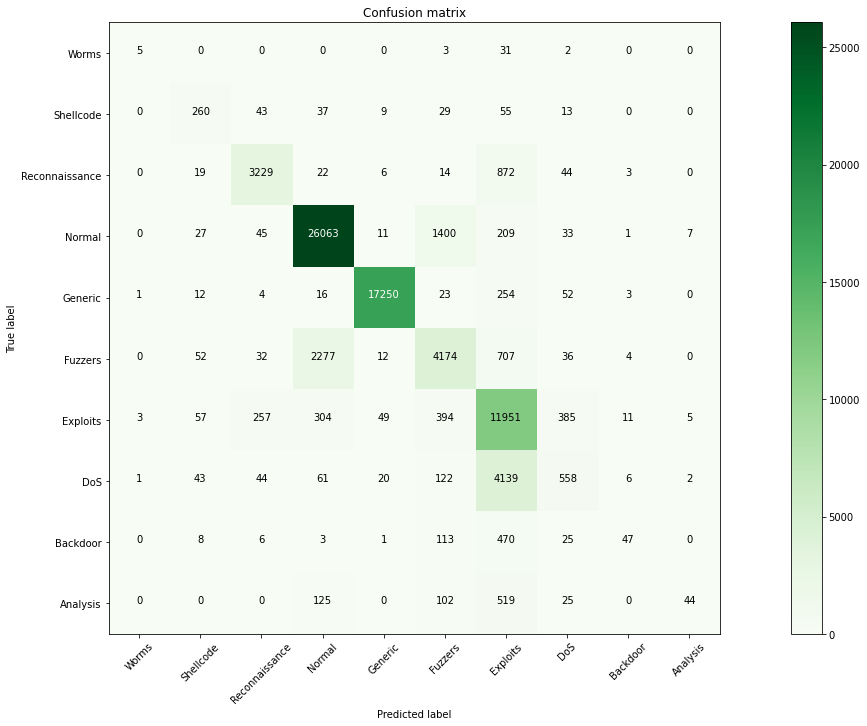
|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.10 | 0.05 | 0.76 | 0 |
| 0.13 | 0.07 | 0.63 | 1 |
| 0.18 | 0.11 | 0.48 | 2 |
| 0.73 | 0.89 | 0.62 | 3 |
| 0.61 | 0.57 | 0.65 | 4 |
| 0.99 | 0.98 | 0.99 | 5 |
| 0.92 | 0.94 | 0.90 | 6 |
| 0.82 | 0.77 | 0.88 | 7 |
| 0.56 | 0.58 | 0.54 | 8 |
| 0.2 | 0.12 | 0.5 | 9 |

همچنین معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی دو کلاسه به صورت زیر است.

**جدول 15- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی دو کلاسه**

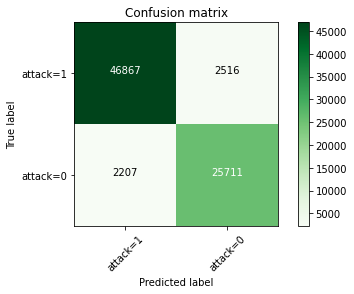
|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.92 | 0.92 | 0.91 | 0 |
| 0.95 | 0.95 | 0.96 | 1 |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه به صورت زیر است



**شکل 16- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه به صورت زیر است.



**شکل 17- ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه**

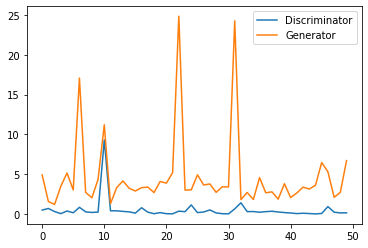
## 5-7- نتایج مدل GAN برای حمله به شبکه عصبی عمیق

نتایج حاصله از اجرای این کار به صورت جدول زیر است.

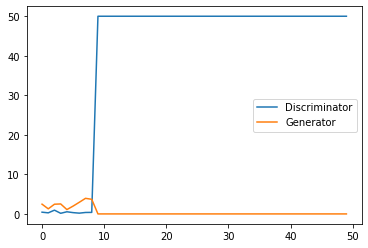
**جدول 17- نتایج مدل GAN برای حمله به شبکه عصبی عمیق**

|  |  |  |
| --- | --- | --- |
| معیار | طبقه‌بندی دودویی | طبقه‌بندی چند کلاسه |
| خطای Generator | 0.0000 | 2.7442 |
| خطای Discriminator | 50.0000 | 0.1510 |

نمودار خطای طبقه‌بندی چند کلاسه و دو کلاسه به صورت زیر است .



**شکل 18- نمودار خطای طبقه‌بندی چند کلاسه**



**شکل 20- نمودار خطای طبقه‌بندی دو کلاسه**

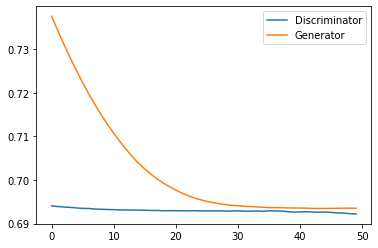
## 5-8- نتایج مدل GAN برای حمله به شبکه عصبی بازگشتی

نتایج حاصله از اجرای این کار به صورت جدول زیر است.

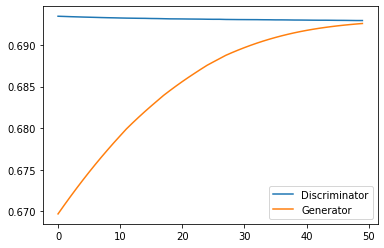
**جدول 19- نتایج مدل GAN برای حمله به شبکه عصبی بازگشتی**

|  |  |  |
| --- | --- | --- |
| معیار | طبقه‌بندی دودویی | طبقه‌بندی چند کلاسه |
| خطای Generator | 0.6935 | 0.6925 |
| خطای Discriminator | 0.6924 | 0.6929 |

همچنین نمودار خطای طبقه‌بندی چند کلاسه و دو کلاسه به صورت زیر است .



**شکل 22- نمودار خطای طبقه‌بندی دو کلاسه**



**شکل 21- نمودار خطای طبقه‌بندی چند کلاسه**

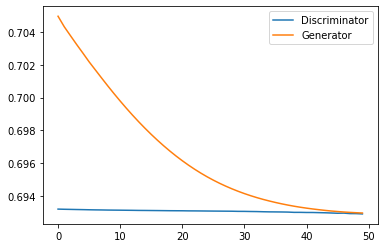
## 5-9- نتایج مدل GAN برای حمله به ترکیب دو شبکه عصبی عمیق و شبکه عصبی بازگشتی

نتایج حاصله از اجرای این کار به صورت جدول زیر است.

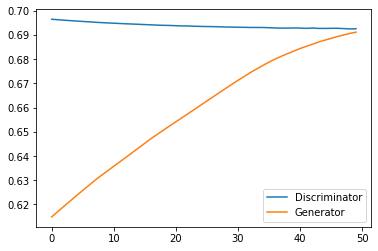
**جدول 21-** **نتایج مدل GAN برای حمله به ترکیب دو شبکه عصبی عمیق و شبکه عصبی بازگشتی**

|  |  |  |
| --- | --- | --- |
| معیار | طبقه‌بندی دودویی | طبقه‌بندی چند کلاسه |
| خطای Generator | 0.6906 | 0.6930 |
| خطای Discriminator | 0.6926 | 0.6929 |

همچنین نمودار خطای طبقه‌بندی چند کلاسه و دو کلاسه به صورت زیر است .



**شکل 23- نمودار خطای طبقه‌بندی چند کلاسه**



**شکل 24- نمودار خطای طبقه‌بندی دو کلاسه**

## 5-10- نتایج ارزابی سه مدل برای آزمایش اول

در هر آزمایش، از بهترین مدل‌هایی که ساختیم و بیشترین دقت را داشته اند استفاده میکنیم. مجموعا سه مدل از شبکه عصبی عمیق و شبکه عصبی بازگشتی و ترکیب این دو را برای آزمایش انتخاب میکنیم.

در آزمایش اول هر سه مدل را به صورت معمولی آموزش میدهیم. برای بخش ارزیابی مدل، از مجموعه داده جعلی استفاده میکنیم تا نتایج آن را ببینیم.

**جدول 22- نتایج ارزابی سه مدل برای آزمایش اول**

|  |  |  |
| --- | --- | --- |
| مدل | طبقه‌بندی چند کلاسه | طبقه‌بندی دو کلاسه |
| شبکه عصبی عمیق | **0.0%** | **69.99%** |
| شبکه عصبی بازگشتی | **0.0%** | **66.92%** |
| ترکیب شبکه عصبی عمیق و بازگشتی | **0.0%** | **72.16%** |

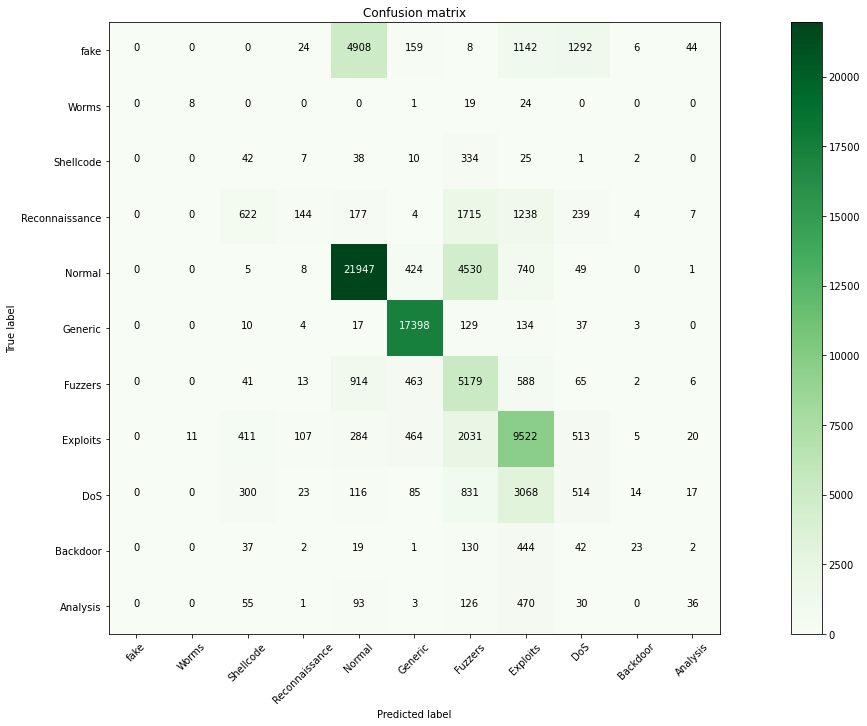
## 5-11- نتایج ارزابی سه مدل برای آزمایش دوم

در آزمایش دوم هر سه مدل را به صورت معمولی آموزش میدهیم. برای بخش ارزیابی مدل، از ترکیب مجموعه داده جعلی و واقعی استفاده میکنیم تا نتایج آن را ببینیم.

**جدول 23- نتایج ارزابی سه مدل برای آزمایش دوم**

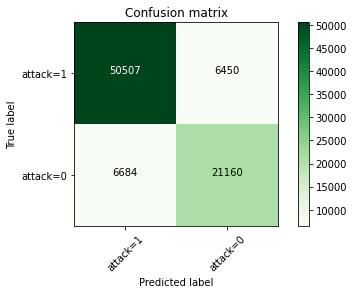
|  |  |  |
| --- | --- | --- |
| مدل | طبقه‌بندی چند کلاسه | طبقه‌بندی دو کلاسه |
| شبکه عصبی عمیق | **64.64%** | **84.51%** |
| شبکه عصبی بازگشتی | **66.63%** | **83.62%** |
| ترکیب شبکه عصبی عمیق و بازگشتی | **65.65%** | **85.05%** |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه مدل شبکه عصبی عمیق به صورت زیر است.



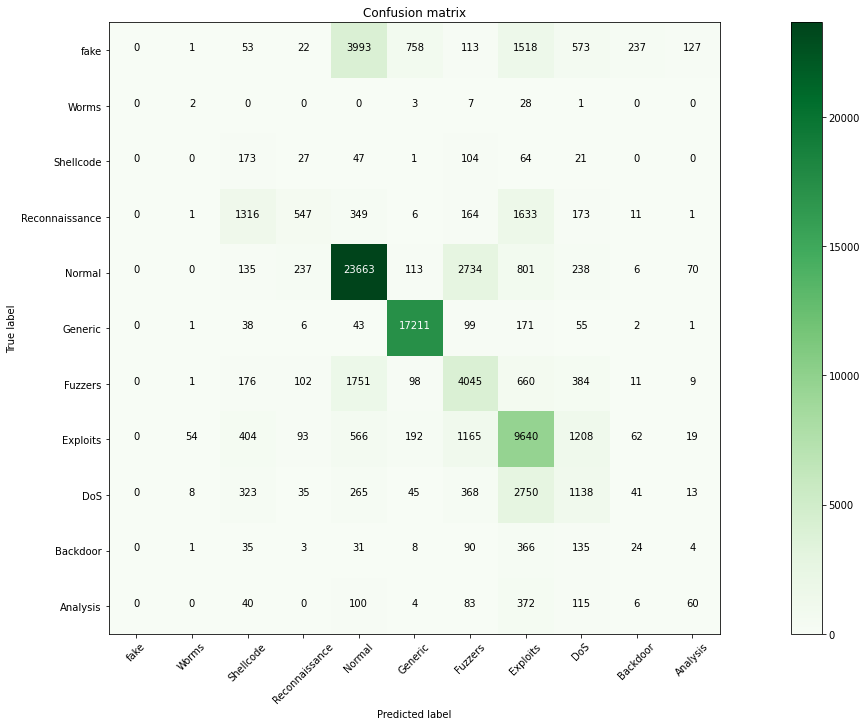
**شکل 26- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه مدل شبکه عصبی عمیق به صورت زیر است.



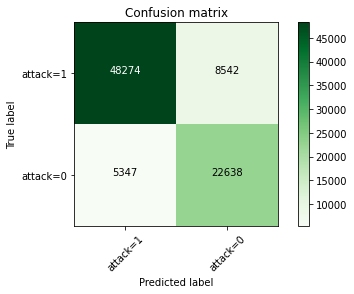
**شکل 27- ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه**

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه مدل شبکه عصبی بازگشتی به صورت زیر است.



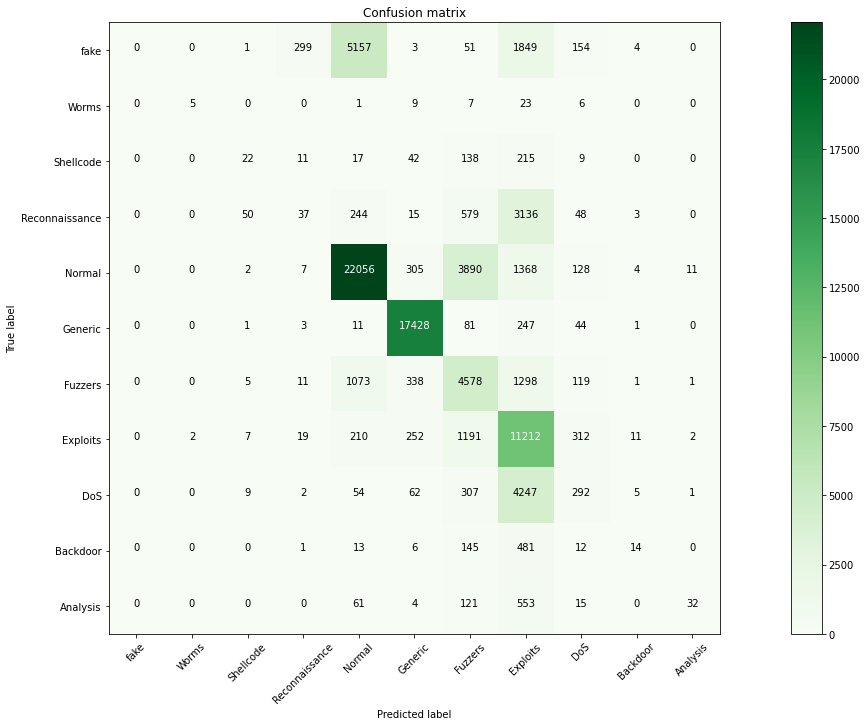
**شکل 28- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه مدل شبکه عصبی بازگشتی به صورت زیر است.



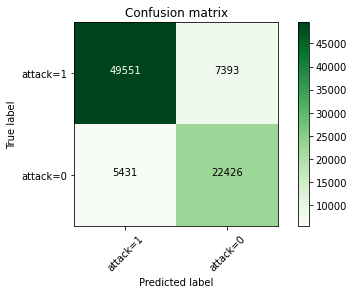
**شکل 29- ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه**

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه مدل شبکه عصبی عمیق و شبکه عصبی بازگشتی به صورت زیر است.



**شکل 30- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه مدل شبکه عصبی عمیق و شبکه عصبی بازگشتی به صورت زیر است.



**شکل 31- ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه**

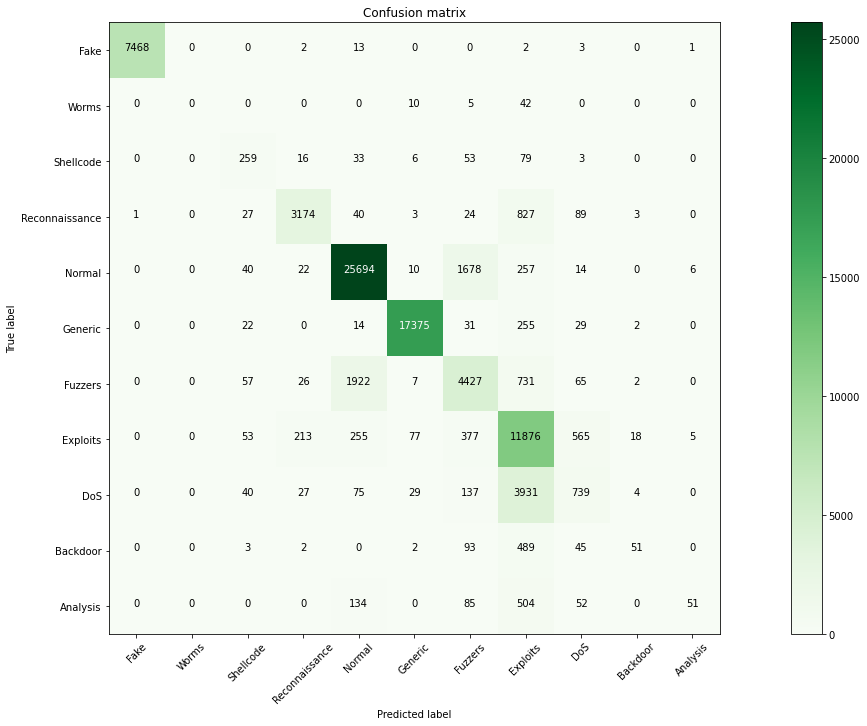
## 5-12- نتایج ارزابی سه مدل برای آزمایش سوم

در آزمایش سوم هر سه مدل را به صورت ترکیب مجموعه داده واقعی و جعلی آموزش میدهیم. برای بخش ارزیابی مدل هم، از ترکیب مجموعه داده جعلی و واقعی استفاده میکنیم تا نتایج آن را ببینیم.

**جدول 24- نتایج ارزابی سه مدل برای آزمایش سوم**

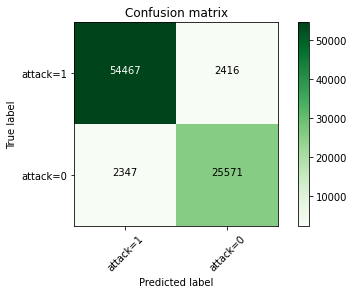
|  |  |  |
| --- | --- | --- |
| مدل | طبقه‌بندی چند کلاسه | طبقه‌بندی دو کلاسه |
| شبکه عصبی عمیق | **83.86%** | **94.38%** |
| شبکه عصبی بازگشتی | **83.98%** | **94.25%** |
| ترکیب شبکه عصبی عمیق و بازگشتی | **83.87%** | **94.46%** |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه مدل شبکه عصبی عمیق به صورت زیر است.



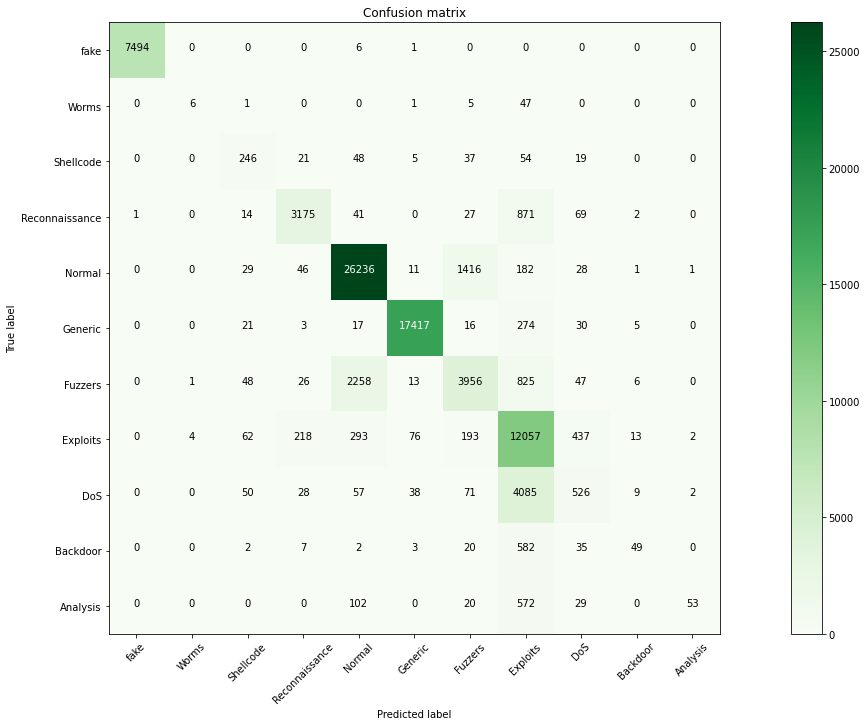
**شکل 32- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه مدل شبکه عصبی عمیق به صورت زیر است.



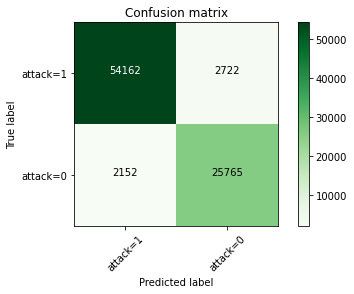
**شکل 33- ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه**

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه مدل شبکه عصبی بازگشتی به صورت زیر است.



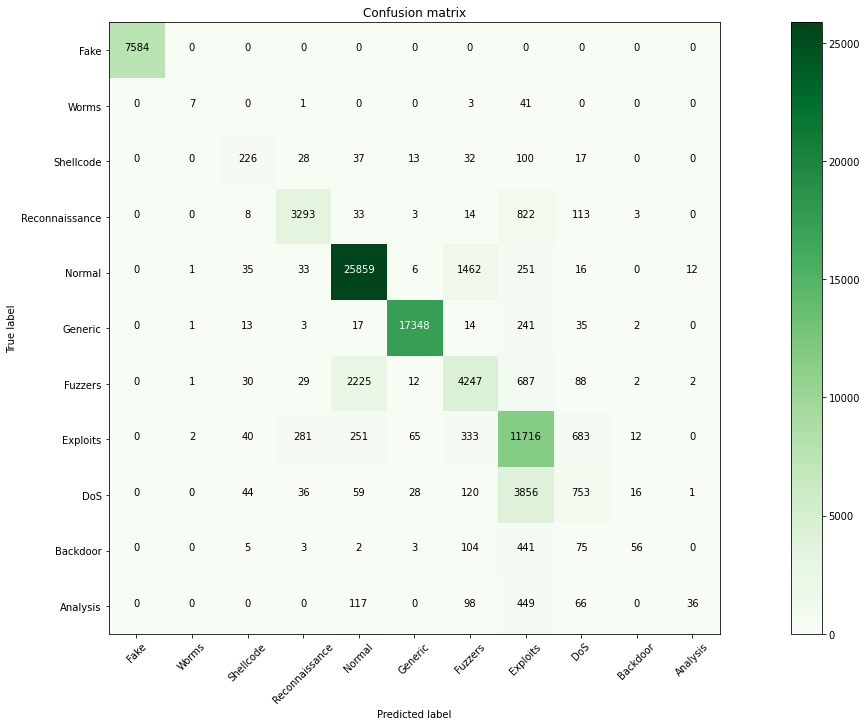
**شکل 34- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه مدل شبکه عصبی بازگشتی به صورت زیر است.



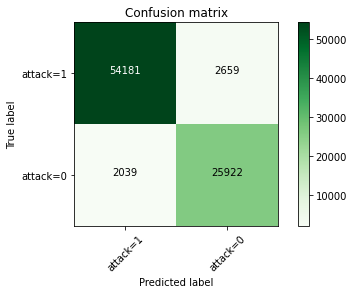
**شکل 35- ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه**

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه مدل شبکه عصبی عمیق و شبکه عصبی بازگشتی به صورت زیر است.



**شکل 36- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

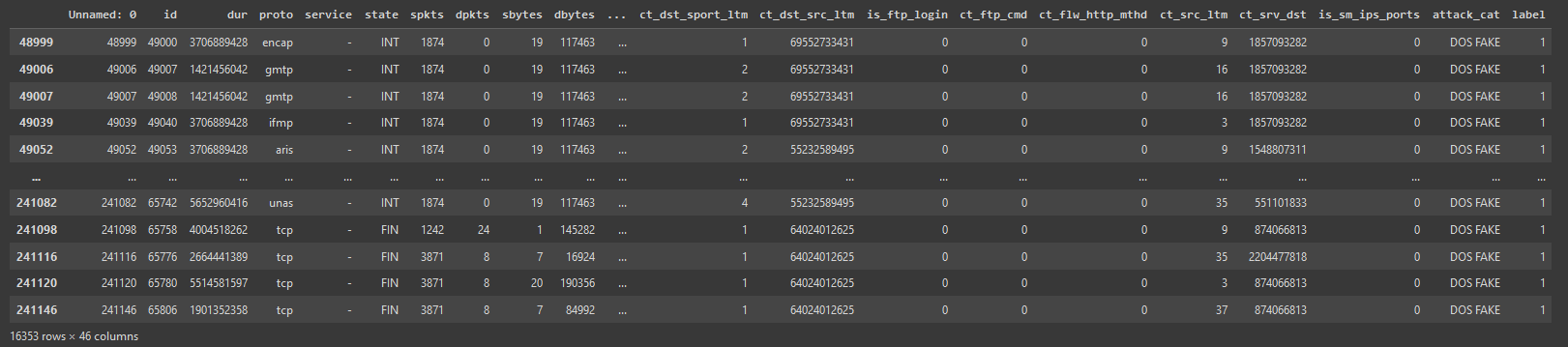
ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه مدل شبکه عصبی عمیق و شبکه عصبی بازگشتی به صورت زیر است.



**شکل 37- ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه**

## 5-13- نتایج ارزابی سه مدل برای آزمایش چهارم

در آزمایش چهارم به جای استفاده از مجموعه داده جعلی که قبلا درست کردیم، حملات DOS که یکی از ده نوع حمله است را انتخاب میکنیم که به صورت زیر است و با تغییر در یک یا چند ستون آن سعی میکنیم که دقت ارزیابی مدل را پایین بیاوریم.



**شکل 38- مجموعه داده جعلی DOS**

در این کار فقط طبقه‌بندی چند کلاسه را مورد بررسی قرار میدهیم. سپس با استفاده از داده های واقعی، یکی از مدل‌ها را آموزش میدهیم و با مجموعه داده جعلی DOS تولید شده مدل را ارزیابی میکنیم. مجموعه داده جعلی DOS با مجموعه داده واقعی DOS فقط در چند ستون فرق دارد و از نظر نوع حمله هم در دسته‌ی حملات DOS قرار میگیرد. هدف ما این است که ببینیم در چند درصد موارد، مدل آموزش داده شده نتواند تشخیص دهد که داده‌های تولید شده جعلی هستند و فکر کند که واقعی بودند. و چون اشتباه کرده، دقت پایین می‌آید.

چندین مرتبه در ستون‌های مجموعه داده DOS تغییرات ایجاد میکنیم تا با بررسی هر بار تغییرات، دقت مدل را بسجیم.

تغییرات اول به صورتی است که هفت تا از ستون های مختلف از مجموعه داده را با اعدادی به صورت تصادفی پر کردیم که در بعضی از آن ها مقدارهای کمینه و بیشینه هر سوتون پر میشود.

dos['spkts'] = dos.groupby('spkts')['spkts'].transform(lambda x: randint(1,8669))

dos['sbytes'] = dos.groupby('sbytes')['sbytes'].transform(lambda x: randint(1,20))

dos['ct\_src\_ltm'] = dos.groupby('ct\_src\_ltm')['ct\_src\_ltm'].transform(lambda x: randint(1,45))

dos['dbytes'] = dos.groupby('dbytes')['dbytes'].transform(lambda x: randint(100,200000))

dos['ct\_srv\_dst'] = dos.groupby('ct\_srv\_dst')['ct\_srv\_dst'].transform(lambda x: randint(453235,2366436436))

dos['dur'] = dos.groupby('dur')['dur'].transform(lambda x: randint(35,6787586867))

dos['ct\_dst\_src\_ltm'] = dos.groupby('ct\_dst\_src\_ltm')['ct\_dst\_src\_ltm'].transform(lambda x: randint(1324324,77776666666))

**کد 1**

تغییرات در ستون های بالا موجب رسیدن دقت مدل به 7 درصد میرسد.

تغییرات دوم به صورتی است که نه تا از ستون های مختلف از مجموعه داده را با اعدادی به صورت تصادفی پر کردیم که در بعضی از آن ها مقدارهای کمینه و بیشینه هر سوتون پر میشود.

dos['spkts'] = dos.groupby('spkts')['spkts'].transform(lambda x: randint(1,8669))

dos['sbytes'] = dos.groupby('sbytes')['sbytes'].transform(lambda x: randint(1,20))

dos['ct\_src\_ltm'] = dos.groupby('ct\_src\_ltm')['ct\_src\_ltm'].transform(lambda x: randint(1,45))

dos['dbytes'] = dos.groupby('dbytes')['dbytes'].transform(lambda x: randint(100,200000))

dos['ct\_srv\_dst'] = dos.groupby('ct\_srv\_dst')['ct\_srv\_dst'].transform(lambda x: randint(453235,2366436436))

dos['dur'] = dos.groupby('dur')['dur'].transform(lambda x: randint(35,6787586867))

dos['ct\_dst\_src\_ltm'] = dos.groupby('ct\_dst\_src\_ltm')['ct\_dst\_src\_ltm'].transform(lambda x: randint(1324324,77776666666))

dos['is\_sm\_ips\_ports'] = dos.groupby('is\_sm\_ips\_ports')['is\_sm\_ips\_ports'].transform(lambda x: randint(3,200))

dos['ct\_dst\_sport\_ltm'] = dos.groupby('ct\_dst\_sport\_ltm')['ct\_dst\_sport\_ltm'].transform(lambda x: randint(98,987654321))

**کد 2**

تغییرات در ستون های بالا موجب رسیدن دقت مدل به 23 درصد میرسد.

تغییرات سوم به صورتی است که سه تا از ستون های مختلف از مجموعه داده را با اعدادی به صورت تصادفی پر کردیم که در بعضی از آن ها مقدارهای کمینه و بیشینه هر سوتون پر میشود.

dos['spkts'] = dos.groupby('spkts')['spkts'].transform(lambda x: randint(1,8669))

dos['sbytes'] = dos.groupby('sbytes')['sbytes'].transform(lambda x: randint(1,20))

dos['ct\_src\_ltm'] = dos.groupby('ct\_src\_ltm')['ct\_src\_ltm'].transform(lambda x: randint(1,45))

**کد 3**

تغییرات در ستون های بالا موجب رسیدن دقت مدل به 6.49 درصد میرسد.

تغییرات چهارم به صورتی است که 4 تا از ستون های مختلف از مجموعه داده را با اعدادی به صورت تصادفی پر کردیم که در بعضی از آن ها مقدارهای کمینه و بیشینه هر سوتون پر میشود.

dos['spkts'] = dos.groupby('spkts')['spkts'].transform(lambda x: randint(1,8669))

dos['sbytes'] = dos.groupby('sbytes')['sbytes'].transform(lambda x: randint(1,20))

dos['ct\_src\_ltm'] = dos.groupby('ct\_src\_ltm')['ct\_src\_ltm'].transform(lambda x: randint(1,45))

dos['dbytes'] = dos.groupby('dbytes')['dbytes'].transform(lambda x: randint(100,200000))

**کد 4**

تغییرات در ستون های بالا موجب رسیدن دقت مدل به 9.91 درصد میرسد.

تغییرات پنجم به صورتی است که پنج تا از ستون های مختلف از مجموعه داده را با اعدادی به صورت تصادفی پر کردیم که در بعضی از آن ها مقدارهای کمینه و بیشینه هر سوتون پر میشود.

dos['spkts'] = dos.groupby('spkts')['spkts'].transform(lambda x: randint(1,8669))

dos['sbytes'] = dos.groupby('sbytes')['sbytes'].transform(lambda x: randint(1,20))

dos['ct\_src\_ltm'] = dos.groupby('ct\_src\_ltm')['ct\_src\_ltm'].transform(lambda x: randint(1,45))

dos['dbytes'] = dos.groupby('dbytes')['dbytes'].transform(lambda x: randint(100,200000))

dos['ct\_srv\_dst'] = dos.groupby('ct\_srv\_dst')['ct\_srv\_dst'].transform(lambda x: randint(453235,2366436436))

**کد 5**

تغییرات در ستون های بالا موجب رسیدن دقت مدل به 8.21 درصد میرسد.

تغییرات ششم به صورتی است که شش تا از ستون های مختلف از مجموعه داده را با اعدادی به صورت تصادفی پر کردیم که در بعضی از آن ها مقدارهای کمینه و بیشینه هر سوتون پر میشود.

dos['spkts'] = dos.groupby('spkts')['spkts'].transform(lambda x: randint(1,8669))

dos['sbytes'] = dos.groupby('sbytes')['sbytes'].transform(lambda x: randint(1,20))

dos['ct\_src\_ltm'] = dos.groupby('ct\_src\_ltm')['ct\_src\_ltm'].transform(lambda x: randint(1,45))

dos['dbytes'] = dos.groupby('dbytes')['dbytes'].transform(lambda x: randint(100,200000))

dos['ct\_srv\_dst'] = dos.groupby('ct\_srv\_dst')['ct\_srv\_dst'].transform(lambda x: randint(453235,2366436436))

dos['dur'] = dos.groupby('dur')['dur'].transform(lambda x: randint(35,6787586867))

**کد 6**

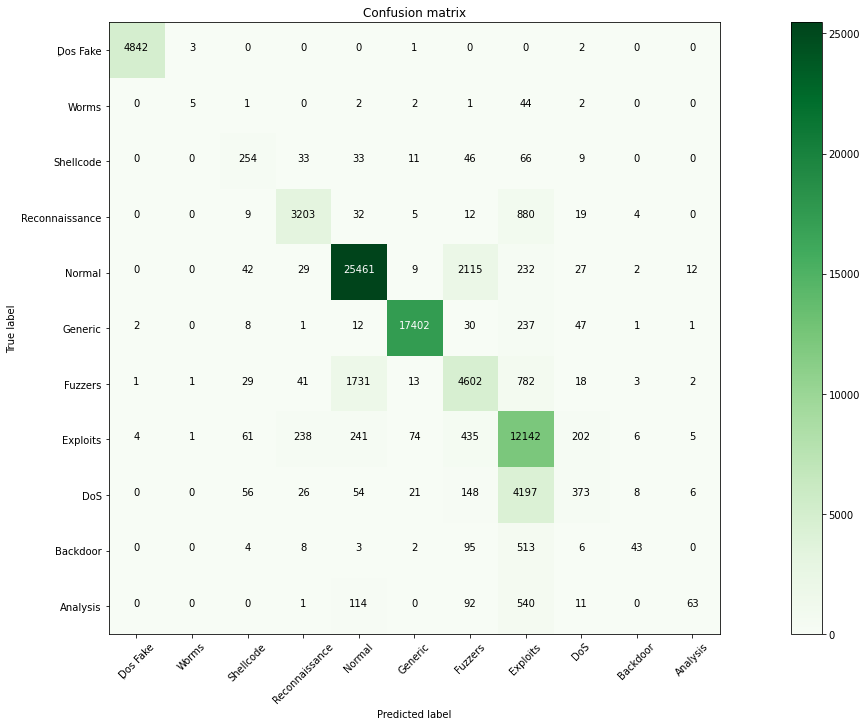
تغییرات در ستون های بالا موجب رسیدن دقت مدل به 9.88 درصد میرسد.

در انتهای آزمایش چهارم، همانند آزمایش قبلی، مدل را به صورت ترکیب مجموعه داده واقعی و جعلی آموزش میدهیم. برای بخش ارزیابی مدل هم، از ترکیب مجموعه داده جعلی و واقعی استفاده میکنیم تا نتایج آن را ببینیم.

**جدول 25- نتایج ارزابی سه مدل برای آزمایش چهارم**

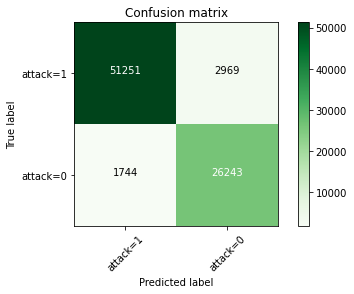
|  |  |  |
| --- | --- | --- |
| مدل | طبقه‌بندی چند کلاسه | طبقه‌بندی دو کلاسه |
| شبکه عصبی عمیق | **83.19%** | **94.27%** |

ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه مدل به صورت زیر است.



**شکل 39- ماتریس درهم ریختگی برای طبقه‌بندی چند کلاسه**

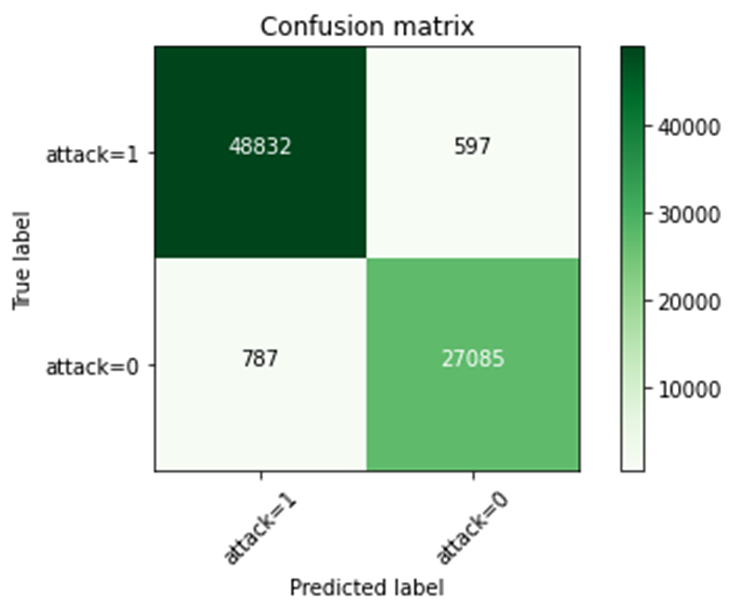
ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه مدل به صورت زیر است.



**شکل 40- ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه**

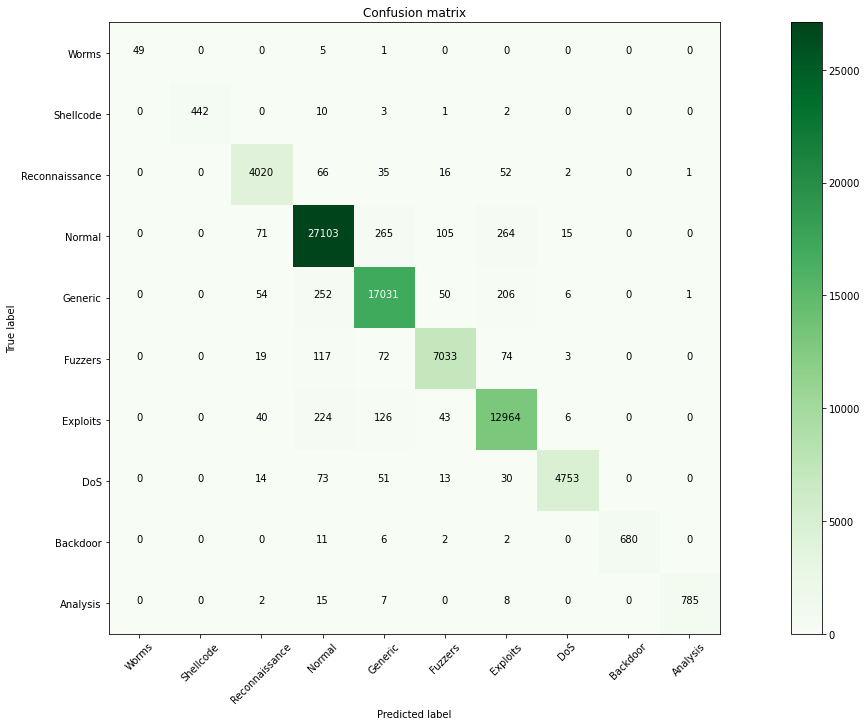
## 5-14- نتایج اعمال تروجان

مقدار دقت برای طبقه بندی دو کلاسه برابر با 95.8500 و ماتریس درهم ریختگی برای کلاس دودویی به صورت زیر است.



**شکل 41- ماتریس درهم ریختگی برای طبقه‌بندی دو کلاسه با تروجان**

مقدار دقت برای طبقه بندی چند گروهه برابر با 83.7544 و ماتریس درهم ریختگی برای کلاس چند گروهه به صورت زیر است.



**شکل 42- ماتریس درهم ریختگی برای طبقه‌بندی چند گروهه با تروجان**

معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه به صورت زیر است.

**جدول 14- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی چند کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.97 | 0.95 | 0.99 | 0 |
| 0.97 | 0.94 | 1 | 1 |
| 0.97 | 0.96 | 0.99 | 2 |
| 0.97 | 0.97 | 0.95 | 3 |
| 0.97 | 0.96 | 0.97 | 4 |
| 0.97 | 0.98 | 0.97 | 5 |
| 0.97 | 0.98 | 0.97 | 6 |
| 0.96 | 0.97 | 0.95 | 7 |
| 0.98 | 0.97 | 1 | 8 |
| 1 | 1 | 1 | 9 |

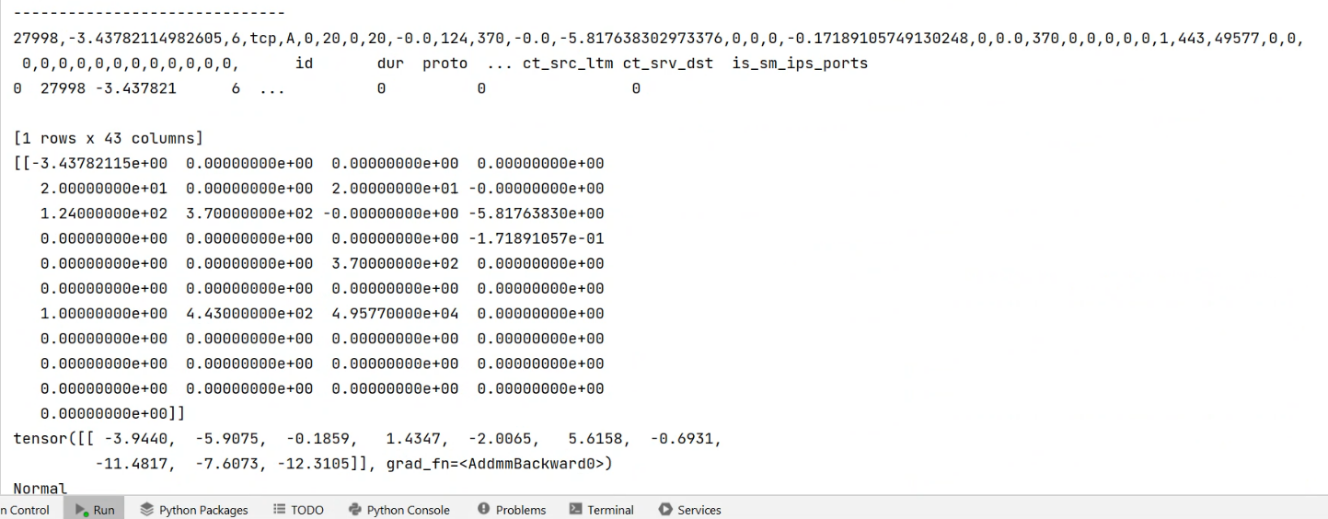
همچنین معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی دو کلاسه به صورت زیر است.

**جدول 15- معیارهای ارزیابی Recall - Precision و F1-score برای طبقه‌بندی دو کلاسه**

|  |  |  |  |
| --- | --- | --- | --- |
| f1-score | recall | precision | کلاس |
| 0.97 | 0.97 | 0.98 | 0 |
| 0.99 | 0.99 | 0.98 | 1 |

همانطور که مشاهده میشود میزان دقت در تشخیص حملات بیشتر شده است که این یک نوآوری در کار ما است.

## 5-15- نتایج تست نهایی



**شکل 43- خروجی نهایی**

همانطور که مشاهده میشود در ابتدا پکتی که از شبکه گرفته شده است نمایش داده میشود. سپس مقادیر نرمال شده و به قسمت ارزیابی مدل داده میشود و تعیین میشود که نوع حمله از چه نوعی بوده است

1. واژه‏نامه فارسي-انگليسي

| Equivalent English | واژه فارسی |
| --- | --- |
| Case-Based Reasoning | استدلال مبتنی بر مورد |
| Cross Site Scripting | اسکریپت متقابل سایت |
| Distributed Denial-of-Service | انکار خدمات توزیع‌شده |
| Information Gain | به دست آوردن اطلاعات |
| Hidden Naïve Bayes | بیز ساده پنهان |
| Backpropagation | پس انتشار |
| Radial Basis Function | تابع پایه شعاعی |
| Sign Intrusion Detection | تشخیص نفوذ مبتنی بر امضا |
| Anomaly-based Intrusion Detection | تشخیص نفوذ مبتنی بر ناهنجاری |
| Denial-of-Service | خودداری از خدمات |
| Intrusion Detection System | سیستم تشخیص نفوذ |
| Network Intrusion Detection System | سیستم تشخیص نفوذ شبکه |
| Host Intrusion Detection System | سیستم تشخیص نفوذ میزبان |
| Collaborative Intrusion Detection Network | شبکه تشخیص نفوذ مشارکتی |
| Convolutional Neural Network | شبکه عصبی کانولوشنال |
| Artificial Neural Network | شبکه‌های عصبی مصنوعی |
| User-to-Root | کاربر به ریشه |
| Support Vector Machine | ماشین بردار پشتیبانی |
| Finite-State Machine | ماشین حالت متناهی |
| Fog Computing | محاسبات مه |
| Hidden Markov Model | مدل مارکوف پنهان |
| K-Nearest Neighbors | نزدیک‌ترین همسایه‌ها K- |
| Correlation | همبستگی |
| Machine learning | یادگیری ماشین |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  | فهرست مراجع  [1] Abdullahi, Mujaheed, "Detecting Cybersecurity Attacks in Internet of Things Using Artificial Intelligence Methods: A Systematic Literature Review.", Electronics, no.11.2, pp.198, 2022  [2] Imran, Muhammad,. "An intelligent and efficient network intrusion detection system using deep learning." Computers and Electrical Engineering, no.99, pp.107764, 2022  [3] Tuan, Tong Anh, et al. "Performance evaluation of Botnet DDoS attack detection using machine learning." Evolutionary Intelligence no. 13 pp. 283-294, 2020‏  [4] Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., & Ahmad, R. “CNN-LSTM: hybrid deep neural network for network intrusion detection system”. IEEE Access, no. 10, pp. 99837-99849, 2022  [5] Gwon, Hyeokmin, et al. "Network intrusion detection based on LSTM and feature embedding." arXiv preprint arXiv:1911.11552, 2019  [6] Aleesa, Ahmed, et al. "Deep-intrusion detection system with enhanced UNSW-NB15 dataset based on deep learning techniques." Journal of Engineering Science and Technology, no. 16.1, pp. 711-727, 2021  [7] https://howsam.org/generative-adversarial-network  [8] Fonseca, Jose, Marco Vieira, and Henrique Madeira. "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks," In 13th Pacific Rim international symposium on dependable computing (PRDC 2007), pp. 365-372, IEEE, 2007.  [9] <https://www.dnsstuff.com/intrusion-detection-system#how-does-network-intrusion-work>  [10] Khraisat, Ansam, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. "Survey of intrusion detection systems: techniques, datasets and challenges," Cybersecurity, vol. 2, no. 1, pp. 1-22, 2019.  [11] Cepheli, Özge, Saliha Büyükçorak, and Güneş Karabulut Kurt. "Hybrid intrusion detection system for ddos attacks," Journal of Electrical and Computer Engineering, vol. 2016,2016.  [12] Abbas, Sara Abdalelah, and Mahdi S. Almhanna. "Distributed denial of service attacks detection system by machine learning based on dimensionality reduction," In Journal of Physics: Conference Series, vol. 1804, no. 1, p. 012136. IOP Publishing, 2021.  [13] Kshirsagar, Deepak, and Sandeep Kumar. "A feature reduction based reflected and exploited DDoS attacks detection system." Journal of Ambient Intelligence and Humanized Computing vol. 13, no. 1, pp. 393-405, 2022.  [14] Karar, Anjali. "Preventing SQL-BasedAttacks Using Intrusion Detection System," International Journal of Science and Engineering Applications 2, no. 6, pp. 145-150, 2013.  [15] Lakshmi, A. Anna, S. Anandkumar, G. Nagarajan, and K. R. Valluvan. "Comparative Analysis of Architectures for Intrusion Detection Systems against DoS Attacks in MANETs based on Chi-Square Test." International Journal of Computer Applications vol. 87, no. 4 2014.  [16] Gavrilovic, Nadja, Vladimir Ciric, and Nikola Lozo. "Snort IDS system visualization interface for alert analysis," SJEE 19, no. 1, pp. 67-78, 2022.  [17] Dr. Pathan Hussain Basha, Dr. G. Sai Chatanya Kumar, Dr. N. Raghavendra Sai, “SIGNATURE TRANSFER IN SECURED WAY FOR DISTRIBUTED INTRUSION DETECTION SYSTEM BY USING BLOCKCHAIN TECHNOLOGY”, 2022  [18] Wang, Yu, Weizhi Meng, Wenjuan Li, Jin Li, Wai-Xi Liu, and Yang Xiang. "A fog-based privacy-preserving approach for distributed signature-based intrusion detection." Journal of Parallel and Distributed Computing 122, pp. 26-35, 2018.  [19] Stency, V. S., and N. Mohanasundaram. "A Study on XSS Attacks: Intelligent Detection Methods." Journal of Physics: Conference Series. vol. 1767, no. 1, IOP Publishing, 2021  [20] <https://www.ossec.net/>  [21] <https://www.tripwire.com>  [22] https://research.unsw.edu.au/projects/unsw-nb15-dataset |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Abstract**

In this thesis, we …



**K. N. Toosi University of Technology**

**Faculty of Computer Engineering**

**A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science (M.Sc.)**

**in Computer Engineering - Software**

Detection of attacks: an adversarial machine learning method

**By:**

Taha AkhlaghPasandi

**Supervisor:**

dr Mohamad hadi alayian

**Advisor:**

[advisor name(s)]

Spring 2023

1. Machine Learning [↑](#footnote-ref-2)
2. Supervised Learning [↑](#footnote-ref-3)
3. Unsupervised Learning [↑](#footnote-ref-4)
4. Deep Learning [↑](#footnote-ref-5)
5. Data Mining [↑](#footnote-ref-6)
6. Natural Language Processing [↑](#footnote-ref-7)
7. Intrusion Detection System [↑](#footnote-ref-8)
8. Malware [↑](#footnote-ref-9)
9. Worm [↑](#footnote-ref-10)
10. Artificial Neural Network [↑](#footnote-ref-11)
11. Deep Neural Network [↑](#footnote-ref-12)
12. Feed-forward [↑](#footnote-ref-13)
13. Recurrent Neural Network [↑](#footnote-ref-14)
14. Generative Adversarial Networks [↑](#footnote-ref-15)
15. Long short-term memory [↑](#footnote-ref-16)
16. Learning Rate [↑](#footnote-ref-17)
17. Activation Function [↑](#footnote-ref-18)
18. Loss Function [↑](#footnote-ref-19)
19. Trojan [↑](#footnote-ref-20)
20. Trojan Trigger [↑](#footnote-ref-21)
21. Intrusion Detection System [↑](#footnote-ref-22)
22. Host Intrusion Detection System [↑](#footnote-ref-23)
23. Network Intrusion Detection System [↑](#footnote-ref-24)
24. Sign Intrusion Detection [↑](#footnote-ref-25)
25. Anomaly-based Intrusion Detection [↑](#footnote-ref-26)
26. Finite-State Machine [↑](#footnote-ref-27)
27. Convolutional Neural Network [↑](#footnote-ref-28)
28. Overfitting [↑](#footnote-ref-29)
29. underfitting [↑](#footnote-ref-30)