گزارش پروژه ی دوم
شناسایی حملات در داده‌های شبکه با استفاده از روش‌های داده کاوی

فاطمه کریمی(۴۰۰۷۲۳۱۳۱)
بیست و سوم اردیبهشت ماه ۱۴۰۱

چکیده

در این پروژه با قصد داریم با در اختیار داشتن ویژگی بسته های ورودی به سرور، امکان وقوع حمله ی ddos را تشخیص دهیم. دیتاست به کار گرفته شده، CICddos2019 است که شامل داده ی بسته ها با برچسب benign برای بسته های عادی، و نوع حمله برای بسته های حمله ی ddos می باشد. نوع حمله های ddos به دسته های LDAP، MSSQL، NetBios، Portmap،Syn، UDP و UDPLag تقسیم می شود. در این پروژه نوع حمله ی ddos را نادیده گرفتیم و تنها تشخیص حمله یا عادی بودن یک بسته را بررسی می کنیم.

۱- مجموعه ی داده

دیتاست CICddos2019 اطلاعات حدود بیست میلیون و سی صد هزار بسته را در بر دارد. در جدول ۱ تعداد داده‌ها برای هر برچسب آمده است. با در نظر گرفتن برچسب‌ها به دو دسته ی benign و attack، بیست میلیون رکورد متعلق به برچسب attack و تنها پنجاه هزار رکورد متعلق به دسته ی benign می باشد. بنابراین نسبت برچسب benign به attack ۱ به ۴۰۰ است که توزیعی نامتوازن می‌باشد و می بایست این عدم توازن در گام نمونه گیری از میان برود.

جدول ۱: توزیع داده ها برای برچسب های مختلف

Label	Count	Percentage
LDAP	1915122	9.40421
NetBIOS	3657497	17.96014
BENIGN	56965	0.27973
MSSQL	5787453	28.41929
Portmap	186960	0.91807
Syn	4891500	24.01971
UDP	367155	18.98966
UDPLag	1873	0.0092
TOTAL	**20364525**	**100**

۲- نمونه گیری

به دلیل حجم بالای دیتاست، نمونه گیری از رکوردها لازم است. تعداد رکورد های در نظر گرفته شده برای دیتاست نمونه، دویست هزار رکورد است که ۱٪ داده ی اصلی را تشکیل می دهد. برای متوازن کردن تعداد رکوردها با برچسب attack و benign، پنجاه درصد نمونه گیری از رکوردهای benign و پنجاه درصد دیگر از رکوردهای attack خواهد بود. از آنجایی که تعداد کل رکوردهای benign پنجاه هزار رکورد است، در این نمونه گیری هر رکورد benign تقریباً دو بار انتخاب می شود. برای نمونه گیری از رکوردهای attack، از stratified sampling بهره می‌گیریم و صدهزار رکورد attack، با رعایت توزیع نوع حملات در داده ی ورودی نمونه گیری خواهد شد. درصد رکوردهای یک حمله ی خاص در نمونه ی نهایی از فرمول ۱ به دست می آید.

$$\frac{number\ of\ records\ with\ the\ specified\ label}{total\ number\ of\ records} \times percentage\ of\ attack\ label\ in\ the\ sample\ dataset$$

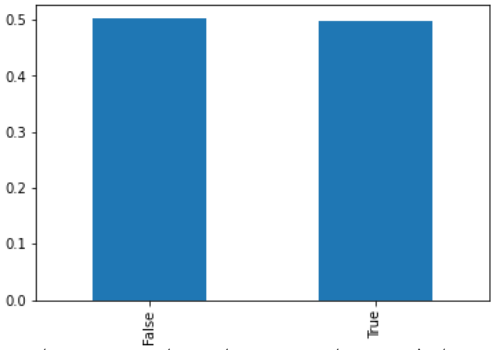
فرمول ۱: محاسبه ی درصد یک نوع حمله ی خاص در دیتاست نمونه

بنابراین طبق فرمول ۱، اگر ۵۰٪ داده‌های نمونه را benign در نظر بگیریم و ۵۰٪ دیگر را attack، درصد رکوردها با برچسب LDAP در داده ی نهایی، برابر با۴.۷٪ خواهد بود.

به طور خلاصه، درصد برچسب‌های تشکیل‌دهنده ی نمونه ی نهایی، در جدول ۲ آمده است.

جدول ۲: درصد برچسب های تشکیل دهنده ی نمونه

Label	**Percentage**
LDAP	4.7%
NetBios	8.9%
Benign	50%
MSSQL	14.2%
Portmap	0.45%
Syn	12%
UDP	9.4%
UDPLag	0.0045%



نمودار ۱: توزیع نهایی برچسب ها در دیتاست نمونه. ۰ برای benign و ۱ برای attack است

۳- پیش پردازش

گام دوم پیش پردازش است که بر روی دیتاست نمونه انجام می شود. این گام شامل حذف متغییرهای ناکارآمد، حذف رکوردها با مقادیر نامعتبر، حذف ویژگی‌ها با واریانس صفر، حذف ویژگی‌های تکراری به کمک همبستگی و شناسایی و حذف داده‌های پرت می باشد.

۳-۱ حذف متغییرهای ناکارآمد

در نخستین گام پیش پردازش، متغییرهای ناکارآمدی که در تحلیل اثری نخواند گذاشت حذف می شوند. این متغییرها عبارتند از: flow_id، source_ip، source_port، destination_ip، destination_port، timestamp، protocol، similar_http و inbound هستند.

۳-۲ حذف رکوردها با مقادیر نامعتبر

مقدار برخی از متغییرها در بعضی رکوردها با nan، inf و یا اعداد منفی پر شده است. همه ی این رکوردها از داده ی نمونه حذف می شوند.

برخی از متغییرها، با مقادیر نامعتبر زیادی پر شده اند. این متغییرها نیز از داده ی نمونه حذف می شوند. دو متغییر flow_packets_s و flow_bytes_s دو متغییری هستند که به این دلیل از داده ی نمونه حذف می شوند.

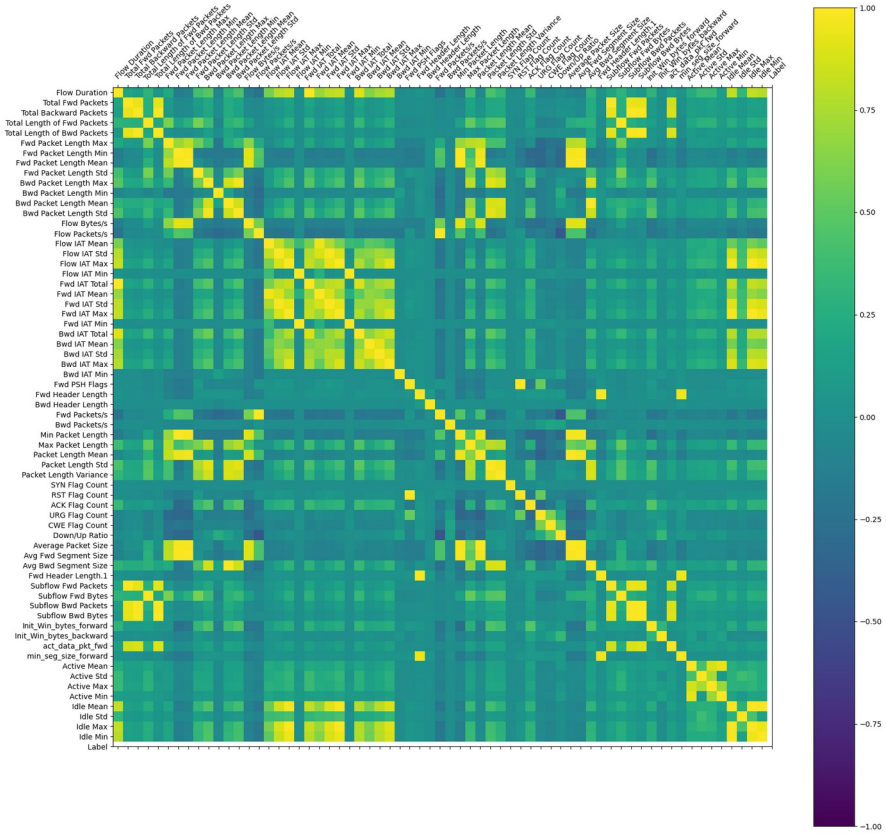
۳-۳ حذف متغییرها با واریانس صفر

برای تحلیل درست، لازم است متغییرهایی که برای همه ی رکوردها مقدار ثابتی دارند، حذف شوند. چنین متغییرهایی واریانس صفر دارند. پس از انجام این کار، متغییرهای Fwd URG Flags, Bwd PSH Flags, Bwd Avg Bulk Rate, ECE Flag Count, Fwd Avg Bulk Rate, Fwd Avg Packets/Bulk, PSH Flag Count, Bwd Avg

در این Packets/Bulk, Bwd URG Flags, Bwd Avg Bytes/Bulk, FIN Flag Count, Fwd Avg Bytes/Bulk مرحله از دیتاست نمونه حذف شدند.

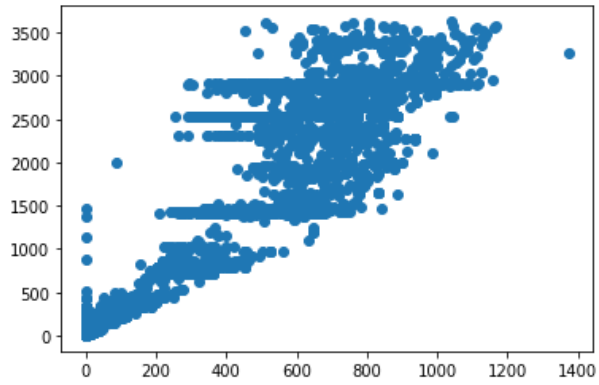
٤-٣ حذف متغییرهای تکراری با محاسبه ی همبستگی

برای محاسبه ی همبستگی میان متغییرها، به دلیل numeric بودن همه ی متغییرهای باقی مانده، روش Pearson را به کار می گیریم. نمودار heatmap در شکل ۱ آمده است.

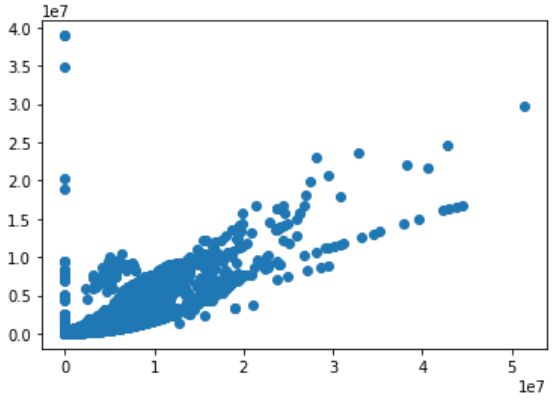


شکل ۱: نمودار همبستگی میان متغییرها

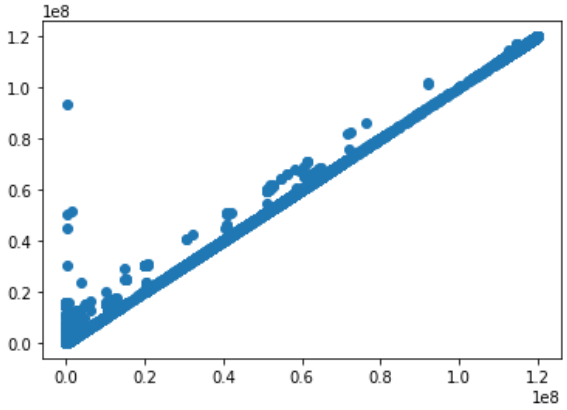
برای درک بهتر همبستگی های محاسبه شده،نمودار scatter برخی از آن‌ها رسم شده است.



نمودار ۲: همبستگی میان دو متغییر Bwd Packet Length Max و Fwd Packet Length Std



نمودار ۳: همبستگی میان دو متغییر Flow IAT Std و Flow IAT mean



نمودار ۴: همبستگی میان Fwd IAT Totalو Flow Duration

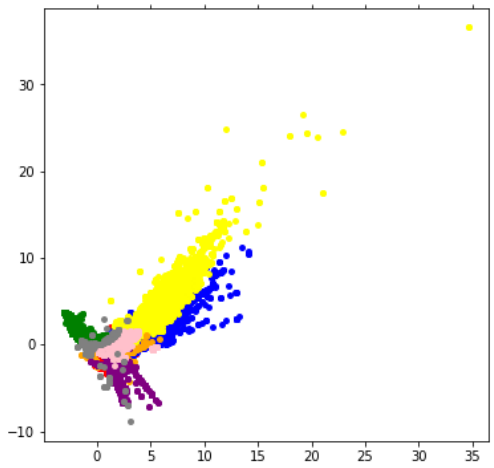
متغییرهایی که در این فاز از داده ی ورودی حذف شدند عبارتند از:

Subflow Fwd Packets, RST Flag Count, Idle Max, Max Packet Length, Flow IAT Std, Fwd IAT Std, Total Length of Bwd Packets, Packet Length Std, Fwd IAT Mean, Bwd IAT Std, Packet Length Variance, Fwd Header Length.1, Bwd Packet Length Mean, Bwd Packet Length Std, Fwd IAT Total, Idle Mean, min_seg_size_forward, Packet Length Mean, act_data_pkt_fwd, Subflow Bwd Packets, Bwd IAT Total, Active Min, Fwd IAT Min, Subflow Fwd Bytes, Fwd Packet Length Mean, Total Backward Packets, Subflow Bwd Bytes, Fwd IAT Max, Flow IAT Max, Avg Fwd Segment Size, Avg Bwd Segment Size, Min Packet Length, Active Max, Average Packet Size, Idle Min, Bwd IAT Max

از میان حدود ۸۰ متغییر اولیه که در دیتاست وجود داشت، تنها ۲۷ متغییر باقی ماند و بقیه در گامهای یاد شده حذف شدند.

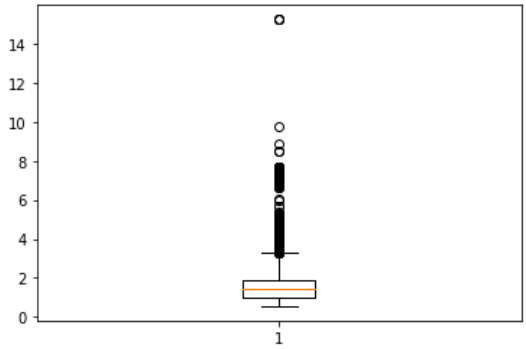
٤-۳ حذف داده‌های پرت

برای حذف داده‌های پرت، از الگوریتم K-means که یک روش clustering است، بهره می گیریم. روش کار به این صورت است که پس از استاندارد سازی داده‌ها به روش z-score، الگوریتم k-means را بر روی داده‌ها اجرا می‌کنیم و تعداد کلاسترها برای این الگوریتم را برابر با تعداد برچسب‌ها یعنی ۸ قرار می دهیم. پس از به دست آمدن کلاسترها، تنها به منظور به تصویر کشیدن کلاسترها، با به کارگیری الگوریتم PCA، داده‌ها را در دو بعد نمایش می دهیم. نمودار ۵ کلاسترهای به دست آمده را نشان می دهد.



نمودار ۵: کلاسترهای به دست آمده از الگوریتم k-means

پس از مشخص شدن کلاستر هر رکورد، فاصله ی اقلیدسی هر رکورد را تا مرکز کلاسترش حساب می کنیم. سپس با به کار گیری روش whisker-box، داده‌های پرت را برای هرکلاستر محاسبه می کنیم. برای مثال، نمودار ۶، box-whisker رسم شده برای کلاستر ۳ را نشان می دهد.



نمودار ۶: شناسایی داده های پرت به کمک box-whisker برای کلاستر ۳

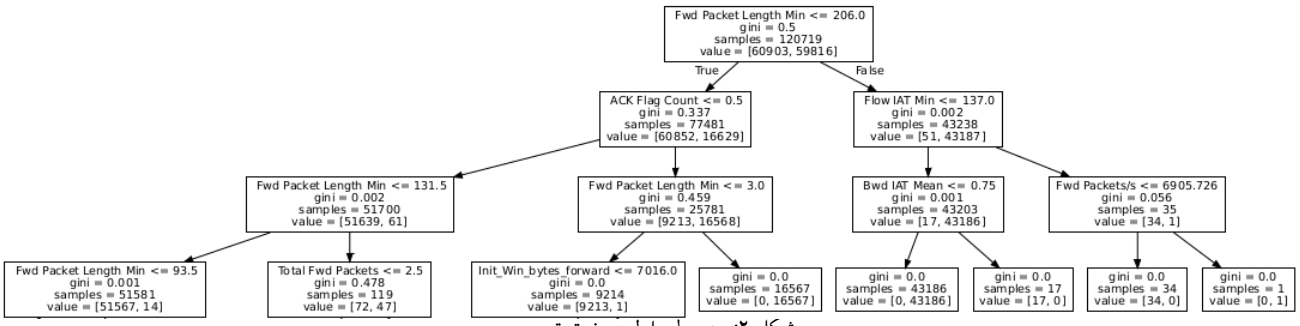
در دیتاست نمونه، حدود بیست و شش هزار رکورد به عنوان داده ی پرت شناسایی و از دیتاست حذف شد.

٤- شناسایی حملات DDOS

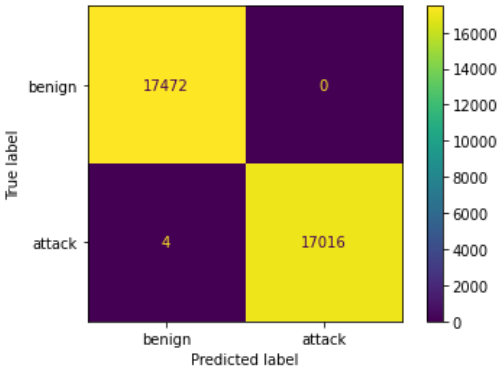
پس از آماده سازی داده ها، به سه روش naive Bayes، decision tree و neural networks مدل را ساخته و معیارهای ارزیابی آن را بررسی می کنیم.

٤-١ decision tree

درخت تصمیم از جمله الگوریتم های clasification است که تفسیرپذیری بالایی دارد. برای پیاده‌سازی این الگوریتم، ازکلاس DecisionTreeClassifier در کتابخانه ی sklearn بهره می گیریم. داده‌های ورودی را به کمک تابع split_train_test به داده‌های آموزش و آزمون تقسیم می کنیم به طوری که ۲۰٪ داده‌های ورودی داده‌های آزمون را تشکیل دهند. در درخت تصمیم، داده‌های ورودی نیازی به نرمالسازی ندارند. درخت تصمیم به دست آمده عمقی برابر با ۱۱ دارد که ۳ سطح اول آن در شکل ۲ نمایش داده شده است. دقت آن بر روی داده ی تست برابر با ۹۹.۹۹٪ است. ماتریس درهم ریختگی این مدل در نمودار ۷ آمده است. هم چنین معیار های precision, recall و f1-score در جدول ۳ محاسبه شده است.



شکل ۲: سه سطح اول درخت تصمیم



نمودار ۷: ماتریس درهم ریختگی درخت تصمیم

جدول ۳: مقادیر precision recall و f-score برای درخت تصمیم

	Precision	Recall	f1-score
0	1.0	1.0	1.0
1	1.0	1.0	1.0
accuracy			1.0
Macro average	1.0	1.0	1.0
Weighted average	1.0	1.0	1.0

به کمک درخت تصمیم، می‌توان ویژگی‌هایی که در شناسایی برچسب داده‌ها مؤثر تر بوده اند، را به دست آورد. با توجه به درخت تصمیم، در این تحلیل ۵ ویژگی مؤثر در جدول ۴ آورده شده است.

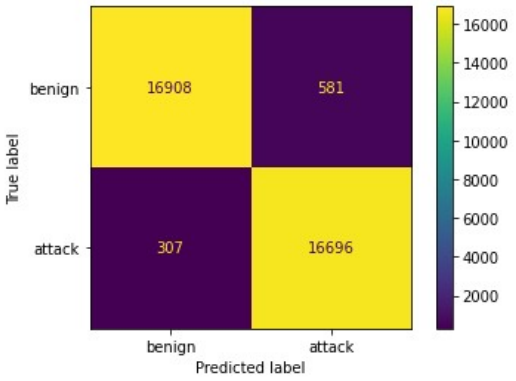
جدول ۴: ۵ ویژگی مهم در درخت تصمیم

Feature	importance
Fwd packet len min	76%
ACK flag count	23%
Flow IAT Min	0.1%
Total length of Fwd Packets	0.08%
Total Fwd Packets	0.05%

با توجه به جدول ۴، دو متغییر Fwd packet len min و ACK flag count اهمیت بسیاری در دسته بندی دارند و میزان اهمیت سایر متغییرها، ناچیز است.

٤-۲ naive bayes

برای پیاده‌سازی naive bayes، از کتابخانه ی sklearn استفاده شده است. داده ی تست ۲۰٪ کل داده‌های ورودی را تشکیل می دهد. لازم است که برای اجرای درست این الگوریتم، داده نرمالسازی شود. نرمال سازی داده در اینجا به روش min-max انجام شد تا همه ی داده‌های ورودی در بازه ی [۱،۰] قرار گیرند. دقت مدل برابر با ۹۷٪ شد و ماتریس درهم ریختگی آن در نمودار ۸ آمده است. همچنین در جدول ۵، مقادیر precision, recall و f1-score برای آن محاسبه شده است.



نمودار ۸: ماتریس درهم ریختگی الگوریتم naive bayes

جدول ۵: مقادیر precision recall و f1score برای الگوریتم naive bayes

	Precision	Recall	f1-score
0	0.98	0.97	0.97
1	0.97	0.98	0.97
accuracy			0.97
Macro average	0.97	0.97	0.97
Weighted average	0.97	0.97	0.97

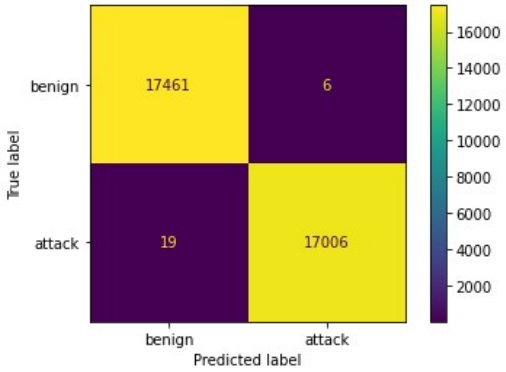
٤-۳ neural network

برای انجام دسته بندی به کمک شبکه‌های عصبی، تنها موثرترین متغییرهایی را که از درخت تصمیم به دست آوردیم را نگاه داشته و بقیه ی متغییرها را حذف می کنیم. این کار از روش‌های feature selection است و کمک می کند تا با حذف متغییرهای بی اثر، کارایی شبکه ی عصبی را بهبود بخشیم. متغییرهایی که در پایان باقی ماندند عبارتند از:

Fwd Packet Length Min, ACK Flag Count, Init_Win_bytes_forward, Total Fwd Packets, Idle Std

اکنون داده‌های باقی‌مانده را به روش min-max نرمالسازی می کنیم. برچسب‌ها نیز باید به فرم one-hot encoding تبدیل شوند. پس از آن داده‌های train و test را تقسیم‌بندی می کنیم. همانند دو حالت قبل، داده‌های تست، ۲۰٪ داده‌های کل را تشکیل خواهند داد. ساختار شبکه ی عصبی به این صورت تعریف می‌شود که لایه ی اول دارای ۵ گره با تابع فعال سازی relu است. برای لایه ی دوم نیز دو گره با تابع فعالسازی softmax در نظر گرفته شده است. همین لایه است که در نهایت برچسب پیش‌بینی شده را مشخص می کند. Loss function مدل categorical crossentropy و معیار آن binary accuracy انتخاب شده است. برای train کردن مدل، تعداد epochها برابر با ۴۰ و batch size برابر

با ۳۲ قرار داده شده است. این تنظیمات مطابق با آنچه که در [۱] آمـده است، می باشـد. دقت مـدل شـبکه ی عصـبی پس از آخرین epoch برابر با ۹۹.۹۳٪ شد. ماتریس درهم ریختگی برای شبکه ی عصبی در نمودار ۹ آمده است.



نمودار ۹: ماتریس درهم ریختگی شبکه ی عصبی

همچنین مقادیر recall ،precision و f1-score برای آن در جدول ۶ آمده است.

جدول ۶: مقادیر precision recall و f1-score برای شبکه ی عصبی

	Precision	Recall	f1-score
0	1.0	1.0	1.0
1	1.0	1.0	1.0
accuracy			1.0
Macro average	1.0	1.0	1.0
Weighted average	1.0	1.0	1.0

۵- نتیجه‌گیری

با توجه به ارزیابی های انجام شده، الگوریتم های درخت تصمیم و شبکه عصبی با دقت بهتری توانستند نوع حمله در شبکه را تشخیص دهند. همچنین، زمان اجرای الگوریتم درخت تصمیم پایین‌تر از شبکه ی عصبی بود. بنابراین به نظر می‌رسد الگورتیم درخت تصمیم برای این دیتاست از دیگر الگوریتم های پیاده‌سازی شده بهینه‌تر باشد.

٦- مراجع

[1] M. S. Elsayed, N. -A. Le-Khac, S. Dev and A. D. Jurcut, "DDoSNet: A Deep-Learning Model for Detecting Network Attacks," *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2020, pp. 391-396, doi: 10.1109/WoWMoM49955.2020.00072.