کنفرانس بین المللی پژوهش ها و فناوری های نوین در مهندسی برق
دانشگاه علم و فرهنگ
10 اسفند ماه 1401

پایگاه استنادی علوم جهان اسلام
۰۱۲۲۱-۱۶۷۲۳

بهبود عملکرد دستگاه های اعلام حریق اماکن جهت پیش‌بینی احتمال وقوع آتش سوزی با استفاده از الگوریتم های یادگیری ماشین

سجاد رضوانی خالدی[1]، سیدرضا جهادی[2] و هادی اشعریون[3]
[1] دانشکده مهندسی برق دانشگاه شهید بهشتی تهران، s.rezvanikhaledi@mail.sbu.ac.ir
[2] دانشکده مهندسی برق دانشگاه شهید بهشتی تهران، s.jahadihoseini@mail.sbu.ac.ir
[3] دانشکده مهندسی برق دانشگاه شهید بهشتی تهران، asharioun@sbu.ac.ir

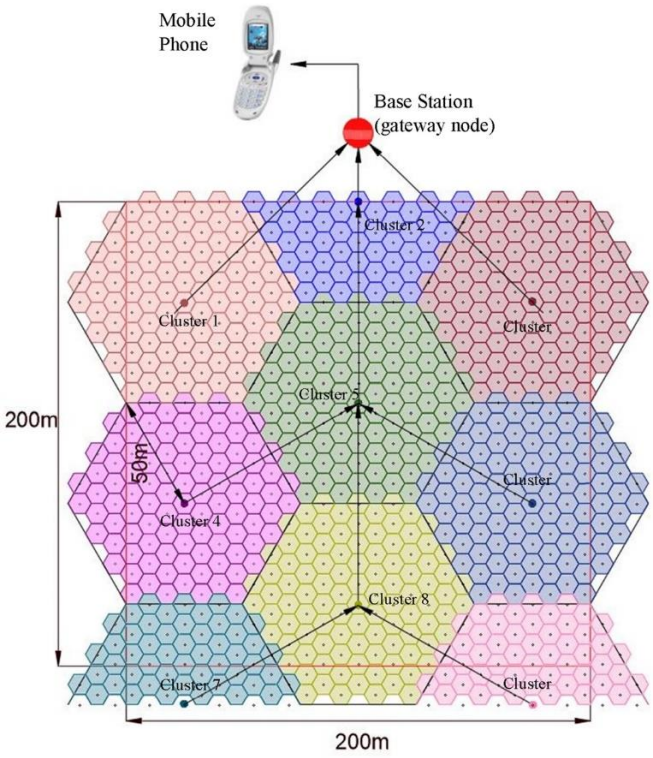
چکیده – دستگاه های اولیه اعلام حریق در ابتدا به صورت تکی و سپس به صورت شبکه ای ساخته شده‌اند تا آتش‌سوزی را به شکل بهتری تشخیص دهند. اگرچه پیشرفت های اخیر در طراحی دستگاه های اعلام حریق باعث تشخیص بهتر آتش سوزی ها شده است، ولی دستگاه های کنونی در آتش‌سوزی های ناگهانی یا در حجم زیاد آتش‌سوزی ناتوان می‌باشند. بهترین کار جهت آمادگی برای این تیپ از حوادث، پیشبینی احتمال آتش سوزی می‌باشد. دستگاهی که ما ارائه کرده‌ایم، به صورت شبکه ای متصل بهم از دستگاه های تشخیص دود و گاز های خطرناک به محوریت یک دستگاه واحد که معماری *Master-Slave* را شکل می‌دهند، می‌باشد که گره های فرعی (*nodes*) این شبکه در اصل وظیفه جمع آوری دیتاست کل محیط ساختمان را دارند و گره اصلی (*master*) با استفاده از این دیتاست و الگوریتم های یادگیری ماشین به بررسی احتمال وقوع آتش‌سوزی می‌پردازد. در این مقاله ابتدا به بررسی ساز و کار شبکه اعلام حریق پرداخته و سپس دو سناریوی بانظارت و بدون نظارت را در نظر گرفته‌ایم و سه مدل یادگیری ماشین را پیاده و دقت خروجی هر یک از این مدل ها را با هم مقایسه کرده‌ایم. در نهایت به این نتیجه رسیدیم که وجود این الگوریتم ها در دستگاه اعلام حریق، قابلیت تشخیص احتمال آتش‌سوزی را به وضوح بهبود داده‌اند.

کلید واژه- اعلام حریق، اعلام حریق های اولیه، پیش بینی، معماری Master-Slave، یادگیری ماشین.

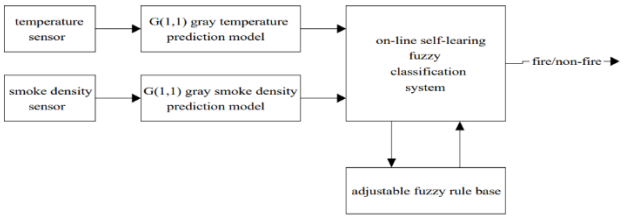
1- مقدمه

نسل اولیه دستگاه های اعلام حریق (1930) گازهای سمی محیط را تشخیص می‌دادند[1]. در سال 1960 دانشمندان کانادایی با انجام آزمایش در 342 خانه مسکونی به این نتیجه رسیدند که دستگاه های تشخیص دود تا 41٪ تعداد مرگ و میر را کاهش داده‌اند[2]. در ادامه، جهت بهبود تشخیص آتش‌سوزی و افزایش قدرت مانور و مانیتورینگ تشخیص حادثه، از دستگاه های چند سنسوره متشکل از نور، دود، دما و رطوبت به جای دستگاه های تک سنسور استفاده شد[3]. در محیط های بزرگ ساختمانی یا حتی اماکن وسیع همچون جنگل ها که طبیعتا یک دستگاه به تنهایی پاسخگو نمی‌باشد، شبکه ای از دستگاه های وایرلس ابداع شد (شکل 1) تا به صورت شبکه ای به مانیتورینگ محیط بپردازند[4]. در ادامه برای بهبود عملکرد این سیستم ها و خارج شدن از حالت solid (تشخیص آتشسوزی صرفا براساس کم و زیاد شدن مقادیر سنسور ها و بدون هیچ منطق خاصی)، دستگاه های هوشمند تشخیص آتش سوزی را داشتیم که بر اساس الگوریتم grey-fuzzy و به وسیله تنها دو سنسور به تشخیص آتش سوزی می پرداختند که نسبت به سری های اولیه پیشرفت چشمگیری بود[5]. در این الگوریتم در نهایت عمل تریگر 30 تا 60 ثانیه زودتر از نمونه های قبلی انجام می‌شود. در شکل 2 طرح کلی این الگوریتم را مشاهده می‌کنید.

پایگاه استنادی علوم جهان اسلام
۰۱۲۲۱-۱۶۷۲۳



شکل 1: ساختار شبکه‌ای سیستم اعلام حریق[4]



شکل 2: ساختار سیستم تشخیص آتش‌سوزی grey-fuzzy [5]

سیستم طراحی شده توسط ما درواقع سیستمی با یک دستگاه اصلی (master) و به تعداد لازم (برحسب بزرگی اتاق‌ها و مکان) دستگاه فرعی (slave) است که در یک شبکه master-slave با هم در ارتباط هستند و در اتاق های مختلف ساختمان نصب می‌شوند. ما از معماری master-slave به این دلیل استفاده کرده‌ایم که گره های فرعی نیازی به دریافت و پردازش اطلاعات ندارند و صرفا وظیفه جمع آوری دیتاست مورد نیاز گره های master را دارند که این رویه باعث بهینه سازی مصرف انرژی و همچنین نگه داری طولانی مدت بسته های داده (Data Packets) می‌شود[6]. گره های فرعی (دستگاه) ویژگی های محیطی (features) را جمع آوری می‌کنند و به گره های master جهت پیاده سازی مدل های یادگیری ماشین ارسال می‌کنند. هر گره فرعی یک محیط محدود را در بر می‌گیرد که به صورت یک Cluster می‌باشد و دیتاها در داخل شبکه بین یکدیگر جابه جا می‌شوند[4].

برای بهبود عملکرد دستگاه های اعلام حریق و مقایسه عملکرد سیستم پیاده سازی شده به وسیله مدل های یادگیری ماشین با سیستم پیاده سازی شده ای که صرفا براساس یک سری لبه (thershold) کار می‌کند ([5] [4] [3])، ما دو سناریو یادگیری بانظارت و بدون نظارت در نظر گرفته ایم که سه الگوریتم یادگیری ماشین به نام های Neural Network و Random Forest در سناریو اول و Anomally Detection در سناریو دوم پیاده کرده‌ایم. در این مقاله ما خطا و دقت هر یک را به صورت مجزا به دست آورده و با حالت solid (دستگاه بدون الگوریتم) مقایسه و مشاهده می‌کنیم که عملکرد دستگاه به صورت چشمگیری افزایش پیدا کرده است.

2- معماری Master-Slave

معماری Master-Slave به شبکه ای از دستگاه Master (ارباب) و دستگاه های Slave (برده) گفته می‌شود که ارباب به عنوان گره اصلی و برده ها به عنوان گره های فرعی می‌باشند. از مزیت های اصلی این معماری می‌توان به این موارد اشاره کرد: 1) بررسی اطلاعات به صورت همزمان و برخط در نقاط مختف صورت میگیرد و نتایج در آخر به یک واحد مرکزی منتقل می‌شوند. 2) اگر یک گره فرعی از کار بیوفتد یا از دسترس گره اصلی خارج شود ، هیچ وقفه ای در کارِ کل شبکهِ ایمنی اتفاق نمی‌افتد و باقی گره ها همچنان در حال جمع آوری اطلاعات می‌باشند. 3) این معماری از یک مخزن متمرکز تشکیل شده است که کار جمع آوری و آنالایز اطلاعات محیطی را تسهیل می‌بخشد. 4) شبکه قابلیت تکثیر و توسعه دارد. 5) باعث بهینه سازی مصرف انرژی می‌شود. 6) نگه داری طولانی مدت بسته های داده یا Data Packets.

هر کدام از گره های فرعی دارای چند سنسور می‌باشند که برای ما ویژگی (features) های زیر را جمع آوری می‌کنند:
1) رطوبت 2) دما 3) سنسور دود MQ139 4) TVOC 5) eCO2

پایگاه استنادی علوم جهان اسلام
۰۱۲۲۱-۱۶۷۲۳

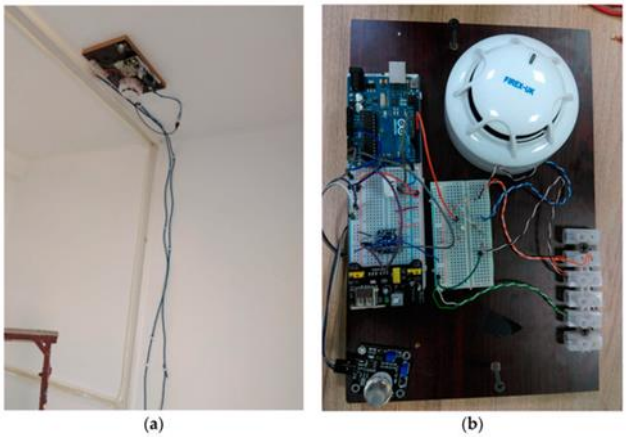
هر دستگاه فرعی یک شماره شناسایی یا ID دارد که به واسطه آن، دستگاه master متوجه می‌شود که دیتاست دریافتی مربوط به کدام دستگاه و در نتیجه کدام قسمت ساختمان دارد. در نهایت خروجی مدل های یادگیری ماشین تشخیص می‌دهند که کدام بخش ساختمان احتمالا دچار آتش‌سوزی می‌شود تا اقدامات لازم از پیش تعیین شود. گره های فرعی دارای باتری می‌باشند تا نیازی به زیر ساخت برقی و برق کشی ساختمان نباشد و همچنین شبکه به راحتی قابل توسعه باشد. نحوه ارتباط گره های معماری master-slave و ارسال اطلاعات به یکدیگر از طرح [6] الهام گرفته شده است. نتایج خروجی بدست آمده توسط گره Master به بخش مانیتورینگ ارسال می‌شوند. در ادامه به بخش یادگیری ماشین دستگاه اعلام حریق می‌پردازیم.

3- یادگیری ماشین

در این قسمت در ابتدا به پیش پردازش (Preprocessing) دیتا پرداخته و سپس مدل های یادگیری ماشین را پیاده می‌کنیم. کدنویسی و پیاده سازی الگوریتم های توضیح داده شده در این مقاله و همچنین نتایج آن (نمودار ها و شکل ها) در [7] قابل مشاهده می‌باشند.

3-1- پیش پردازش داده ها

برای دیتای ورودی الگوریتم ها که روی آنان یادگیری مدل ها انجام می‌شود از دیتابیس Indoor Laboratory Fire Dataset از دانشگاه Zayed University - Abu Dhabi Campus استفاده کرده‌ایم[8] . برای بدست آوردن این دیتابیس از دستگاه شکل 3 استفاده شده است که مشابه باقی دستگاه های معمول تشخیص آتش سوزی از سنسورهای تشخیص دما و رطوبت و دود تشکیل شده است. طبق دیتای خود این دیتابیس می توان دقت (recall) دستگاه را بدست آورد تا پس از آموزش مدل های ماشین لرنینگ با دقت های بدست آمده مقایسه کرد.


(a) (b)

شکل 3: دستگاه ساخته شده برای جمع آوری دیتاست [7]

این دیتابیس از 8 دیتاست مختلف تشکیل شده که دوتا مربوط به آتش سوزی کارتن، دوتا پارچه، چهارتا الکتریکال است. برای کلی تر کردن (generalize) مدل، این 8 دیتاست را باهم جمع (concatenate) کرده‌ایم و یک دیتاست نهایی با 11767 سطر (رکورد) و 8 ستون بدست آمد که فرآیند یادگیری مدل ها را با آن انجام می دهیم و از اینجا به بعد از آن به عنوان دیتاست نام میبریم.

در ابتدا ستون های هر دیتاست (features) را توضیح می‌دهیم:

Time: زمان ثبت هر رکورد

Reading ID: آی دی مشخص هر رکورد

Humidity: درصد رطوبت محیط

Temperature: دمای محیط

MQ139: گاز های VOC (بیشترین حساسیت به سطح گاز آمونیاک و فرون)

TVOC: مجموع سطح ترکیبات ارگانیگ فرار در هوا

eCO2: سطح گاز دی اکسید کربن محیط

Detector: گزارش دستگاه از وقوع آتش سوزی

Status: دارای سه حالت: 0: عدم وقوع آتش سوزی و هشدار غیر فعال 1: وقوع آتش سوزی ولی عدم فعال شدن هشدار 2: وقوع آتش و فعال شدن هشدار.

از این دو ستون آخر علاوه بر این که وقوع و عدم وقوع آتش سوزی بدست می آید که لیبل (lable) ها دیتا هستند می توان حالات (رکورد) هایی که آتش سوزی رخ داده ولی دستگاه تشخیص نداده بدست آورد که این همان False_Negetive های دستگاه

پایگاه استنادی علوم جهان اسلام
۰۱۲۲۱-۱۶۷۲۳

است. لذا می توان به عددی برای دقت (recall) دستگاه ساخته شده اولیه دست پیدا کرد. با توجه به این که دیتاست در ستون status به تعداد 6665 مقدار 1 و 2831 مقدار 0 و 2301 مقدار 2 دارد، recall دستگاه 0.2566 بدست می آید.

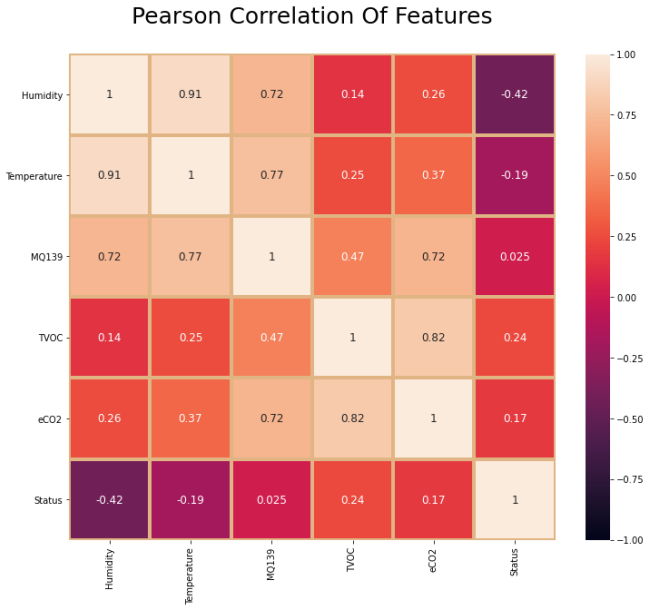
همانطور که در شکل 4 مشاهده می‌کنید، برای محاسبه recall که یکی از معیار های تعیین دقت مدل است از میزان TP (true_positive) و FN (false_negetive) استفاده می‌شود که به ترتیب میزان دفعاتی است که دستگاه وقوع آتش سوزی را به درستی تشخیص داده و دفعاتی که آتش سوزی اتفاق افتاده بوده ولی دستگاه به خطا عدم وقوع آتش سوزی اعلام کرده است که به ترتیب در دیتاست های آتش سوزی پارچه، سپس در کارتن و در آخر در الکتریکال از همه کمتر است و همین موضوع باعث کاهش آن در دیتاست نهایی میشود.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

شکل 4: فرمول Recall مدل [9]

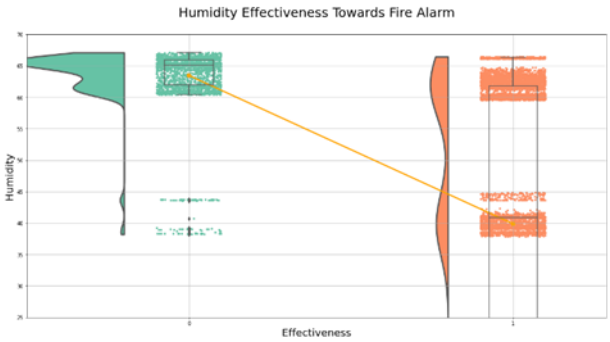
در ادامه این میزان recall محاسبه شده برای دستگاه را، recall دستگاه اولیه یا معمولی می‌نامیم و با recall بدست آمده از مدل ها مقایسه می‌کنیم. پس از حذف ستون های اضافی و بدست آوردن سطرهای تکراری، کرلیشن پیرسون (pearson correlation) ویژگی‌ها (features) را باهم حساب می کنیم که معیاری برای سنجش ستون هایی است که روند آنها ارتباط بیشتری باهم دارند.

طبق شکل 5 همانطور که مشاهده می‌کنید ستون status که لیبل دیتا است بیشترین رابطه معکوس را با humidity و بیشترین رابطه مستقیم را با TVOC دارد. حال نمودار Raincloud این دو ستون دیتاست را رسم می‌کنیم. در هریک از شکل های 6 و 7 به ترتیب توزیع رطوبت و TVOC برای دو کلاس label status =1 & 0 که نماینده وقوع یا عدم وقوع آتش سوزی است، همراه با نمودار شمعی هر توزیع رسم شده است. در نمودار شمعی چهارک اول تا چهارم هر یک از توزیع ها مشخص شده است. همچنین در هر نمودار خط نارنجی اتصال که میانگین دوتوزیع 0و1 را به هم وصل می‌کند.



شکل 5: کرلیشن پیرسون دوبه دو ستون های دیتاست

شکل 6: نمودار RainCloud ستون رطوبت نشان دهنده توزیع داده بر دو کلاس status 1و0



شکل 7: نمودار RainCloud ستون TVOC نشان دهنده توزیع داده بر دو کلاس status 1و0

پایگاه استنادی علوم جهان اسلام
۰۱۲۲۱-۱۶۷۲۳

در نظر داشته باشید قصد داریم از این دیتاست برای رسیدن به دو هدف در دو سناریو مختلف از فرایند کاری دستگاه استفاده کنیم.

سناریو اول: بدست آوردن مدلی اولیه برای قراردادن بر دستگاه به منظور پیشبینی و تشخیص وقوع آتش‌سوزی در هر مکان پیش از نصب دستگاه و بدست آوردن دیتای بخصوص خود آن محیط. اگرچه این دیتاست های مورد استفاده، مختصِ همان محیطِ بخصوصِ جمع آوری آن است، ولی از آنجایی که توزیع دیتا با همین موضوع در مکانهای دیگر مشابه است، این دیتا برای بدست مدل اولیه با نظارت (Supervised) (از آنجایی که دیتای موجود لیبل دارد) برای استفاده در دستگاه ها، تقریب خوبی است و بهتر از فرآیند فعلی (که صرفا براساس یک سری لبه (thershold) کار می‌کند [3] [4] [5]) مورد استفاده در دستگاه های معمولی موجود عمل می‌کند.

سناریو دوم: بافرض اینکه دستگاه در محیط نصب شده و به صورت مداوم در حال ذخیره اطلاعات است و دیتاستی از همان محیط بدست آورده‌ایم، می‌خواهیم مشاهده کنیم که با فیت کردن مدل بدون نظارت (Unsupervised) به این دیتای محیطی، چه نتیجه ای حاصل می‌شود و چه بهبودی در عملکرد دستگاه از آن به بعد به نسبت دستگاه اولیه ایجاد می‌شود؟ یعنی فرض می کنیم دستگاه نصب شده ما در یک محیط، همان دستگاه ساخته شده‌ای است که دیتای محیط را جمع آوری و ذخیره کرده و بررسی می‌کنیم که با فیت کردن مدل یادگیری ماشین به چه دقتی می‌توان دست پیدا کرد.

3-2 پیاده سازی مدل های یادگیری ماشین

سناریو دوم: بافرض اینکه دستگاه در محیط نصب شده و به صورت مداوم در حال ذخیره اطلاعات است و دیتاستی از همان محیط بدست. در اینجا برای پیاده سازی مدل از سه روش پرکاربرد classification استفاده می‌کنیم:

1 - Anomaly Detection : برای این قسمت از کتابخانه sklearn کمک گرفتیم ولی بسیاری قسمت ها مانند محاسبه ε بصورت کاملا دستی (scratch) صورت گرفته است. این روش یکی از روش های بدون نظارت unsupervised است. علت استفاده از این روش این است که ذات مسئله‌ی تشخیص حادثه در سناریو دوم بدون نظارت است، زیرا توزیع دیتا انحراف دارد (skewed) و زمانی که مدل اولیه آموزش داده می‌شود و در محل دستگاه مورد استفاده قرار می‌گیرد و در آنجا به ذخیره دیتای محیط می‌پردازد، وقوع حالات اتفاق افتادن حادثه بسیار قریب الوقوع تر از عدم وقوع حادثه است. از طرفی دیگر، ممکن است در مواقع تست حالاتی از وقوع حادثه رخ دهد که در دیتاست اولیه وجود نداشته است. در این مواقع بهتر است که یک مدل بدون نظارت (unsupervised) استفاده شود و از آن معدود دیتای لیبل خورده برای دیتای تست و کراس ولیدیشن (CV) جهت تایین مقدار حدی ε استفاده شود[10].

در این دیتاست تمام دیتای نرمال که 2831 عدد بودند برای آموزش مدل قرار دادیم. باقی دیتا را که دیتای آنرمال و مربوط به وقوع حادثه بودند، به دو دسته Cross Validation برای تعیین حد ε و Test برای تست و تعیین دقت نهایی مدل، تقسیم می‌کنیم.

در این روش یک توزیع نرمال (زنگوله ای) بر دیتای نرمال فیت می‌کنیم و سپس با توجه به بهینه شدن دقت مدل بر روی دیتای کراس ولیدیشن حد مرزی ε تایین می‌شود که در صورتی که رکوردی بر این مدل مقادیر کمتر از ε داشت، انرمال (anomal) تشخیص داده می‌شود. با مقدار ε بدست آمده رو دیتای CV برابر با 0.27 بدست می‌آید که میزان F1_score مدل 0.97062 و میزان recall آن 0.95035 بدست می‌آید و شکل confusion matrix آن در شکل 8 نمایش داده شده است.

TP: 312	FN: 119
FP: 0	TN: 1966

شکل 8: confusion matrix

همانطور که مشاهده می‌کنید این recall با اختلاف از recall خود دستگاه اولیه (0.2566) بهتر است. بنابرین این الگوریتم به خوبی می‌تواند در سناریو دوم با جمع آوری دیتای مختص هر محیط بعد از نصب دستگاه در آن محل به یادگیری مدل با استفاده از دیتای همان محیط بپردازد.

2 - Random Forest : برای این قسمت از کتابخانه sklearn کمک گرفتیم. این روش بانظارت درواقع از تکامل روش decision tree بوسیله‌ی sample with replacement بدست آمده است. برای

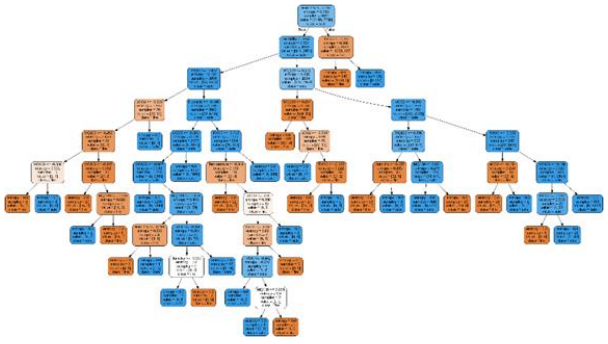
پایگاه استنادی علوم جهان اسلام
۰۱۲۲۱-۱۶۷۲۳

این روش و روش بعد دیتا را به نسبت 0.2 به دسته Train,Test تقسیم میکنیم. 9437 رکورد برای train و 2360 رکورد برای test. در این روش در هر مرحله یکی از خصوصیت های دیتا انتخاب می‌شود و تمام دیتای آن مرحله براساس آن خصوصیت به دو دسته تقسیم بندی می‌شود. در هر مرحله بی‌نظمی (entropy) آن دسته از رابطه شکل 9 بدست می‌آید.

$$E = -p*\log2(p) - q*\log2(q)$$

شکل 9: فرمول محاسبه آنتروپی هر دسته[11]

اختلاف بی¬نظمی بین دو مرحله میزان نظم (information gain) را نشان می‌دهد که هرچه بیشتر باشد خصوصیت بهتری را دسته بندی گروه ها انتخاب کرده‌ایم. پارامتر های این الگوریتم را نیز n_estimators=200 و random_state = 1899 تنظیم می‌کنیم. درخت تصمیم نهایی بدست آمده برای دسته بندی حالات به شکل 10 در می‌آید: در هر دسته هر چه انتروپی به 0.5 نزدیکتر باشه بی‌نظمی حداکثر است و رنگ آن دسته به سفید نزدیکتر است. هرچه میزان انتروپی به 0 یا 1 نزدیکتر باشد رنگ آن دسته به آبی و نارنجی نزدیکتر است که نشان دهنده جداسازی کاملتر دسته ها از هم است.



شکل 10: شکل درخت نهایی بدست آمده برای کلاس بندی دیتا

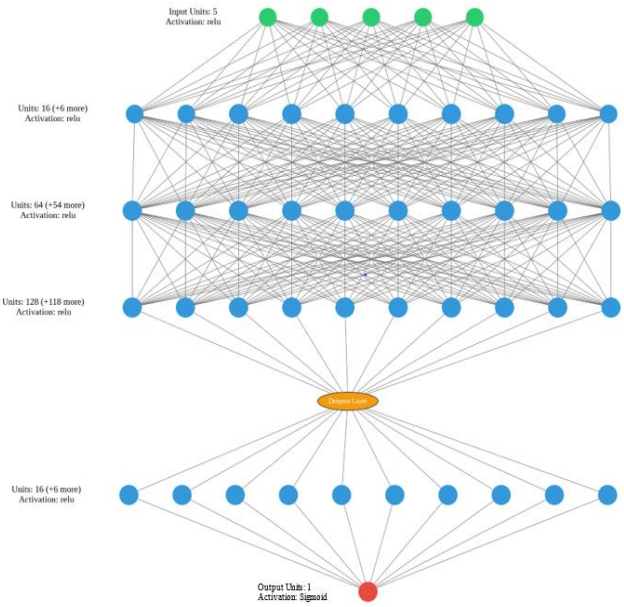
در نهایت f1_score این مدل0.99872 و recall آن 0.994 بدست می آید(شکل 11).

TP: 570	FN: 3
FP: 0	TN: 1787

شکل 11:Confusion Matrix

3- Neural Network : برای این قسمت از کتابخانه keras و tensor_flow کمک گرفتیم.

تعریف مدل: در این روش یک شبکه عصبی تعریف می‌کنیم که دارای 7 لایه است. این شبکه به ترتیب دارای یک لایه ورودی با 5 نود، چهار لایه Dense به ترتیب دارای 16، 64، 128 و16 نود با تابع فعال سازی relu (activation function) و 1 نود خروجی با تابع فعال سازی sigmoid (activation function) است. همچنین قبل از لایه Dense از BatchNormalization استفاده کرده‌ایم. علت این امر، نرمالایز کردن داده های خروجی هر لایه قبل از دادن آن به لایه های بعدی درنتیجه جلوگیری از Gradient vanishing است. همچنین قبل از لایه پنجم نام برده، از یک لایه drop out با ضریب 0.5 استفاده کرده ایم تا بصورت رندم بعضی از نود هارا غیرفعال کند و از Overfitting جلوگیری کند. شکل 12.



شکل 12: ساختار شبکه عصبی طراحی شده

پایگاه استنادی علوم جهان اسلام
۰۱۲۲۱-۱۶۷۲۳

کامپایل مدل: از آنجایی که مسئله کلاس بندی است از loss function Binary cross entropy استفاده می‌کنیم، برای کاهش خطای محاسباتی، در هنگام تعریف مدل در Sequential برخلاف باقی لایه ها که activation function شان را تعریف می‌کنیم، لایه آخر را linear رها می‌کنیم و در هنگام کامپایل مدل که loss فانکشن را تعریف میکنیم، از پارامتر from_logits = True استفاده می‌کنیم تا اعمال تابع sigmoid در آخرین لایه و محاسبه loss با هم صورت گیرد که خطای محاسباتی در مقایسه با محاسبه جدا جدای تابع sigmoid و سپس محاسبه loss، کاهش میابد چون یک مرحله رند سازی کم می‌شود.

برای بهینه سازی مدل از optimizer Adam با پارامتر 0.0003 استفاده می‌کنیم که پارامتری از اینرسی مدل است. همچنین برای پارامتر metrics تعیین کننده معیار هایی است که در زمان یادگیری مدل نمایش داده می‌شود، علاوه بر Accuracy از AUC_ROC و F1score استفاده می‌کنیم که معیارهایی بر اساس بهینه بودن precision و recall محاسبه می شوند. Auc_ROC نیز همان مساحت زیر منحنی presicio-recall است که با F1_score رابطه مستقیم دارد. ضمنا از آنجا که keras خود تابعی از پیش تعریف شده باید برای محاسبه F1_score ندارد، آن را با استفاده از تعریف recall و precision در شکل 13 تعریف کردیم:

$$F1\ Score = 2 \times \frac{recall \times precision}{recall + precision}$$

شکل 13: محاسبه f1_score با recall و precision [12].

فیت کردن مدل: در این قسمت دیتای ورودی شبکه را که همان x_train,y_train است را تعیین می‌کنیم. همچنین پارامتر های epochs = 150 و batch_size=256 را تعیین می‌کنیم. در قسمت فانکشن های call_back از دوتابع استفاده کرده ایم: Early, callback_obj. اولی برای early stopping است که با تعیین پارامتر های آن بر mode='max',patience=25 مشخص می‌کنیم که پس از میزانی مشخص که val_accuracy در حداکثر می‌ماند آموزش مدل متوقف شود تا مدل دچار overfitting نشود. دومی آبجکتی از کلاس mycallback است که در این کلاس می‌توانیم توابعی تعریف کنیم که در اول یا آخر هر epoc متغیری از آموزش مدل را چک کند و عملی را انجام دهد. در اینجا ما از برای قطع آموزش یادگیری زمانی که accuracy == 1 شد استفاده کردیم.

با اجرای کد بعد از این سه مرحله آموزش مدل انجام می‌شود. اگرچه مقدار epoces=150 تعیین کرده‌ایم به دلیل Earlly stopping که بالاتر توضیح داده شد در epoc 77ام فرآیند آموزش به پایان می‌رسد و نتایج زیر از دقت مدل بدست می‌آید:

loss: 0.1686
accuracy:0.9870
f1: 0.9914
auc_17: 0.9994
recall_15: 0.9958
val_loss:0.1580
val_accuracy:0.9890
val_f1: 0.9933
val_auc:0.9998
val_recall: 0.9994

حال می‌توان مقادیر پارامتر های شبکه را مشاهده کرد و در قالب فایل .h5 خروجی گرفت. سپس نمودار های آموزش مدل را طبق شکل 14 رسم می‌کنیم.

شکل 14: نمودار accuracy و loss

همانطور که مشاهده می‌کنید recall بدست آمده در این روش بر داده تست 0.9994 بسیار بالاتر از recall دستگاه اولیه است که به این معناست این با استفاده از این روش ها learning میزان false_negetive دستگاه را به طرز چشم گیری می‌توان کاهش داد. این روش شبکه عصبی که در کنار روش Random forest در سناریو اول قرار می‌گیرد، از دقت بالاتری (0.99872<0.9994) با اختلافی ناچیز از Random forest دقیق تر است هرچند که آموزش شبکه عصبی از لحاظ زمانی، زمانبر تر است ولی در مسئله ما اولویتی ایجاد نمی‌کند.

پایگاه استنادی علوم جهان اسلام
۰۱۲۲۱-۱۶۷۲۳

4- نتیجه‌گیری

در این مقاله ما به پیاده سازی سه مدل یادگیری ماشین در دو سناریو متفاوت پرداختیم و نشان دادیم که با استفاده از روش Anomally detection در سناریو دوم و روش های Random forest و Neural Network در سناریو اول دقت بالاتری نسبت به حالت عادی خواهیم داشت. پس وجود این الگوریتم ها به وضوح برتری دستگاه را نسبت به دستگاه های کنونی که الگوریتم آن ها به صورت fuzzy می باشد و در threshold های مورد نظر تریگر میکنند، نشان می‌دهد.

مراجع

[1] Bistrović, Miroslav & Kezić, Danko & Komorčec, Domagoj. (2013). Historical development of fire detection system technology on ships. 60. 127-133.
[2] Ahrens, M, Fire Analysis and Research Division National Fire Protection Association, Quince, 2007th (Available on www.nfpa.org/ ~ / media Files / Research / OSHomes.ashx)
[3] Ding Q, Peng Z, Liu T, Tong Q. Multi-Sensor Building Fire Alarm System with Information Fusion Technology Based on D-S Evidence Theory.Algorithms.2014;7(4):523-537. https://doi.org/10.3390/a7040523
[4] Dampage, U., Bandaranayake, L., Wanasinghe, R. et al. Forest fire detection system using wireless sensor networks and machine learning. Sci Rep 12, 46 (2022). https://doi.org/10.1038/s41598-021-03882-9
[5] H.C. Kuo, H.K. Chang,A real-time shipboard fire-detection system based on grey-fuzzy algorithms,Fire Safety Journal,Volume 38, Issue 4,2003,Pages341-363,ISSN0379-7112,https://doi.org/10.1016/S0379-7112(02)00088-7.
[6] Jan S, Yafi E, Hafeez A, Khatana HW, Hussain S, Akhtar R, Wadud Z. Investigating Master-Slave Architecture for Underwater Wireless Sensor Network. Sensors (Basel). 2021 Apr 25;21(9):3000. doi: 10.3390/s21093000. PMID: 33922886; PMCID: PMC8123114.
[7] S. Rezvani (28/1/2023) safehomepro. https://github.com/Sajjad-RK/safehomepro
[8] Amril Nazir, Zayed University - Abu Dhabi Campus, Indoor LaboratoryFireDataset,12February2021,DOI:10.17632/f3mjnbm9b3.1
[9] Lavanya Gupta, Precision-Recall Tradeoff in Real-World Use Cases, Feb 19, 2021
[10] N. Goernitz, M. Kloft, K. Rieck, U. Brefeld,Toward Supervised Anomaly Detection,DOI: 10.1613/jair.3623
[11] Aditya Kumar Pandey, Decision Tree Entropy|Entropy Calculation, Aug 13, 2020
[12] TOM KELDENICH, Recall, Precision, F1 Score – Simple Metric Explanation Machine Learning, 2 SEPTEMBER 2021