

# Xây dựng mô hình Machine Learning dự báo cháy rừng ở các tỉnh Tây Nguyên dựa vào dữ liệu lịch sử thời tiết.

Nguyễn Đại Kỳ  
19521731

Văn Viết Hiếu Anh  
19521225

Lê Văn Phước  
19522054

Ngày 18 tháng 8 năm 2021

Môn học: CS114 - Máy học  
Giảng viên hướng dẫn: Lê Đình Duy  
Phạm Nguyễn Trường An

## Mục lục

<b>1 Tổng quan</b>	<b>2</b>
1.1 Mô tả bài toán . . . . .	2
1.2 Mô tả dữ liệu . . . . .	2
1.2.1 Weather Data . . . . .	2
1.2.2 Fire Data . . . . .	3
<b>2 Các nghiên cứu trước</b>	<b>3</b>
2.1 Predicting Australia wildfires with weather data a Call For Code spot challenge of IBM [6] . . . . .	3
2.2 AI and Climate data for predicting fire frequency in California [3] . . . . .	4
<b>3 Xây dựng bộ dữ liệu</b>	<b>4</b>
3.1 Quá trình thu thập dữ liệu . . . . .	5
3.1.1 weather.com . . . . .	5
3.1.2 worldweatheronline.com . . . . .	9
3.1.3 firewatchvn.kiehlam.org.vn . . . . .	12
3.2 Xây dựng bộ dữ liệu . . . . .	15
3.2.1 Imputation of Data . . . . .	15
3.2.2 Creation of Dataset . . . . .	17
<b>4 Training và đánh giá modelods</b>	<b>17</b>
4.1 Thử nghiệm trên dữ liệu 5 tỉnh Tây Nguyên . . . . .	17
4.1.1 Model dự đoán cháy rừng . . . . .	17
4.1.2 Model dự đoán cấp độ cháy rừng . . . . .	19
4.2 Thực hiện trên dữ liệu 3 tỉnh Tây Nguyên (Gia Lai, Lâm Đồng, Đắk Lắk) . . . . .	21
4.2.1 Model dự đoán cháy rừng . . . . .	22
4.2.2 Model dự đoán cấp độ cháy rừng . . . . .	25
4.2.3 Tổng kết . . . . .	30
<b>5 Ứng dụng và hướng phát triển</b>	<b>31</b>
5.1 Ứng dụng . . . . .	31
5.2 Hướng phát triển . . . . .	31

# 1 Tổng quan

Bài viết là về quá trình thực nghiệm nghiên cứu các model Machine Learning với mục đích chọn ra mô hình tối ưu để dự đoán mức độ cháy rừng dựa vào dữ liệu thời tiết trong lịch sử của từng địa phương. Với mục đích hỗ trợ trong việc dự đoán để phục vụ trong công tác phòng chống cháy rừng ở nước ta. Vì bài viết là ghi chép của quá trình thực nghiệm nên sẽ có nhiều phương pháp được đưa ra sử dụng.

## 1.1 Mô tả bài toán

Ở nước ta có 3 thảm họa lớn nhất, gây thiệt hại lớn hàng năm về cả người và của. Cùng với lũ lụt và hạn hán, cháy rừng là một thảm họa gây thiệt hại không chỉ về kinh tế mà còn cả con người và hệ sinh thái. Theo thống kê của Cục Kiểm lâm từ năm 1992 đến 2006, trung bình mỗi năm xảy ra 1254 vụ cháy rừng gây thiệt hại khoảng 6646 ha rừng, trong đó có 2854 ha là rừng tự nhiên và 3791 ha là rừng trồng. Bên cạnh việc nâng cao năng lực phòng cháy chữa cháy rừng (PCCCR) cho lực lượng kiểm lâm như đầu tư trang thiết bị, cơ sở vật chất, xây dựng cơ chế điều hành phối hợp và tuyên truyền nâng cao nhận thức trách nhiệm của chủ rừng và người dân, công tác cảnh báo nguy cơ cháy rừng cũng như tổ chức phát hiện sớm và thông báo kịp thời điểm cháy rừng là rất cần thiết.

Từ đầu năm 2007, Cục Kiểm lâm (Bộ Nông nghiệp và Phát triển Nông thôn) đã lắp đặt và vận hành trạm thu ảnh viễn thám MODIS tại Hà Nội với mục đích chính là phát hiện sớm các điểm cháy rừng (hotspots) trên toàn lãnh thổ Việt Nam. Hệ thống trạm thu của TeraScan đã tự động thu nhận, xử lý và sao lưu dữ liệu ảnh MODIS hàng ngày từ 2 vệ tinh TERRA và AQUA với mô-đun Vulcan tự động xử lý và tạo ra dữ liệu các điểm cháy sử dụng thuật toán ATBD-MOD14 [5].

Hệ thống này cung cấp dữ liệu về điểm cháy ghi nhận được từ vệ tinh và lưu lại thời gian và tọa độ cháy. Từ khi bắt đầu lắp đặt đến nay hệ thống dữ liệu cháy của cục kiểm lâm được ghi lại được gần 1 triệu điểm cháy. Nhờ lượng dữ liệu này việc xây dựng một hệ thống tự động phân tích mức độ cháy rừng dựa vào các đặc trưng cơ bản của dữ liệu khí tượng thủy văn là hoàn toàn có cơ sở và khả quan.

Input của bài toán là dữ liệu lịch sử thời tiết của địa phương trong vòng 1 tháng trước ngày xảy ra vụ cháy.

Output của bài toán là:

- Model 1 cho ra kết quả một trong hai giá trị 0 hoặc 1 đại diện cho địa phương đó có xảy ra cháy rừng hay không?
- Model 2 cho ra kết quả là cấp độ cháy rừng gồm 3 nhãn là nhãn 1 (cháy rừng cấp độ 1), nhãn 2 (cháy rừng cấp độ 2), nhãn 3 (cháy rừng cấp độ 3).

## 1.2 Mô tả dữ liệu

Nguồn dữ liệu của nhóm đến từ các website gồm 3 website chính:

- firewatchvn.kiemlam.org.vn: là Hệ thống theo dõi cháy rừng trực tuyến thuộc Cục Kiểm Lâm - Tổng cục Lâm Nghiệp
- weather.com: là website của The Weather Channel (TWC) - IBM [8]
- worldweatheronline.com

Trong 3 nguồn dữ liệu thì chỉ có worldweatheronline.com sử dụng SSR(Server-side render) còn 2 nguồn còn lại đều sử dụng Ajax để truyền dữ liệu qua lại giữa server.

### 1.2.1 Weather Data

*Việc tìm các nguồn dữ liệu khác về thời tiết ngoài 2 nguồn trên đã được thực hiện song các nguồn này đều có những điểm thiếu rất quan trọng ví dụ như API của website chỉ cung cấp trong 1 năm trở lại hay các website này không cung cấp đủ nhiều địa phương mà chỉ cung cấp dữ liệu ở những thành phố cụ thể.*

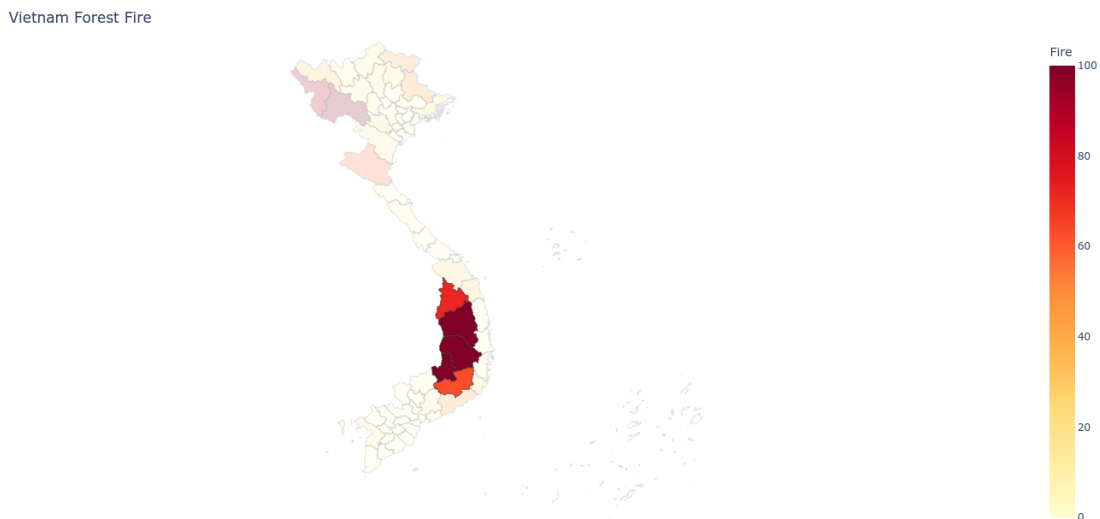
Có thể nói việc lấy dữ liệu thời tiết là công đoạn gây ra nhiều khó khăn nhất. Đa phần dữ liệu lịch sử thời tiết là rất lớn và các công ty hay tập đoàn công nghệ đều dùng để bán chứ không public trên website của họ. Ngay cả trên giao diện chính của weather.com của IBM cũng chỉ hiển thị dữ liệu thời tiết trong 2 năm trở lại (tức là 2021 và 2020).

Tuy nhiên vì sử dụng Ajax nên sau khi phân tích và ghi lại các request mà website gửi đi cũng như các response nhận về, việc có thể tìm được các cổng API và phương thức giao tiếp với server, từ đó dùng vào việc khai thác dữ liệu tự động trong nhiều năm trước nữa là hoàn toàn có hy vọng.

Về việc tìm nguồn dữ liệu tương tự đã được khai thác trước, nhóm đã từng thử tìm kiếm nhưng để đạt được yêu cầu chi tiết đến từng địa phương với thời gian kéo dài thì không tìm được dữ liệu nào đạt yêu cầu. Ngay cả khi join vào Slack của Call For Code năm nay để xin hỗ trợ vì đề tài này có liên quan đến cuộc thi thì phía ban tổ chức cuộc thi cũng trả lời rằng dữ liệu này không cung cấp cho thí sinh. Ngoài ra nhóm cũng đã thử gửi mail cho Trung tâm Dự báo khí tượng thủy văn quốc gia nhưng cũng không nhận được phản hồi. Hiển nhiên, việc tự đi thu thập dữ liệu là tất yếu.

### 1.2.2 Fire Data

firewatchvn.kiemlam.org.vn là website tạo ra ý tưởng cho nhóm. Website này cung cấp giao diện tra cứu dữ liệu về các điểm cháy vào từng thời gian cụ thể. Tuy nhiên điểm yếu của website này là xây dựng quá nhiều tính năng và sử dụng Ajax nên dùng những công cụ như BeautifulSoup thì không thể thu thập còn nếu dùng những công cụ như Selenium hay Puppeteer thì tốc độ quá chậm (dữ liệu này kéo dài từ 1/1/2008 đến 5/12/2020 nếu tra cứu từng ngày trên 700 quận, huyện, thành phố thuộc tỉnh,... thì sẽ mất rất nhiều thời gian). Điều này bắt buộc nhóm phải phân tích API mà website đã lấy dữ liệu điểm cháy để tăng tốc độ lấy dữ liệu. Bởi vì những thông tin như bản đồ của địa điểm lấy dữ liệu là không cần thiết, ta hoàn toàn có thể lấy bản đồ địa hình của cả trái đất chỉ cần dùng tọa độ. Việc lấy những thông tin nặng như bản đồ cần thời gian tải rất lâu nên việc tìm ra API mà website sử dụng cũng là cần thiết.



Hình 1: Dữ liệu cháy biểu thị trên bản đồ

## 2 Các nghiên cứu trước

### 2.1 Predicting Australia wildfires with weather data a Call For Code spot challenge of IBM [6]

Mục đích của cuộc thi này là dự đoán kích thước của khu vực cháy tính bằng km<sup>2</sup> theo khu vực ở Úc cho mỗi ngày trong tháng 2 năm 2021 bằng cách sử dụng dữ liệu có sẵn cho đến ngày 29 tháng 1. Bài toán chuỗi thời gian dựa trên dữ liệu hàng ngày do Pairs Geoscope cung cấp. Dữ liệu được cung cấp bao gồm:

- Những trận cháy rừng trong lịch sử
- Thời tiết lịch sử
- Các dự báo thời tiết lịch sử
- Chỉ số thảm thực vật lịch sử
- Các loại đất của các địa điểm xảy ra cháy rừng.

Trong đó dữ liệu thời tiết lịch sử bao gồm các trường như **vùng**, **thời gian**, **lượng mưa** (mm/ngày), **Độ ẩm tương đối** (%), **Hàm lượng nước trong đất** ( $m^3/m^3$ ), **bức xạ mặt trời** (MJ/ngày), **hiệu độ** (C), **tốc độ gió** (m/s). Đây là dữ liệu chính áp dụng vào bài toán của nhóm. Tuy nhiên vì khó khăn trong việc tìm kiếm dữ liệu nên nhóm chỉ đáp 4/6 tiêu chí mà các chuyên gia đã đưa ra đó là lượng mưa, độ ẩm, hiệu độ, tốc độ gió.

Từ những nghiên cứu các chuyên gia đưa ra các biến ảnh hưởng đến cháy rừng như:

- Lãnh thổ:
  - Khu vực: Số lượng và cường độ đám cháy khác nhau ở các khu vực khác nhau. Các sự kiện ở các khu vực lân cận có thể ảnh hưởng đến cháy rừng trong một lãnh thổ nhất định. Thành phần tự phục hồi( sự diện diện của cháy rừng trong những ngày trước đó).
  - Tính theo mùa: Cháy rừng đặc biệt dữ dội trong “Mùa cháy rừng” kéo dài từ tháng 10 đến tháng 12. Quan sát này sẽ ảnh hưởng đến cách thức phân chia tập dữ liệu huấn luyện và kiểm tra.
- Điều kiện đất và khí quyển:
  - Thời tiết và đất đai: Thời tiết và Hạn hán có liên quan chặt chẽ đến hỏa hoạn. So sánh giữa lịch sử và dự báo thời tiết trong lịch sử.
  - Thảm thực vật: Có mối tương quan thuận giữa sự thay đổi chỉ số thảm thực vật và cường độ cháy.
  - Sử dụng đất: Việc sử dụng đất có thể liên quan đến việc dự đoán sự kéo dài của cháy rừng. Tuy nhiên, dữ liệu này chỉ có sẵn dưới dạng một hàng – cho mọi khu vực, do đó nó không được đưa vào mô hình.

Theo báo cáo tổng kết cuộc thi thì mô hình được sử dụng là Convolutional Neural Network (Windowing Dataset, Conv1D Layers,...) cho ra kết quả RMSE: 19.96, MAE: 6.94, TOT: 9.54 Qua nghiên cứu của cuộc thi này nhóm rút ra được các biến ảnh hưởng đến cháy rừng, cách khai thác lấy dữ liệu từ thực tế, mô hình đào tạo phù hợp với bài toán.

## 2.2 AI and Climate data for predicting fire frequency in California [3]

Mục đích của nghiên cứu này là dự đoán tần suất xảy ra cháy rừng có thể giúp lập kế hoạch khẩn cấp và chủ động quản lý rủi ro thiên tai. Dữ liệu về đám cháy được thu từ Kaggle(<https://www.kaggle.com/rtatman/188-million-us-wildfires>), bao gồm khoảng 1,9 triệu vụ cháy rừng ở Mỹ được tham chiếu trong giai đoạn 1992-2015. Ngoài ra, dữ liệu khí hậu từ Copernicus ERA5 được tải xuống và sử dụng cho bài toán này.

Bài nghiên cứu này cũng đưa ra kết luận về việc cháy rừng thường xảy ra hàng tháng hoặc theo mùa. Ở đây, các chuyên gia đưa ra các trường dữ liệu thời tiết được lấy là **total\_precipitation** (tổng lượng mưa), **2m\_temperature** (hiệu độ), **2m\_dewpoint\_temperature** (hiệu độ điểm sương), **10m\_wind\_speed** (tốc độ gió), **volumetric\_soil\_water** (lượng nước trong đất), **potential\_evaporation** (khả năng bốc hơi).

Qua nghiên cứu, bài báo đã tính toán hệ số tương quan giữa tần suất các đám cháy và điều kiện khí hậu trung bình. Phân tích chỉ ra mối tương quan cao đáng kể giữa hiệu độ mùa hè và PET với số lượng các sự kiện cháy ở CA.

Mô hình dự đoán trong nghiên cứu này khá đơn giản và nó chỉ sử dụng 6 biến khí hậu được tính trung bình trên toàn California để dự đoán tần suất cá đám cháy trong tiểu bang. Sử dụng tensorflow và scikit-learning để phát triển mạng nơ-ron nhân tạo dựa trên hồi quy.

Từ đánh giá mô hình của bài nghiên cứu cho ra sai số tuyệt đối trung bình tương đối hàng năm (RMAE) khoảng 10%.

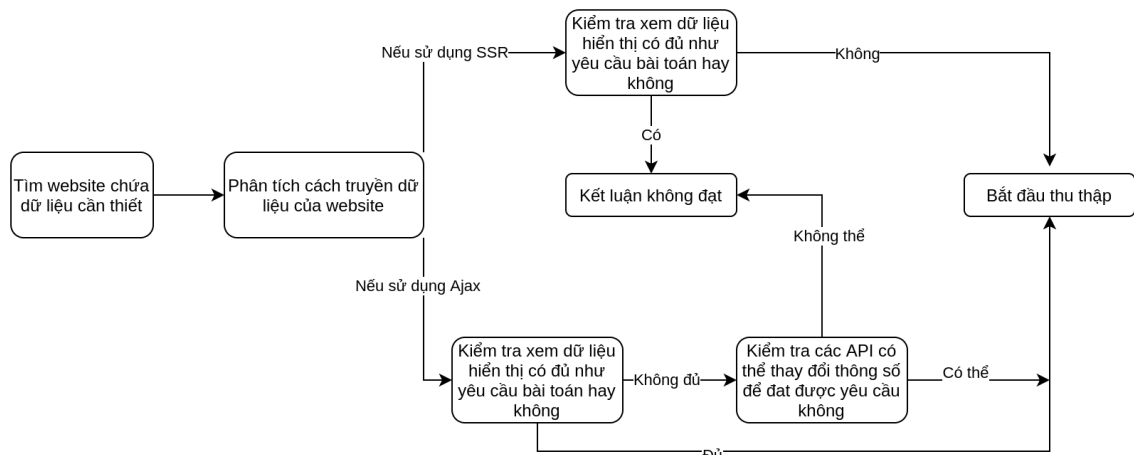
Qua nghiên cứu trên, nhóm cũng rút ra được các biến dữ liệu cần thiết và tác động lớn đến cháy rừng, các phương pháp xử lý số liệu và mô hình được đề xuất sử dụng cho loại bài toán này là ANN.

## 3 Xây dựng bộ dữ liệu

*Lưu ý ở phần này Tỉnh là thuật ngữ để chỉ chung đơn vị hành chính cấp 1 trực thuộc quốc gia, thay cho "tỉnh thành" hoặc chính xác hơn là "đơn vị hành chính cấp tỉnh" [2]. Tương tự với "Huyện" là đơn vị hành chính bậc hai, và xã là cấp ba.*

### 3.1 Quá trình thu thập dữ liệu

Tất cả dữ liệu của nhóm đều bắt đầu bằng 1 quy trình chung để kiểm tra xem nguồn dữ liệu có đáp ứng các yêu cầu hay không.



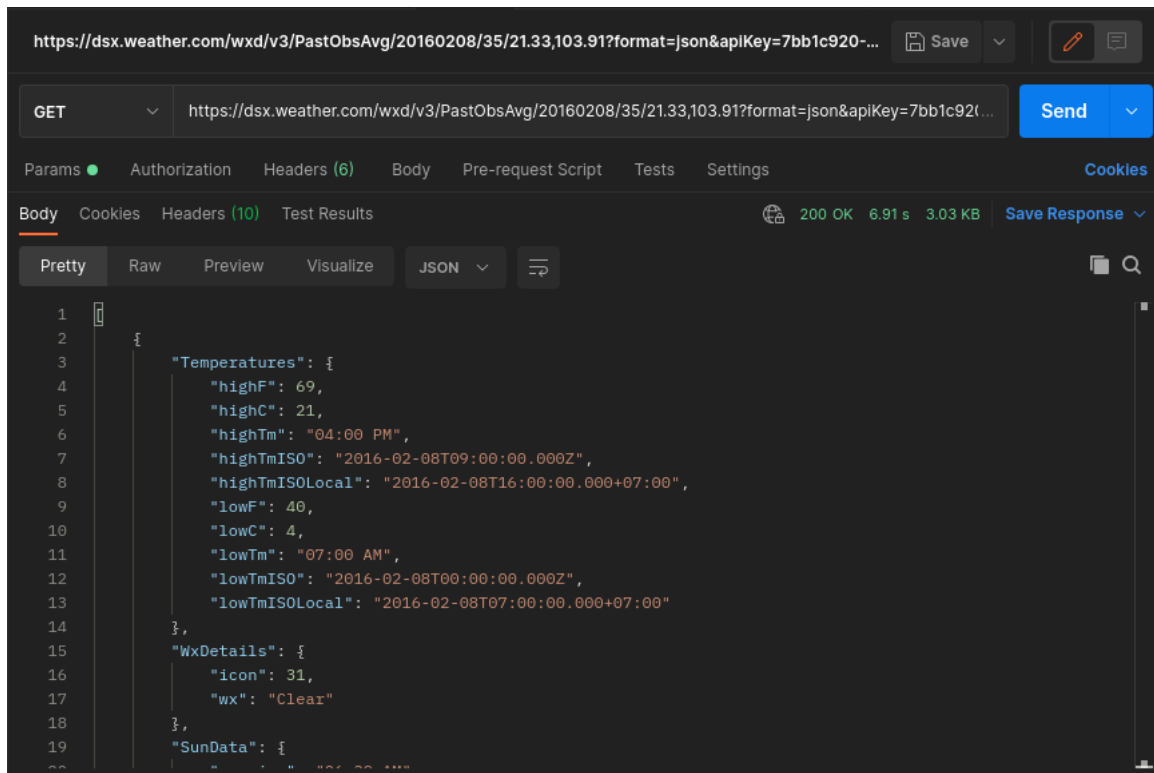
#### 3.1.1 weather.com

Để tìm ra API phục vụ cho việc crawl ta vào trang Monthly của một địa phương cụ thể sau đó navigate đến các trang chứa dữ liệu của các tháng trước (việc này giúp cho website sử dụng các API yêu cầu dữ liệu thời tiết quá khứ). Kiểm tra các response trong filter XHR ta có thể tìm ra những response tốt nhất chứa dữ liệu cần thiết.

Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	dsx.weather...	12.74,107.94?format=json&apiKey=7bb1c920-70...	eum.min.js.28 (f...	plain	4.59 KB (raced)	35.07 ...
200	GET	api.weather.c...	astro?geocode=12.74,107.94&days=30&date=20...	eum.min.js.28 (f...	json	23.23 KB	91.50 ...
200	GET	api.weather.c...	astro?geocode=12.74,107.94&days=30&date=20...	eum.min.js.28 (f...	json	23.23 KB	91.42 ...
200	GET	api.weather.c...	dateTime?geocode=12.74,107.94&format=json&i...	eum.min.js.28 (f...	json	cached	107 B
404	GET	api.weather.c...	45day?geocode=12.74,107.94&units=m&startDay...	eum.min.js.28 (f...	json	662 B	191 B
200	GET	dsx.weather...	12.74,107.94?format=json&apiKey=7bb1c920-70...	eum.min.js.28 (f...	plain	3.93 KB	29.26 ...
200	POST	amplitude.we...	/	eum.min.js.23 (...)	html	1.44 KB	7 B
200	GET	api.weather.c...	astro?geocode=12.74,107.94&days=30&date=20...	eum.min.js.29 (f...	json	cached	91.50 ...
200	GET	api.weather.c...	dateTime?geocode=12.74,107.94&format=json&i...	eum.min.js.29 (f...	json	cached	107 B
404	GET	api.weather.c...	45day?geocode=12.74,107.94&units=m&startDay...	eum.min.js.29 (f...	json	662 B	191 B
200	GET	api.weather.c...	astro?geocode=12.74,107.94&days=30&date=20...	eum.min.js.29 (f...	json	22.73 KB (raced)	91.42 ...
200	GET	dsx.weather...	12.74,107.94?format=json&apiKey=7bb1c920-70...	eum.min.js.29 (f...	plain	3.93 KB (raced)	29.26 ...
200	POST	amplitude.we...	/	eum.min.js.23 (...)	html	1.44 KB	7 B
200	GET	api.weather.c...	astro?geocode=12.74,107.94&days=30&date=20...	eum.min.js.29 (f...	json	cached	91.50 ...
200	GET	api.weather.c...	astro?geocode=12.74,107.94&days=30&date=20...	eum.min.js.29 (f...	json	cached	91.42 ...
200	GET	api.weather.c...	dateTime?geocode=12.74,107.94&format=json&i...	eum.min.js.29 (f...	json	cached	107 B
404	GET	api.weather.c...	45day?geocode=12.74,107.94&units=m&startDay...	eum.min.js.29 (f...	json	662 B	191 B
200	GET	dsx.weather...	12.74,107.94?format=json&apiKey=7bb1c920-70...	eum.min.js.29 (f...	plain	3.86 KB	29.25 ...

Hình 2: Thông tin của response chứa những thông tin cần thiết

Tiếp theo đó khi đã xác nhận API này đủ điều kiện ra kiểm tra ý nghĩa của các thông số trong request, thử thay đổi thông số để kiểm tra dữ liệu kéo dài được đến bao lâu.



Hình 3: Thử nghiệm với các tham số khác

Vì dữ liệu chấp nhận đến 2014 nên sẽ tiến hành crawl dữ liệu. Nhóm sử dụng Scrappy để crawl vì công cụ này hỗ trợ async cho phép gửi nhiều request cùng lúc cùng với đó là có pipeline để lưu dữ liệu nên giảm rất nhiều quá trình cài đặt.

Đầu tiên ta tạo tải dữ liệu danh sách chứa tất cả các xã cần crawl dữ liệu, làm tròn các tọa độ đến 2 chữ số thập phân(theo yêu cầu của API, xử lý bằng numpy trước sẽ nhanh hơn việc xử lý trong vòng lặp).

```

class IBMWeatherScapper(Spider):
    name = 'ibm-weather'

    df_ward_taynguyen_fire = pandas.read_csv(
        'https://raw.githubusercontent.com/.../taynguyen_wards_longlat.csv')
    df_ward_taynguyen_fire['long'] = numpy.round(
        df_ward_taynguyen_fire['long'], 2)
    df_ward_taynguyen_fire['lat'] = numpy.round(
        df_ward_taynguyen_fire['lat'], 2)

```

Trong hàm start\_requests của Spider với mỗi xã ta tạo 1 request vào ngày 1/1/2014 với tọa độ của xã đó, cùng với đó là lưu kèm 1 meta để nhận diện xã đó trong parse function.

```

def start_requests(self):
    requests = []
    for i, ward in self.df_ward_taynguyen_fire.iterrows():
        requests.append(request.Request(
            url=f"https://dsx.weather.com/wxd/v3/PastObsAvg/20140101/35/\
{ward['lat']:.2f},{ward['long']:.2f}\
?format=json\
&apiKey=7bb1c920-7027-4289-9c96-ae5e263980bc&\
fbclid=IwAR1IgpD8qPU6ZaHqDnZT1tM195Y4G8gfGvIYmpM3CGGIqyjwQeaAmbZZ8SE",
            meta={
                'ward_code': ward['ward_code'],
                'long': ward['long'],
                'lat': ward['lat']
            }
        ),

```

```

        callback=self.parse
    ))
    return requests

```

Vì đây là API nên dữ liệu được thể hiện dưới dạng json rất rõ ràng.

```

{
  "Temperatures": {
    "highF": 72,
    "highC": 22,
    "highTm": "01:00 PM",
    "highTmISO": "2014-01-01T06:00:00.000Z",
    "highTmISOLocal": "2014-01-01T13:00:00.000+07:00",
    "lowF": 68,
    "lowC": 20,
    "lowTm": "01:00 AM",
    "lowTmISO": "2013-12-31T18:00:00.000Z",
    "lowTmISOLocal": "2014-01-01T01:00:00.000+07:00"
  },
  "WxDetails": { "icon": 27, "wx": "Mostly Cloudy" },
  "SunData": {
    "sunrise": "06:05 AM",
    "sunset": "05:26 PM",
    "sunriseISO": "2013-12-31T23:05:00.000Z",
    "sunsetISO": "2014-01-01T10:26:00.000Z",
    "sunriseISOLocal": "2014-01-01T06:05:00.000+07:00",
    "sunsetISOLocal": "2014-01-01T17:26:00.000+07:00"
  },
  "Moon": {
    "moonriseLocal": "2014-01-01T05:34:00.000+07:00",
    "moonsetLocal": "2014-01-01T17:25:00.000+07:00",
    "moonriseISO": "2013-12-31T22:34:00.000Z",
    "moonsetISO": "2014-01-01T10:25:00.000Z"
  }
}

```

Việc còn lại là lấy ra và lưu lại dưới file csv. Sau đó lấy ngày cuối cùng trong dữ liệu trả về xem có bằng ngày hôm nay không, nếu không thì tiếp tục tải dữ liệu còn nếu có thì dừng lại.

```

def parse(self, response, **kwargs):
    json = response.json()
    ward_code = response.meta['ward_code']
    long = response.meta['long']
    lat = response.meta['lat']
    for d in json:
        yield {
            'ward': ward_code,
            'date': d.get('Temperatures', {}).get('highTmISOLocal', '')[:10],
            'highC': d.get('Temperatures', {}).get('highC', ''),
            'lowC': d.get('Temperatures', {}).get('lowC', ''),
            'sun_rise': d.get('SunData', {}).get('sunrise', ''),
            'sun_set': d.get('SunData', {}).get('sunset', ''),
            'sevenDayPrecipCm': d.get('Precips', {}).get('sevenDayPrecipCm', ''),
            'mtdPrecipCm': d.get('Precips', {}).get('mtdPrecipCm', ''),
            'precip24Cm': d.get('Precips', {}).get('precip24Cm', ''),
        }

    last = json[-1]
    recentDate = datetime.strptime(last['Temperatures']['highTmISO'][:10], '%Y-%M-%d')
    next = recentDate + timedelta(days=1)

```

```

if recentDate.date() == datetime.today():
    return
else:
    yield response.follow(
        url=f"https://dsx.weather.com/wxd/v3/PastObsAvg/ \
        {next.strftime('%Y%M%d')}/35/{lat:.2f},{long:.2f}\
        ?format=json\
        &apiKey=7bb1c920-7027-4289-9c96-ae5e263980bc\
        &fbclid=IwAR1IgpD8qPU6ZaHqDnZT1tM195Y4G8gfGvIYmpM3CGGIqyjwQeaAmbZZ8SE",
        meta={
            'ward_code': ward_code,
            'long':long, 'lat':lat
        },
        callback=self.parse
    )

```

Dữ liệu từ website này cung cấp có độ chính xác đến từng tọa độ, có nghĩa là chỉ cần cung cấp tọa độ (làm tròn đến 2 chữ số thập phân) thì server sẽ trả về thời tiết tại điểm đó tùy vào thời gian mà ta muốn. Tuy nhiên điểm yếu của dữ liệu này là chỉ cung cấp các đặc tính cơ bản nhất của thời tiết tại địa điểm đó gồm: nhiệt độ cao nhất và thấp nhất trong ngày, thời gian mặt trời mọc và lặn, lượng mưa (tích lũy trong 7 ngày, trong 1 tháng hoặc chỉ ngày hôm đó).

Sau khoảng nhiều ngày khai thác và xử lý, nhóm đã lấy được dữ liệu của 5 tỉnh Tây Nguyên vào từng xã từng ngày kéo dài từ 1/1/2014 đến 8/6/2021. Dữ liệu gồm các trường cơ bản sau (lưu ý các trường này đã được đổi tên so với khi crawl dữ liệu để thống nhất các bộ dữ liệu với nhau nhằm dễ dàng cho việc nghiên cứu):

- **ward:** Mã xã. Vì sẽ có những địa điểm trùng tên nên nhóm sử dụng mã để phân biệt các địa phương (mã này cung cấp bởi API của [firewatchvn.kiemlam.org.vn](http://firewatchvn.kiemlam.org.vn) [4])
- **date:** là ngày mà record được ghi lại.
- **max/min:** là nhiệt độ cao nhất và thấp nhất được ghi nhận trong ngày(celcius).
- **sunrise/sunset:** là thời gian mặt trời mọc và lặn.
- **7\_rain:** lượng mưa tổng tính từ ngày chủ nhật gần nhất trước đó (cm)
- **m\_rain:** lượng mưa tổng tính từ ngày 1 của tháng đó(cm)
- **24\_rain:** lượng mưa ghi nhận trong ngày(cm)

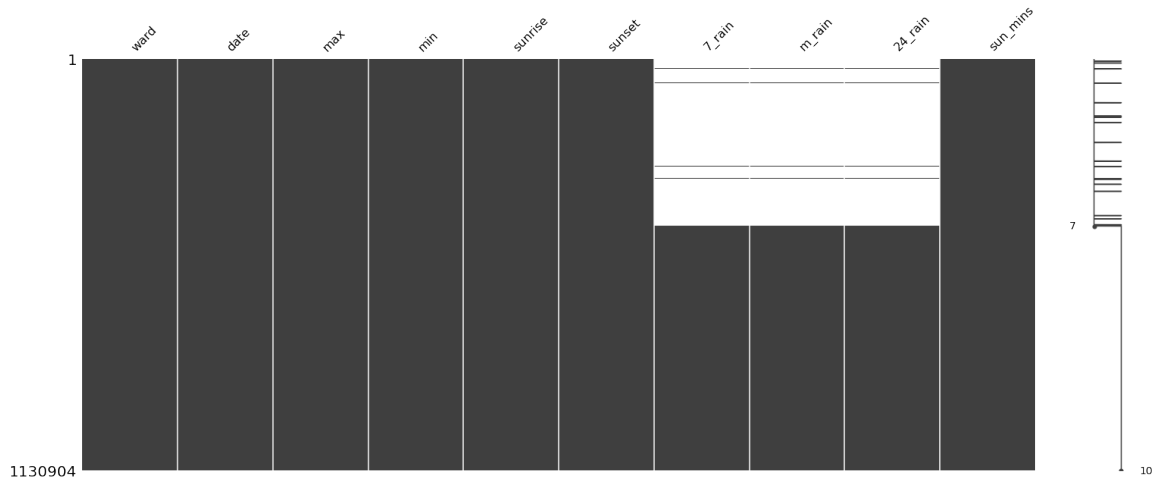
Dưới đây là một đoạn mẫu trong dữ liệu.

ward	date	max	min	sunrise	sunset	7_rain	m_rain	24_rain
24727.0	2016-10-17	30	25	05:36 AM	05:28 PM	5.94	30.91	1.82
24727.0	2016-10-18	30	25	05:36 AM	05:28 PM	7.54	32.66	1.75
24727.0	2016-10-19	30	25	05:36 AM	05:27 PM	7.54	32.96	0.3
24761.0	2019-10-14	32	24	05:36 AM	05:30 PM	0.02	0.07	0.0
24761.0	2019-10-15	32	25	05:36 AM	05:30 PM	0.02	0.07	0.0
24761.0	2019-10-16	32	25	05:36 AM	05:29 PM	0.02	0.07	0.02
24761.0	2019-10-17	32	25	05:36 AM	05:29 PM	0.02	0.07	0.0
24761.0	2019-10-18	32	25	05:36 AM	05:28 PM	0.02	0.07	0.0
24761.0	2019-10-19	32	26	05:36 AM	05:28 PM	0.02	0.07	0.0
24761.0	2019-10-20	32	26	05:36 AM	05:27 PM	0.02	0.1	0.0
24761.0	2019-10-21	33	25	05:36 AM	05:27 PM	0.02	0.1	0.0
24761.0	2019-10-22	32	25	05:37 AM	05:26 PM	0.02	0.1	0.0
24761.0	2019-10-23	33	25	05:37 AM	05:26 PM	0.0	0.1	0.0
24761.0	2019-10-24	33	25	05:37 AM	05:25 PM	0.0	0.1	0.0

Dưới đây là các thông số của bộ dữ liệu



	ward	max	min	7_rain	m_rain	24_rain
count	1130904	1130904	1130904	679514	679514	679514
mean		30.60013	24.66924	33.813765	82.088195	4.797731
std		2.836824	2.207649	71.482188	153.732438	16.574482
min		18	14	0	0	0
25%		29	23	0.5	2.2	0
50%		31	25	8.1	21.3	0
75%		32	26	34.7	88.6	2.7
max		46	36	903.7	1300.7	419.1



Hình 4: Dữ liệu sau khi sắp xếp theo ngày và địa phương (theo thứ tự ngày trước địa phương sau, tăng dần). Phần được tô là phần chứa dữ liệu, phần màu trắng là phần dữ liệu trống

### 3.1.2 worldweatheronline.com

Ở trang này dữ liệu thời tiết chỉ hiển thị ở 40 tỉnh thành, các điểm dữ liệu này được đặt title theo trung tâm hành chính của tỉnh thành phố đó.

Danh sách 40 thành phố:

Bac Lieu	Ho Chi Minh City	Tam Ky	Ben Tre
Hoa Binh	Tan An	Bien Hoa	Hong Gai
Thai Nguyen	Buon Me Thuot	Hue	Thanh Hoa
Ca Mau	Long Xuyen	Tra Vinh	Cam Pha
My Tho	Tuy Hoa	Cam Ranh	Nam Dinh
Uong Bi	Can Tho	Nha Trang	Viet Tri
Chau Doc	Phan Rang	Vinh	Da Lat
Phan Thiet	Vinh Long	Ha Noi	Play Cu
Vung Tau	Hai Duong	Qui Nhon	Yen Bai
Hai Phong	Rach Gia	Hanoi	Soc Trang

```

<li>
  ::marker
  <a href="https://www.worldweatheronline.com/buon-me-
  thuot-weather/vn.aspx" title="Buon Me Thuot holiday
  weather">Buon Me Thuot</a>
</li>

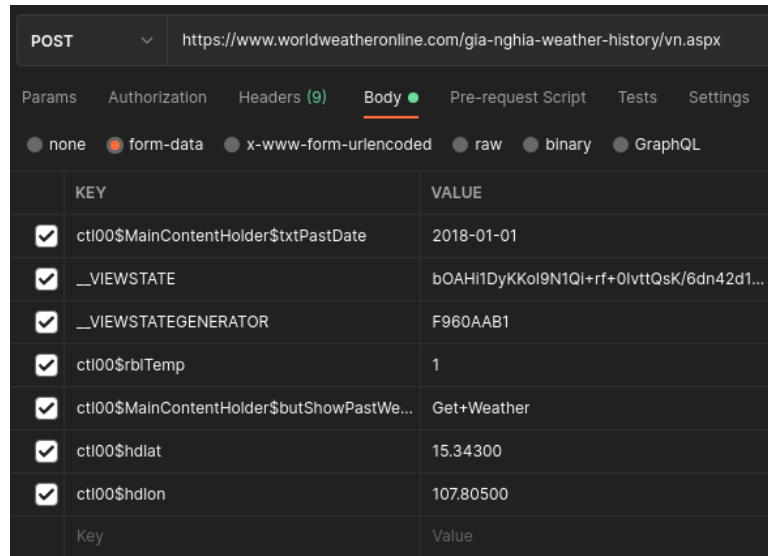
```

Hình 5: Bố cục danh sách các tỉnh của website, kèm theo đường dẫn trong thẻ a của các tỉnh đó

Vì bố cục này đơn giản và hiển thị đầy đủ danh sách trên website nên chỉ cần sử dụng Javascript để lấy danh sách các đường dẫn đến các trang lấy dữ liệu để nhanh hơn trong quá trình xử lý. Trong console của trình duyệt

```
let a = document.querySelectorAll(".country_part li >a")
let cities = Array.from(a)
cities = cities.map(e => {return {title: e.innerHTML, url:e.href}})
// Dùng stringify để tạo json string, chuỗi này chính là mảng chứa danh sách
→ các thành phố, ta chỉ cần khai báo như một biến list của dictionary trong
→ python.
JSON.stringify(cities)
```

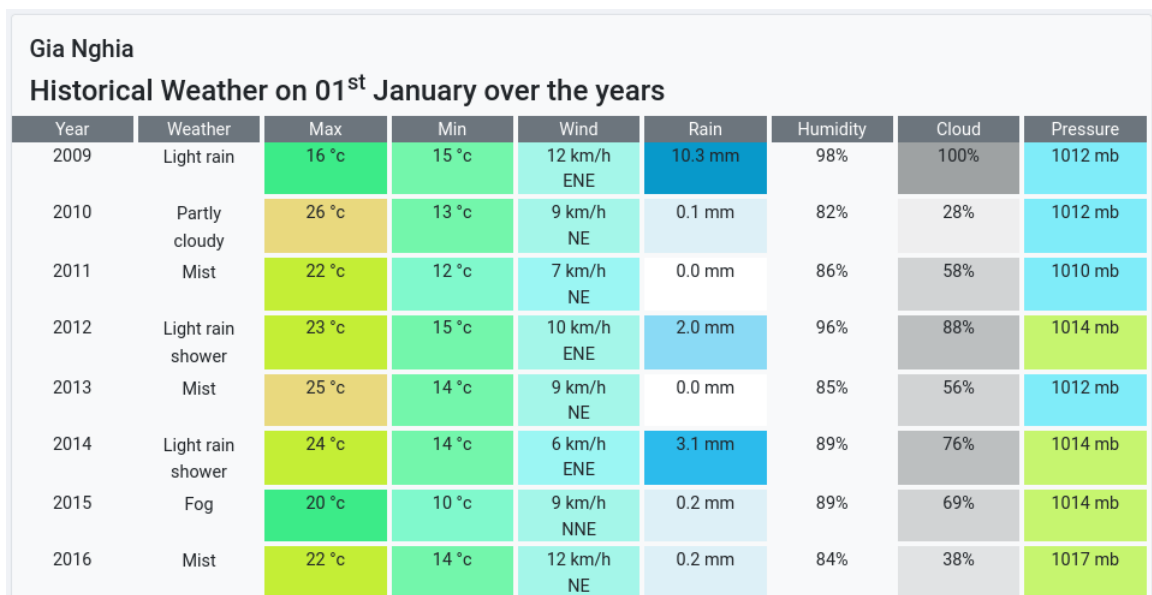
Dùng phương pháp phân tích tương tự khi phân tích weather.com để kiểm tra thông tin gửi về cho server ta sẽ biết được những thông tin để nhận về thời tiết vào một ngày cụ thể.



KEY	VALUE
ctl00\$MainContentHolder\$txtPastDate	2018-01-01
__VIEWSTATE	bOAHi1DyKKol9N1Qi+rf+0lvttQsK/6dn42d1...
__VIEWSTATEGENERATOR	F960AAB1
ctl00\$rbtTemp	1
ctl00\$MainContentHolder\$butShowPastWe...	Get+Weather
ctl00\$hdlat	15.34300
ctl00\$hdlon	107.80500
Key	Value

Hình 6: Những thông số server cần để gửi trả về dữ liệu cần thiết

Vì server của website này render hầu như đầy đủ thông tin nên ta chỉ cần lấy thông tin từ mẫu response dưới đây là đạt yêu cầu. Ở đây có thể thấy rằng có một bảng dữ liệu thể hiện sự khác biệt qua từng năm của dữ liệu. Nên ta chỉ cần gửi 365 request là có thể lấy tất cả dữ liệu cần cho mỗi tỉnh.



Year	Weather	Max	Min	Wind	Rain	Humidity	Cloud	Pressure
2009	Light rain	16 °c	15 °c	12 km/h ENE	10.3 mm	98%	100%	1012 mb
2010	Partly cloudy	26 °c	13 °c	9 km/h NE	0.1 mm	82%	28%	1012 mb
2011	Mist	22 °c	12 °c	7 km/h NE	0.0 mm	86%	58%	1010 mb
2012	Light rain shower	23 °c	15 °c	10 km/h ENE	2.0 mm	96%	88%	1014 mb
2013	Mist	25 °c	14 °c	9 km/h NE	0.0 mm	85%	56%	1012 mb
2014	Light rain shower	24 °c	14 °c	6 km/h ENE	3.1 mm	89%	76%	1014 mb
2015	Fog	20 °c	10 °c	9 km/h NNE	0.2 mm	89%	69%	1014 mb
2016	Mist	22 °c	14 °c	12 km/h NE	0.2 mm	84%	38%	1017 mb

Hình 7: Những dữ liệu response hiển thị

Từ những dữ liệu này ta bắt tay vào viết spider.

```

# Sử dụng numpy để tạo một mảng chứa tất cả các ngày trong năm
BASE = date(2018, 1, 1)
DATES = numpy.array(
    [(BASE + timedelta(days=i)).strftime("%Y-%m-%d") for i in range(365)]
)

# Các thông số cơ bản của Spider
class WorldWeatherOnlineScraper(Spider):
    name = 'world-weather-online'

    provinces = [
        {
            "url":
                ↪ "https://www.worldweatheronline.com/bac-lieu-weather-history/vn.aspx",
            "title": "Bac Lieu"
        },
        ...
        {
            "url":
                ↪ "https://www.worldweatheronline.com/ho-chi-minh-city-weather-history/vn.aspx",
            "title": "Ho Chi Minh City"
        }
    ]

    def start_requests(self):
        # Ở mỗi tỉnh thành ta gửi 365 request cho 365 ngày trong năm
        for province in self.provinces:
            for date in DATES:
                formData = {
                    # Các thông số khác như đơn vị nhiệt độ hay tọa độ gửi request
                    ↪ đều giống nhau nên sẽ không ghi chú ở đây
                    'ctl00$MainContentHolder$txtPastDate': date,
                }

                yield FormRequest(province['url'],
                                formdata=formData,
                                meta={
                                    'date': date,
                                    'province': province['title']
                                }, callback=self.parse
                )

    def parse(self, response):
        # Tách bảng history
        history = response.css('.row.text-center.woo-tabular')

        # Với mỗi cell trong bảng ta lấy tất cả ra và gom thành một mảng lớn chung
        # 9 element đầu tiên là tên của 9 cột nên ta loại ra
        records = history.css('.col.mr-1::text').getall()[9:]

        # Việc cuối cùng là lặp qua tất cả các dòng để ghi lại dữ liệu
        for i in range(0, len(records), 9):
            yield{
                'province': response.meta['province'],
                'day': response.meta['date'][8:10],
                'month': response.meta['date'][5:7],
                'year': records[i],
                'max': records[i+1],
                # so on ...
            }

```

```
    'pressure': records[i+8],
}
```

Sau khoảng 1 tiếng chạy và điều chỉnh lại ta có bảng dữ liệu sau:

- Thời gian: 1/1/2009 to 18/06/2021
- Địa lý: 40 thành phố, danh sách đã ghi lại ở trên
- Các trường dữ liệu:
  - **province**: Tên thành phố (*không dấu*)
  - **date**: ngày
  - **max**: nhiệt độ cao nhất trong ngày (*celcius*)
  - **min**: nhiệt độ thấp nhất trong ngày (*celcius*)
  - **wind**: tốc độ gió (*km/h*)
  - **wind\_d**: hướng gió direction
  - **rain**: lượng mưa (*mm*)
  - **humidi**: độ ẩm (%)
  - **cloud**: mây (%)
  - **pressure**: áp suất(Bar)

Dưới đây là một đoạn trong dữ liệu:

province	max	min	wind	wind_d	rain	humidi	cloud	pressure	date
Bac Lieu	27	22	17	NNE	6.9	90	71	1010	2009-01-01
Bac Lieu	31	25	20	ENE	0.0	64	24	1010	2010-01-01
Bac Lieu	29	24	14	E	0.0	75	45	1008	2011-01-01
Bac Lieu	30	24	30	E	0.0	79	52	1012	2012-01-01
Bac Lieu	31	25	20	ENE	0.0	70	24	1010	2013-01-01

Các thông số cơ bản của dữ liệu (lưu ý rằng gió đã được đổi ra các góc theo radian)

	max	min	wind	wind_d	rain	humidi	cloud	pressure
count	181960	181960	181960	181960	181960	181960	181960	181960
mean	29.837277	23.277874	11.038657	2.67885	6.56713	77.083068	41.721268	1010.229127
std	4.571345	3.945381	5.311807	1.241257	13.602055	9.288553	23.875067	4.635714
min	4	2	1	0	0	23	0	988
25%	28	21	7	1.570796	0.1	71	23	1008
50%	31	24	10	2.356194	1.8	78	38	1010
75%	33	26	14	3.926991	7.5	83	58	1012
max	46	32	54	5.890486	596.4	100	100	1038

### 3.1.3 firewatchvn.kiemlam.org.vn

Tương tự với 2 trang ở trên, sau khi phân tích dữ liệu nhận về từ các response, ta biết được các cổng API sau:

Domain: [firewatchvn.kiemlam.org.vn](http://firewatchvn.kiemlam.org.vn)

#### 1. Thông tin cấu trúc hành chính

- Route: `/fwdata/hanhchinh/province-code/district-code`
- Parameters:
  - **province-code**: mã tỉnh, nếu để là 0 thì sẽ trả về danh sách tất cả các tỉnh trên cả nước
  - **district-code**: mã huyện, nếu để là 0 thì sẽ trả về danh sách tất cả các huyện của tỉnh hiện tại, nếu **province-code** là 0 thì tham số này không có ý nghĩa
- Response:

```
// Response from "/fwdata/hanhchinh/0/0"
[
  {"ma": "89", "ten": "An Giang"},
  {"ma": "24", "ten": "Bắc Giang"},
  ...,
  {"ma": "15", "ten": "Yên Bái"}
]
```

## 2. Thông tin điểm cháy

(a) Route: `/fwdata/search/diaphuong/{province-code}/{district-code}/{ward-code}/{start-date}/{end-date}/1/100`

(b) Parameters:

- **province-code**: mã tỉnh, nếu để là 0 thì sẽ trả về thống kê bằng tổng số các vụ cháy trên cả nước ở các tỉnh
- **district-code**: mã huyện, nếu để là 0 thì sẽ trả về thống kê bằng tổng số các vụ cháy trên cả nước ở các huyện, nếu **province-code** là 0 thì tham số này không có ý nghĩa
- **ward-code**: mã xã, nếu để là 0 thì sẽ trả về thống kê bằng tổng số các vụ cháy trên cả nước ở các xã, nếu **province-code** hoặc **district-code** là 0 thì tham số này không có ý nghĩa
- **start-date**: ngày bắt đầu, format dưới dạng dd!mm!yyyy
- **end-date**: ngày kết thúc, format dưới dạng dd!mm!yyyy

(c) Response:

```
// Response from
↪ "/fwdata/search/diaphuong/49/512/0/01!10!2019/01!10!2019/1/100"
[
  {
    "sdc": 3,
    "x": 108.023503269442,
    "y": 15.5811098358411,
    "xa": "Hiệp Hòa",
    "hp": [
      {
        "x": 108.02951,
        "y": 15.56155,
        "idhotspot": "20758-_263b067"
      },
      {
        "x": 108.03237,
        "y": 15.553405,
        "idhotspot": "20758-!ab49a26"
      },
      {
        "x": 108.03386,
        "y": 15.58714,
        "idhotspot": "20758-_c73ffe4"
      }
    ]
  },
  ...,
  {
    "sdc": 1,
    "x": 108.014536842611,
    "y": 15.537556163829,
    "xa": "Sông Trà",
    "hp": [
      {
        "x": 108.05701,
```

```

        "y": 15.53117,
        "idhotspot": "20770-_594ec71"
    }
}
]
}
]

```

Với API này ta có 2 hướng giải quyết:

1. Tạo request cho tất cả các huyện vào mỗi ngày, việc này sẽ giúp quét hết toàn bộ điểm cháy chi tiết đến từng tọa độ theo ngày. Trong bộ dữ liệu có mã của 710 đơn vị hành chính cấp huyện nếu vậy để quét qua từ 1/1/2008 đến 31/12/2020 (khoảng 4749 ngày) ta cần gửi khoảng 3371790 request
2. Tạo request thống kê trên ngày cho cả nước, gom tất cả các huyện có cháy trong ngày đó, sau đó tra theo mã huyện đã nhận. Phương pháp này có thể giảm nhưng cũng có thể làm tăng số lượng request tùy các các điểm cháy phân bố.

Sau khi so sánh 2 phương pháp bằng cách request 30 ngày thì phương pháp thứ 2 cho lượng request chỉ bằng 70% so với phương pháp thứ nhất. Vì vậy nhóm quyết định chọn phương pháp thứ 2.

Dưới đây là kết quả tổng kết từ crawler (đã lược bỏ, chi tiết xem tại đây)

```

{
  'downloader/request_count': 3127864,
  'downloader/response_count': 3127864,
  'downloader/response_status_count/200': 3127863,
  'downloader/response_status_count/404': 1,
  'elapsed_time_seconds': 68738.104826,
  'finish_reason': 'finished',
  'item_scraped_count': 997218,
}

```

Vậy ta đã giảm được 7.23% lượng request (243926, khoảng 1.489 tiếng thời gian thực thi)

Đặc điểm chung của bộ dữ liệu này là không thiếu sót ở các cell vì nếu đã được ghi lại ở server thì chắc chắn dữ liệu này đầy đủ thông tin.

- Thời gian: 1/1/2008 đến 31/12/2020
- Địa lý: Cả nước
- Các trường dữ liệu:
  - date: Ngày ghi nhận dưới dạng yyyy-mm-dd
  - province: Tỉnh ghi nhận
  - district: Huyện ghi nhận
  - ward: Xã ghi nhận
  - long: Kinh độ
  - lat: Vĩ độ

Một đoạn mẫu trong dữ liệu

date	province	district	ward	long	lat
2020-01-06	Bắc Giang	Yên Dũng	Tiền Phong	106.172	21.238
2020-05-26	Lào Cai	Văn Bàn	Nậm Tha	104.41616	21.99459
2020-05-26	Sóc Trăng	Mỹ Tú	Hưng Phú	105.70397	9.64698
2020-05-26	Sơn La	Văn Hồ	Tân Xuân	104.70561	20.6593
2020-05-26	Bạc Liêu	Bạc Liêu	Vĩnh Trạch	105.76225	9.30437
2020-05-26	Bình Dương	Bàu Bàng	Lai Uyên	106.63067	11.23651
2020-05-26	Hà Tĩnh	TX. Kỳ Anh	P. Kỳ Long	106.41985	18.04084

Lưu ý rằng ở một xã nhất định vào một ngày có thể có nhiều điểm cháy, do thuật toán nhận diện dựa trên ảnh từ vệ tinh nên nếu cháy trên diện rộng sẽ nhận diện được nhiều điểm

Đắk Lắk



Buôn Đôn

Tất cả

Từ ngày: 23/02/2018

Đến ngày: 23/02/2018

Tìm kiếm

#	Huyện	Số điểm cháy	
1	Krông Na	848	 

Hình 8: Ví dụ về ngày phát hiện nhiều điểm cháy nhất ở một xã

### 3.2 Xây dựng bộ dữ liệu

Do việc thiếu dữ liệu nên nhóm tạo 2 bộ dữ liệu có những đặc trưng tối ưu riêng (về không gian và thời gian) để có thể tận dụng hết được các trường dữ liệu có ảnh hưởng lớn đến bài toán và dễ dàng hơn trong việc imputation.

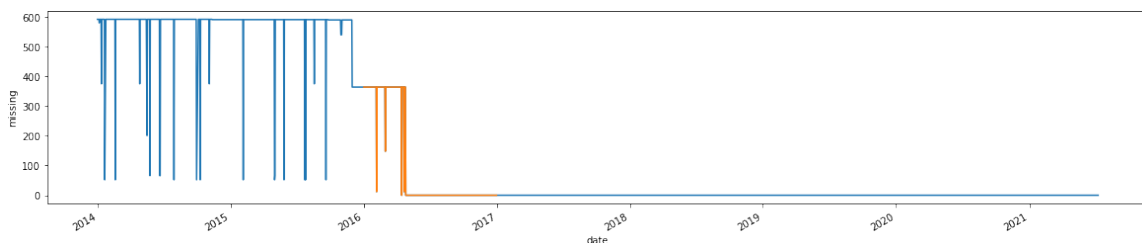
Như đã nói ở trên dữ liệu lượng mưa vốn có ảnh hưởng lớn đến thông tin cháy bị thiếu rất nhiều ở thời gian đầu của bộ dữ liệu. Nhóm đã tham khảo 2 phương pháp:

- Sử dụng phương pháp FFT(Fast Fourier transform) để tìm ra các chu kỳ lặp lại của dữ liệu để dự đoán ngược dữ liệu.
- Sử dụng dữ liệu từ các thành phố trung tâm từ bộ dữ liệu từ World Weather Online để lấp bù vào các xã, bị thiếu.

Vì lượng khí hậu thay đổi theo từng năm do biến đổi khí hậu trong thời gian gần đây, nên mặc dù các chu kỳ lặp lại qua từng năm thì sự phản hồi qua phép FFT cũng bị nhiễu loạn rất nhiều. Thêm vào đó biến đổi khí hậu còn là một trong những nguyên nhân gây nên cháy rừng, nên một yếu tố bị ảnh hưởng bởi biến đổi khí hậu như lượng mưa thì việc sử dụng FFT để dự đoán theo chu kỳ là không hợp lý. Vì vậy nhóm quyết định sử dụng dữ liệu từ thành phố trung tâm (hành chính) của các tỉnh để lấp vào phần lượng mưa còn trống. Tuy nhiên bộ dữ liệu thời tiết từ WWO chỉ có 3 tỉnh ở Tây Nguyên nên nhóm chia thành 2 bộ dữ liệu như sau:

- Gồm 5 Tỉnh Tây Nguyên từ 1/1/2017 đến 31/12/2020
- Gồm 3 Tỉnh Gia Lai, Lâm Đồng, Đắk Lắk từ 01/01/2014 đến 31/12/2020

Bộ dữ liệu đầu tiên gồm tất cả tỉnh thành nhưng chỉ lấy từ năm 2017 vì điểm đứt gãy dữ liệu (điểm mà dữ liệu bắt đầu thiếu là trước thời gian này)



Hình 9: Biểu đồ thể hiện dữ liệu bị thiếu, trục tung là số lượng xã bị mất dữ liệu lượng mưa

#### 3.2.1 Imputation of Data

Bộ dữ liệu số 1 đầy đủ các thông tin cần thiết trong thời gian yêu cầu, nên việc imputation là không cần thiết.

Như đã nói thì phương pháp imputation ở đây là sử dụng dữ liệu lượng mưa ở thành phố trung tâm cho tất cả các xã bị mất dữ liệu. Ví dụ tất cả các xã của tỉnh Lâm Đồng bị mất dữ liệu lượng

mưa/nhiệt độ trong ngày 20/8/2020 sẽ được lấp bằng dữ liệu lượng mưa/nhiệt độ của Đà Lạt ngày 20/8/2020.

Để thực hiện Imputation bộ dữ liệu thứ 2, ta map toàn bộ các tên các thành phố trong bộ dữ liệu WWO thành tên Tỉnh mà thành phố đó trực thuộc

```
df_wwo_weather_tn = df_wwo_weather[df_wwo_weather['province'].isin(['Buon Me
↪ Thuot', 'Play Cu', 'Da Lat'])].copy()
df_wwo_weather_tn['province'] = df_wwo_weather_tn['province'].replace({
    'Buon Me Thuot': 'Đắk Lắk',
    'Play Cu': 'Gia Lai',
    'Da Lat': 'Lâm Đồng',
})
```

Sau đó ta chuyển tên tỉnh đó thành mã tỉnh. Tạo một bảng chứa toàn bộ mã xã cùng với thời tiết của 3 tỉnh này từ bảng WWO. Merge bảng này với bảng dữ liệu từ weather.com bằng `left join` sau đó imputation từ cột của bảng WWO sang cột của bảng IBM weather ta sẽ được một bảng dữ liệu Imputation

```
# Create replacement df contain data imputation from wwo data to each ward of that
↪ province, drop province_code because don't need more
df_replacement = pd.merge(df_wwo_weather_tn,
↪ df_administrative_tn[['province_code', 'ward_code']], on=['province_code'],
↪ how='outer').dropna().drop(['province_code'], axis=1)

# Merge with ibm weather the for show missing data, drop ward because ward columns
↪ of IBM data will be NaN if ward do not have data in that date
df = pd.merge(df_ibm_weather, df_replacement, how='right',
↪ left_on=['ward', 'date'], right_on=['ward_code', 'date']).drop(['ward'], axis=1)

df['max_x'] = df['max_x'].fillna(df['max_y'])
df['min_x'] = df['min_x'].fillna(df['min_y'])
df['24_rain'] = df['24_rain'].fillna(df['rain'])
```

ward_code	date	fires	max	24_rain	...	wind	wind_d	cloud	pressure	doy
23557	2009-01-01	0	22	8.7	...	22	1.178097	84	1014	1
23557	2009-01-02	0	22	1.6	...	26	1.178097	66	1015	2
23557	2009-01-03	0	23	1.4	...	20	0.785398	63	1016	3
23557	2009-01-04	0	22	0	...	18	0.785398	31	1014	4
23557	2009-01-05	0	26	0	...	15	0.785398	29	1014	5

Danh sách các trường dữ liệu

- ward\_code: mã xã
- date: ngày ghi nhận
- fires: số điểm cháy
- max: nhiệt độ cao nhất
- min: nhiệt độ thấp nhất
- 24\_rain: lượng mưa
- sun\_mins: số phút nắng trong ngày
- wind: vận tốc gió
- wind\_d: hướng gió (radian)
- humidi: độ ẩm
- cloud: lượng mây
- pressure: áp suất
- doy: ngày trong năm



### 3.2.2 Creation of Dataset

Việc tạo dữ liệu cho timeseries đã có hàm dựng sẵn từ tensorflow nên cách tạo dataset rất đơn giản, ta shuffle mã xã sau đó chia ra tùy vào mã xã. Lấy từng khoảng dữ liệu theo mã xã đã sắp xếp theo thời gian, đưa vào hàm `timeseries_dataset_from_array`, sử dụng hàm `concatenate` ta sẽ được các generator sử dụng cho việc training.

Tùy vào mục đích cho từng model ta sẽ cần mở rộng chiều của dữ liệu để phù hợp, ví dụ Conv2D sẽ yêu cầu dữ liệu 4 chiều. Việc sử dụng Generator sẽ kéo dài việc training song sẽ giúp quá trình tạo dataset vốn rất lớn gần như tức thời.

**Dữ liệu đào tạo** Sau khi hoàn thành bước Imputation of Data ta được một bộ dữ liệu thời tiết hoàn chỉnh từ 1/1/2014 đến 31/12/2020 với các trường dữ liệu như trên.

Để việc đào tạo hiệu quả chúng em đã tham khảo một số phương pháp xử lý số liệu thời tiết ở time series của tensorflow [7] trong đó họ có đề cập việc thay đổi trường vận tốc gió và hướng gió tạo thành vector gió theo chiều x và chiều y sẽ mang lại kết quả tốt hơn.

```
df_weather['wind_x'] = df_weather['wind']*np.cos(df_weather['wind_d'])
df_weather['wind_y'] = df_weather['wind']*np.sin(df_weather['wind_d'])
```

Sau khi đã đủ các trường để đào tạo model, nhóm tiếp tục nghiên cứu và thảo luận rồi đưa ra quyết định về việc xây dựng dữ liệu thời tiết trước 30 ngày của ngày đưa ra dự đoán để đảm bảo quá trình hình thành cháy rừng. Từ đó tiến hành xây dựng data windowing.

**Xác định nhãn** Với dữ liệu wildfire thì ta xác định được thời gian xảy ra cháy rừng cũng như địa điểm cháy. Từ đó ta xác định được dữ liệu chứa nhãn 1 đối với model dự đoán cháy rừng. Ngoài ra thì khi ta xác định được dữ liệu này rồi thì ta tiếp tục chia dữ liệu đào tạo này dựa trên số lượng điểm cháy để phân ra thành các nhãn cấp độ cháy đối với model dự đoán cấp độ cháy rừng.

Để xác định nhãn 0 của model dự đoán cháy rừng, nhóm đã xem xét trên toàn bộ dữ liệu cháy rừng đã thu thập được và từ các nghiên cứu của các chuyên gia trước đó đã xác định được cháy rừng thì xảy ra theo mùa, vì thế để lấy được dữ liệu chứa nhãn 0, nhóm xác định các tháng ít xảy ra cháy rừng trên địa bàn Tây Nguyên từ đó xác định ngày và địa điểm để tạo dữ liệu.

## 4 Training và đánh giá modelods

### 4.1 Thử nghiệm trên dữ liệu 5 tỉnh Tây Nguyên

**Mô tả** Từ việc phân tích dữ liệu ở trên, ta thấy dữ liệu thời tiết bị thiếu dữ liệu lượng mưa trước năm 2017. Vì vậy, nhóm tiến hành lấy dữ liệu từ 2017 đến 2020 để thử nghiệm một vài model trước khi thực hiện trên toàn bộ tập dữ liệu. Phần thử nghiệm này thực hiện trên dữ liệu từ 2017 đến 2020, gồm 553 xã thuộc 5 tỉnh Tây Nguyên. Thử nghiệm trên 2 loại model. Loại 1 là Dự đoán cháy rừng với output là 0 và 1, tương ứng với không cháy và cháy. Loại 2 là Dự đoán mức độ cháy rừng với output đầu ra là con số tương ứng với mức độ cháy.

#### 4.1.1 Model dự đoán cháy rừng

**Xử lý và chia dữ liệu** Từ bộ dữ liệu ta tiến hành xử lý và chia dữ liệu:

Từ bộ dữ liệu thời tiết IBM đã được xử lý ở phần trước ta tiến hành lấy những features cần dùng và drop những features không cần thiết.

Những features cần dùng cho training:

- max: nhiệt độ cao nhất (độ C)
- rain: lượng mưa trong 24h (mm)
- wind: vận tốc gió (km/h)
- humidi: độ ẩm (%)
- doy: ngày trong năm (1-365)
- ward code: mã xã

```
# drop unnecessary features
df_ibm_weather_5 =
↳ df_ibm_weather_5.drop(['min_x', 'sun_mins', 'wind_d', 'cloud', 'pressure', 'sunset', 'sunrise', '7_ra
↳ axis=1)
df_ibm_weather_5 =
↳ df_ibm_weather_5.rename(columns={'max_x': 'max', '24_rain': 'rain'})

# get data after 2017 and before 2021
df_ibm_weather_5 = df_ibm_weather_5.loc[df_ibm_weather_5['date'].dt.year >= 2017]
df_ibm_weather_5 = df_ibm_weather_5.loc[df_ibm_weather_5['date'].dt.year <= 2020]
```

Ta tiến hành đếm tổng các điểm cháy trong ngày tại mỗi xã và merge với dữ liệu thời tiết tương ứng theo ward\_code và date.

```
# count fires
df_fire_count = df_wildfire_tn.groupby(['ward_code', 'date']).size().reset_index(name='fires')

# merge fire with weather
df_dataset_5 = pd.merge(df_fire_count, df_ibm_weather_5, how='right', on=['ward_code', 'date'])
```

Scale dữ liệu thời tiết bằng MinMaxScale. Việc này giúp model hội tụ tốt hơn.

```
def MinMaxScale(df, columns):
    for c in columns:
        df[c] = (df[c] - df[c].min()) / (df[c].max() - df[c].min())
    return df
```

Sau khi xử lý bộ dữ liệu xong ta tiến hành windowing. Ở đây em dùng vòng lặp để thực hiện. Sau khi hoàn tất xử lý và tạo bộ dữ liệu. Ta chia bộ dữ liệu thành 3 phần train, validation và test tương ứng với tỉ lệ 70%, 20%, 10%.

## Thiết lập, Training

**Dense** Khởi tạo Sequential() của Keras. Ở đây em dùng gồm 3 lớp Dense với hàm kích hoạt là relu, lớp cuối có 1 node với hàm sigmoid để phân loại binary ở output.

```
denseModel = tf.keras.Sequential()
denseModel.add(Input(shape=(X_train_binary.shape[1],)))
denseModel.add(Dense(128, activation='relu'))
denseModel.add(tf.keras.layers.Dropout(0.2))
denseModel.add(Dense(64, activation='relu'))
denseModel.add(Dense(32, activation='relu'))
denseModel.add(Dense(1, activation='sigmoid'))
```

Compile denseModel với optimizer là adam, loss là binary\_crossentropy và accuracy. Để tránh mất nhiều thời gian, ta thiết lập early stopping để kết thúc sớm việc training trong trường hợp val loss không giảm.

```
denseModel.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, mode='min')
```

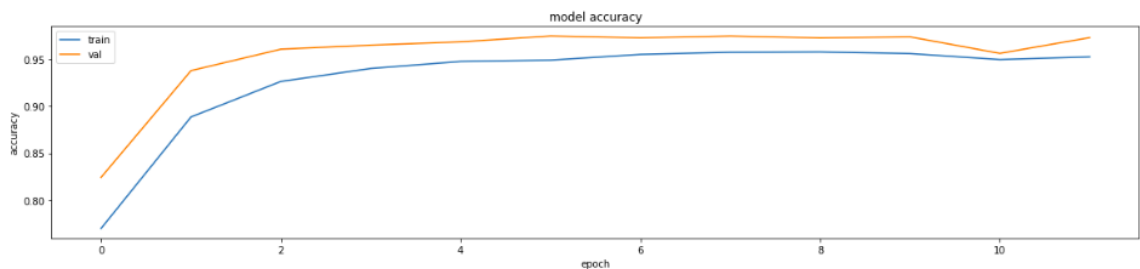
Sau khi thiết lập xong các thông số, ta tiến hành training.

```
history = denseModel.fit(X_train_binary, y_train_binary, batch_size=256,
↳ epochs=20, callbacks=[early_stopping], validation_data=(X_val_binary,
↳ y_val_binary), verbose=1)
```

Các số liệu về training:

- Epochs: 20
- Batch size: 256

- Thời gian train: 3 phút 20 giây
- Thời gian test: 12 giây
- Accuracy:
  - validation: 0.9730
  - test: 0.9742
- Loss:
  - validation: 0.0912
  - test: 0.0890



Hình 10: Biểu đồ quá trình training

#### 4.1.2 Model dự đoán cấp độ cháy rừng

**Xử lý và chia dữ liệu** Việc xử lý dữ liệu thực hiện tương tự như phần trước. Nhưng đặc biệt ở đây là bộ dataset được tạo bằng bộ tạo time series dataset của tensorflow trong hàm bên dưới. Hàm `timeseries_dataset_from_array` của tensorflow sẽ thực hiện phân chia features và label, lấy dữ liệu 30 ngày trước, chia `batch_size` và shuffle dữ liệu.

```
def concatDataset(data, ward_code_column ,wards,columns):
    for i, w in enumerate(wards):
        df = data[data[ward_code_column] == w][columns].to_numpy()
        inputs, targets = df[:, :-1], df[:, -1:]

        ds = tf.keras.preprocessing.timeseries_dataset_from_array(data=inputs,
            ↪ targets=targets, sequence_length=30, sequence_stride=1, sampling_rate=1,
            ↪ batch_size=256, shuffle=True, seed=20)
        ds = ds.map(expand)
        if i == 0:
            dataset = ds
        else:
            dataset = dataset.concatenate(ds)
    return dataset
```

**Thiết lập, Training** Sau khi tạo xong bộ dữ liệu ta tiến hành áp dụng các model cho việc training.

**CNN** Ta khởi tạo `Sequential()` của Keras. Thiết lập layer `Conv1D` với kernel size là 6 và hàm kích hoạt là `relu`. Tiếp đến là 2 lớp `Dense` với 1 node ở output đầu ra. Sử dụng hàm `relu` để lấy các giá trị từ 0, 1, 2,... Đó là các mức độ cháy rừng.

```
cnnModel = tf.keras.Sequential()
cnnModel.add(tf.keras.layers.Conv1D(32,kernel_size=(6),activation='relu'))
cnnModel.add(Dense(32,activation='relu'))
cnnModel.add(Dense(units=1,activation='relu'))
```

Sử dụng optimizer là Adam, loss là Mean Squared Error. Tương tự, ta cũng thiết lập early stopping để dừng training khi cần thiết.

```

cnnModel.compile(optimizer='adam',loss=tf.keras.losses.MeanSquaredError(),metrics=['accuracy'])
early_stopping =
↳ tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=2,mode='min')
history =
↳ cnnModel.fit(train_dataset_5,validation_data=val_dataset_5,epochs=20,callbacks=[early_stopping])

```

Các số liệu về training:

- Epochs: 20
- Batch size: 256
- Thời gian train: 1 giờ 2 phút
- Thời gian test: 11 phút
- Accuracy:
  - validation: 0.9740
  - test: 0.9942
- Loss:
  - validation: 9.5859
  - test: 1.321

**RNN** Recurrent Neural Network là một mạng neural được đánh giá là phù hợp nhất cho time series data. Tương tự, ta cũng khởi tạo Sequential() của Keras. Sau đó add 2 lớp SimpleRNN, thiết lập return sequential = True cho phép model return một output cho mỗi input đầu vào. Cuối cùng là một lớp Dense 1 node với hàm kích hoạt là relu để nhận kết quả trả về 0, 1, 2,... Tương ứng với mức độ cháy. Các thông số compile ta thiết lập tương tự như mạng CNN trên.

```

rnnModel = tf.keras.Sequential()
rnnModel.add(tf.keras.layers.SimpleRNN(32,return_sequences=True))
rnnModel.add(tf.keras.layers.SimpleRNN(32))
rnnModel.add(Dense(1,activation='relu'))
rnnModel.compile(optimizer='adam', loss=tf.keras.losses.MeanSquaredError(), metrics=['accuracy'])
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=2,mode='min')

```

Các số liệu về training:

- Epochs: 20
- Batch size: 256
- Thời gian train: 33 phút
- Thời gian test: 6 phút
- Accuracy:
  - validation: 0.9740
  - test: 0.9942
- Loss:
  - validation: 9.5859
  - test: 1.321

**LSTM** Long Short-Term Memory là một dạng của mạng RNN, độ hiệu quả cao hơn CNN và phù hợp cho các dữ liệu thời gian dài. Tương tự, ta cũng khởi tạo `Sequential()` của Keras. Sau đó add lớp LSTM và thiết lập `return_sequences = True`. Lớp Dense output và thiết lập compile tương tự như mạng RNN phía trên.

```
lstmModel = tf.keras.Sequential()
lstmModel.add(LSTM(32,return_sequences=True))
lstmModel.add(Dense(units=1,activation='relu'))
lstmModel.compile(loss=tf.keras.losses.MeanSquaredError(),optimizer='adam',metrics=['accuracy'])
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=2,mode='min')
```

Các số liệu về training:

- Epochs: 20
- Batch size: 256
- Thời gian train: 39 phút
- Thời gian test: 6 phút
- Accuracy:
  - validation: 0.9740
  - test: 0.9942
- Loss:
  - validation: 9.5859
  - test: 1.3218

**Đánh giá** Từ các model thử nghiệm trên cho ta độ chính xác khá cao trên 98%. Từ đó, ta cũng có thể đánh giá được khả năng dự đoán cháy của model trên bộ dữ liệu nhóm đã thu thập. Tuy nhiên, với độ chính xác quá cao cho thấy dấu hiệu overfitting. Vì vậy, trong phần tiếp theo, nhóm sẽ thực hiện trên toàn bộ tập dữ liệu. Dữ liệu sẽ đầy đủ và dài hơn sau khi đã imputation những dữ liệu còn thiếu, bên cạnh đó thực hiện với nhiều thuật toán khác nhau để đưa ra kết quả tối ưu hơn.

## 4.2 Thực hiện trên dữ liệu 3 tỉnh Tây Nguyên (Gia Lai, Lâm Đồng, Đắk Lắk)

Dữ liệu thời tiết được lấy từ năm 2014 đến 2020. Mỗi ngày lấy các thông tin thời tiết gồm các trường sau:

- max\_temp: Nhiệt độ cao nhất trong ngày
- min\_temp: Nhiệt độ thấp nhất trong ngày
- wind\_x: Vector gió theo chiều x (được tạo thành từ hướng gió và tốc độ gió)
- wind\_y: Vector gió theo chiều y
- rain: Lượng mưa
- humidity: Độ ẩm
- Cloud: Tỷ lệ mây che phủ.

Với mỗi địa điểm xảy ra cháy trong một ngày nhất định thì ta lấy dữ liệu thời tiết (gồm các trường dữ liệu như trên) từ ngày trước khi xảy ra vụ cháy trở về trước 30 ngày. Với một ngày gồm 7 trường dữ liệu thời tiết trên ta nhân cho 30 ngày thì thu được 210 trường dữ liệu dùng để training.

	address	day	month	year	label	wildfire_in_a_day	max_temp_d1	min_temp_d1	wind_x_d1	wind_y_d1	rain_d1	humidi_d1	cloud_d1	max_temp_d2	min_temp_d2	wind_x_d2	wind_y_d2	rain_d2	humidi_d2	cloud_d2
0	Krông Nà Lặc, Đắk Lắk	5	12	2020	1	1	25	18	4.209518	10.162675	0.2	82	2	24	17	4.592201	11.086554	0.2	81	2
1	Cư Pao, MĐrắk, Đắk Lắk	5	12	2020	1	2	25	18	4.209518	10.162675	0.2	82	2	24	17	4.592201	11.086554	0.2	81	2

2 rows × 21 columns

Hình 11: Dữ liệu dùng để đào tạo model

Để thuận tiện cho việc dự báo cháy rừng và mang lại độ chính xác cao, ta tiến hành chia thành 2 model riêng biệt. Model đầu tiên dùng để dự đoán cháy rừng, đầu vào là dữ liệu thời tiết như ở trên, đầu ra là 2 nhãn 0 (không xảy ra cháy rừng) và nhãn 1 (xảy ra cháy rừng). Model thứ 2 dùng để dự đoán cấp độ cháy rừng, cấp độ càng cao thì càng có độ tin cậy về cháy rừng càng cao. Đầu vào của model 2 này là dữ liệu thời tiết như trên nhưng mà có xảy ra cháy rừng, sau đó ta tiến hành chia dữ liệu thành 3 nhãn: nhãn 1 (cháy rừng cấp 1), nhãn 2 (cháy rừng cấp độ 2), nhãn 3 (cháy rừng cấp độ 3).

#### 4.2.1 Model dự đoán cháy rừng

**Chia dữ liệu** Từ bộ dữ liệu đã được tạo ở trên, ta tiến hành chia dữ liệu thành các phần training, validation, testing. Ban đầu ta sẽ chia bộ dữ liệu thành training (70%) và validation + testing (30%). Sau đó từ bộ dữ liệu validation + testing, ta tiến hành chia dữ liệu thêm 1 lần nữa với tỷ lệ tương ứng là validation (70%) và testing (30%)

Sau khi chi dữ liệu xong ta có số lượng các bộ dữ liệu như sau:

- Training dataset: 32921
- Validation dataset: 9876
- Testing dataset: 4233

#### Thiết lập, Training

**Support Vector Machines (SVM)** Ta sử dụng pipeline của thư viện sklearn để tiền xử lý dữ liệu và khởi tạo model SVM.

```
clf = make_pipeline(StandardScaler(), SVC(C=10, gamma='auto', kernel='rbf'))
```

Trong SVC() ta thiết lập các thông số:

1. C=1, gamma='auto', kernel='linear'
  - Thời gian train: 9 min 28 second
  - Thời gian validation: 15 second
  - Thời gian test: 7 second
  - Accuracy:
    - validation: 0.9590927501012556
    - testing: 0.9581856839121191
  - Mean Squared Error:
    - validation: 0.04090724989874443
    - testing: 0.041814316087880936
2. C=1, gamma='auto', kernel='rbf'
  - Thời gian train: 41 second
  - Thời gian validation: 9 second
  - Thời gian test: 4 second
  - Accuracy:
    - validation: 0.987444309437019
    - testing: 0.9877155681549729

- Mean Squared Error:
  - validation: 0.012555690562980963
  - testing: 0.012284431845027168

3. C=10, gamma='auto', kernel='rbf'

- Thời gian train: 1 min 32 second
- Thời gian validation: 8 second
- Thời gian test: 4 second
- Accuracy:
  - validation: 0.9920008100445524
  - testing: 0.9943302622253721
- Mean Squared Error:
  - validation: 0.00799918995544755
  - testing: 0.005669737774627924

Mean squared error được tính bằng công thức:

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2 \quad (1)$$

Trong đó:

- $\hat{y}$  là một tập các nhãn dự đoán
- $y$  là một tập các nhãn đúng
- $\hat{y}$  là giá trị dữ đoán của mẫu thứ  $i$
- $y_i$  là giá trị thực tương ứng với giá trị dự đoán trên
- $n_{samples}$  là số lượng các giá trị trong tập  $y$  (hoặc  $\hat{y}$ )

**Dense** Đầu tiên, ta khởi tạo model Sequential() của thư viện keras. Ta tiếp tục tạo lớp Input (đầu vào), và các lớp ẩn và cuối cùng là tạo lớp đầu ra. Trong lớp đầu vào ta cần khởi tạo số nút tương ứng với kích thước của bộ dữ liệu. Trong các lớp ẩn thì ta cũng khởi tạo số nút cho mỗi lớp ẩn và thiết lập loại hình kích hoạt. Cuối cùng, trong lớp tạo số nút đầu ra và cũng thiết lập loại hình kích hoạt.

```
model = Sequential()
model.add(Input(shape=(210,)))
model.add(Dense(128, activation='relu', name='hidden_layer_1'))
model.add(Dense(64, activation='relu', name='hidden_layer_2'))
model.add(Dense(1, activation='sigmoid', name='output_layer'))
```

Tiếp theo ta thiết lập hàm call back Early Stopping dùng để dừng quá trình train sớm, sau khi mà thông số chúng ta quan sát (có thể là validation accuracy hay validation loss) không "khá lên" sau một vài epochs. Đây cũng là một kỹ thuật hay được dùng để tránh Overfit.

```
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5,
↪ mode='min')
```

Tiếp tục ta tiếp hành lựa chọn các thuật toán tối ưu (optimizers). Về cơ bản, thuật toán tối ưu là cơ sở để xây dựng mô hình neural network với mục đích học được các features (hay pattern) của dữ liệu đầu vào, từ đó có thể tìm được một cặp weights và bias phù hợp để tối ưu hóa model. Trong các thuật toán tối ưu hóa ta tiến hành chọn learning rate (tỉ lệ học tập) sao cho phù hợp với bài toán.

```
sgd = SGD(learning_rate=0.01)
```

Sau khi xây dựng model trên xong thì ta tiến hành compile nó có tác dụng biên tập lại toàn bộ model của chúng ta đã xây dựng. Ở đây ta có thể lựa chọn các tham số để training model như: thuật toán training thông qua tham số optimize, hàm loss, chọn metrics hiện thị khi model được train.

```
model.compile(optimizer=sgd, loss='categorical_crossentropy',  
              metrics=['accuracy'])
```

Tiếp đến ta dùng hàm fit để đưa data vào training để tìm tham số model.

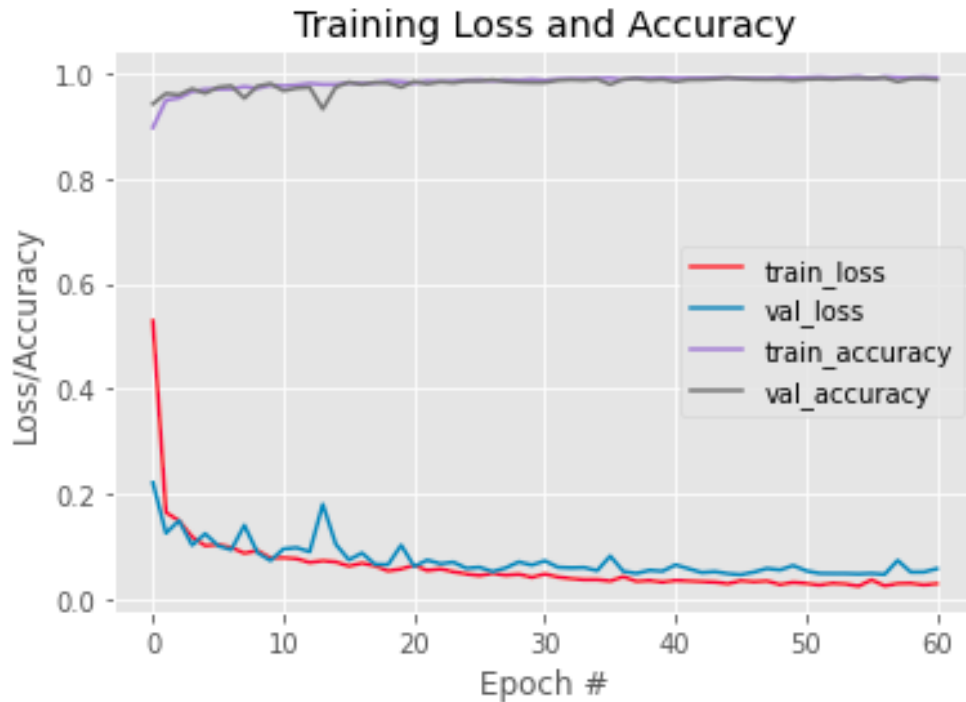
```
history = model.fit(X_train, y_train, batch_size=64, epochs=200, validation_data=(X_valid,
```

Từ model đã được xây dựng trên, ta tiến hành chỉnh sửa các thông số của model:

1. Hai lớp ẩn (số nút tương ứng là 128 và 64, activation="relu"), SGD(learning\_rate=0.01), loss="binary\_crossentropy"
  - Thời gian train: 5 min 13 second
  - Thời gian validation: 27 second
  - Thời gian test: 5 second
  - Accuracy:
    - validation: 0.9533211588859558
    - testing: 0.9588943719863892
  - Loss:
    - validation: 0.15693733096122742
    - testing: 0.1464112251996994
  - Mean Squared Error:
    - validation: 0.042361213827923364
    - testing: 0.03825463792437562
2. Hai lớp ẩn (số nút tương ứng là 128 và 64, activation="relu"), optimizer="adam", loss="binary\_crossentropy"
  - Thời gian train: 5 min 13 second
  - Thời gian validation: 7 second
  - Thời gian test: 5 second
  - Accuracy:
    - validation: 0.9870392680168152
    - testing: 0.9893692135810852
  - Loss:
    - validation: 0.059234291315078735
    - testing: 0.046542562544345856
  - Mean Squared Error:
    - validation: 0.011526199238160804
    - testing: 0.009204111539974925
3. Ba lớp ẩn (số nút tương ứng là 128, 64 và 32, activation="relu"), optimizer="adam", loss="binary\_crossentropy"
  - Thời gian train: 3 min 23 second
  - Thời gian validation: 2 second
  - Thời gian test: 1 second
  - Accuracy:
    - validation: 0.990481972694397
    - testing: 0.9922041296958923
  - Loss:
    - validation: 0.0414808988571167



- testing: 0.037396930158138275
- Mean Squared Error:
  - validation: 0.00846018838163131
  - testing: 0.00734131887253742



Hình 12: Biểu đồ thể hiện quá trình training

**Đánh giá** Từ hai thuật toán phân loại phân loại trên ta thấy được model hình dự đoán khá tốt khi dự báo cháy rừng trên dữ liệu thời tiết. Cụ thể, đối với model SVM cho ra accuracy đối với tập dữ liệu validation là 99,2% và dữ liệu testing thì 99,43%. Còn model Dense thì cho ra accuracy đối với tập dữ liệu validation là 99,05% và dữ liệu testing thì 99,22%

#### 4.2.2 Model dự đoán cấp độ cháy rừng

Model này dùng để dự đoán các cấp độ cháy rừng sau khi đã có kết quả của model trên. Nếu model dự báo cháy rừng trên cho ra kết quả là 1 (tương ứng với việc có xảy ra cháy rừng) thì ta tiếp tục sử dụng model này để phân loại các cấp độ cháy rừng, để có thể kịp thời ứng phó với nguy cơ cháy rừng. Còn nếu model trên dự đoán ra 0 thì ta không cần phải dự đoán tiếp model này.

**Chia dữ liệu** Tương tự như model trên việc chia dữ liệu cũng giống tương tự. Sau khi chi dữ liệu xong ta có số lượng các bộ dữ liệu như sau:

- Training dataset: 17311
- Validation dataset: 5193
- Testing dataset: 2226

#### Thiết lập, Training

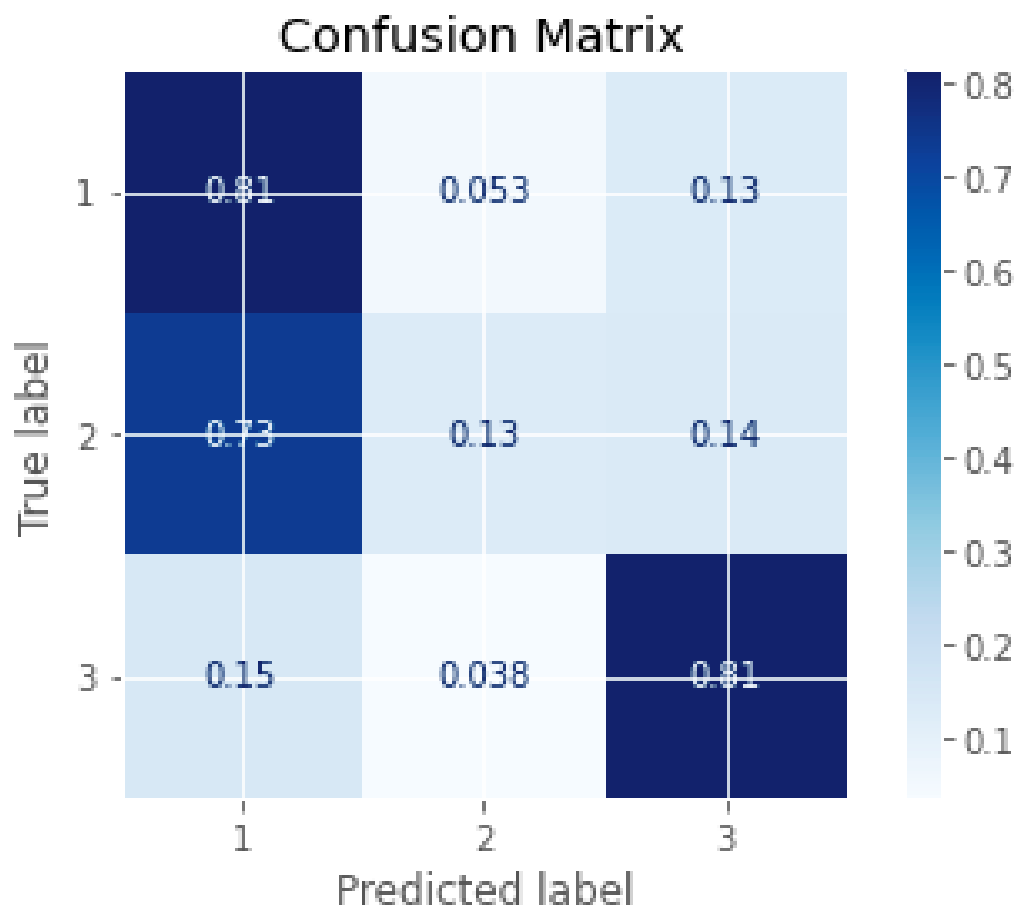
**Support Vector Machines (SVM)** Tương tự như SVM của model trên ta cũng tiến hành thiết lập model và thay đổi các thông số để cho ra kết quả mong muốn:

1. C=1, gamma='auto', kernel='poly'

- Thời gian train: 2 min 8 second
- Thời gian validation: 25 second
- Thời gian test: 12 second
- Accuracy:
  - validation: 0.6187175043327556
  - testing: 0.6154537286612758
- Mean Squared Error:
  - validation: 0.7232813402657423
  - testing: 0.7497753818508536

2. C=1, gamma='auto', kernel='rbf'

- Thời gian train: 1 min 56 second
- Thời gian validation: 27 second
- Thời gian test: 11 second
- Accuracy:
  - validation: 0.6347005584440593
  - testing: 0.6302785265049416
- Mean Squared Error:
  - validation: 0.6761024455998459
  - testing: 0.7133872416891285



Hình 13: Confusion Matrix

**Dense** Ta thiết lập model tương tự model trên:

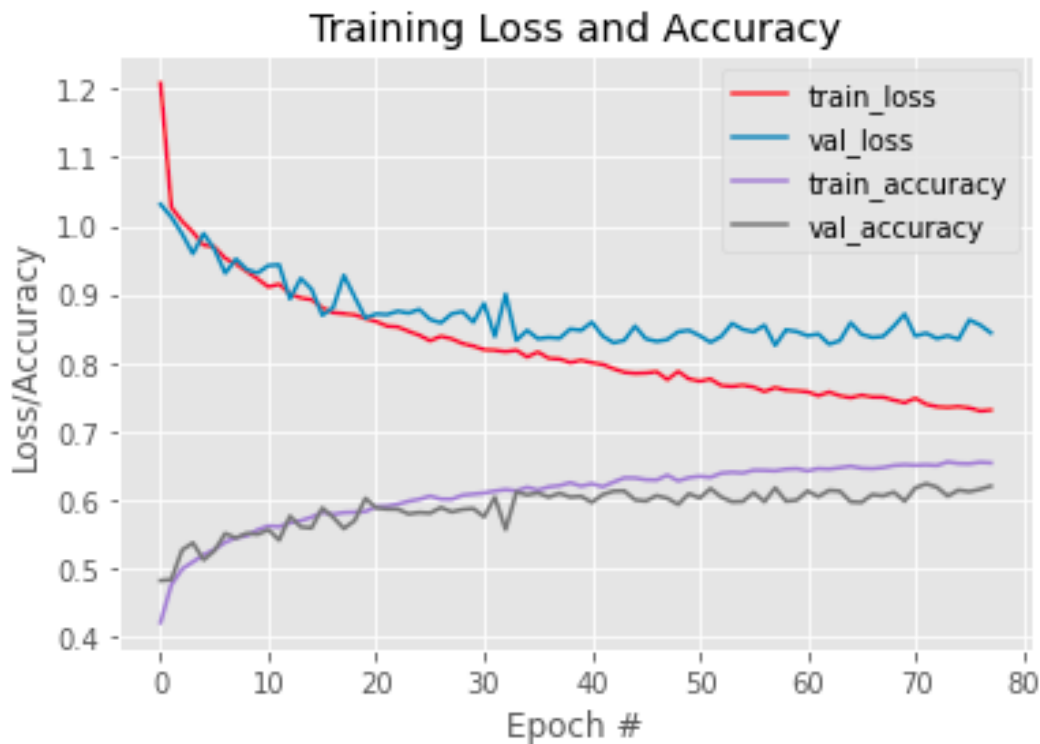
```
model = Sequential()
model.add(Dense(128, input_shape=(210,), activation="relu",
↪ name='hidden_layer_1'))
model.add(Dense(64, activation="relu", name='hidden_layer_2'))
model.add(Dense(32, activation="relu", name='hidden_layer_3'))
model.add(Dense(16, activation="relu", name='hidden_layer_4'))
model.add(Dense(3, activation="sigmoid", name='output_layer'))

callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10,
↪ mode='min')

model.compile(optimizer='adam', loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(X_train, y_train, batch_size=64, epochs=200,
↪ validation_data=(X_valid, y_valid), callbacks=callback, verbose=1)
```

- Thời gian train: 1 min 4 second
- Thời gian validation: 5 second
- Thời gian test: 2 second
- Accuracy:
  - validation: 0.6192951798439026
  - testing: 0.6145552396774292]
- Loss:
  - validation: 0.8435102105140686
  - testing: 0.8677495121955872
- Mean Squared Error:
  - validation: 0.16747459320485225
  - testing: 0.17110031298128617



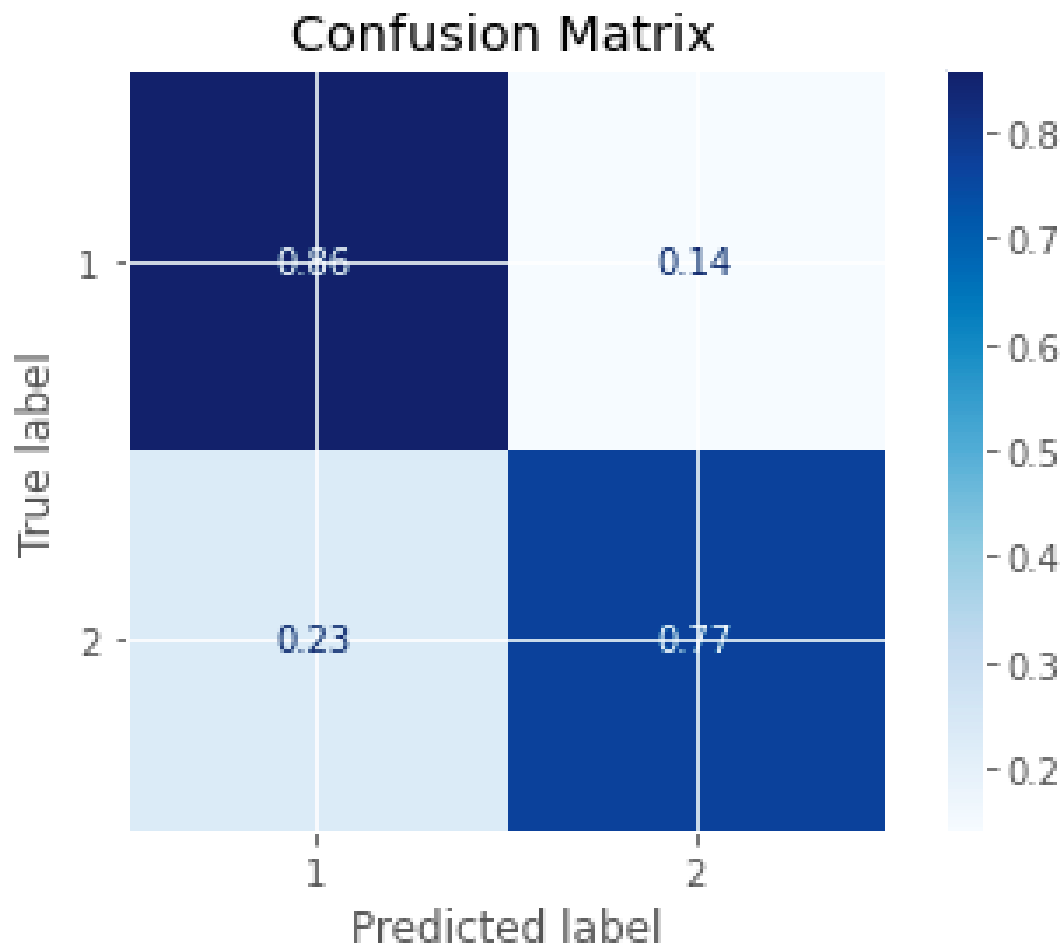
Hình 14: Biểu đồ thể hiện quá trình training

Từ model trên, ta thấy được nhãn 2 dự đoán đúng rất thấp, chủ yếu lệch về phía nhãn 1 rất nhiều, vì vật để khắc phục vấn đề này ta tiến hành giảm lớp class xuống, chỉ còn 2 lớp (nhãn 1, 2) và tiến hành phân chia lại dữ liệu. Thực ra, ban đầu nhóm chia dữ liệu cháy thành 3 lớp là dự trên số lượng các điểm cháy xảy ra trong ngày ở một xã nhất định. Vì theo quan điểm của nhóm thì số lượng điểm cháy xảy ở một xã trong một ngày càng nhiều thì cho thấy dữ liệu thời tiết đó càng có nguy cơ cao xảy ra cháy rừng. Nhưng sau khi phân chia dữ liệu, tiến hành train thì xảy ra hiện tượng trên, dữ liệu không khớp khi với nhãn 2, phần lớn những dữ liệu đó thì dễ nhầm với nhãn 1 hơn nên nhóm mới quyết định gộp nhãn 1 với nhãn 2 cũ thành nhãn 1 mới và chuyển nhãn 3 thành nhãn 2. Ta tiến hành train với dữ liệu mới. Sau khi train với nhiều model khác nhau, ta có kết quả các model như sau:

**Support Vector Machines (SVM)** Trong SVC() ta thiết lập các thông số:

1. C=1, gamma='auto', kernel='rbf'
  - Thời gian train: 1 mini 29 second
  - Thời gian validation: 19 second
  - Thời gian test: 8 second
  - Accuracy:
    - validation: 0.8239938378586559
    - testing: 0.8221024258760108
  - Mean Squared Error:
    - validation: 0.17600616214134412
    - testing: 0.1778975741239892
2. C=10, gamma='auto', kernel='rbf'
  - Thời gian train: 2 min 20 second
  - Thời gian validation: 17 second
  - Thời gian test: 7 second
  - Accuracy:

- validation: 0.8234161371076449
- testing: 0.8274932614555256
- Mean Squared Error:
  - validation: 0.1765838628923551
  - testing: 0.1725067385444744

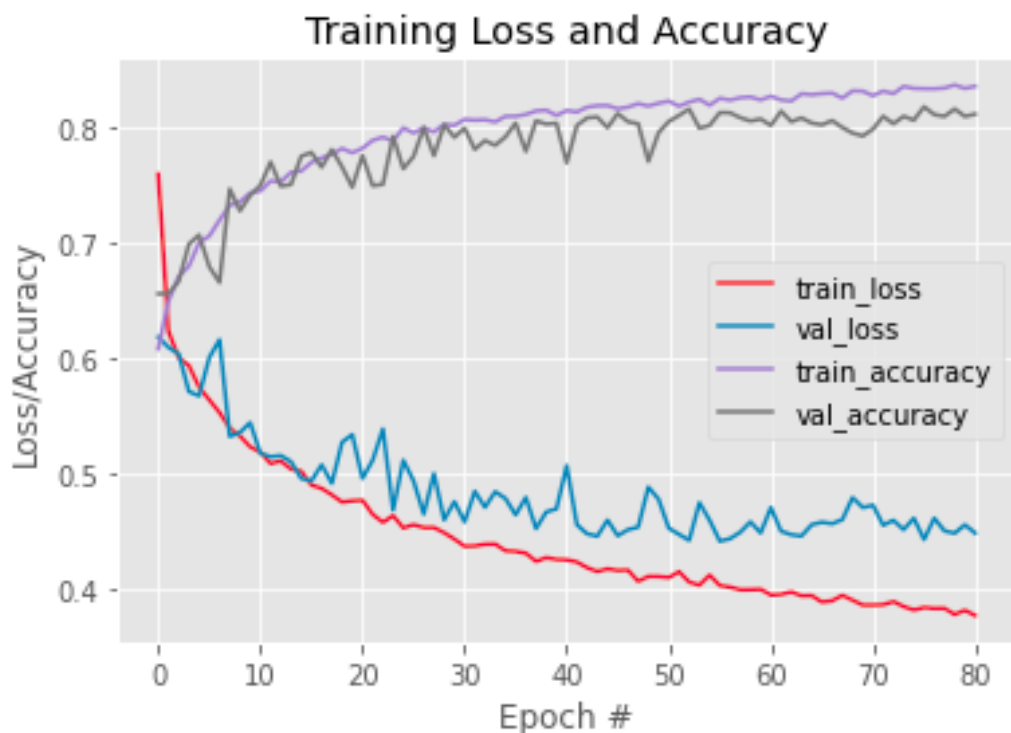


Hình 15: Confusion Matrix

**Dense** Tương tự, ta chỉ xét đến các thông số sau khi train xong model:

- Thời gian train: 2 min 6 second
- Thời gian validation: 5 second
- Thời gian test: 2 second
- Accuracy:
  - validation: 0.8137878179550171
  - testing: 0.8126684427261353
- Loss:
  - validation: 0.4617549777030945
  - testing: 0.46853387355804443
- Mean Squared Error:

- validation: 0.14147366984777401
- testing: 0.14031667554728067



Hình 16: Biểu đồ thể hiện quá trình training

**Đánh giá** Nhìn chung model được chia thành 3 lớp dựa trên số điểm cháy của xã trong một ngày ban đầu cho ra kết quả khá tệ, nó chỉ dự đoán tốt trên nhãn 1 và nhãn 3 (81% và 81%) còn nhãn 2 thì cho ra kết quả khá tệ (13%), các dự đoán gần như là rơi vào nhãn 1 (chiếm đến 73%).

Khi xây dựng lại bộ dữ liệu 2 lớp thì cho ra kết quả khả quan hơn với accuracy là 82,39% đối với dữ liệu validation và 82,21% đối với dữ liệu testing. Mặc dù khi giảm số class thì accuracy sẽ tăng lên, dẫn đến phân loại đúng hơn. Nhưng cũng phải chấp nhận phương pháp trên, việc phân chia số lượng class trên còn phải dựa vào số lượng dữ liệu điểm cháy nữa nên không thể tạo thành nhiều lớp hơn khi gộp dữ liệu nhãn 1 và 2 trên lại với nhau. Vì vậy, nhóm quyết định chỉ làm 2 lớp để phân loại.

#### 4.2.3 Tổng kết

Qua quá trình đào tạo 2 mô hình khác nhau, với độ dữ đoán chính xác là khác nhau. Mô hình đầu tiên cho kết quả tốt trên cả tập train, tập validation và tập test. Loss thấp và mean squared error cũng khá thấp, cho thấy khả năng học tập và đào tạo của mô hình là khá tốt. Từ đó thấy được mô hình dự đoán cháy rừng có thể áp dụng thực tế được. Mặc dù, nhóm chưa thử kiểm tra với dữ liệu năm 2021 (vì dữ liệu thời tiết thì có thể thu thập, nhưng dữ liệu cháy rừng năm 2021 thì chưa có nên khi dự đoán thì không thể kiểm tra được nên nhóm chưa trên khai kiểm tra thử) nhưng với độ chính xác cao thì nó vẫn có khả năng lớn là cho ra kết quả dự đoán đúng về việc xảy ra cháy rừng trên địa bàn tỉnh Tây Nguyên.

Mô hình thử hai dự đoán cấp độ cháy rừng cho kết quả thấp với việc phân chia dữ liệu thành 3 lớp, nhưng khi thấy vấn đề ở việc phân chia dữ liệu như vậy, thì nhóm quyết định giảm lớp phân loại xuống còn hai lớp và tiến hành phân chia lại dữ liệu thì cho ra kết quả khả quan hơn với độ chính xác trên 80%, điều này làm cho việc kiểm tra phân loại cấp độ cháy cho độ chính xác cao hơn và có thể thử nghiệm thử trong thực tế.

## 5 Ứng dụng và hướng phát triển

Tuy đề tài là một ứng dụng rất cần thiết trong thực tế và nhóm nhận thấy rằng đã xử lý nhiều phương diện cơ bản của một đề án Máy học (phân tích website để thu thập dữ liệu, xử lý thiếu sót, mất cân bằng dữ liệu, xử lý các model cơ bản,...). Song vẫn phải thừa nhận kết quả đạt được còn nhiều điều hạn chế, thiếu sót trong việc ứng dụng.

### 5.1 Ứng dụng

Phải thành thật thừa nhận rằng tính ứng dụng của đề án chưa cao, đó là điều mà nhóm đã nhận thấy từ lời khuyên ban đầu của thầy. Có rất nhiều nguyên nhân trong đó có cả việc những tài nguyên (input) đưa vào model không đủ để phân tích. Vì có rất nhiều vấn đề vĩ mô lẫn vi mô liên quan mật thiết đến cháy rừng như: mật độ thực vật, độ đa dạng, hình thái phân bố thực vật,... Kinh nghiệm còn nhiều thiếu sót cũng là một điểm yếu của nhóm.

Tuy nhiên, còn điểm yếu là còn có thể cải thiện, đề tài này tuy có tính khả quan rất thấp nhưng lại có tính cần thiết rất cao vì tất cả chúng ta đều biết những nguy tại của thiên tai có thể ảnh hưởng lớn thế nào đến con người.

### 5.2 Hướng phát triển

Như đã nói ở trên, nếu có được các nghiên cứu hơn về cấu trúc rừng ở nước ta (rừng trồng hay rừng tự nhiên), những yếu tố liên quan đến phân bố thực vật (loại thực vật nào sinh trưởng chủ yếu ở những vùng nào), thậm chí là thu nhập của người dân ở từng vùng. Là những yếu tố thực tế gắn liền với rừng hiện nay hơn là thời tiết ta vẫn có hy vọng một kết quả nghiên cứu tốt hơn.

Ngoài ra một yếu tố như bản đồ hoàn chỉnh của nước ta về các bản đồ với những đặc trưng trên có thể áp dụng tốt những cấu trúc model khác như sử dụng phép Convolution [1] trên bản đồ để trích rút ra được đặc trưng trên phương diện địa lý cũng là một ý tưởng.

## Tài liệu

- [1] Convolution - wikipedia, July 2021.
- [2] Tỉnh - wikipedia, January 2021.
- [3] Ali Ahmadalipour. Ai and climate data for predicting fire frequency in california, December 2020.
- [4] Cục Kiểm Lâm Tổng cục Lâm Nghiệp. <http://firewatchvn.kiemlam.org.vn/>.
- [5] Christopher Justice, Louis Giglio, Luigi Boschetti, David Roy, Ivan Csiszar, Jeffrey Morisette, and Yoram Kaufman. Atbd mod14 - algorithm technical background document - modis fire products (eos id numbers 2741), October 2006.
- [6] Susan Malaika. Call for code spot challenge for wildfires, November 2020.
- [7] TensorFlow. Time series forecasting, 2021.
- [8] Wikipedia. The weather channel, July 2021.