ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA HỆ THỐNG THÔNG TIN

○○🕮○○

BÁO CÁO ĐỒ ÁN

KHAI THÁC DỮ LIỆU

ĐỀ TÀI:

ỨNG DỤNG MÔ HÌNH DỰ ĐOÁN DỮ LIỆU TAI NẠN GIAO THÔNG LIÊN QUAN ĐẾN KHẢ NĂNG XẢY RA VA CHẠM

Lớp: IS252.O21

Giảng viên hướng dẫn: Mai Xuân Hùng

Phạm Nguyễn Thanh Bình

Sinh viên thực hiện: Hồ Quang Đỉnh - 21520190

Tp.Hồ Chí Minh, năm 2024

LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn sâu sắc đến quý thầy cô trường Đại học Công Nghệ Thông Tin, đặc biệt là những giảng viên trong ngành đã trang bị cho em có được những kiến thức căn bản vững chắc để có thể thực hiện đồ án lần này.

Em chân thành cảm ơn thầy Phạm Nguyễn Thanh Bình đã giúp đỡ em trong suốt quá trình học tập và thực hiện đồ án, sự quan tâm, giảng dạy tận tình của thầy đã hỗ trợ em rất nhiều trong thời gian vừa qua và qua đó em đã học hỏi được nhiều kiến thức hơn. Một lần nữa em xin gửi lời cảm ơn sâu sắc đến thầy.

Trong quá trình nghiên cứu và thực hiện đồ án "Ứng dụng mô hình dự đoán dữ liệu tai nạn giao thông liên quan đến khả năng xảy ra va chạm", em đã kết hợp giữa những kiến thức căn bản và những gì được thầy trao đổi và truyền đạt trên lớp để cố gắng hoàn thiện đồ án tốt nhất có thể.Tuy nhiên, trong đồ án của em vẫn còn một vài thiếu sót nhưng nó là kết quả của sự nỗ lực và cố gắng của em cũng như sự giúp đỡ từ bạn bè và Thầy Cô. Nhóm rất mong nhận được sự góp ý từ phía Thầy Cô nhằm rút ra được những kinh nghiệm quý báu và hoàn thiện vốn kiến thức của mình để có thể tiếp tục hoàn thành được những đồ án khác ở trong tương lai.

Lời cuối cùng, em xin chúc quý Thầy Cô thật nhiều sức khỏe và niềm vui để có thể tiếp tục giảng dạy và truyền đạt thật nhiều kiến thức bổ ích đến cho những sinh viên khác. Em xin chân thành cảm ơn!

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN

1.1. Lý do chọn đề tài

Bộ dữ liệu Nashville Vehicle Collision Datas là một bộ dữ liệu chứa thông tin chi tiết về các vụ tai nạn giao thông xảy ra trên nhiều quận và bang tại Hoa Kỳ. Dữ liệu bao gồm 214,601 vụ tai nạn được ghi nhận, cung cấp một nguồn tài nguyên phong phú để nghiên cứu và phân tích các yếu tố liên quan đến an toàn giao thông. Ngoài ra bộ dữ liệu này cung cấp cái nhìn tổng quan về tình hình tai nạn giao thông và mối liên quan giữa các yếu tố khác nhau trong các vụ tai nạn. Nó giúp các nhà nghiên cứu, nhà hoạch định chính sách và các cơ quan chức năng có cái nhìn sâu sắc về các yếu tố ảnh hưởng đến tai nạn giao thông và đưa ra các biện pháp cải thiện an toàn giao thông hiệu quả. Bộ dữ liệu tai nạn xe hơi của Nashville có thể dùng để tìm hiểu về mặt địa lý trong Nashville và các khu vực lân cận:

- Tai nạn xảy ra ở đâu nhiều nhất?
- Có bao nhiêu người tử vong?
- Có bao nhiêu người bị thương?
- Loại va chạm nào xảy ra thường xuyên hơn?
- Ban ngày & hoặc ban đêm có ảnh hưởng đến tai nạn không?

1.2. Giới thiệu về Dataset

1.2.1. Giới thiệu nguồn dữ liệu

1.2.1.1. Tên bộ dữ liệu

- Tên dataset: Nashville Vehicle Collision Datas
- Ngày cập nhật gần nhất: 04/2022

1.2.1.2. Đơn vị cung cấp

- Đây là bộ dữ liệu mô tả tai nạn xe hơi của Nashville được ghi bởi Nina Flirp

1.2.1.3. Mô tả số dòng, cột, thời gian thu thập

- Bộ dữ liệu gồm 214.601 dòng dữ liệu và 25 cột thuộc tính
- Dữ liệu được thu thập tới 08/04/2022

1.2.1.4. Nguồn tải dataset

- Link dataset: Nashville Vehicle Collision Dataset (kaggle.com)

1.2.2. Mô tả các cột thuộc tính

STT	Tên thuộc tính	Ý nghĩa	Kiểu dữ liệu
1	Accident Number	Mã số tai nạn	Int
2	Date and Time	Ngày và giờ xảy ra tai nạn	Object
3	Number of Motor Vehicles	Số lượng phương tiện liên quan	Float

4	Number of Injuries	Số người bị thương	Int
5	Number of Fatalities	Số người tử vong	Int
6	Property Damage	Thiệt hại tài sản	Object
7	Hit and Run	Vụ tai nạn bỏ chạy	Object
8	Reporting Officer	Mã số nhân viên báo cáo	Float
9	Collision Type Code	Mã loại va chạm	Float
10	Collision Type Description	Mô tả loại va chạm	Object
11	Weather Code	Mã thời tiết	Float
12	Weather Description	Mô tả thời tiết	Object
13	Illumination Code	Mã khung giờ chiếu sáng	
14	Illumination Description	Mô tả khung giờ chiếu sáng	Object
15	Harmful Code	Mã mô tả tổn hại	Object
16	Harmful Description	Mô tả tổn hại	Object
17	Street Address	Địa chỉ đường phố	Object
18	City	Thành phố	Object
19	State	Bang	Object
20	ZIP	Mã bưu điện	Float
21	RPA	Mã khu vực	Float
22	Precinct	Đơn vị	Object
23	Latitude	Vĩ độ	Float
24	Longitude	Kinh độ	Float
25	Mapped Location	Vị trí được xác định	Object

Hình 12

1.2.3. Phát biểu bài toán

- Dựa vào các thuộc tính có sẵn trong bộ dữ liệu tai nạn giao thông, chúng ta có thể dự đoán và phân tích mối quan hệ giữa các điều kiện khách quan và các vụ tai nạn. Mục tiêu là xác định các điều kiện cụ thể có thể làm tăng khả năng xảy ra tai nạn, từ đó đề xuất các biện pháp phòng ngừa và cải thiện an toàn giao thông.

1.2.4. Công cụ khai thác dữ liệu

- Công cụ khai thác dữ liệu: Jupyter notebook
- Phiên bản: Python 3.10.4

1.2.5. Các thư viện kèm theo

1.2.5.1. Pandas

- Để xử lý và tiền xử lý dữ liệu một cách mượt mà, tôi đã dựa vào thư viện pandas. Công cụ đa năng này giúp tải dữ liệu và cho phép xử lý dữ liệu hiệu quả, đảm bảo rằng bộ dữ liệu được chuẩn bị tốt cho các bước mô hình hóa tiếp theo.

1.2.5.2. Matplotlib

- Cung cấp các hàm để tạo biểu đồ, biểu đồ cột, biểu đồ đường, và nhiều loại biểu đồ khác.

1.2.5.3. Numpy

- Hỗ trợ các phép toán với mảng, ma trận, và cung cấp nhiều hàm toán học hiệu quả.

1.2.5.4. Sklearn

- Tận dụng chức năng toàn diện của thư viện scikit-learn, tôi đã triển khai các mô hình học máy một cách liền mạch, cho phép quá trình huấn luyện, kiểm tra và đánh giá diễn ra hiệu quả.

1.2.5.5. Seaborn

- Sử dụng thư viện seaborn, tôi đã trực quan hóa dữ liệu để có được những hiểu biết có giá trị về phân phối và các mẫu trong dữ liệu. Công cụ trực quan hóa này đóng vai trò quan trọng trong việc nâng cao hiểu biết của tôi về bộ dữ liệu và định hướng các quyết định tiền xử lý tiếp theo.

CHƯƠNG 2. TIỀN XỬ LÝ DỮ LIỆU

2.1. Xóa các thuộc tính không cần thiết trong quá trình khai thác

- Import các thư viện của Python

```
[1]  %matplotlib inline
     import matplotlib.pyplot as plt
     import numpy as np
     import pandas as pd
     import seaborn as sns
     import csv
     from sklearn.model_selection import train_test_split
     from sklearn import tree
     from sklearn.naive_bayes import GaussianNB
     from sklearn import metrics
     from sklearn.preprocessing import OneHotEncoder
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.ensemble import RandomForestClassifier
```

- Tiến hành đọc file .csv

```
[3]  df = pd.read_csv('/content/drive/MyDrive/Week3/Traffic_Accidents.csv')
```

=> Dataset có 214601 dòng dữ liệu và 25 cột thuộc tính

```
# Hiển thị thông tin tổng quan về dataset
df
```

	Accident Number	Date and Time	Number of Motor Vehicles	Number of Injuries	Number of Fatalities	Property Damage	Hit and Run	Reporting Officer	Collision Type Code	Collision Type Description	...	Harmful Description	Street Address	City	State	ZIP	RPA	Precinct	Latitude	Longitude	Mapped Location
0	20200606233	September 21, 2020 06:37 PM	1.0	0	0	NaN	False	311344.0	0.0	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	...	TREE,Ran Off Road-Left	LEWIS ST & FAIN ST	NASHVILLE	TN	37210.0	8207.0	HERMIT	36.1518	-86.7580	POINT (-86.758 36.1518)
1	20200603908	September 20, 2020 12:00 AM	1.0	0	0	True	True	409155.0	0.0	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	...	MAIL BOX	RICHMOND HILL DR & CARTERWOOD DR	NASHVILLE	TN	37207.0	2005.0	MADISO	36.2323	-86.7733	POINT (-86.7733 36.2323)
2	20200604395	September 20, 2020 07:20 PM	1.0	1	0	NaN	False	330419.0	0.0	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	...	OTHER ANIMAL	SHUTE LN & SAUNDERSVILLE RD	OLD HICKORY	TN	37138.0	9717.0	HERMIT	36.2267	-86.6032	POINT (-86.6032 36.2267)
3	20200605021	September 21, 2020 05:58 AM	2.0	0	0	NaN	False	256569.0	5.0	SIDESWIPE - SAME DIRECTION	...	MOTOR VEHICLE IN TRANSPORT	MURFREESBORO PKE & WILHAGAN RD	NASHVILLE	TN	37217.0	8801.0	SOUTH	36.1325	-86.7193	POINT (-86.7193 36.1325)
4	20200605988	September 21, 2020 03:27 PM	2.0	0	0	NaN	False	179606.0	11.0	Front to Rear	...	MOTOR VEHICLE IN TRANSPORT	NOLENSVILLE PKE & WOODYCREST AV	NASHVILLE	TN	37211.0	8041.0	MIDTOW	36.1312	-86.7562	POINT (-86.7562 36.1312)
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
214596	20210529661	October 10, 2021 08:53 PM	2.0	0	0	NaN	False	4000726.0	4.0	ANGLE	...	MOTOR VEHICLE IN TRANSPORT,UTILITY POLE	PEABODY ST & 6TH AVS	NASHVILLE	TN	37203.0	4039.0	CENTRA	36.1549	-86.7767	POINT (-86.7767 36.1549)
214597	20210529129	October 10, 2021 01:24 AM	2.0	0	0	NaN	True	4004451.0	5.0	SIDESWIPE - SAME DIRECTION	...	PARKED MOTOR VEHICLE	NEILL AV & MANSFIELD CT	NASHVILLE	TN	37206.0	1915.0	EAST	36.1816	-86.7600	POINT (-86.76 36.1816)
214598	20210529262	October 10, 2021 02:30 PM	3.0	1	0	NaN	False	4001100.0	11.0	Front to Rear	...	MOTOR VEHICLE IN TRANSPORT	MM 77 5 I 65	NASHVILLE	TN	37220.0	86042.0	MIDTOW	36.0775	-86.7663	POINT (-86.7663 36.0775)
214599	20210529382	October 10, 2021 12:00 AM	2.0	1	0	NaN	False	4001002.0	9.0	UNKNOWN	...	MOTOR VEHICLE IN TRANSPORT,UNKNOWN MOST HARMFU...	MM 58 2 I 24	ANTIOCH	TN	37013.0	87060.0	SOUTH	36.0564	-86.6761	POINT (-86.6761 36.0564)
214600	20210528648	October 10, 2021 02:00 AM	1.0	0	0	NaN	True	342952.0	0.0	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	...	CONCRETE TRAFFIC BARRIER	MM 220 3 I 40	HERMITAGE	TN	37076.0	94020.0	HERMIT	36.1678	-86.6098	POINT (-86.6098 36.1678)

214601 rows × 25 columns

- Hiển thị thông tin các thuộc tín: tên, số lượng dòng không bị null, kiểu dữ liệu

```
# Hiển thị thông tin tổng quan về dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214601 entries, 0 to 214600
Data columns (total 25 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Accident Number             214601 non-null  int64
 1   Date and Time               214601 non-null  object
 2   Number of Motor Vehicles    214600 non-null  float64
 3   Number of Injuries          214601 non-null  int64
 4   Number of Fatalities        214601 non-null  int64
 5   Property Damage             15827 non-null   object
 6   Hit and Run                 214584 non-null  object
 7   Reporting Officer           214591 non-null  float64
 8   Collision Type Code         214585 non-null  float64
 9   Collision Type Description  214585 non-null  object
 10  Weather Code                209302 non-null  float64
 11  Weather Description         209302 non-null  object
 12  Illumination Code           214312 non-null  float64
 13  Illumination Description    214312 non-null  object
 14  Harmful Code                214601 non-null  object
 15  Harmful Description         212361 non-null  object
 16  Street Address              214593 non-null  object
 17  City                        214601 non-null  object
 18  State                       214601 non-null  object
 19  ZIP                         213506 non-null  float64
 20  RPA                         213499 non-null  float64
 21  Precinct                    213482 non-null  object
 22  Latitude                    214220 non-null  float64
 23  Longitude                   214220 non-null  float64
 24  Mapped Location             214220 non-null  object
dtypes: float64(9), int64(3), object(13)
memory usage: 40.9+ MB
```

- Loại bỏ các thuộc tính không cần thiết trong quá trình khai thác
    - Xóa thuộc tính “Accident Number” vì cột này không có ý nghĩa
    - Xóa thuộc tính “Collision Type Code”, “Weather Code”, “Illumination Code”, “Harmful Code” vì các thuộc tính này không cần thiết cho quá trình phân tích dữ liệu.
    - Xóa dữ liệu Reporting Officer, ZIP, RPA, Precinct, Latitude, Longitude , Mapped Location vì không có ý nghĩa trong việc tìm kiếm mối liên hệ giữa các vụ tai nạn và các điều kiện khách quan (thời gian, thời tiết, ánh sáng, đường phố.
    - Xóa thuộc tính Property Damage vì không có ý nghĩa trong việc tìm kiếm mối liên hệ giữa các vụ tai nạn và các điều kiện khách quan (thời gian, thời tiết, ánh sáng, đường phố...) nên ta sẽ loại bỏ.
    - Xóa các thuộc tính Street Address, Harmful Description, Date and Time vì không cần thiết tròn quá trình xử lí dữ liệu.

Dữ liệu sau khi được xóa:

	Number of Motor Vehicles	Number of Injuries	Number of Fatalities	Hit and Run	Collision Type Description	Weather Description	Illumination Description	City	State
0	1.0	0	0	False	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	CLEAR	DUSK	NASHVILLE	TN
1	1.0	0	0	True	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	CLEAR	DAYLIGHT	NASHVILLE	TN
2	1.0	1	0	False	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	NaN	DARK - LIGHTED	OLD HICKORY	TN
3	2.0	0	0	False	SIDESWIPE - SAME DIRECTION	NaN	DAWN	NASHVILLE	TN
4	2.0	0	0	False	Front to Rear	NaN	DAYLIGHT	NASHVILLE	TN
...	...	...	...	...	...	...	...	...	...
214596	2.0	0	0	False	ANGLE	CLEAR	DARK - LIGHTED	NASHVILLE	TN
214597	2.0	0	0	True	SIDESWIPE - SAME DIRECTION	CLEAR	DARK - NOT LIGHTED	NASHVILLE	TN
214598	3.0	1	0	False	Front to Rear	CLEAR	DAYLIGHT	NASHVILLE	TN
214599	2.0	1	0	False	UNKNOWN	CLEAR	DAYLIGHT	ANTIOCH	TN
214600	1.0	0	0	True	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	CLEAR	DARK - LIGHTED	HERMITAGE	TN

214601 rows × 9 columns

- Kiểm tra dữ liệu sau khi xóa đi những cột không cần thiết

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214601 entries, 0 to 214600
Data columns (total 9 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   Number of Motor Vehicles    214600 non-null  float64
 1   Number of Injuries          214601 non-null  int64  
 2   Number of Fatalities        214601 non-null  int64  
 3   Hit and Run                 214584 non-null  object 
 4   Collision Type Description  214585 non-null  object 
 5   Weather Description         209302 non-null  object 
 6   Illumination Description    214312 non-null  object 
 7   City                        214601 non-null  object 
 8   State                       214601 non-null  object 
dtypes: float64(1), int64(2), object(6)
memory usage: 14.7+ MB
```

2.2. Xóa các thuộc tính Null hoặc Unknow

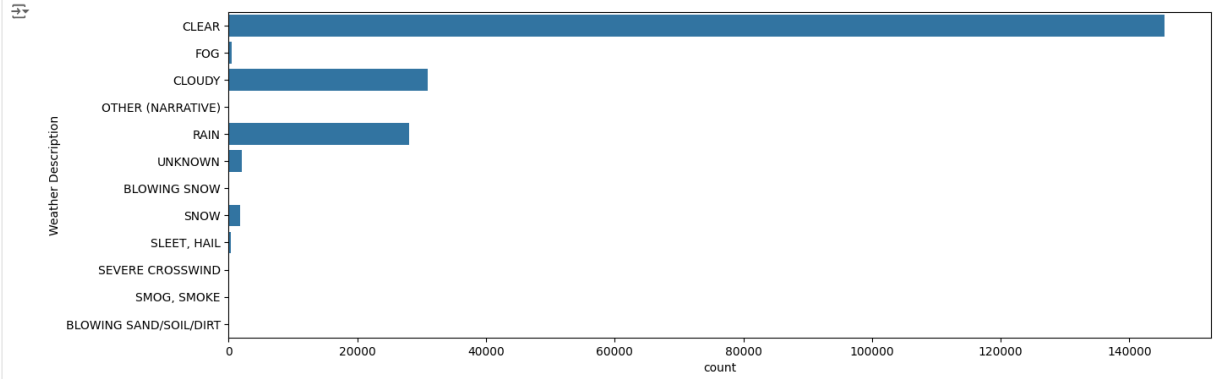
- Kiểm tra các dữ liệu bị thiếu

```
#Xử lí dữ liệu bị thiếu
total = df.isnull().sum().sort_values(ascending=False) #Tìm trong dữ liệu giá null, tính tổng chúng lại, sắp xếp theo tăng dần  = False tức là sắp giảm
percent_1=df.isnull().sum()/df.isnull().count()*100 #Tổng giá trị null của một cột/ số dòng của cột null đó
percent_2 = (round(percent_1,1)).sort_values(ascending = False) #Làm tròn 1 chữ số và sort giảm dần
missing_data = pd.concat([total,percent_2],axis=1,keys=['Total','%'])
missing_data.head(11)
```

	Total	%
Weather Description	5299	2.5
Illumination Description	289	0.1
Hit and Run	17	0.0
Collision Type Description	16	0.0
Number of Motor Vehicles	1	0.0
Number of Injuries	0	0.0
Number of Fatalities	0	0.0
City	0	0.0
State	0	0.0

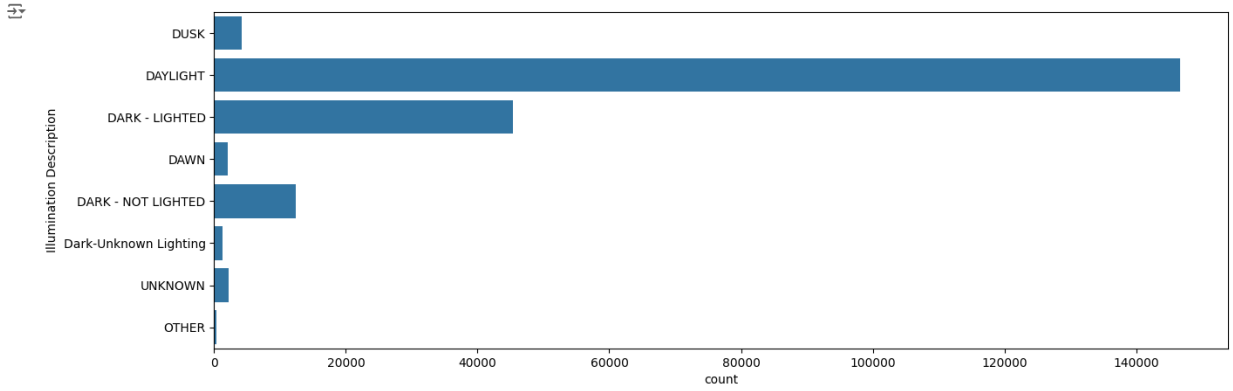
- Xem thuộc tính Weather Description. Ta nhận thấy số lượng giá trị UNKNOWN là rất ít. Do đó, có thể thay thế các giá trị UNKNOWN bằng giá trị phổ biến nhất (CLEAR) hoặc loại bỏ các dòng có giá trị UNKNOWN để tối ưu hóa dữ liệu.

```
#Xem thuộc tính Weather Description
plt.figure(1,figsize=(15,5))
sns.countplot(y = 'Weather Description',data =df)
plt.show()
```



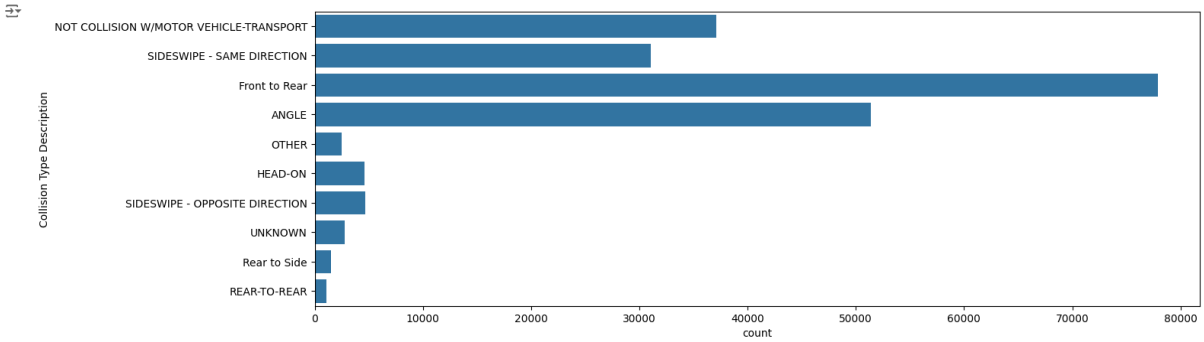
- Xem thuộc tính Illumination Description. Ta nhận thấy số lượng giá trị UNKNOWN là rất ít. Do đó, có thể thay thế các giá trị UNKNOWN bằng giá trị phổ biến nhất (DAYLIGHT) hoặc loại bỏ các dòng có giá trị UNKNOWN để tối ưu hóa dữ liệu.

```
#Xem thuộc tính Illumination Description
plt.figure(1,figsize=(15,5))
sns.countplot(y = 'Illumination Description',data =df)
plt.show()
```



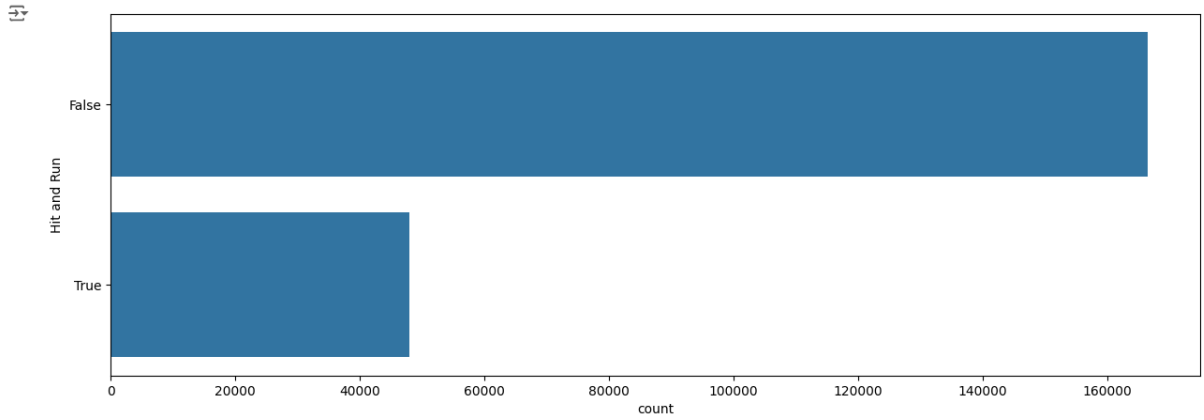
- Xem thuộc tính Collision Type Description. Ta nhận thấy số lượng giá trị UNKNOWN là rất ít. Do đó, có thể thay thế các giá trị UNKNOWN bằng giá trị phổ biến nhất (Front to Rear) hoặc loại bỏ các dòng có giá trị UNKNOWN để tối ưu hóa dữ liệu.

```
#Xem thuộc tính Collision Type Description
plt.figure(1,figsize=(15,5))
sns.countplot(y = 'Collision Type Description',data =df)
plt.show()
```



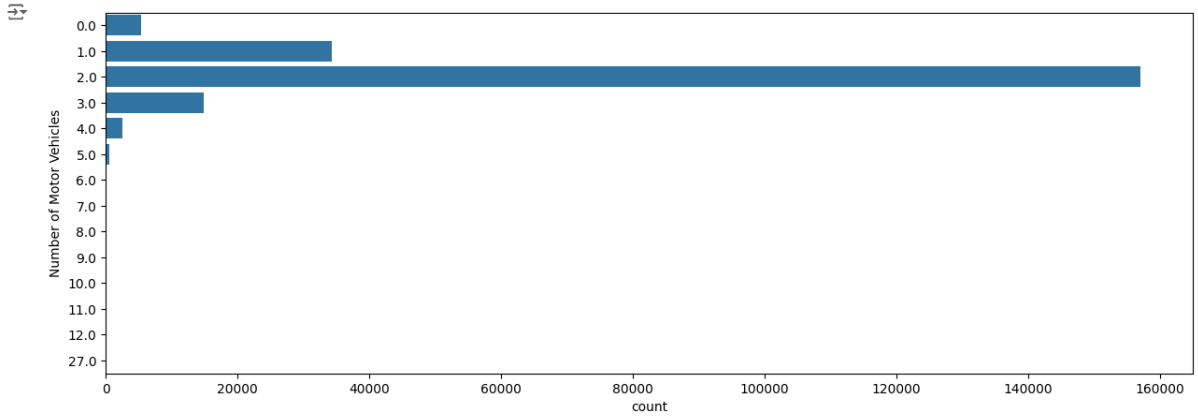
- Xem thuộc tính Hit and Run.

```
#Xem thuộc tính Hit and Run
plt.figure(1,figsize=(15,5))
sns.countplot(y = 'Hit and Run',data =df)
plt.show()
```



- Xem thuộc tính Number of Motor Vehicles.

```
#Xem thuộc tính Number of Motor Vehicles
plt.figure(1,figsize=(15,5))
sns.countplot(y = 'Number of Motor Vehicles',data =df)
plt.show()
```



- Thay thế thuộc tính Unknown hoặc UNKNOWN thành NaN.

```
#Thay thế thuộc tính có Unkown bằng NaN
df.replace('Unknown',np.nan,inplace=True)
df.replace('UNKNOWN',np.nan,inplace=True)
df
```

	Number of Motor Vehicles	Number of Injuries	Number of Fatalities	Hit and Run	Collision Type Description	Weather Description	Illumination Description	City	State
0	1.0	0	0	False	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	CLEAR	DUSK	NASHVILLE	TN
1	1.0	0	0	True	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	CLEAR	DAYLIGHT	NASHVILLE	TN
2	1.0	1	0	False	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	NaN	DARK - LIGHTED	OLD HICKORY	TN
3	2.0	0	0	False	SIDESWIPE - SAME DIRECTION	NaN	DAWN	NASHVILLE	TN
4	2.0	0	0	False	Front to Rear	NaN	DAYLIGHT	NASHVILLE	TN
...	...	...	...	...	...	...	...	...	...
214596	2.0	0	0	False	ANGLE	CLEAR	DARK - LIGHTED	NASHVILLE	TN
214597	2.0	0	0	True	SIDESWIPE - SAME DIRECTION	CLEAR	DARK - NOT LIGHTED	NASHVILLE	TN
214598	3.0	1	0	False	Front to Rear	CLEAR	DAYLIGHT	NASHVILLE	TN
214599	2.0	1	0	False	NaN	CLEAR	DAYLIGHT	ANTIOCH	TN
214600	1.0	0	0	True	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	CLEAR	DARK - LIGHTED	HERMITAGE	TN

214601 rows × 9 columns

- Xóa thuộc tính NaN.

```
test = df.dropna()
test
```

	Number of Motor Vehicles	Number of Injuries	Number of Fatalities	Hit and Run	Collision Type Description	Weather Description	Illumination Description	City	State
0	1.0	0	0	False	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	CLEAR	DUSK	NASHVILLE	TN
1	1.0	0	0	True	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	CLEAR	DAYLIGHT	NASHVILLE	TN
5	2.0	0	0	False	Front to Rear	CLEAR	DAYLIGHT	NASHVILLE	TN
6	2.0	0	0	True	Front to Rear	CLEAR	DAYLIGHT	NASHVILLE	TN
8	2.0	0	0	True	ANGLE	CLEAR	DARK - NOT LIGHTED	HERMITAGE	TN
...	...	...	...	...	...	...	...	...	...
214595	3.0	0	0	False	Front to Rear	CLEAR	DAYLIGHT	NASHVILLE	TN
214596	2.0	0	0	False	ANGLE	CLEAR	DARK - LIGHTED	NASHVILLE	TN
214597	2.0	0	0	True	SIDESWIPE - SAME DIRECTION	CLEAR	DARK - NOT LIGHTED	NASHVILLE	TN
214598	3.0	1	0	False	Front to Rear	CLEAR	DAYLIGHT	NASHVILLE	TN
214600	1.0	0	0	True	NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	CLEAR	DARK - LIGHTED	HERMITAGE	TN

204574 rows × 9 columns

- Xuất dữ liệu ra file csv.

```
test.to_csv('data_daxuly.csv')
df2 = pd.read_csv('data_daxuly.csv')
df2
```

CHƯƠNG 3. XỬ LÝ MISSING DATA

3.1 Những kiểu missing data

3.1.1 Missing at Random - Dữ liệu khuyết ngẫu nhiên

Sự mất mát dữ liệu ở đây là ngẫu nhiên, tuy nhiên vẫn có mối quan hệ hệ thống giữa dữ liệu bị mất và dữ liệu được quan sát. Ví dụ ở hình dưới những vụ tai nạn không có phương tiện vận chuyển bị khuyết dữ liệu về điều kiện thời tiết, có nghĩa là có một mối quan hệ hệ thống giữa biến weather description và collision type description.

Collision Type Description	Weather Code	Weather Description
NOT COLLISION W/MOTOR VEHICLE-TRANSPORT		
NOT COLLISION W/MOTOR VEHICLE-TRANSPORT		
NOT COLLISION W/MOTOR VEHICLE-TRANSPORT		
REAR-TO-REAR	21	CLEAR
ANGLE	21	CLEAR
NOT COLLISION W/MOTOR VEHICLE-TRANSPORT		
ANGLE	21	CLEAR
ANGLE	21	CLEAR
NOT COLLISION W/MOTOR VEHICLE-TRANSPORT		

3.1.2 Missing Completely at Random - Dữ liệu thiếu hoàn toàn ngẫu nhiên

Sự mất mát dữ liệu ở đây là hoàn toàn ngẫu nhiên, và không có bất kỳ một mối quan hệ hay sự liên quan nào giữa dữ liệu và bất kì dữ liệu nào, missing hoặc dữ liệu quan sát. Ở ví dụ dưới đây không hề tìm thấy một mối quan hệ nào giữa giá trị bị thiếu và giá trị được giữ nguyên.

Collision Type Description	Weather Code	Weather Description
NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	21	CLEAR
NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	21	CLEAR
NOT COLLISION W/MOTOR VEHICLE-TRANSPORT		
SIDESWIPE - SAME DIRECTION		
Front to Rear		
Front to Rear	21	CLEAR
Front to Rear	21	CLEAR
Front to Rear		
ANGLE	21	CLEAR
SIDESWIPE - SAME DIRECTION	21	CLEAR
Front to Rear	21	CLEAR
Front to Rear	21	CLEAR
ANGLE	21	CLEAR
SIDESWIPE - SAME DIRECTION		
ANGLE	21	CLEAR
ANGLE		
SIDESWIPE - SAME DIRECTION		
ANGLE		
SIDESWIPE - SAME DIRECTION		

3.1.3 Missing Not at Random - Dữ liệu khuyết không ngẫu nhiên

MNAR: Sự mất mát dữ liệu không phải là ngẫu nhiên mà có một mối quan hệ xu hướng giữa giá trị bị missing và giá trị không bị missing trong một biến. Ví dụ ở hình dưới - những những vụ front to rear sẽ bị missing còn lại thì không bị thiếu, có nghĩa là có sự liên quan giữa 2 giá trị missing và không missing trong chính biến weather description.

Collision Type Description	Weather Code	Weather Description
NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	21	CLEAR
NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	21	CLEAR
NOT COLLISION W/MOTOR VEHICLE-TRANSPORT	21	CLEAR
SIDESWIPE - SAME DIRECTION	21	CLEAR
Front to Rear		
Front to Rear		
Front to Rear		
Front to Rear		
ANGLE	21	CLEAR
SIDESWIPE - SAME DIRECTION	21	CLEAR

3.2 Giải quyết vấn đề missing

3.2.1 Tìm kiếm missing data trong dataset

Trên thực tế có rất nhiều kiểu dữ liệu missing xuất hiện: có thể là một chuỗi rống, có thể là NA, N/A, Non, -1, 99 hoặc 999. Cách tốt nhất để giải quyết mising values là hiểu rõ được dataset: Hiểu được cách dữ liệu missing đang được biểu diễn, cách data được thu thập, dữ liệu bị missing thuộc trường nào, ...

Có thể loại bỏ dữ liệu missing khi chúng ta nhận ra thiếu dữ liệu hoàn toàn ngẫu nhiên (MCAR). Tuy nhiên với MAR và MNAR thì việc loại bỏ sẽ làm ảnh hưởng đến độ chính xác của mô hình.

3.2.2 Xác định giá trị missing trong data

Tìm trong dữ liệu có giá trị Null hay không

In [19]:
```
#Xử lí dữ liệu bị thiếu
total = df.isnull().sum().sort_values(ascending=False) #Tìm trong dữ liệu giá null, tính tổng chúng lại, sắp xếp theo tăng dần
percent_1=df.isnull().sum()/df.isnull().count()*100 #Tổng giá trị null của một cột/ số dòng của cột null đó
percent_2 = (round(percent_1,1)).sort_values(ascending = False) #Làm tròn 1 chữ số và sort giảm dần
missing_data = pd.concat([total,percent_2],axis=1,keys=['Total','%'])
missing_data.head(11)
```

Out[19]:

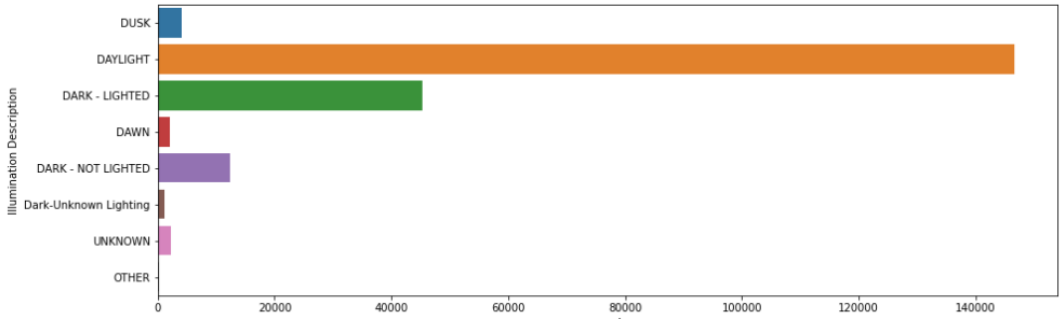
	Total	%
Weather Description	5299	2.5
Illumination Description	289	0.1
Hit and Run	17	0.0
Collision Type Description	16	0.0
Number of Motor Vehicles	1	0.0
Number of Injuries	0	0.0
Number of Fatalities	0	0.0
City	0	0.0
State	0	0.0

3.2.3 Removing Data

3.2.3.1 Listwise deletion

- Nếu dữ liệu missing trong tập dữ liệu là MCAR và số lượng missing values không nhiều, chúng ta sẽ xóa đi những giá trị missing đó. những giá trị bị thiếu ở biến Illumination Description drop nó đi. Ta thấy giá trị UNKNOWN rất bé nên ta có thể xóa những dòng có giá trị UNKNOWN đi.

```
In [46]: #Xem thuộc tính Illumination Description
plt.figure(1,figsize=(15,5))
sns.countplot(y = 'Illumination Description',data =df)
plt.show()
```



3.2.3.2 Dropping variable

Có rất nhiều trường hợp xảy ra khi thiếu data, nếu trong trường hợp một biến có nhiều giá trị bị thiếu và chúng ta có thể phán đoán rằng biến bị thiếu đó thật sự không quan trọng nếu không xuất hiện trong dữ liệu, thì có thể xóa luôn cả biến đó đi. Thông thường, khi dữ liệu của một biến bị thiếu khoảng 60 ~70 % nên xem xét đến việc loại bỏ hoàn toàn biến đó đi.

3.2.4 Data Imputation

3.2.4.1 Thay thế những dữ liệu missing với các giá trị: -1, -99, -999

Với những features có tính liên tục thì việc chúng ta thay thế những giá trị missing bằng các giá trị -1, -99, -999, ... sẽ giúp cho những mô hình cây như (RF - Random Forest) hoạt động tốt hơn bởi khi thay thế bằng những giá trị ở trên thì các mô hình này có thể giải thích cho việc thiếu dữ liệu thông qua việc encoding này. Nhược điểm của nó làm giải hiệu suất của mô hình tuyến tính sẽ bị ảnh hưởng.

3.2.4.2 Thay thế bằng giá trị mean, midian, mode

Với phương pháp này chúng ta điền những giá trị bị thiếu bằng giá trị** mean** hay median của một vài biến nếu biến liên tục và điền mode nếu biến là biến categorical. Tuy phương pháp này nhanh nhưng lại làm giảm phương sai của dữ liệu. Bên cạnh đó khi thực hiện cách này thì nó phù hợp với mô hình tuyến tính đơn giản và NN. Nhưng đối với những bài toán dựa trên tree thì có vẻ không phù hợp lắm.

CHƯƠNG 4. THUẬT TOÁN

4.1. K-Nearest Neighbours

4.1.1. Định nghĩa

K-Nearest Neighbors (K-NN) phân loại một điểm dựa trên nhóm 'k' các điểm lân cận xung quanh nó. Nó sử dụng sự tương đồng về đặc trưng để dự đoán nhóm mà điểm mới sẽ thuộc về.

4.1.2. Đặc điểm thuật toán

K-Nearest Neighbours (KNN) là một thuật toán đa dụng, nó bao gồm đặc trưng học lười biếng(Lazy learning) và phương pháp phi tham số. Nó được sử dụng cho cả phân loại và hồi quy. Học lười biếng có nghĩa là thuật toán này gần như không mất thời gian để học vì nó chỉ lưu trữ dữ liệu của phần huấn luyện và sử dụng nó để đánh giá các điểm truy vấn mới. Là một mô hình phi tham số, KNN tránh việc giả định các phân phối cụ thể mà dựa vào một tham số do người dùng cung cấp K đại diện cho số điểm được xem xét để phân tích so sánh.

4.1.3. Phép đo khoảng cách

Có nhiều phương pháp để đo khoảng cách như Hamming, Euclid, Manhattan hoặc Khoảng cách Minkowski nhưng phổ biến nhất là khoảng cách Euclid (cho dữ liệu có kích thước nhỏ) và khoảng cách Manhattan.

- Tính khoảng cách Euclid:

Khoảng cách Euclid được định nghĩa là khoảng cách giữa hai vector giá trị thực. Khoảng cách Euclid được tính là căn bậc hai của tổng bình phương hiệu giữa hai vector.

$$d_{pq} = d_{qp} = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

Trong đó p và q là hai điểm khác nhau có n chiều.

- Tính khoảng cách Manhattan:

Khoảng cách này còn được gọi là khoảng cách taxi hoặc khoảng cách dãy phố vì cách tính khoảng cách này. Khoảng cách giữa hai điểm là tổng các giá trị tuyệt đối của sự khác biệt giữa các tọa độ Cartesian của chúng.

$$d_{pq} = d_{qp} = \sum_{i=1}^{n} |q_i - p_i|$$

Trong đó p và q là hai điểm khác nhau có n chiều.

4.1.4. Các quy trình
Trong giai đoạn huấn luyện (training), mô hình sẽ chỉ lưu trữ các điểm dữ liệu.

Trong giai đoạn kiểm thử (testing), tất cả các khoảng cách từ điểm truy vấn đến các điểm khác từ giai đoạn huấn luyện được tính toán để phân loại mỗi điểm trong tập dữ liệu kiểm thử.

4.1.5. Ưu điểm và nhược điểm
Ưu điểm:

- Thuật toán đơn giản và dễ hiểu, dễ triển khai.
- Giai đoạn huấn luyện của phân loại KNN nhanh hơn nhiều so với các thuật toán phân loại khác vì không cần phải huấn luyện một mô hình để tổng quát hóa. Đây là lý do tại sao KNN được gọi là thuật toán học dựa trên ví dụ đơn giản.
- KNN cung cấp độ chính xác tương đối cao so với các thuật toán khác.
- KNN có thể hữu ích trong trường hợp dữ liệu phi tuyến tính. Nó có thể được sử dụng với bài toán hồi quy. Giá trị đầu ra cho đối tượng được tính bằng giá trị trung bình của k giá trị lân cận gần nhất.

Nhược điểm:

- Không hoạt động tốt với tập dữ liệu lớn: Trong các tập dữ liệu lớn, chi phí tính toán khoảng cách giữa điểm mới và từng điểm hiện có rất lớn, làm giảm hiệu suất của thuật toán.
- Không hoạt động tốt với dữ liệu có nhiều chiều: Thuật toán KNN không hoạt động tốt với dữ liệu có nhiều chiều vì với số chiều lớn, thuật toán gặp khó khăn trong việc tính toán khoảng cách trong mỗi chiều.
- Nhạy cảm với dữ liệu nhiễu, giá trị thiếu và giá trị ngoại lai: KNN nhạy cảm với nhiễu trong tập dữ liệu. Chúng ta cần phải điền giá trị thiếu bằng tay và loại bỏ các giá trị ngoại lai.

4.1.6. Tham số
K-Nearest Neighbors chỉ có một tham số 'K' đại diện cho số lân cận cần thiết cho mục đích tính toán. Một điều cần chú ý là nếu giá trị của K là số chẵn, nó có thể tạo ra vấn đề khi lấy đa số phiếu vì dữ liệu có số lớp chẵn. Do đó, chọn K là số lẻ khi dữ liệu có số lớp chẵn và số chẵn khi dữ liệu có số lớp lẻ.

4.1.7. Các bước thực hiện
Bước 1: Chọn một giá trị cho K. K nên là một số lẻ vì chúng ta chỉ có hai lớp 'Có' hoặc 'Không'

Bước 2: Tìm khoảng cách từ điểm mới đến mỗi điểm dữ liệu huấn luyện.

Bước 3: Đếm số lân cận gần nhất K đến điểm dữ liệu mới.

Bước 4: Đối với phân loại, đếm số điểm dữ liệu trong mỗi danh mục trong số các lân cận K. Điểm dữ liệu mới sẽ thuộc về một lớp có nhiều số phiếu. Đối với hồi quy, giá trị cho điểm dữ liệu mới sẽ là giá trị trung bình của K lân cận.

4.2. Decision Tree

Thuật toán Cây Quyết Định thuộc loại học có giám sát. Không giống như các thuật toán học có giám sát khác, thuật toán cây quyết định có thể được sử dụng để giải quyết các vấn đề hồi quy và phân loại. Mục tiêu của việc sử dụng Cây Quyết Định là tạo ra một mô hình huấn luyện có thể được sử dụng để dự đoán lớp hoặc giá trị của biến mục tiêu bằng cách học các quy tắc quyết định đơn giản được suy ra từ dữ liệu trước (dữ liệu huấn luyện).

4.2.1. Quy trình

Phát triển cây :

- Bắt đầu với toàn bộ tập dữ liệu tại nút gốc.
- Chọn thuộc tính tốt nhất sử dụng các biện pháp lựa chọn thuộc tính (ASM) như "Độ lợi thông tin" (Information Gain) hoặc "Chỉ số Gini" để xác định thuộc tính nào cung cấp sự phân chia tốt nhất cho dữ liệu.
- Chia tập dữ liệu thành các tập con dựa trên thuộc tính đã chọn.
- Lặp lại quá trình cho mỗi tập con (đệ quy) cho đến khi gặp điều kiện dừng.

Điều kiện dừng:

- Dừng chia khi đạt độ sâu xác định trước của cây.
- Dừng nếu số lượng mẫu trong một nút dưới một ngưỡng nhất định.
- Dừng nếu cải thiện điểm độ tinh khiết dưới một ngưỡng nhất định.
- Dừng nếu việc chia một nút sẽ dẫn đến cây vượt quá độ sâu tối đa.

Tạo nút lá:

- Khi gặp điều kiện dừng hoặc một tập con không còn mẫu để chia, tạo một nút lá.
- Gán nút lá một nhãn lớp (cho phân loại) hoặc một giá trị số (cho hồi quy).
- Nhãn lớp có thể được xác định bởi lớp chiếm đa số trong tập con hoặc như đã đề cập trong tuyên bố ban đầu của bạn bởi lớp phổ biến nhất trong toàn bộ tập dữ liệu.

Quá trình tiếp tục cho đến khi một trong các điều kiện dừng này được đáp ứng cho mỗi nhánh của cây. Ngoài ra, một số thuật toán cây quyết định có thể có các biến thể trong chi tiết của tiêu chí kết thúc nhưng ý tưởng cơ bản vẫn nhất quán với ba tiêu chí kết thúc trên.

4.2.2. Các độ đo lựa chọn thuộc tính

Độ lợi thông tin (Information Gain)

"Độ lợi thông tin" là một chỉ số định lượng sự giảm không chắc chắn về biến mục tiêu mang lại bởi việc chia tập dữ liệu dựa trên một thuộc tính cụ thể. Nó được tính bằng cách đo sự khác biệt về entropy trước và sau khi chia. Entropy trong ngữ cảnh này đại diện cho mức độ rối loạn hoặc không thể đoán trước trong tập dữ liệu. Một "Độ lợi thông tin" cao hơn cho thấy thuộc tính hiệu quả hơn cho việc chia.

Công thức:

Tính độ lợi thông tin ở tất cả các thuộc tính để chọn ra thuộc tính ở nút gốc:

$$G(S, A) = E(S) - \sum_{i=1}^{m} f_s(A_i) E\left(s_{A_i}\right)$$

Trong đó:

- $G(S, A)$ là độ lợi thông tin của tập S khi chia theo thuộc tính A
- $E(S)$ là độ bất định (Entropy) của tập S:

$$E(S) = -\sum_{j=1}^{n} f_s(A_j)\, log_2\left(f_s(A_j)\right)$$

- m là số giá trị khác nhau của thuộc tính A đang xét
- A_i là số mẫu tương ứng với mỗi giá trị $\dot{l}$ của thuộc tính A
- $f_s(A_i)$ là tỷ lệ của số mẫu có thuộc tính A_i với S
- S_{A_i} là một tập con của S chứa tất cả các mẫu có giá trị A_i

Chỉ số Gini (Gini Index)

Chỉ Số Gini là một chỉ số đo lường độ không tinh khiết trong một tập dữ liệu. Nó đánh giá mức độ thường xuyên mà một phần tử được chọn ngẫu nhiên sẽ bị phân loại sai. Chỉ Số Gini càng thấp, thuộc tính càng tốt trong việc phân loại dữ liệu. Chỉ Số Gini được tính bằng cách tổng các xác suất bình phương của mỗi lớp được chọn nhân với xác suất sai lầm tương ứng của chúng.

Công thức:

Chỉ số Gini của tập huấn luyện S:

$$Gini(S) = 1 - \sum_{j} p(j|s)^2$$

Với $p(j|S)$ là tần suất của lớp j trong S

Khi phân chia nút A thành k nhánh, chất lượng của phép chia được tính bằng công thức:

$$Gini_A(s) = \sum_{i=1}^{k} \frac{n_i}{n} Gini(i)$$

Trong đó:

- n_i là số mẫu trong nút i
- n là số mẫu trong nút A

4.2.3. Thuật ngữ trong Cây quyết định

- **Nút Gốc:** Đại diện cho tập dữ liệu ban đầu chia thành các tập đồng nhất.
- **Chia Đôi:** Quá trình chia một nút thành các nút con.
- **Nút Quyết Định:** Nút con kết quả từ việc chia đôi thêm, định nghĩa tiêu chí quyết định.
- **Nút Lá / Nút Cuối:** Điểm kết thúc của cây nơi không còn chia đôi.
- **Tỉa Cây:** Loại bỏ các nút con để tối ưu hóa cấu trúc cây.
- **Nhánh / Cây Con:** Phần của cây đại diện cho một chuỗi cụ thể của các nút.
- **Nút Cha và Nút Con:** Nút cha chia thành các nút con, mô tả mối quan hệ phân cấp.

4.2.4. Các tham số

```
DecisionTreeClassifier(
    criterion='gini',
    splitter='best',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    max_features=None,
```

```
    max_leaf_nodes=None
)
```

Tham số liên quan đến cấu trúc cây:

- **criterion:** Xác định các Biện Pháp Lựa Chọn Thuộc Tính (ASM). Hai tùy chọn phổ biến là “gini” và “entropy”.
- **splitter:** Xác định chiến lược chọn thuộc tính để chia tại mỗi nút. Hai tùy chọn phổ biến là “best” (chọn thuộc tính tốt nhất) và “random” (chọn một thuộc tính ngẫu nhiên).
- **max_depth:** Giới hạn độ sâu tối đa của cây quyết định. Nếu không xác định, cây sẽ phát triển cho đến khi không thể chia thêm hoặc đạt đến số mẫu tối thiểu mỗi lá. Điều này giúp giới hạn độ sâu để kiểm soát độ phức tạp của mô hình và ngăn ngừa quá khớp.
- **min_samples_split:** Số lượng mẫu tối thiểu cần thiết để thực hiện một lần chia. Nếu một nút có ít hơn số mẫu này, nó sẽ không được chia. Điều này giúp kiểm soát quá khớp bằng cách ngăn ngừa các lần chia trên các tập dữ liệu nhỏ.
- **min_samples_leaf:** Số lượng mẫu tối thiểu cần thiết để ở một nút lá. Nếu một nút lá có ít hơn số mẫu này, sẽ không thực hiện lần chia cho nút lá đó. Điều này giúp ngăn ngừa quá khớp bằng cách đảm bảo các lá có đủ dữ liệu.
- **max_features:** Số lượng thuộc tính tối đa để xem xét khi tìm kiếm sự chia tốt nhất. Điều này giúp kiểm soát độ phức tạp của mô hình và giảm thời gian huấn luyện.

Tham số liên quan đến cân bằng lớp:

- **class_weight:** Trọng số liên kết với các lớp dưới dạng {nhãn_lớp: trọng số}. Điều này được sử dụng để xử lý các lớp không cân bằng bằng cách cho trọng số nhiều hơn cho các lớp thiểu số.

Các tham số bổ sung:

- **max_leaf_nodes:** Giới hạn số lượng nút lá tối đa trong cây quyết định. Nếu không xác định, cây sẽ phát triển cho đến khi không thể chia thêm hoặc đạt đến số mẫu tối thiểu mỗi lá.
- **min_impurity_decrease:** Một nút sẽ được chia nếu việc chia này gây ra giảm độ không tinh khiết lớn hơn hoặc bằng giá trị này.

4.2.5. Ưu điểm và nhược điểm

Ưu điểm:

- **Dễ Hiểu:** Một trong những ưu điểm lớn nhất của cây quyết định là tính minh bạch của chúng. Bạn có thể dễ dàng theo dõi các nhánh để hiểu cách dự đoán được thực hiện, làm cho chúng lý tưởng để giải thích các mối quan hệ phức tạp cho các bên liên quan không có kỹ thuật.
- **Chuẩn Bị Dữ Liệu Tối Thiểu:** Không giống như một số thuật toán, cây quyết định yêu cầu chuẩn bị dữ liệu tối thiểu. Thiếu dữ liệu và việc chuẩn hóa thường không phải là vấn đề lớn.
- **Linh Hoạt:** Chúng có thể xử lý cả nhiệm vụ phân loại và hồi quy, làm cho chúng trở thành công cụ đa năng cho các ứng dụng dự đoán đa dạng.

Nhược điểm:

- **Tính Phức Tạp:** Các cây sâu hơn có thể trở nên phức tạp và khó hiểu mặc dù cấu trúc cơ bản là minh bạch.
- **Không Ổn Định:** Những thay đổi nhỏ trong dữ liệu huấn luyện có thể dẫn đến các cây quyết định khác nhau đáng kể, làm cho chúng có thể không ổn định. Các phương pháp tập hợp như rừng ngẫu nhiên có thể cải thiện tính ổn định.
- **Tầm Quan Trọng của Thuộc Tính:** Mặc dù chúng có thể xác định các thuộc tính quan trọng, việc định lượng tầm quan trọng chính xác của chúng có thể là thách thức.

4.3. Naive Bayes

4.3.1. Định nghĩa

Bộ phân loại Naive Bayes là một loại "bộ phân loại xác suất" cơ bản trong thống kê, dựa trên việc sử dụng định lý Bayes' với các giả định độc lập mạnh mẽ giữa các đặc trưng. Chúng nằm trong số các mô hình mạng Bayesian cơ bản nhất, nhưng khi kết hợp với ước lượng mật độ kernel, chúng có thể đạt được mức độ chính xác cao. Các bộ phân loại Naive Bayes cực kỳ có khả năng mở rộng, với các tham số tỷ lệ thuận với số lượng biến (đặc trưng/dự đoán) trong một nhiệm vụ học tập. Huấn luyện tối đa hợp lý có thể được thực hiện trong thời gian tuyến tính bằng cách đánh giá một biểu thức dạng đóng, trái ngược với phương pháp ước lượng lặp lại tốn kém được sử dụng cho nhiều loại bộ phân loại khác. Các mô hình Naive Bayes được biết đến với nhiều tên gọi trong lĩnh vực thống kê, bao gồm Bayes đơn giản và Bayes độc lập. Tất cả những tên gọi này đều ám chỉ việc sử dụng định lý Bayes trong quy tắc quyết định của bộ phân loại, tuy nhiên Naive Bayes không nhất thiết phải là một phương pháp Bayesian.

4.3.2. Định lý Bayes

Trước khi chuyển sang công thức cho Naive Bayes, điều quan trọng là phải biết về định lý Bayes'. Định lý Bayes' tìm xác suất xảy ra của một sự kiện với điều kiện đã biết xác suất của một sự kiện khác đã xảy ra. Định lý Bayes' được biểu diễn toán học bằng phương trình sau:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Ở đây, X và y là các sự kiện và P(X) ≠ 0.

Cơ bản, chúng ta đang cố gắng tìm xác suất của sự kiện y, với điều kiện sự kiện X là đúng. Sự kiện X cũng được gọi là bằng chứng.

P(y) là xác suất tiên nghiệm của y (xác suất trước khi thấy bằng chứng, tức là xác suất của sự kiện trước khi bằng chứng được quan sát).

P(X) là xác suất tiên nghiệm của dự báo của lớp.

P(y|X) là xác suất hậu nghiệm của y, tức là xác suất của sự kiện sau khi bằng chứng được quan sát.

P(X|y) là khả năng, tức là xác suất của dự báo của lớp đã cho.

Nói cách khác, sử dụng định lý Bayes', chúng ta có thể tìm xác suất xảy ra của y, với điều kiện rằng X đã xảy ra. Ở đây, X là bằng chứng và y là giả thuyết. Giả định được đưa ra ở đây là các đặc trưng/dự báo là độc lập với nhau. Tức là, sự hiện diện của một đặc trưng cụ thể không ảnh hưởng đến các đặc trưng khác. Do đó, nó được gọi là "naive".

4.3.3. Giả định Naive

Áp dụng giả định Naive vào định lý Bayes', tức là giả định độc lập giữa các đặc trưng. Vì vậy, chúng ta sẽ tách bằng chứng thành các phần độc lập.

Nếu hai sự kiện y và X độc lập, thì:

$$P(y, X) = P(y).P(X)$$

Bằng cách thay thế cho X và mở rộng sử dụng quy tắc chuỗi, chúng ta có:

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

Điều này có thể được biểu diễn như sau:

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$$

Chúng ta có thể thu được các giá trị cho mỗi phần bằng cách nhìn vào tập dữ liệu và thay thế chúng vào trong phương trình. Đối với tất cả các mục trong tập dữ liệu, mẫu số không thay đổi. Do đó, mẫu số có thể được loại bỏ và một tỷ lệ có thể được biểu diễn như sau.

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Chúng ta cần tạo một mô hình bộ phân loại. Để làm điều này, chúng ta tìm xác suất của một tập hợp các đầu vào cho tất cả các giá trị có thể của biến lớp y và chọn ra đầu ra có xác suất lớn nhất. Điều này có thể được biểu diễn toán học như sau:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Vậy cuối cùng, chúng ta còn lại nhiệm vụ tính toán P(y) và P(xi|y).

Lưu ý rằng P(y) cũng được gọi là xác suất lớp và P(xi|y) được gọi là xác suất điều kiện.

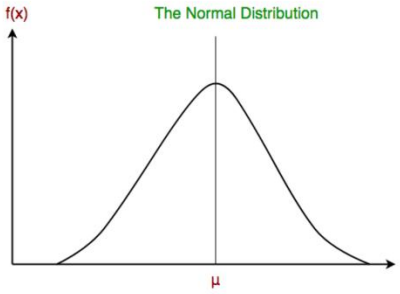
Các bộ phân loại Naive Bayes khác nhau chủ yếu bởi các giả định về phân phối của P(xi|y).

4.3.4. Gaussian Naive Bayes

Phương pháp chúng ta đã thảo luận ở trên áp dụng cho dữ liệu rời rạc. Trong trường hợp dữ liệu liên tục, chúng ta cần đưa ra một số giả định về phân phối của các giá trị của mỗi đặc trưng. Các bộ phân loại Naive Bayes khác nhau chủ yếu khác nhau dựa trên các giả định về phân phối của P(xi|y).

Bây giờ, chúng ta sẽ thảo luận về một trong các bộ phân loại này. Trong Gaussian Naive Bayes, giá trị liên tục liên quan đến mỗi đặc trưng được giả định phân phối theo phân phối Gaussian, còn được gọi là phân phối Normal. Khi vẽ biểu đồ, phân phối này tạo ra một đường cong hình chuông đối xứng với giá trị trung bình của các đặc trưng.

Xác suất điều kiện được giả định là Gaussian; do đó, xác suất có điều kiện được cho bởi:



$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

4.3.5. Đa thức Naive Bayes

Các vector đặc trưng đại diện cho tần suất mà các sự kiện nhất định được tạo ra bởi một phân phối đa thức. Đây là mô hình sự kiện thường được sử dụng cho phân loại tài liệu, để xác định xem một tài liệu thuộc về thể loại thể thao, chính trị, công nghệ, v.v.

Các đặc trưng/dự đoán được sử dụng bởi bộ phân loại là tần suất của các từ xuất hiện trong tài liệu.

4.3.6. Bernoulli Naive Bayes

Trong mô hình sự kiện Bernoulli đa biến, các đặc trưng là các biến Boolean độc lập (biến nhị phân) mô tả các đầu vào. Giống như mô hình đa thức, mô hình này cũng phổ biến cho các nhiệm vụ phân loại tài liệu, nơi mà các đặc trưng là sự xuất hiện nhị phân của các từ được sử dụng thay vì tần suất từ. Các tham số mà chúng ta sử dụng để dự đoán biến lớp chỉ có giá trị là có hoặc không, ví dụ, một từ có xuất hiện trong văn bản hay không.

4.3.7. Ưu và nhược điểm

❖ **Ưu điểm:**

- Các bộ phân loại Naive Bayes đã hoạt động rất tốt trong nhiều tình huống thực tế, nổi tiếng là phân loại tài liệu và lọc thư rác.
- Chúng yêu cầu một lượng nhỏ dữ liệu huấn luyện để ước lượng các tham số cần thiết.
- Các bộ học và bộ phân loại Naive Bayes có thể cực kỳ nhanh so với các phương pháp phức tạp hơn.
- Naive Bayes hoạt động tốt với các biến đầu vào phân loại so với các biến số. Đối với biến số, giả định phân phối là phân phối chuẩn.

❖ **Nhược điểm:**

- Yêu cầu các biến dự đoán phải độc lập. Trong hầu hết các trường hợp thực tế, các biến dự đoán thường phụ thuộc lẫn nhau, điều này làm giảm hiệu suất của bộ phân loại.
- Nếu một biến phân loại có một danh mục (trong tập dữ liệu kiểm tra) mà không xuất hiện trong tập dữ liệu huấn luyện, mô hình sẽ gán xác suất bằng 0 và không thể đưa ra dự đoán. Điều này thường được gọi là "Tần số Bằng Không" (Zero Frequency).
- Naive Bayes cũng được biết đến như là một bộ ước lượng kém, vì vậy các đầu ra xác suất từ chức năng dự đoán (predict proba) không nên được xem trọng quá mức.

4.3.8. Tham số

Trong các thuật toán Naive Bayes, Scikit-learn cung cấp một số tham số hỗ trợ xây dựng mô hình Naive Bayes.

Đối với Gaussian Naive Bayes:

- **priors:** mảng có dạng (số lớp,), mặc định là None: Tham số này đại diện cho các xác suất tiên nghiệm của các lớp. Nếu chúng ta chỉ định tham số này khi khớp dữ liệu, thì các xác suất tiên nghiệm sẽ không được điều chỉnh theo dữ liệu.
- **var smoothing:** float, mặc định là 1e-9: Tham số này đại diện cho phần của phương sai lớn nhất của các đặc trưng được thêm vào phương sai để ổn định tính toán.

Đối với Multinomial Naive Bayes:

- **alpha:** float hoặc mảng có dạng (số đặc trưng,), mặc định là 1.0: Tham số làm mịn bổ sung (Laplace/Lidstone) (đặt alpha=0 và force alpha=True, để không làm mịn).
- **force alpha:** bool, mặc định là False: Nếu False và alpha nhỏ hơn 1e-10, nó sẽ đặt alpha thành 1e-10. Nếu True, alpha sẽ không thay đổi. Điều này có thể gây ra lỗi số học nếu alpha quá gần với 0.
- **fit prior:** bool, mặc định là True: Có học các xác suất tiên nghiệm của lớp hay không. Nếu False, sẽ sử dụng xác suất tiên nghiệm đồng đều.
- **class prior:** mảng có dạng (số lớp,), mặc định là None: Xác suất tiên nghiệm của các lớp. Nếu được chỉ định, các xác suất tiên nghiệm sẽ không được điều chỉnh theo dữ liệu.

Đối với Bernoulli Naive Bayes:

- **alpha:** float hoặc mảng có dạng (số đặc trưng,), mặc định là 1.0: Tham số làm mịn bổ sung (Laplace/Lidstone) (đặt alpha=0 và force alpha=True, để không làm mịn).
- **force alpha:** bool, mặc định là False: Nếu False và alpha nhỏ hơn 1e-10, nó sẽ đặt alpha thành 1e-10. Nếu True, alpha sẽ không thay đổi. Điều này có thể gây ra lỗi số học nếu alpha quá gần với 0.
- **binarize:** float hoặc None, mặc định là 0.0: Ngưỡng để nhị phân hóa (chuyển đổi thành giá trị Boolean) của các đặc trưng mẫu. Nếu None, đầu vào được giả định là đã bao gồm các vector nhị phân.
- **fit prior:** bool, mặc định là True: Có học các xác suất tiên nghiệm của lớp hay không. Nếu False, sẽ sử dụng xác suất tiên nghiệm đồng đều.
- **class prior:** mảng có dạng (số lớp,), mặc định là None: Xác suất tiên nghiệm của các lớp. Nếu được chỉ định, các xác suất tiên nghiệm sẽ không được điều chỉnh theo dữ liệu.

4.3.9. Điều chỉnh siêu tham số

Các phương pháp điều chỉnh siêu tham số liên quan đến việc chọn các ứng viên kiến trúc mô hình tiềm năng từ tập hợp các giá trị siêu tham số có thể có. Điều này được gọi là "tìm kiếm" không gian siêu tham số để tìm ra các giá trị tốt nhất. Chúng ta phải tìm kiếm không gian siêu tham số để tìm ra các giá trị siêu tham số mang lại điểm số cao nhất. Có nhiều phương pháp để điều chỉnh siêu tham số, bao gồm tìm kiếm lưới (grid search), tìm kiếm ngẫu nhiên (random search), tối ưu hóa Bayesian (Bayesian optimization), và các phương pháp khác.

Trong dự án này, chúng ta cần xây dựng một mô hình, cụ thể là mô hình Naive Bayes, có thể giải quyết vấn đề đã đề cập với độ chính xác cao mà không bị overfitting với dữ liệu huấn luyện. Một số siêu tham số cần xem xét để đạt được điều này bao gồm:

- **var smoothing: float, mặc định là 1e-9** (cho Gaussian Naive Bayes).
- **alpha: float hoặc mảng có dạng (số đặc trưng,), mặc định là 1.0** (cho Bernoulli Naive Bayes và Multinomial Naive Bayes).

Tối ưu hóa Bayesian được sử dụng để xác định các siêu tham số "tốt nhất" hỗ trợ phát triển mô hình và ngăn chặn overfitting, nhằm tìm ra các siêu tham số cho vấn đề đã đề cập. Với phương pháp này, chúng ta chỉ cần tạo một mô hình cho mỗi sự kết hợp có thể của các giá trị siêu tham số đã cung cấp, đánh giá từng mô hình và chọn thiết kế mang lại kết quả tốt nhất.

Các mô hình Naive Bayes của Scikit-learn sẽ được xây dựng bằng cách sử dụng các siêu tham số khác nhau như sau:

Đối với Gaussian Naive Bayes:

- var smoothing: np.logspace(0, -9)

Đối với Bernoulli Naive Bayes và Multinomial Naive Bayes:

- alpha: np.linspace(0, 100)

4.3.10. Quy trình

Bộ phân loại Naive Bayes hoạt động như sau:

1. Giả sử bộ dữ liệu va chạm xe Nashville là tập dữ liệu huấn luyện kèm theo nhãn lớp. Mỗi bộ dữ liệu được biểu diễn bởi một vector phần tử n chiều.

$$X = (x_1, x_2, x_3, ..., x_n)$$

2. Xem xét rằng có m lớp y1, y2, y3, ..., ym. Giả sử chúng ta muốn phân loại một bộ dữ liệu chưa biết X, thì bộ phân loại sẽ dự đoán rằng X thuộc về lớp có xác suất hậu nghiệm cao hơn, dựa trên X. Tức là, bộ phân loại Naive Bayes gán bộ dữ liệu chưa biết X cho lớp yi nếu và chỉ nếu P(yi|X) > P(yj|X) For $1 \leqslant j \leqslant m$ và $I \neq j$. Các xác suất hậu nghiệm trên được tính toán bằng cách sử dụng định lý Bayes.

4.4. Random Forest

Random Forest là một thuật toán học có giám sát. " Forest " mà nó tạo ra là một tập hợp các cây quyết định, mỗi cây trong forest được xây dựng độc lập, sử dụng một tập hợp ngẫu nhiên các đặc trưng và các mẫu bootstrapped từ tập dữ liệu, còn được gọi là phương pháp bagging. Ý tưởng chung của phương pháp bagging là sự kết hợp của các mô hình học sẽ tăng cường kết quả tổng thể.

Quá trình xây dựng forest ngẫu nhiên:

- Tạo tập dữ liệu bootstrap: Chọn ngẫu nhiên các mẫu từ tập dữ liệu gốc. Chúng ta có thể chọn cùng một mẫu nhiều lần.
- Tạo một cây quyết định: Tạo cây sử dụng tập dữ liệu bootstrap, nhưng chỉ sử dụng một tập hợp ngẫu nhiên các biến (hoặc cột) tại mỗi bước.
- Lặp lại: Tạo một tập dữ liệu bootstrap mới và sử dụng nó để xây dựng một cây mới cho đến khi đạt đến một số cây phù hợp. Sự kết hợp của các cây này tạo thành Random Forest.
- Chúng ta có thể điều chỉnh bất kỳ đặc tính nào (số lượng mẫu, số lượng biến trong tập dữ liệu bootstrap,...) trong quá trình này để xây dựng một mô hình chính xác hơn.

4.4.1. Tham số

*Class: sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None)*

- **n_estimators:** Số lượng cây trong forest. (mặc định = 100)
- **criterion:** Hàm để đo lường chất lượng của một phép chia. {"gini", "entropy", "log_loss"}, (mặc định = "gini").
- **max_depth:** Độ sâu tối đa của cây. (mặc định = None).
- **min_samples_split:** Số lượng mẫu tối thiểu cần thiết để chia một nút nội (mặc định = 2).
- **min_samples_leaf:** Số lượng mẫu tối thiểu cần thiết để ở một nút lá (mặc định = 1).
- **min_weight_fraction_leaf:** Tỷ lệ trọng số tối thiểu của tổng trọng số của tất cả các mẫu đầu vào cần thiết để ở một nút lá (mặc định = 0.0).
- **max_features:** Số lượng tính năng được xem xét khi tìm phép chia tốt nhất (mặc định = "sqrt").
- **max_leaf_nodes:** Số lượng nút lá tối đa. Nếu None thì không giới hạn số lượng nút lá (mặc định = None).
- **min_impurity_decrease:** Một nút sẽ được chia nếu phép chia này gây ra giảm độ không thuần khiết lớn hơn hoặc bằng giá trị này.
- **bootstrap:** Xác định liệu mẫu có được bootstrap hay không khi xây dựng cây. Nếu False, toàn bộ tập dữ liệu được sử dụng để xây dựng mỗi cây (mặc định = True).

- **oob_score:** Xác định liệu có sử dụng mẫu ngoài tập huấn luyện để ước lượng điểm số tổng quát không (mặc định = False).
- **n_jobs:** Số lượng công việc chạy song song (mặc định = None).
- **random_state:** Kiểm soát cả sự ngẫu nhiên của việc bootstrapping của các mẫu được sử dụng khi xây dựng cây (nếu bootstrap=True) và việc lấy mẫu các tính năng để xem xét phép chia tốt nhất tại mỗi nút (mặc định = None).
- **verbose:** Kiểm soát tính chất chatty khi phù hợp và dự đoán (mặc định = 0).
- **warm_start:** Khi được đặt thành True, sử dụng lại giải pháp của cuộc gọi trước đó để phù hợp và thêm nhiều bộ ước lượng vào tập hợp, nếu không chỉ cần phù hợp với một rừng mới hoàn toàn (mặc định = False).
- **class_weight:** Trọng số liên kết với các lớp theo dạng {nhãn_lớp: trọng_số}. Nếu không được cung cấp, tất cả các lớp được giả định có trọng số là một (mặc định = None).
- **ccp_alpha:** Tham số phức tạp được sử dụng cho Cắt tỉa Chi phí Tối ưu Tối thiểu (mặc định = 0.0).
- **max_samples:** Nếu bootstrap là True, số lượng mẫu để lấy từ X để huấn luyện mỗi bộ ước lượng cơ sở (mặc định = None).

4.4.2. Ưu điểm

- Random Forest có khả năng học các ranh giới quyết định phi tuyến tính: Random Forest là một thuật toán học tập tập hợp sử dụng nhiều cây quyết định để đưa ra dự đoán. Nó có thể mô hình các mối quan hệ phức tạp, phi tuyến tính giữa các đặc trưng và biến mục tiêu.
- Linh hoạt và mạnh mẽ: Random Forest có thể xử lý nhiều loại dữ liệu khác nhau bao gồm dữ liệu số và phân loại. Nó có thể xử lý các giá trị ngoại lệ và giá trị thiếu, và không đòi hỏi phải tỉ mỉ tính đặc trưng vì nó sử dụng một phương pháp dựa trên quy tắc thay vì tính toán khoảng cách.
- Quan trọng của đặc trưng: Random Forest cung cấp thông tin về sự quan trọng của mỗi đặc trưng trong dữ liệu, điều này có thể rất hữu ích trong việc hiểu các mẫu cơ bản.
- Khả năng mở rộng: Random Forest có thể xử lý các bộ dữ liệu lớn với số chiều cao, khiến nó trở thành lựa chọn phổ biến trong nhiều ngành công nghiệp.
- Xử lý song song: Các cây có thể được tạo ra song song, vì không có sự phụ thuộc giữa các lần lặp, điều này làm tăng tốc độ huấn luyện.
- Độ chính xác cao: Nó giúp giảm các vấn đề quá mức và giúp cải thiện độ chính xác. Nó giảm phương sai dự đoán so với cây quyết định đơn lẻ.

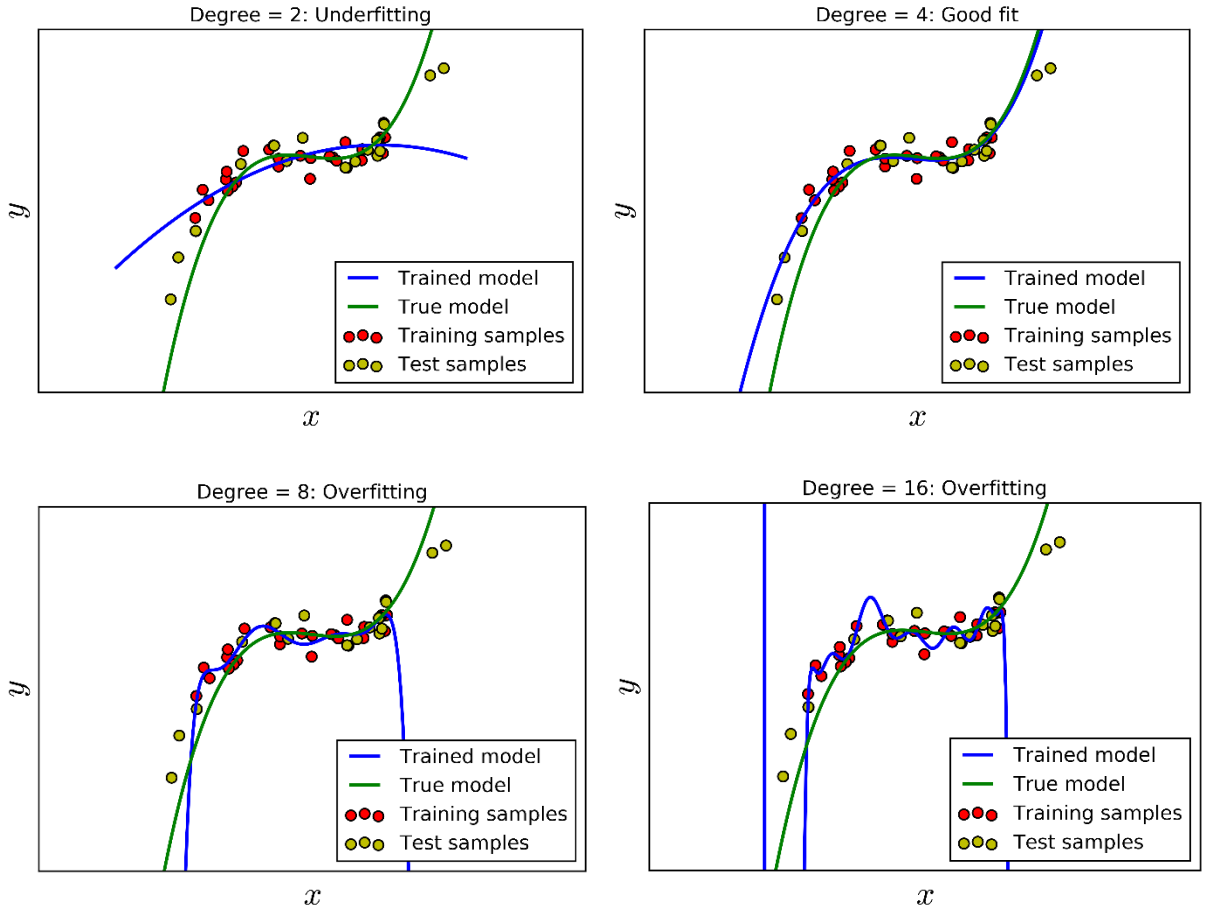
4.4.3. Nhược điểm

- Khả năng giải thích: Ít có thể giải thích hơn so với một cây quyết định đơn lẻ, vì dự đoán không thể được giải thích chỉ bằng một biểu đồ duy nhất. Tuy nhiên, sự quan trọng của biến vẫn có thể được trích xuất.
- Độ phức tạp tính toán: Random Forest có thể tốn nhiều thời gian tính toán, đặc biệt là khi làm việc với các bộ dữ liệu lớn. Nó đòi hỏi nhiều bộ nhớ, điều này có thể là một ràng buộc khi làm việc với tài nguyên hạn chế.
- Nhạy cảm với nhiễu: Mặc dù Random Forest thường khá mạnh mẽ trong việc ngăn chặn vượt mức, nhưng vẫn có thể gặp phải tình trạng này trong một số trường hợp, đặc biệt là khi làm việc với dữ liệu có độ nhiễu cao.

CHƯƠNG 5. Overfitting và Đánh giá hệ thống phân lớp

5.1 Overfitting

- Overfitting là hiện tượng mô hình tìm được quá khớp với dữ liệu training. Việc quá khớp này có thể dẫn đến việc dự đoán nhầm nhiễu, và chất lượng mô hình không còn tốt trên dữ liệu test nữa. Dữ liệu test được giả sử là không được biết trước, và không được sử dụng để xây dựng các mô hình Machine Learning.
- Về cơ bản, overfitting xảy ra khi mô hình quá phức tạp để mô phỏng training data. Điều này đặc biệt xảy ra khi lượng dữ liệu training quá nhỏ trong khi độ phức tạp của mô hình quá cao.
- Sự thật là nếu một mô hình quá fit với dữ liệu thì nó sẽ gây phản tác dụng. Hình dưới là ví dụ về 50 điểm dữ liệu được tạo bằng một đa thức bậc ba cộng thêm các dữ liệu sai lệch hoàn toàn so với bộ mẫu. Tập dữ liệu này được chia làm hai, 30 điểm dữ liệu màu đỏ cho training data, 20 điểm dữ liệu màu vàng cho test data. Đồ thị của đa thức bậc ba này được cho bởi đường màu xanh lục. Bài toán là giả sử ta không biết mô hình ban đầu mà chỉ biết các điểm dữ liệu, hãy tìm một mô hình "tốt" để mô tả dữ liệu đã cho.



- Theo đồ thị các hình trên ta có thể thấy được khi sử dụng đa thức bậc cao để đồ thị càng khớp với mẫu trainning thì lại càng sai lệch với các mẫu test và cuối cùng thì ta tìm được đa thức bậc 4 mới là mô hình khớp với mô hình ban đầu nhất

5.1.1 Các đại lượng đánh giá chất lượng của mô hình

5.1.1.1 Train error

- Thường là hàm mất mát áp dụng lên training data. Hàm mất mát này cần có một thừa số $\frac{1}{Ntrain}$ để tính giá trị trung bình, tức mất mát trung bình trên mỗi điểm dữ liệu. Với Regression, đại lượng này thường được định nghĩa:

$$train\ error\ =\ \frac{1}{Ntrain}\sum_{training\ set}\ \|\ y-y\hat{}\ \|_p^2$$

- với p thường bằng 1 hoặc 2.
- Với Classification, trung bình cộng của cross entropy có thể được sử dụng.

5.1.1.2 Test error

- Tương tự như train error nhưng áp dụng mô hình tìm được vào test data. Chú ý rằng, khi xây dựng mô hình, ta không được sử dụng thông tin trong tập dữ liệu test. Dữ liệu test chỉ được dùng để đánh giá mô hình. Với Regression, đại lượng này thường được định nghĩa:

$$test\ error\ =\ \frac{1}{Ntest}\sum_{test\ set}\ \|\ y-y\hat{}\ \|_p^2$$

- với p giống như p trong cách tính train error phía trên.
- Việc lấy trung bình là quan trọng vì lượng dữ liệu trong hai tập hợp training và test có thể chênh lệch rất nhiều.
- Một mô hình được coi là tốt (fit) nếu cả train error và test error đều thấp. Nếu train error thấp nhưng test error cao, ta nói mô hình bị overfitting. Nếu train error cao và test error cao, ta nói mô hình bị underfitting. Nếu train error cao nhưng test error thấp, tôi không biết tên của mô hình này, vì cực kỳ may mắn thì hiện tượng này mới xảy ra, hoặc có chỉ khi tập dữ liệu test quá nhỏ.

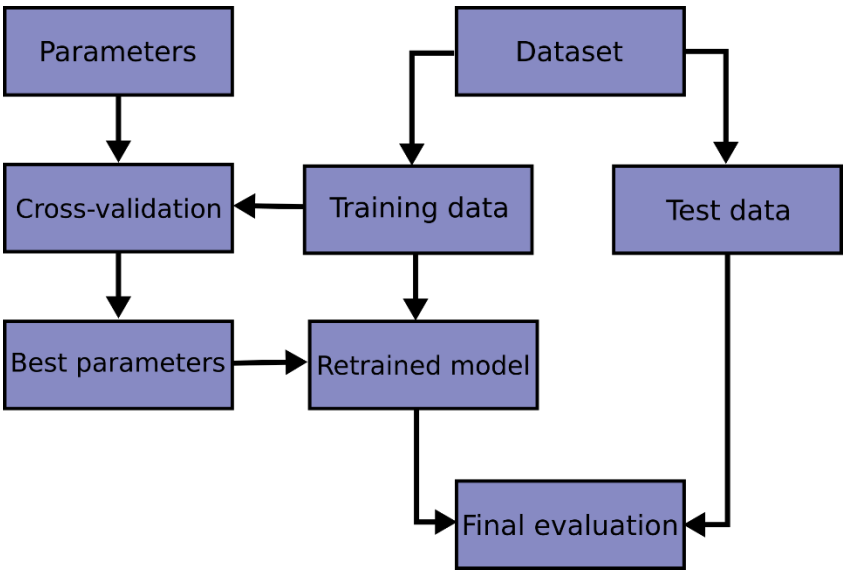
5.1.2 Các phương pháp tránh overfitting

5.1.2.1 Validation

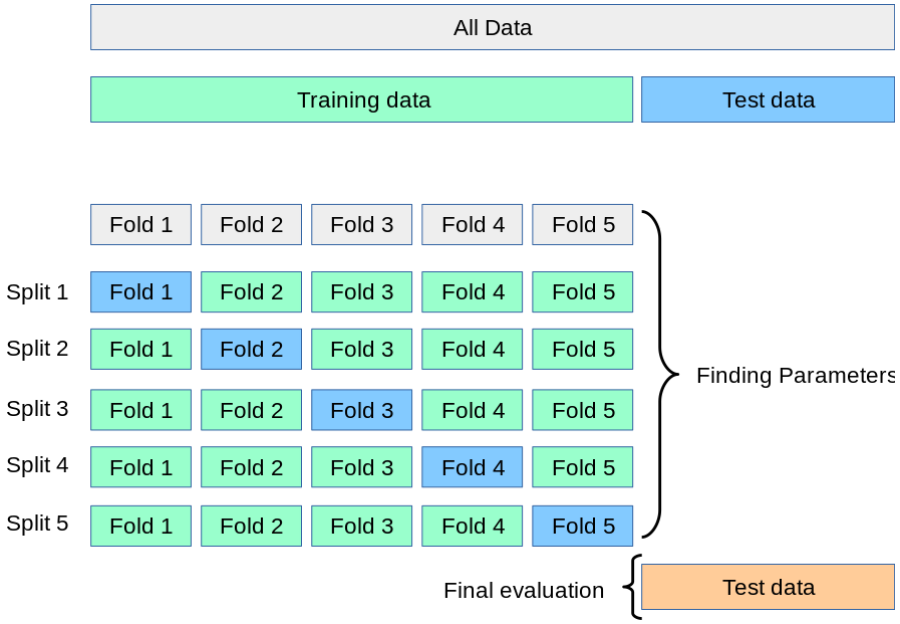
- Phương pháp đơn giản nhất là trích từ tập training data ra một tập con nhỏ và thực hiện việc đánh giá mô hình trên tập con nhỏ này. Tập con nhỏ được trích ra từ training set này được gọi là validation set. Lúc này, training set là phần còn lại của training set ban đầu. Train error được tính trên training set mới này, và có một khái niệm nữa được định nghĩa tương tự như trên validation error, tức error được tính trên tập validation.
- Với khái niệm mới này, ta tìm mô hình sao cho cả train error và validation error đều nhỏ, qua đó có thể dự đoán được rằng test error cũng nhỏ. Phương pháp thường được sử dụng là sử dụng nhiều mô hình khác nhau. Mô hình nào cho validation error nhỏ nhất sẽ là mô hình tốt.
- Thông thường, ta bắt đầu từ mô hình đơn giản, sau đó tăng dần độ phức tạp của mô hình. Tới khi nào validation error có chiều hướng tăng lên thì chọn mô hình ngay trước đó. Chú ý rằng mô hình càng phức tạp, train error có xu hướng càng nhỏ đi.

5.1.2.2 Cross-validation

- Trong nhiều trường hợp, chúng ta có rất hạn chế số lượng dữ liệu để xây dựng mô hình. Nếu lấy quá nhiều dữ liệu trong tập training ra làm dữ liệu validation, phần dữ liệu còn lại của tập training là không đủ để xây dựng mô hình. Lúc này, tập validation phải thật nhỏ để giữ được lượng dữ liệu cho training đủ lớn. Tuy nhiên, một vấn đề khác nảy sinh. Khi tập validation quá nhỏ, hiện tượng overfitting lại có thể xảy ra với tập training còn lại.
- Cross validation là một cải tiến của validation với lượng dữ liệu trong tập validation là nhỏ nhưng chất lượng mô hình được đánh giá trên nhiều tập validation khác nhau. Một cách thường đường sử dụng là chia tập training ra k tập con không có phần tử chung, có kích thước gần bằng nhau. Tại mỗi lần kiểm thử , được gọi là run, một trong số tập con được lấy ra làm validate set. Mô hình sẽ được xây dựng dựa vào hợp của $k - 1$ tập con còn lại. Mô hình cuối được xác định dựa trên trung bình của các train error và validation error. Cách làm này còn có tên gọi là k-fold cross validation.
- Khi k bằng với số lượng phần tử trong tập training ban đầu, tức mỗi tập con có đúng 1 phần tử, ta gọi kỹ thuật này là leave-one-out.



- Thước đo hiệu suất được báo cáo bởi xác thực chéo k -fold khi đó là giá trị trung bình của các giá trị được tính toán trong vòng lặp. Cách tiếp cận này có thể tốn kém về mặt tính toán nhưng không lãng phí quá nhiều dữ liệu (như trường hợp sửa một bộ xác thực tùy ý), đây là một lợi thế lớn trong các vấn đề như suy luận nghịch đảo trong đó số lượng mẫu rất nhỏ.



5.2 Đánh giá hệ thống phân lớp

5.2.1 Accuracy

- Cách đơn giản và hay được sử dụng nhất là accuracy (độ chính xác). Cách đánh giá này đơn giản tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử.
- Giả sử, ta có thể đếm được có 6 điểm dữ liệu được dự đoán đúng trên tổng số 10 điểm. Vậy ta kết luận độ chính xác của mô hình là 0.6 (hay 60%). Để ý rằng đây là bài toán với chỉ 3 class, nên độ chính xác nhỏ nhất đã là khoảng 1/3, khi tất cả các điểm được dự đoán là thuộc vào một class nào đó.

5.2.2 Confusion matrix

- Cách tính sử dụng accuracy như ở trên chỉ cho chúng ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, và dữ liệu thuộc lớp nào thường bị phân loại nhầm vào lớp khác. Để có thể đánh giá được các giá trị này, chúng ta sử dụng một ma trận được gọi là confusion matrix.

Total: 10	Predicted as: 0	Predicted as: 1	Predicted as: 2	
True: 0	2	1	1	4
True: 1	1	2	0	3
True: 2	0	1	2	3

- Confusion matrix là một ma trận vuông với kích thước mỗi chiều bằng số lượng lớp dữ liệu. Giá trị tại hàng thứ i, cột thứ j là số lượng điểm lẽ ra thuộc vào class i nhưng lại được dự đoán là thuộc vào class j. Như vậy, nhìn vào hàng thứ nhất (0), ta có thể thấy được rằng trong số bốn điểm thực sự thuộc lớp 0, chỉ có hai điểm được phân loại đúng, hai điểm còn lại bị phân loại nhầm vào lớp 1 và lớp 2.

5.2.3 True/False Positive/Negative

Cách đánh giá này thường được áp dụng cho các bài toán phân lớp có hai lớp dữ liệu. Cụ thể hơn, trong hai lớp dữ liệu này có một lớp nghiêm trọng hơn lớp kia và cần được dự đoán chính xác. Ví dụ, trong bài toán xác định có bệnh ung thư hay không thì việc không bị sót (miss) quan trọng hơn là việc chẩn đoán nhầm âm tính thành dương tính. Trong bài toán xác định có mìn dưới lòng đất hay không thì việc bỏ sót nghiêm trọng hơn việc báo động nhầm rất nhiều. Hay trong bài toán lọc email rác thì việc cho nhầm email quan trọng vào thùng rác nghiêm trọng hơn việc xác định một email rác là email thường.

- TP (True Positive): Số lượng dự đoán chính xác. Là khi mô hình dự đoán đúng một người bị ung thư.
- TN (True Negative): Số lương dự đoán chính xác một cách gián tiếp. Là khi mô hình dự đoán đúng một người không bị ung thư, tức là việc không chọn trường hợp bị ung thư là chính xác.
- FP (False Positive - Type 1 Error): Số lượng các dự đoán sai lệch. Là khi mô hình dự đoán một người bị ung thư và người đó hoàn toàn khỏe mạnh.
- FN (False Negative - Type 2 Error): Số lượng các dự đoán sai lệch một cách gián tiếp. Là khi mô hình dự đoán một người không bị ung thư nhưng người đó bị ung thư, tức là việc không chọn trường hợp bị ung thư là sai.

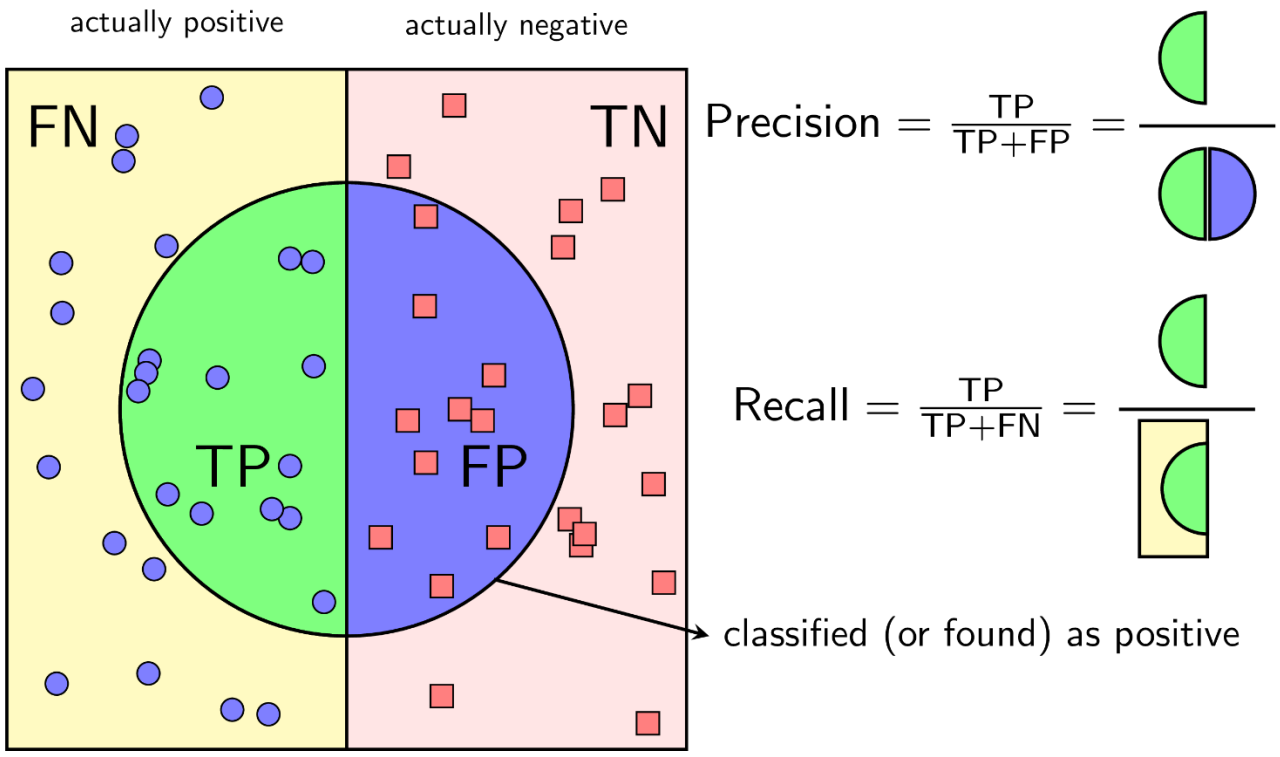
Có thể định nghĩa True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN) dựa trên confusion matrix chưa chuẩn hoá:

	Predicted as Positive	Predicted as Negative
Actual: Positive	True Positive (TP)	False Negative (FN)
Actual: Negative	False Positive (FP)	True Negative (TN)

Chúng ta sẽ quy về các tỉ lệ TPR, FNR, FPR, TNR (R - Rate) dựa trên normalized confusion matrix:

	Predicted as Positive	Predicted as Negative
Actual: Positive	TPR = TP/(TP + FN)	FNR = FN/(TP + FN)
Actual: Negative	FPR = FP/(FP + TN)	TNR = TN/(FP + TN)

5.2.4 Precision và Recall



- Với một cách xác định một lớp là positive, Precision được định nghĩa là tỉ lệ số điểm true positive trong số những điểm được phân loại là positive (TP + FP).
- Recall được định nghĩa là tỉ lệ số điểm true positive trong số những điểm thực sự là positive (TP + FN).
- Một cách toán học, Precison và Recall là hai phân số có tử số bằng nhau nhưng mẫu số khác nhau:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

- TPR và Recall là hai đại lượng bằng nhau. Ngoài ra, cả Precision và Recall đều là các số không âm nhỏ hơn hoặc bằng một.
- Precision cao đồng nghĩa với việc độ chính xác của các điểm tìm được là cao. Recall cao đồng nghĩa với việc True Positive Rate cao, tức tỉ lệ bỏ sót các điểm thực sự positive là thấp.
- Khi Precision = 1, mọi điểm tìm được đều thực sự là positive, tức không có điểm negative nào lẫn vào kết quả. Tuy nhiên, Precision = 1 không đảm bảo mô hình là tốt, vì câu hỏi đặt ra là liệu mô hình đã tìm được tất cả các điểm positive hay chưa. Nếu một mô hình chỉ tìm được đúng một điểm positive mà nó chắc chắn nhất thì ta không thể gọi nó là một mô hình tốt.
- Khi Recall = 1, mọi điểm positive đều được tìm thấy. Tuy nhiên, đại lượng này lại không đo liệu có bao nhiêu điểm negative bị lẫn trong đó. Nếu mô hình phân loại mọi điểm là positive thì chắc chắn Recall = 1, tuy nhiên dễ nhận ra đây là một mô hình cực tồi.
- Một mô hình phân lớp tốt là mô hình có cả Precision và Recall đều cao, tức càng gần một càng tốt. Có hai cách đo chất lượng của bộ phân lớp dựa vào Precision và Reall: Precision-Recall curve và F-score.

5.2.5 F1-score

F1-score là harmonic mean của precision và recall (giả sử rằng hai đại lượng này khác không):

$$\frac{2}{F_1} = \frac{1}{\text{precision}} + \frac{1}{\text{recall}} \text{ hay } F_1 = 2\frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2\frac{\text{precion} \cdot recall}{\text{precision} + \text{recall}}$$

F1-score có giá trị nằm trong nửa khoảng (0,1]. F1 càng cao, bộ phân lớp càng tốt. Khi cả recall và precision đều bằng 1 (tốt nhất có thể), F1=1. Khi cả recall và precision đều thấp, ví dụ bằng 0.1, F1=0.1.

precision	recall	F_1

1	1	1
0.1	0.1	0.1
0.5	0.5	0.5
1	0.1	0.182
0.3	0.8	0.36

Như vậy, một bộ phân lớp với precision = recall = 0.5 tốt hơn một bộ phân lớp khác với precision = 0.3, recall = 0.8 theo cách đo này.

Trường hợp tổng quát của F1-score là Fβ-score:

$$F_\beta = (1 + \beta^2) \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

F1 chính là một trường hợp đặc biệt của Fβ khi β=1. Khi β>1, recall được coi trọng hơn precision, khi β<1, precision được coi trọng hơn. Hai đại lượng β thường được sử dụng là β=2 và β=0.5.

CHƯƠNG 6. THỰC HIỆN BÀI TOÁN

Algorithms	**F1-Score (%)**
	Normal
Decision Tree (DT)	79
Random Forest (RF)	79
Naive Bayes (GNB)	23
K-Nearest Neighbors (KNN)	72

Các thuật toán được đánh giá bao gồm Decision Tree (DT), Random Forest (RF), Naive Bayes (GNB) và K-Nearest Neighbors (KNN).

Ở trạng thái bình thường, thuật toán có F1-score cao nhất là Random Forest (79%) và DT (79%), KNN (72%), NB (23%). Những điểm số này cung cấp dấu hiệu ban đầu về hiệu suất của thuật toán.

Decision Tree và Random Forest, hai thuật toán dựa trên cây quyết định thể hiện hiệu suất mạnh mẽ. NB đối với tập dữ liệu này thật sự rất thấp chứng tỏ một phần thể hiện sự

kém liên quan của các thuộc tính trong tập dữ liệu này. Tóm lại, việc đánh giá F1-score cung cấp cái nhìn toàn diện về hiệu suất của từng thuật toán, trong đó Decision Tree và Random Forest thể hiện hiệu suất mạnh mẽ mà không cần điều chỉnh.

CHƯƠNG 7. TÀI LIỆU THAM KHẢO

[1] https://machinelearningcoban.com/2017/03/04/overfitting/

[2] https://viblo.asia/p/xu-ly-missing-data-trong-data-analysis-maGK7qaAlj2

[3] https://machinelearningcoban.com/2017/01/08/knn/

[4] https://machinelearningcoban.com/2017/08/31/evaluation/

[5] https://machinelearningcoban.com/2017/08/08/nbc/

[6]