

**VIETNAM NATIONAL UNIVERSITY
UNIVERSITY OF INFORMATION TECHNOLOGY
INFORMATION SYSTEMS FACULTY**



DATA MINING

**REPORT: PREDICTING THE VICTIM'S ABILITY TO SURVIVE
BASED ON THE USE OF AIRBAGS AND OTHER FACTORS IN
TRAFFIC ACCIDENTS.**

Teacher's Guide: PhD. Cao Thị Nhạn

PhD. Vũ Minh Sang

Class: IS252.O22.HTCL

Group 10: Nguyễn Dương Chí Tâm – 21520439

Huỳnh Mạnh Huy – 21520259

Đỗ Hiền Thảo – 21520460

Nguyễn Trương Đình Giang - 21520215

Ho Chi Minh City, June 2024

Table of Contents

TEACHER’S COMMENTS	4
CHAPTER 1. IDENTIFICATION OF DATA MINING PROBLEMS.....	6
1. Overview Data	6
2. Attribute Description	6
3. State Problem	7
4. Data Mining Tools	7
5. Common Libraries	7
CHAPTER 2. DATA PREPROCESSING.....	8
1. Data Description	8
2. Data Cleaning	9
2.1 Handle unnecessary attributes	10
2.2 Handle noisy data.....	10
2.3 Handle missing data	11
2.4 Handle duplicate data.....	11
2.5 Data binning	12
2.6 Export processed files.....	13
CHAPTER 3. APPLYING CLASSIFICATION ALGORITHMNS TO THE DATASET	13
1. Process the data set before running the algorithm.....	13
2. Decision Tree ID3 Model.....	20
2.1 Definition.....	20
2.2 Advantage.....	20
2.3 Decision Tree ID3 algorithm	21
3. Decision Tree Cart Model	22
3.1 Definition.....	22
3.2 Advantage.....	22
3.3 Decision Tree Cart algorithm	23
4. Naïve Bayes Model.....	24
4.1 Definition.....	24
4.2 Advantage.....	24
4.3 Naïve Bayes algorithm.....	24
5. Random Forest Model	26
5.1 Definition.....	26

5.2	Advantage.....	26
5.3	Random Forest algorithm.....	26
6.	Bagging Classifier Model.....	28
6.1	Definition.....	28
6.2	Advantage.....	28
6.3	Bagging Classifier algorithm.....	28
7.	K – Nearest Neighbor Model.....	29
7.1	Definition.....	29
7.2	Advantage.....	30
7.3	KNN algorithm	30
8.	Logistic Regression Model	31
8.1	Definition.....	31
8.2	Advantage.....	31
8.3	Logistic Regression algorithm.....	31
9.	Support Vector Machine Model	32
9.1	Definition.....	32
9.2	Advantage.....	33
9.3	SVM algorithm	33
CHAPTER 4. EVALUATE ALGORITHMNS AND FORECASTS.....		35
1.	Runtime Evaluation	35
2.	Compare the accuracy of algorithms	36
3.	Advantages and disadvantages of the algorithm	36
3.1	Decision tree	36
3.2	Random Forest	37
3.3	Bagging Classifier	37
3.4	KNN	37
3.5	Logistic Regression.....	38
3.6	Naïve Bayes.....	38
3.7	Support Vector Machine.....	38
4.	Development	39
CHAPTER 5. ASSIGNMENT TABLE		39
CHAPTER 6. REFERENCES		40

TEACHER'S COMMENTS

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

THANK YOU

First of all, our group are infinitely grateful to University Of Information Technology for providing us with the excellent learning environment that facilitates our research and analysis.

Specially, our team would like to express our deepest gratitude to PhD. Cao Thi Nhan and PhD. Vu Minh Sang, our course instructors, whose guidance, encouragement, and valuable insights helped us successful completion of this project. Your support and expertise have significantly enhanced our understanding and application of “PREDICTING THE VICTIM'S ABILITY TO SURVIVE BASED ON THE USE OF AIRBAGS AND OTHER FACTORS IN TRAFFIC ACCIDENTS”. We would like to express our sincere thanks to our colleagues who have studied, supported, and researched together during the implementation of this project.

Even though we have made the most of what we have learned, it is still difficult to avoid mistakes. Therefore, my group is looking forward to receiving feedback from the teacher to be able to improve in the best way possible. Thereby also accumulating and learning experience to prepare for the future.

Best regards.

CHAPTER 1. IDENTIFICATION OF DATA MINING PROBLEMS

1. Overview Data

- US data, for 1997-2002, from police-reported car crashes in which there is a harmful event (people or property), and from which at least one vehicle was towed. Data are restricted to front-seat occupants, include only a subset of the variables recorded, and are restricted in other ways also.

- **Data sources :**

<https://vincentarelbundock.github.io/Rdatasets/doc/DAAG/nassCDS.html>

<https://vincentarelbundock.github.io/Rdatasets/csv/DAAG/nassCDS.csv>

- A data frame with 26217 observations on the following 16 variables.

2. Attribute Description

No.	Name	Meaning	Type
1	rownames	Row ID	Numeric
2	dvcat	Impact speed.	Categorical
3	weight	Observation weights.	Numeric
4	dead	Whether the victim live or die.	Categorical
5	airbag	Whether the individual is wearing an airbag or not.	Categorical
6	seatbelt	Whether the individual is wearing a seat belt or not.	Categorical
7	frontal	Whether the accident was caused by a direct attack. Two values: 1. 0 : non - frontal 2. 1 : frontal impact	Numeric
8	sex	Gender of the individuals involved in the accident. Two values: 1. f : female 2. m : male	Categorical

9	ageOFocc	Age of the individuals involved in the accident.	Numeric
10	yearacc	Year of accident	Numeric
11	yearVeh	Year of model of vehicle	Numeric
12	abcat	The airbag was deployed or not deployed or unavailed.	Categorical
13	occRole	Is that person the driver or the passenger.	Categorical
14	deploy	Whether an airbag was deployed. Two values: 1. 0 : if an airbag was unavailable or did not deploy 2. 1 : if one or more airbags deployed.	Numeric
15	injSeverity	Severity of the accident 0 : none, 1 : possible injury, 2 : no incapacity, 3 : incapacity, 4 : killed; 5 : unknown, 6 : death	Numeric
16	caseid	The case ID used to identify the vehicle.	String

3. State Problem

Based on the properties of the data set, predict the victim's ability to survive through the use of airbags and other factors in traffic accidents. The purpose is to determine the effectiveness of airbags in supporting the victim's ability to survive.

4. Data Mining Tools

- Data exploration tools:
 - Google Colab
 - Jupyter Notebook.
- Python version: 3.8.

5. Common Libraries

```
import numpy as np
```

```
import pandas as pd
import seaborn as sns
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style
import time
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn import tree
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn.ensemble import BaggingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
```

CHAPTER 2. DATA PREPROCESSING

1. Data Description

- Quantitative statistics: count number of values, maximum value, minimum value, mean, standard deviation and quartiles (first, second, third).

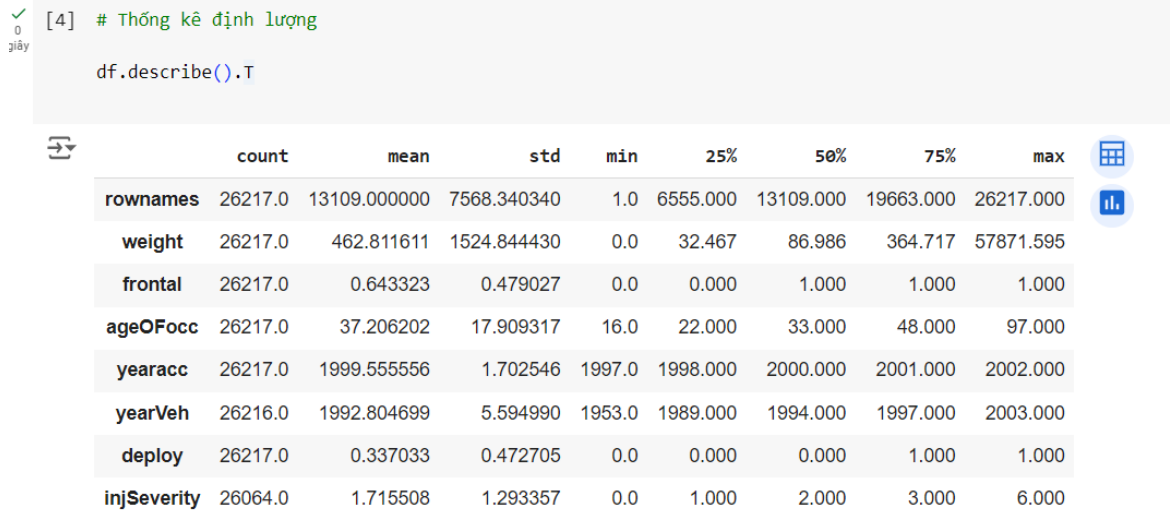


Figure 1.1 Quantitative statistics

- Qualitative statistics: count number of values, the number of different values in each attribute, the value with the highest frequency of occurrence of the attribute and the frequency of that value.

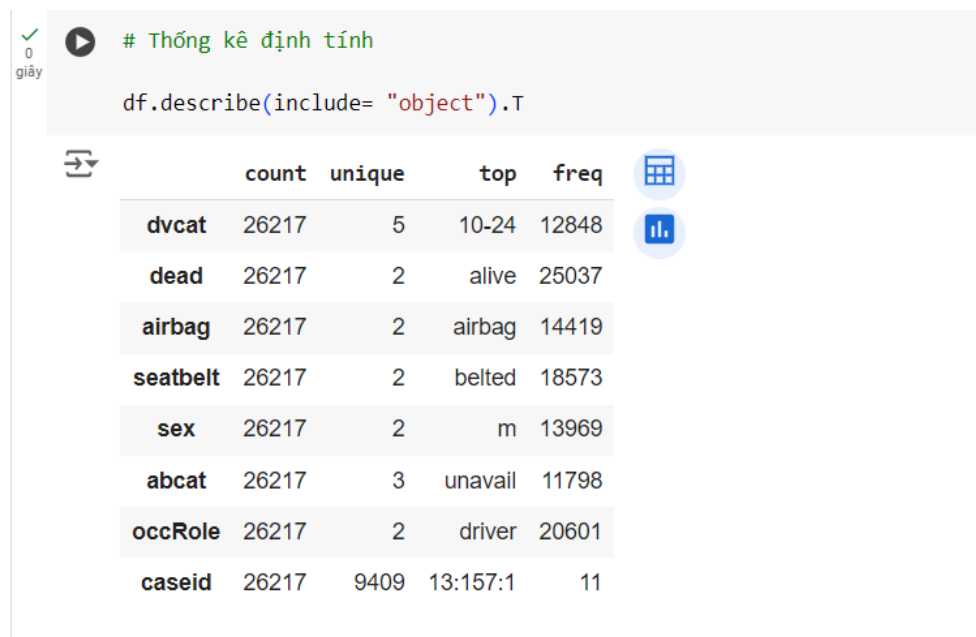


Figure 1.2 Qualitative statistics

2. Data Cleaning

2.1 Handle unnecessary attributes

Handling unnecessary attributes in data mining helps:

- Reduce model complexity: Make the model simpler and easier to understand.
- Improve model performance: Remove noise and increase accuracy.
- Reduce the risk of overfitting: Avoid overfitting the model to the training data.
- Save resources: Reduce memory and computational resources.
- Increased model interpretability: Easier to understand and present results.

✓
0 giây

#2.2.1 xử lý các thuộc tính không khai thác

```
df=df.drop(['rownames'],axis = 1)
df=df.drop(['caseid'],axis = 1)
df=df.drop(['yearVeh'],axis = 1)
df=df.drop(['yearacc'],axis = 1)
df=df.drop(['abcat'],axis = 1)
df=df.drop(['injSeverity'],axis = 1)
df=df.drop(['weight'],axis = 1)
df
```


↔

	dvcat	dead	airbag	seatbelt	frontal	sex	ageOFocc	occRole	deploy
0	25-39	alive	none	belted	1	f	26	driver	0
1	10-24	alive	airbag	belted	1	f	72	driver	1
2	10-24	alive	none	none	1	f	69	driver	0
3	25-39	alive	airbag	belted	1	f	53	driver	1
4	25-39	alive	none	belted	1	f	32	driver	0
...
26212	25-39	alive	none	belted	1	m	17	driver	0
26213	10-24	alive	airbag	belted	1	m	54	driver	0
26214	10-24	alive	airbag	belted	1	f	27	driver	1
26215	25-39	alive	airbag	belted	1	f	18	driver	1
26216	25-39	alive	airbag	belted	1	m	17	pass	1

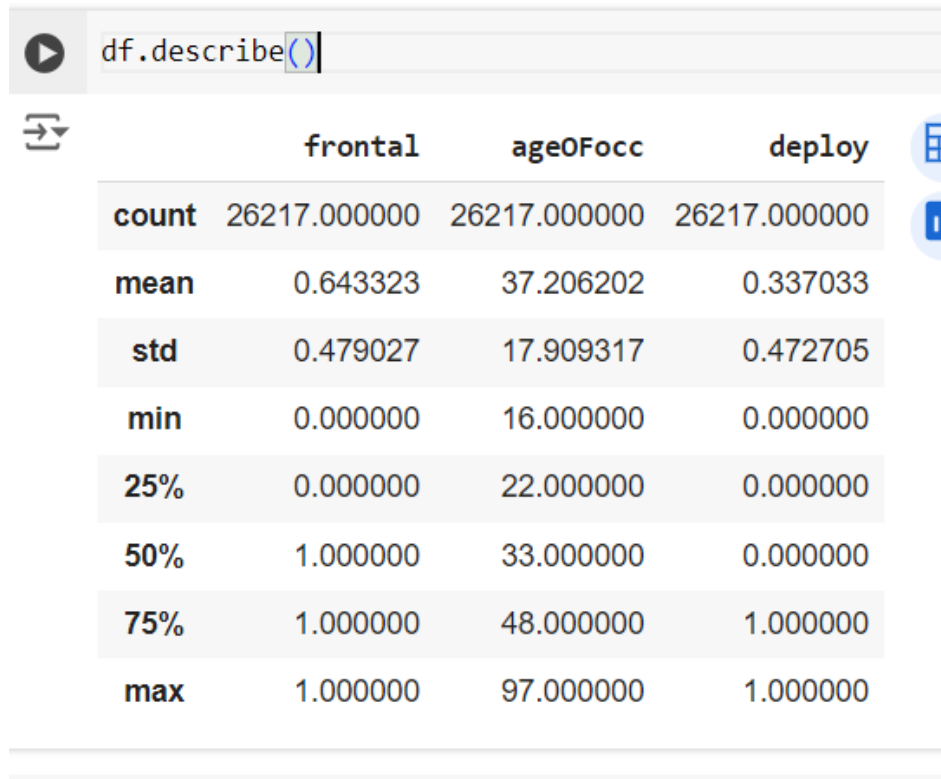
26217 rows × 9 columns

Figure 2.1.1 Use the “drop” function to drop unnecessary attributes

2.2 Handle noisy data

- Using the Winsorization method to eliminate existing outliers in each attribute by replacing A% of the lowest values in the array with the nearest adjacent value and B% of the highest values in the array with the nearest adjacent value.
- In the case of this dataset, we can observe that only the column "ageOFocc" might lead to noise, but examining the highest and lowest values of "ageOFocc", we find that these

values are not sufficiently high to cause noise in this problem's dataset. Therefore, in this case, we do not need to handle noisy data.



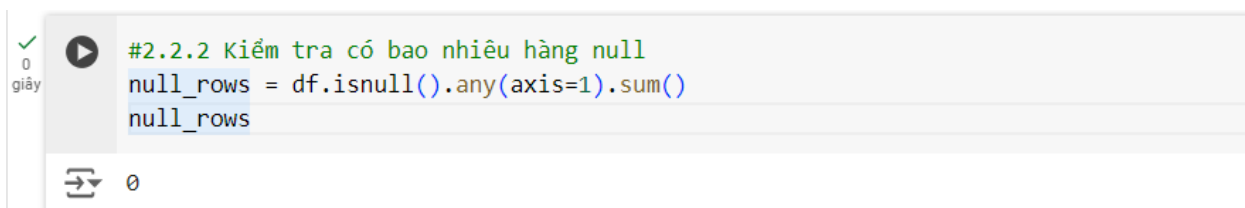
A screenshot of a Jupyter Notebook cell. The code `df.describe()` is entered in the input area. Below the code, the output is displayed as a table with statistics for three columns: `frontal`, `ageOfocc`, and `deploy`. The statistics include count, mean, std, min, 25%, 50%, 75%, and max.

	frontal	ageOfocc	deploy
count	26217.000000	26217.000000	26217.000000
mean	0.643323	37.206202	0.337033
std	0.479027	17.909317	0.472705
min	0.000000	16.000000	0.000000
25%	0.000000	22.000000	0.000000
50%	1.000000	33.000000	0.000000
75%	1.000000	48.000000	1.000000
max	1.000000	97.000000	1.000000

Figure 2.2.1 Handle noisy data

2.3 Handle missing data

To check for missing data, we use the “isnull” function to check by row (axis=1).



A screenshot of a Jupyter Notebook cell. The code `#2.2.2 Kiểm tra có bao nhiêu hàng null` is followed by `null_rows = df.isnull().any(axis=1).sum()` and `null_rows`. The output below the code is `0`.

```
#2.2.2 Kiểm tra có bao nhiêu hàng null
null_rows = df.isnull().any(axis=1).sum()
null_rows
```

0

Figure 2.3.1 Use the “isnull” function to check data

⇒ We see that the returned value is 0, so we conclude that the data is not missing.

2.4 Handle duplicate data

Using the function `duplicated()` to check for duplicates.

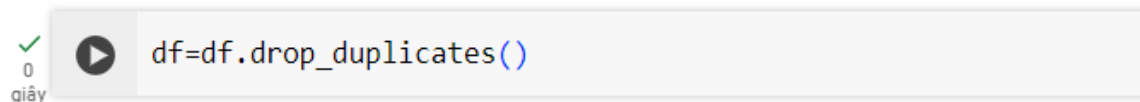


```
#Kiểm tra trùng lặp
df.duplicated().any()
```

True

Figure 2.4.1. The function returns True when there is duplicate data.

Drop the duplicate data.



```
df=df.drop_duplicates()
```

The result is False, meaning there is no duplicate data.



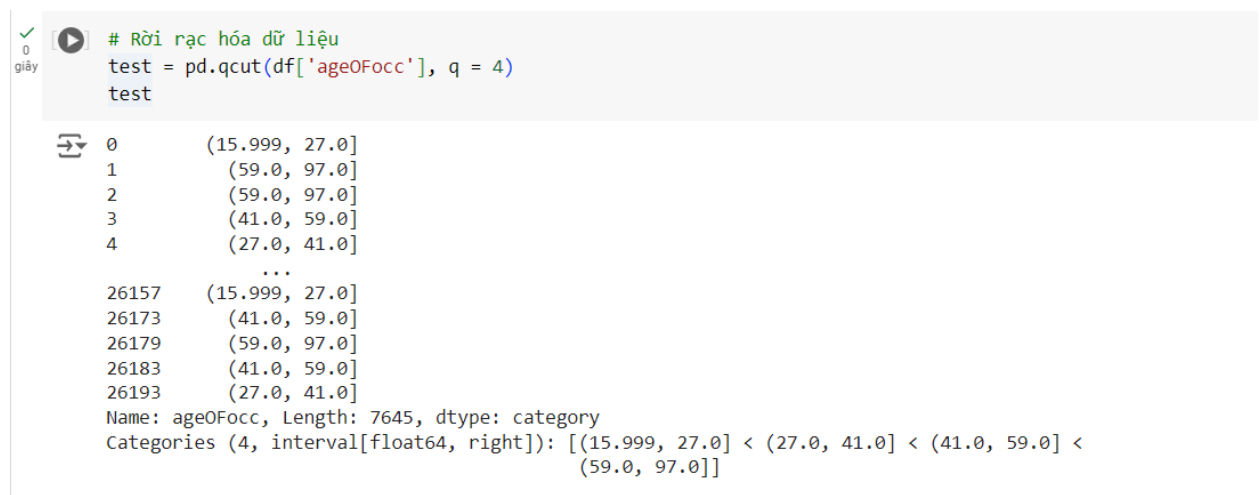
```
df.duplicated().any()
```

False

Figure 2.4.2. The data is no longer duplicated

2.5 Data binning

- ageOFocc attribute: Create age groups and divide age into 4 small parts, use qcut to divide into each group



```
# Rời rạc hóa dữ liệu
test = pd.qcut(df['ageOFocc'], q = 4)
test
```

```
0      (15.999, 27.0]
1      (59.0, 97.0]
2      (59.0, 97.0]
3      (41.0, 59.0]
4      (27.0, 41.0]
...
26157  (15.999, 27.0]
26173  (41.0, 59.0]
26179  (59.0, 97.0]
26183  (41.0, 59.0]
26193  (27.0, 41.0]
Name: ageOFocc, Length: 7645, dtype: category
Categories (4, interval[float64, right]): [(15.999, 27.0] < (27.0, 41.0] < (41.0, 59.0] < (59.0, 97.0]]
```

Figure 2.5.1 Use the “qcut” function to divide age into 4 small parts

```

df['ageOfocc'] = pd.qcut(df['ageOfocc'], q=4, labels=[1, 2, 3, 4])
df

```

<ipython-input-14-c699fdda234d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```

df['ageOfocc'] = pd.qcut(df['ageOfocc'], q=4, labels=[1, 2, 3, 4])

```

	dvcat	dead	airbag	seatbelt	frontal	sex	ageOfocc	occRole	deploy
0	25-39	alive	none	belted	1	f	1	driver	0
1	10-24	alive	airbag	belted	1	f	4	driver	1
2	10-24	alive	none	none	1	f	4	driver	0
3	25-39	alive	airbag	belted	1	f	3	driver	1
4	25-39	alive	none	belted	1	f	2	driver	0
...
26157	55+	dead	airbag	none	0	m	1	pass	1
26173	10-24	alive	airbag	belted	0	f	3	driver	1
26179	10-24	alive	airbag	none	1	m	4	driver	0
26183	25-39	dead	none	none	1	m	3	driver	0
26193	25-39	dead	none	belted	0	f	2	driver	0

7645 rows x 9 columns

Figure 2.5.2 Use the “qcut” function to divide into each groups

2.6 Export processed files

After processing, we can export the processed file to the computer.

```

df.to_csv('data_clear.csv')

```

CHAPTER 3. APPLYING CLASSIFICATION ALGORITHMS TO THE DATASET

1. Process the data set before running the algorithm

We import data after it has been processed and delete the unnamed column.

```

# Nhập dữ liệu
df = pd.read_csv('data_clear.csv')

df = df.drop(['Unnamed: 0'], axis= 1)

```

Figure 3.1.1 import data after it has been processed

We check the data again :

0 giây

df

	dvcat	dead	airbag	seatbelt	frontal	sex	ageOfocc	occRole	deploy
0	25-39	alive	none	belted	1	f	1	driver	0
1	10-24	alive	airbag	belted	1	f	4	driver	1
2	10-24	alive	none	none	1	f	4	driver	0
3	25-39	alive	airbag	belted	1	f	3	driver	1
4	25-39	alive	none	belted	1	f	2	driver	0
...
7640	55+	dead	airbag	none	0	m	1	pass	1
7641	10-24	alive	airbag	belted	0	f	3	driver	1
7642	10-24	alive	airbag	none	1	m	4	driver	0
7643	25-39	dead	none	none	1	m	3	driver	0
7644	25-39	dead	none	belted	0	f	2	driver	0

7645 rows × 9 columns

Figure 3.1.2 Displays data after processing

In order to avoid the situation where highly similar data exists in the data set, we create a heat chart to check.



Figure 3.1.3 Heat maps show the similarity of attributes

⇒ Based on the figure, we do not see any attributes with high similarity, so we do not remove any attributes

We convert non-numeric values into numbers to better match the algorithm input .

We use the “unique” function to check the values in the “dvcat” column.

```
[18] df['dvcat'].unique()
array(['25-39', '10-24', '40-54', '55+', '1-9km/h'], dtype=object)
```

Figure 3.1.4 use unique function to check the values in the “dvcat” column

We use the one-hot Vector method to convert data into digital form.

```

0
iây
▶ dummies= pd.get_dummies(df['dvcat'])
  dummies= dummies.astype(int)
  df = pd.concat([df,dummies],axis=1)
  df=df.drop(['dvcat'],axis=1)
  df

```

	dead	airbag	seatbelt	frontal	sex	ageOfocc	occRole	deploy	1-9km/h	10-24	25-39	40-54	55+
0	alive	none	belted	1	f	1	driver	0	0	0	1	0	0
1	alive	airbag	belted	1	f	4	driver	1	0	1	0	0	0
2	alive	none	none	1	f	4	driver	0	0	1	0	0	0
3	alive	airbag	belted	1	f	3	driver	1	0	0	1	0	0
4	alive	none	belted	1	f	2	driver	0	0	0	1	0	0
...
7640	dead	airbag	none	0	m	1	pass	1	0	0	0	0	1
7641	alive	airbag	belted	0	f	3	driver	1	0	1	0	0	0
7642	alive	airbag	none	1	m	4	driver	0	0	1	0	0	0
7643	dead	none	none	1	m	3	driver	0	0	0	1	0	0
7644	dead	none	belted	0	f	2	driver	0	0	0	1	0	0

7645 rows × 13 columns

We convert the values of the **dead**, **airbag**, **seatbelt** and **sex** columns back to values 0 and 1.

- Dead columns:

```

0
iây
▶ # chỉnh cột dead
  dead_Temp= {'dead':0,'alive':1}

  for dataset in [df] :
    dataset['dead']= dataset['dead'].map(dead_Temp)

  df

```

	dead	airbag	seatbelt	frontal	sex	ageOfocc	occRole	deploy	1-9km/h	10-24	25-39	40-54	55+
0	1	none	belted	1	f	1	driver	0	0	0	1	0	0
1	1	airbag	belted	1	f	4	driver	1	0	1	0	0	0
2	1	none	none	1	f	4	driver	0	0	1	0	0	0
3	1	airbag	belted	1	f	3	driver	1	0	0	1	0	0
4	1	none	belted	1	f	2	driver	0	0	0	1	0	0
...
7640	0	airbag	none	0	m	1	pass	1	0	0	0	0	1
7641	1	airbag	belted	0	f	3	driver	1	0	1	0	0	0
7642	1	airbag	none	1	m	4	driver	0	0	1	0	0	0
7643	0	none	none	1	m	3	driver	0	0	0	1	0	0
7644	0	none	belted	0	f	2	driver	0	0	0	1	0	0

7645 rows × 13 columns

- Airbag columns:

✓ 0 giây

```
# Chỉnh airbag
dead_Temp= {'none':0, 'airbag':1}

for dataset in [df]:
    dataset['airbag']= dataset['airbag'].map(dead_Temp)

df
```

	dead	airbag	seatbelt	frontal	sex	ageOfocc	occRole	deploy	1-9km/h	10-24	25-39	40-54	55+
0	1	0	belted	1	f	1	driver	0	0	0	1	0	0
1	1	1	belted	1	f	4	driver	1	0	1	0	0	0
2	1	0	none	1	f	4	driver	0	0	1	0	0	0
3	1	1	belted	1	f	3	driver	1	0	0	1	0	0
4	1	0	belted	1	f	2	driver	0	0	0	1	0	0
...
7640	0	1	none	0	m	1	pass	1	0	0	0	0	1
7641	1	1	belted	0	f	3	driver	1	0	1	0	0	0
7642	1	1	none	1	m	4	driver	0	0	1	0	0	0
7643	0	0	none	1	m	3	driver	0	0	0	1	0	0
7644	0	0	belted	0	f	2	driver	0	0	0	1	0	0

7645 rows × 13 columns

- Seatbelt columns:

✓ 0 giây

```
# chỉnh seatbelt
seatbelt_Temp= {'none':0, 'belted':1}

for dataset in [df]:
    dataset['seatbelt']= dataset['seatbelt'].map(seatbelt_Temp)

df
```

	dead	airbag	seatbelt	frontal	sex	ageOfocc	occRole	deploy	1-9km/h	10-24	25-39	40-54	55+
0	1	0	1	1	f	1	driver	0	0	0	1	0	0
1	1	1	1	1	f	4	driver	1	0	1	0	0	0
2	1	0	0	1	f	4	driver	0	0	1	0	0	0
3	1	1	1	1	f	3	driver	1	0	0	1	0	0
4	1	0	1	1	f	2	driver	0	0	0	1	0	0
...
7640	0	1	0	0	m	1	pass	1	0	0	0	0	1
7641	1	1	1	0	f	3	driver	1	0	1	0	0	0
7642	1	1	0	1	m	4	driver	0	0	1	0	0	0
7643	0	0	0	1	m	3	driver	0	0	0	1	0	0
7644	0	0	1	0	f	2	driver	0	0	0	1	0	0

7645 rows × 13 columns

- Sex columns:

0 giây

```
sex_Temp= {'m':0, 'f':1}
for dataset in [df]:
    dataset['sex']= dataset['sex'].map(sex_Temp)
df
```

	dead	airbag	seatbelt	frontal	sex	ageOfOcc	occRole	deploy	1-9km/h	10-24	25-39	40-54	55+
0	1	0	1	1	1	1	driver	0	0	0	1	0	0
1	1	1	1	1	1	4	driver	1	0	1	0	0	0
2	1	0	0	1	1	4	driver	0	0	1	0	0	0
3	1	1	1	1	1	3	driver	1	0	0	1	0	0
4	1	0	1	1	1	2	driver	0	0	0	1	0	0
...
7640	0	1	0	0	0	1	pass	1	0	0	0	0	1
7641	1	1	1	0	1	3	driver	1	0	1	0	0	0
7642	1	1	0	1	0	4	driver	0	0	1	0	0	0
7643	0	0	0	1	0	3	driver	0	0	0	1	0	0
7644	0	0	1	0	1	2	driver	0	0	0	1	0	0

7645 rows x 13 columns

We use the “unique” function to check the values in the “occRole” column.

0 giây

```
df['occRole'].unique()
array(['driver', 'pass'], dtype=object)
```

We use the one-hot Vector method to convert data into digital form.

0 giây

```

dummies= pd.get_dummies(df['occRole'])
dummies= dummies.astype(int)
df = pd.concat([df,dummies],axis=1)
df=df.drop(['occRole'],axis=1)
df

```

	dead	airbag	seatbelt	frontal	sex	ageOFocc	deploy	1-9km/h	10-24	25-39	40-54	55+	driver	pass
0	1	0	1	1	1	1	0	0	0	1	0	0	1	0
1	1	1	1	1	1	4	1	0	1	0	0	0	1	0
2	1	0	0	1	1	4	0	0	1	0	0	0	1	0
3	1	1	1	1	1	3	1	0	0	1	0	0	1	0
4	1	0	1	1	1	2	0	0	0	1	0	0	1	0
...
7640	0	1	0	0	0	1	1	0	0	0	0	1	0	1
7641	1	1	1	0	1	3	1	0	1	0	0	0	1	0
7642	1	1	0	1	0	4	0	0	1	0	0	0	1	0
7643	0	0	0	1	0	3	0	0	0	1	0	0	1	0
7644	0	0	1	0	1	2	0	0	0	1	0	0	1	0

7645 rows x 14 columns

We use the “unique” function to check the values in the “ageOFocc” column.

0 giây

```

df['ageOFocc'].unique()

```

array([1, 4, 3, 2])

We use the one-hot Vector method to convert data into digital form.

0 play

```

dummies= pd.get_dummies(df['ageOfOcc'])
dummies= dummies.astype(int)
df = pd.concat([df,dummies],axis=1)
df=df.drop(['ageOfOcc'],axis=1)
df = df.rename(columns={1: 'age_1', 2: 'age_2', 3: 'age_3', 4: 'age_4'})
df

```

	dead	airbag	seatbelt	frontal	sex	deploy	1-9km/h	10-24	25-39	40-54	55+	driver	pass	age_1	age_2	age_3	age_4
0	1	0	1	1	1	0	0	0	1	0	0	1	0	1	0	0	0
1	1	1	1	1	1	1	0	1	0	0	0	1	0	0	0	0	1
2	1	0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	1
3	1	1	1	1	1	1	0	0	1	0	0	1	0	0	0	1	0
4	1	0	1	1	1	0	0	0	1	0	0	1	0	0	1	0	0
...
7640	0	1	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0
7641	1	1	1	0	1	1	0	1	0	0	0	1	0	0	0	1	0
7642	1	1	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1
7643	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	1	0
7644	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0

7645 rows × 17 columns

2. Decision Tree ID3 Model

2.1 Definition

- Before delving into the intricacies of the ID3 algorithm, let's grasp the essence of decision trees. Decision trees mimic human decision-making processes by recursively splitting data based on different attributes to create a flowchart-like structure for classification or regression.
- A well-known decision tree approach for machine learning is the Iterative Dichotomiser 3 (ID3) algorithm. By choosing the best characteristic at each node to partition the data depending on information gain, it recursively constructs a tree. The goal is to make the final subsets as homogeneous as possible. By choosing features that offer the greatest reduction in entropy or uncertainty, ID3 iteratively grows the tree. The procedure keeps going until a halting requirement is satisfied, like a minimum subset size or a maximum tree depth.

2.2 Advantage

- **Intuitive:** Decision trees are very easy to understand and intuitive. Decisions are made step by step from the root to the leaves, making it easy to explain the results.
- **Simple Interpretation:** Decisions in the tree are represented as simple if-then rules, making them easy to understand for non-technical users.
- **Categorical Data Compatibility:** ID3 is designed to handle categorical data effectively, providing clear classifications for different categories.

- Fast Tree Construction: The ID3 algorithm can build decision trees quickly, using a recursive method to choose the best attribute and split the dataset into subsets.
- Handles Multi-Class Classification: ID3 can handle multi-class classification problems without needing to convert data or alter the algorithm structure.

2.3 Decision Tree ID3 algorithm

-Perform algorithm runtime testing and execute the algorithm using the following command lines.

```
df_copy = df.copy()
y = df_copy['dead']
x = df_copy.drop('dead', axis=1) # Các thuộc tính đặc trưng

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3 , random_stat

# Xây dựng cây ID3
# Tính thời gian chạy thuật toán
start_time = time.time()

regressor = tree.DecisionTreeClassifier(criterion="entropy",random_state=0)
regressor.fit(X_train, y_train)
end_time = time.time()
execution_time_ID3 = end_time - start_time
print(f"Thời gian chạy thuật toán: {execution_time_ID3:.4f} giây")
```

Thời gian chạy thuật toán: 0.0112 giây

- Evaluate the model.

```
tree_predict = regressor.predict(X_test)
tree_score = metrics.accuracy_score(y_test,tree_predict)
print("Độ chính xác: ", tree_score)
print("Report: ",metrics.classification_report(y_test,tree_predict))
```

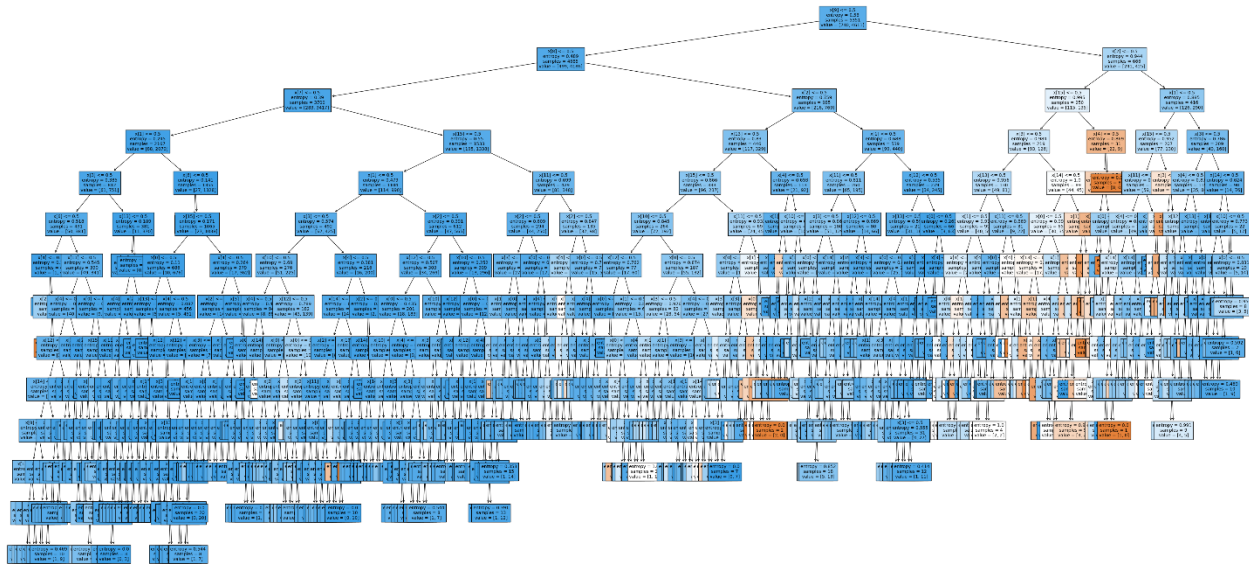
Độ chính xác: 0.8496076721883173

Report:

	precision	recall	f1-score	support
0	0.33	0.13	0.19	302
1	0.88	0.96	0.92	1992
accuracy			0.85	2294
macro avg	0.60	0.55	0.55	2294
weighted avg	0.81	0.85	0.82	2294

-Decision tree ID3 model.

```
[ ] # 4. Hiển thị cây quyết định bằng biểu đồ
fig,ax = plt.subplots(figsize=(50,24))
tree.plot_tree(regressor,filled=True,fontsize=10)
plt.savefig("Decision_tree",dpi=100)
plt.show()
```



3. Decision Tree Cart Model

3.1 Definition

- CART(Classification And Regression Trees) is a variation of the decision tree algorithm. It can handle both classification and regression tasks.
- CART is a predictive algorithm used in Machine learning and it explains how the target variable's values can be predicted based on other matters. It is a decision tree where each fork is split into a predictor variable and each node has a prediction for the target variable at the end.
- The term CART serves as a generic term for the following categories of decision trees:
 - Classification Trees: The tree is used to determine which “class” the target variable is most likely to fall into when it is continuous.
 - Regression trees: These are used to predict a continuous variable's value.

3.2 Advantage

- Versatility: Can be used for both classification and regression tasks.
- Interpretability: Results in a model that is easy to interpret and visualize.
- Non-Parametric: Makes no assumptions about the distribution of the data.

- Handles Both Categorical and Numerical Data: Can process both types of data seamlessly.

3.3 Decision Tree Cart algorithm

-Perform algorithm runtime testing and execute the algorithm using the following command lines.

```
[ ] df_copy = df.copy()
df_copy=df_copy.drop(['deploy'],axis=1)
y = df_copy['dead']
x = df_copy.drop('dead', axis=1) # Các thuộc tính đặc trưng

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3 , random_state=42)

# Xây dựng cây ID3
# Tính thời gian chạy thuật toán
start_time = time.time()
cart = tree.DecisionTreeClassifier(criterion="gini",random_state=0)
cart.fit(X_train, y_train)
end_time = time.time()
execution_time_cart = end_time - start_time
print(f"Thời gian chạy thuật toán: {execution_time_cart:.4f} giây")
```

➡ Thời gian chạy thuật toán: 0.0131 giây

- Evaluate the model.

```
▶ tree2_predict = cart.predict(X_test)
tree2_score = metrics.accuracy_score(y_test,tree2_predict)
print("Độ chính xác: ",tree2_score)
print("Report: ",metrics.classification_report(y_test,tree2_predict))
```

➡ Độ chính xác: 0.8605056669572798

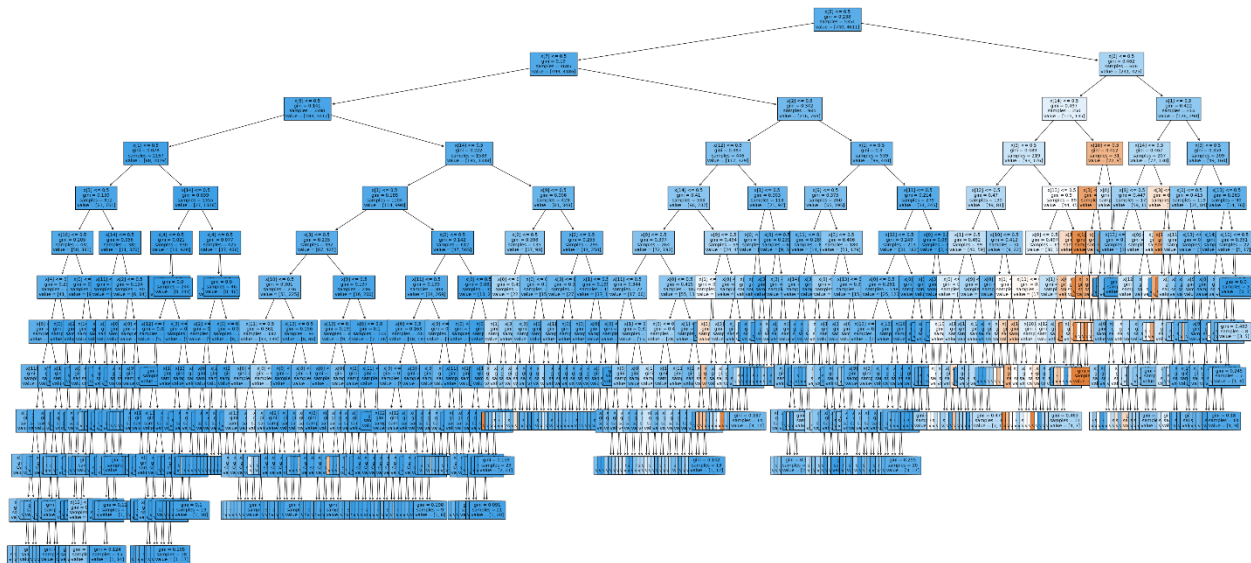
Report:

	precision	recall	f1-score	support
0	0.40	0.11	0.18	302
1	0.88	0.97	0.92	1992
accuracy			0.86	2294
macro avg	0.64	0.54	0.55	2294
weighted avg	0.82	0.86	0.83	2294



```
# 4. Hiện thị cây quyết định bằng biểu đồ
fig,ax = plt.subplots(figsize=(50,24))
tree.plot_tree(cart,filled=True,fontsize=10)
plt.savefig("Decision_tree",dpi=100)
plt.show()
```

- Decision tree cart model.



4. Naïve Bayes Model

4.1 Definition

The Naïve Bayes Model is Naïve Bayes is a classification algorithm modeled on Bayes' theorem in statistical probability, used in many classification tasks. The main purpose is to calculate the probability $\Pr(C_j, d')$, the probability that document d' lies in class C_j .

4.2 Advantage

- Easy to apply and understand.
- Good performance with feature spaces and large data.
- Useful in text classification problems.

4.3 Naïve Bayes algorithm

Perform algorithm runtime testing and execute the algorithm using the following command lines.

```
x = df.drop('dead', axis=1) # Các thuộc tính đặc trưng
y = df['dead']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3 , random_state=42)

# Khởi tạo và huấn luyện mô hình Native Bayes
model = MultinomialNB()

# Đo thời gian huấn luyện
start_time = time.time()

# Huấn luyện mô hình
model.fit(x_train, y_train)

# Tính thời gian huấn luyện
end_time = time.time()
training_time = end_time - start_time

# Đo thời gian dự đoán
start_time = time.time()
y_pred = model.predict(x_test)
end_time = time.time()

# Tính thời gian dự đoán
prediction_time_nb = end_time - start_time
print(f'Thời gian dự đoán: {prediction_time_nb:.4f} giây')
```

Thời gian dự đoán: 0.0070 giây

Evaluate the model.

```
# Đánh giá mô hình
accuracy_nb = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print("Accuracy: ", accuracy_nb)
print("Report: ", report)
```

Accuracy: 0.8679163034001743

Report: precision recall f1-score support

0	0.33	0.00	0.01	302
1	0.87	1.00	0.93	1992

accuracy			0.87	2294
macro avg	0.60	0.50	0.47	2294
weighted avg	0.80	0.87	0.81	2294

5. Random Forest Model

5.1 Definition

The Random Forest model is a machine learning method belonging to the ensemble model category, used for both classification and regression tasks. Random Forest operates by constructing multiple decision trees during the training process and makes predictions (based on classification or regression) by aggregating the results from all those decision trees.

5.2 Advantage

- Flexibility and high accuracy: Random Forest can be used for both classification and regression tasks, often providing accurate predictions even when using default hyperparameters, saving time and effort in model optimization.
- Reduction of overfitting risk: By combining multiple decision trees, Random Forest reduces the risk of overfitting, especially when the number of trees is sufficiently large, ensuring effective model performance on both the training and testing datasets.
- Evaluation of feature importance: The model provides the ability to assess the importance of input features, helping to understand the impact of each feature on the prediction results and improving the model's interpretability.
- One of the major challenges in machine learning is overfitting (the phenomenon where the model learns too much detail from the training data, leading to poor predictions on the testing data). However, due to the structure of Random Forest with multiple decision trees, this issue rarely occurs, especially when the number of trees in the forest is large enough.

5.3 Random Forest algorithm

- Perform algorithm runtime testing and execute the algorithm using the following command lines.

```

df_copy= df.copy()
df_copy=df_copy.drop(['deploy'],axis=1)
y = df_copy['dead']
x = df_copy.drop('dead', axis=1) # Các thuộc tính đặc trưng

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3 , random_state=42)

# Xây dựng RF
# Tính thời gian chạy thuật toán
start_time = time.time()

rf = RandomForestClassifier()
rf.fit(X_train, y_train)

end_time = time.time()
execution_time_rf = end_time - start_time
print(f"Thời gian chạy thuật toán: {execution_time_rf:.4f} giây")

```

Thời gian chạy thuật toán: 0.3741 giây

Figure 5.3.1 Verify the execution time and run the model

- Evaluate the model.

```

# Đánh giá mô hình
rf_predict = rf.predict(X_test)
rf_score = metrics.accuracy_score(y_test,rf_predict)
print("Độ chính xác:" ,rf_score)
print("Báo cáo: ",metrics.classification_report(y_test,rf_predict))

```

Độ chính xác: 0.8657367044463818

Báo cáo:	precision	recall	f1-score	support
0	0.45	0.10	0.16	302
1	0.88	0.98	0.93	1992
accuracy			0.87	2294
macro avg	0.67	0.54	0.54	2294
weighted avg	0.82	0.87	0.83	2294

Figure 5.3.2 Evaluate the model

- Random Forest model.

```
#Mô hình rừng ngẫu nhiên
individual_tree = rf.estimators_[0]
```

```
fig,ax = plt.subplots(figsize=(50,24))
plot_tree(individual_tree,filled=True,fontsize=10)
```

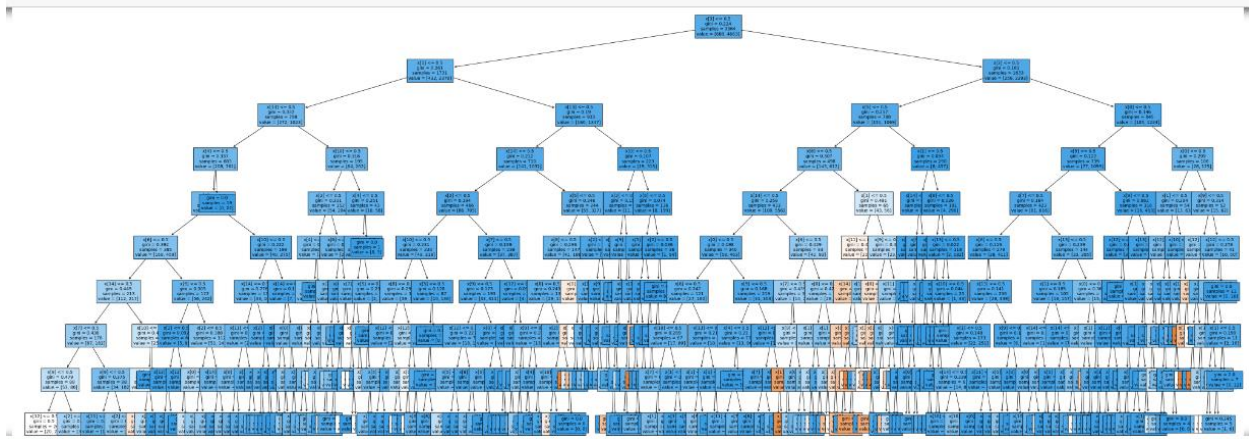


Figure 5.3.3 Random Forest model

6. Bagging Classifier Model

6.1 Definition

- The Bagging Classifier (Bootstrap Aggregating Classifier) model is a powerful ensemble machine learning method primarily used to reduce variance and prevent overfitting, aiming to improve prediction performance. This method is developed based on the idea that the combination of multiple individual machine learning models will yield better results than using a single model. Bagging is a form of bootstrap technique, where the training data is randomly sampled with replacement to create multiple different subsets of data.

6.2 Advantage

- Variance reduction: By combining multiple models trained on different subsets of the data, the Bagging Classifier helps reduce the variance of the model, making predictions more stable and consistent.
- Overfitting prevention: Overfitting occurs when a machine learning model fits too closely to the training data, resulting in poor predictions on the test data. The Bagging Classifier, by using multiple independent machine learning models and combining their predictions, effectively reduces the risk of overfitting.
- High performance with weak learners: The Bagging Classifier performs well even when using weak machine learning models such as simple decision trees. The combination of multiple weak models often creates a stronger ensemble model.
- Easy deployment due to Python libraries like scikit-learn providing readily available modules that you can leverage to optimize your model.

6.3 Bagging Classifier algorithm

- Algorithm runtime testing and execution using the following command lines.

```
df_copy = df.copy()
df_copy = df_copy.drop(['deploy'], axis=1)
y = df_copy['dead']
x = df_copy.drop('dead', axis=1) # Các thuộc tính đặc trưng

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)

# Xây dựng Bagging Classifier
# Tính thời gian chạy thuật toán
start_time = time.time()

bagging_clf = BaggingClassifier()
bagging_clf.fit(X_train, y_train)

end_time = time.time()
execution_time_bc = end_time - start_time
print(f"Thời gian chạy thuật toán: {execution_time_bc:.4f} giây")
```

Thời gian chạy thuật toán: 0.0691 giây

Figure 6.3.1 Verify the execution time and run the model

- Model evaluation.

```
# Đánh giá mô hình
bagging_clf_predict = bagging_clf.predict(X_test)
bagging_clf_score = metrics.accuracy_score(y_test, bagging_clf_predict)
print("Độ chính xác: ", bagging_clf_score)
print("Report: ", metrics.classification_report(y_test, bagging_clf_predict))
```

Độ chính xác: 0.8578901482127289

Report: precision recall f1-score support

0 0.36 0.10 0.16 302

1 0.88 0.97 0.92 1992

accuracy 0.86 2294

macro avg 0.62 0.54 0.54 2294

weighted avg 0.81 0.86 0.82 2294

Figure 6.3.2 Model evaluation

7. K – Nearest Neighbor Model

7.1 Definition

The K - Nearest Neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today.

More clearly, KNN tries to predict the correct class for testing data by calculating the distance between testing data and all training points. Then, select K points whose values match the testing data and calculate the probability that testing data belongs to training data classes K and probability classes. The highest groups will be selected.

7.2 Advantage

- Simplicity: Easy to understand and implement.
- Versatility: Can be used for both classification and regression.
- Faster than other algorithms: KNN is called Lazy Learner because it does not learn from the training set immediately instead it stores the dataset and at the time of classification. It does not build a model until a query is performed on the dataset.
- No Training Phase: Requires no explicit training phase before making predictions, new data can be added to the next row without the algorithm's accuracy being affected.
- It can be more effective if the training data is large.

7.3 KNN algorithm

Perform algorithm runtime testing and execute the algorithm using the following command lines.

```
#KNN
X = df.drop('dead', axis=1) # Các thuộc tính đặc trưng
y = df['dead']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3 , random_state=42)

# Chuẩn hóa đặc trưng
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

start_time = time.time()
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
end_time = time.time()

# Your existing code
execution_time_knn = end_time - start_time
print(f"Thời gian chạy thuật toán: {execution_time_knn:.4f} giây")
```

Thời gian chạy thuật toán: 0.0014 giây

Evaluate the model.

```
# Dự đoán nhãn cho tập kiểm tra
y_pred = knn.predict(X_test)
# Đánh giá mô hình

accuracy_knn = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Độ chính xác: {accuracy_knn}")
print("Report:", report)
```

Độ chính xác: 0.8539668700959023

Report:	precision	recall	f1-score	support
0	0.37	0.16	0.22	302
1	0.88	0.96	0.92	1992
accuracy			0.85	2294
macro avg	0.63	0.56	0.57	2294
weighted avg	0.81	0.85	0.83	2294

8. Logistic Regression Model

8.1 Definition

Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation. The model delivers a binary or dichotomous outcome limited to two possible outcomes: yes/no, 0/1, or true/false.

8.2 Advantage

- Easier to implement, interpret, and train than other machine learning methods.
- Suitable for linearly separable dataset: A linearly separable dataset refers to a graph where a straight line separates the two data classes. In logistic regression, the y variable takes only two values. Hence, one can effectively classify data into two separate classes if linearly separable data is used.
- Provide valuable insight: Logistic regression measures how relevant or appropriate an independent/predictor variable is (coefficient size) and also reveals the direction of their relationship or association (positive or negative).

8.3 Logistic Regression algorithm

Perform algorithm runtime testing and execute the algorithm using the following command lines.

```

X = df.drop('dead', axis=1) # Các thuộc tính đặc trưng
y = df['dead']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3 , random_state=42)

start_time = time.time()
lf = LogisticRegression()
lf.fit(X_train, y_train)
end_time = time.time()

# Thời gian chạy thuật toán
execution_time_lr = end_time - start_time
print(f"Thời gian chạy thuật toán: {execution_time_lr:.4f} giây")

```

Thời gian chạy thuật toán: 0.0246 giây

Evaluate the model.

```

# Dự đoán nhãn cho tập kiểm tra
y_pred = lf.predict(X_test)
# Đánh giá mô hình

accuracy_lr = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(f"Độ chính xác: {accuracy_lr}")
print("Report:", report)

```

Độ chính xác: 0.8718395815170009

Report:	precision	recall	f1-score	support
0	0.62	0.07	0.12	302
1	0.88	0.99	0.93	1992
accuracy			0.87	2294
macro avg	0.75	0.53	0.53	2294
weighted avg	0.84	0.87	0.82	2294

9. Support Vector Machine Model

9.1 Definition

SVM is a supervised algorithm that can be used for either classification or recursion. However, it is mainly used for classification. In this algorithm, we graph the data as points in n dimensions (where n is the number of features you have) with the value of each feature being an associated part. Then we perform a search for the "Hyper-plane" that divides the classes. Hyper-plane is simply a straight line that can divide layers into two separate parts.

9.2 Advantage

This is an algorithm that works effectively with high dimensional spaces. The algorithm consumes little memory because it only uses points in the support set for prediction in the decision function. We can create multiple decision functions from different kernel functions.

9.3 SVM algorithm

Perform algorithm runtime testing and execute the algorithm using the following command lines.

```

x = df.drop('dead', axis=1) # Các thuộc tính đặc trưng
y = df['dead']

# Chia dữ liệu thành tập huấn luyện và tập kiểm tra
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3 , random_state=42)

# Xây dựng pipeline với scaler và mô hình SVC
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('svm', SVC(kernel='rbf')) # Có thể thử các kernel khác như 'linear'
])

# Khởi tạo và huấn luyện mô hình SVM
model = pipeline.fit(x_train, y_train)

# Đo thời gian huấn luyện
start_time = time.time()
end_time = time.time()

# Tính thời gian huấn luyện
training_time = end_time - start_time

# Đo thời gian dự đoán
start_time = time.time()
y_pred = model.predict(x_test)
end_time = time.time()

# Tính thời gian dự đoán
prediction_time_svm = end_time - start_time
print(f'Thời gian dự đoán: {prediction_time_svm:.4f} giây')

```

Thời gian dự đoán: 0.2531 giây

Evaluate the model.

```
# Đánh giá mô hình
accuracy_svm = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print("Accuracy: ", accuracy_svm)
print("Report: ", report)
```

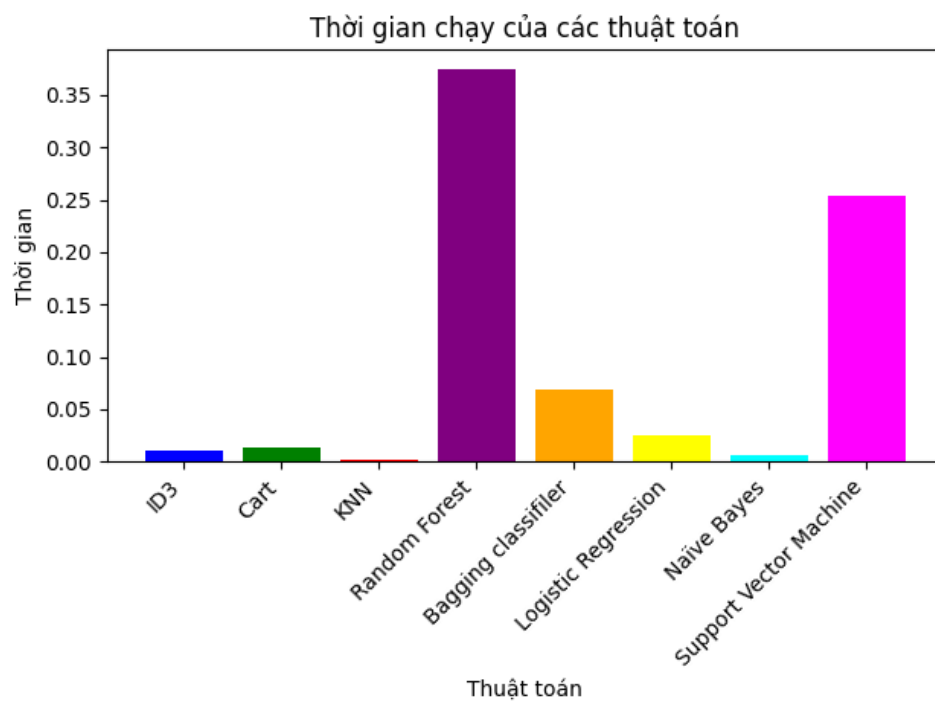
```
Accuracy: 0.8718395815170009
Report:
              precision    recall  f1-score   support

     0       0.79      0.04      0.07       302
     1       0.87      1.00      0.93      1992

 accuracy          0.87          0.87          0.87          2294
 macro avg         0.83          0.52          0.50          2294
 weighted avg      0.86          0.87          0.82          2294
```

CHAPTER 4. EVALUATE ALGORITHMNS AND FORECASTS

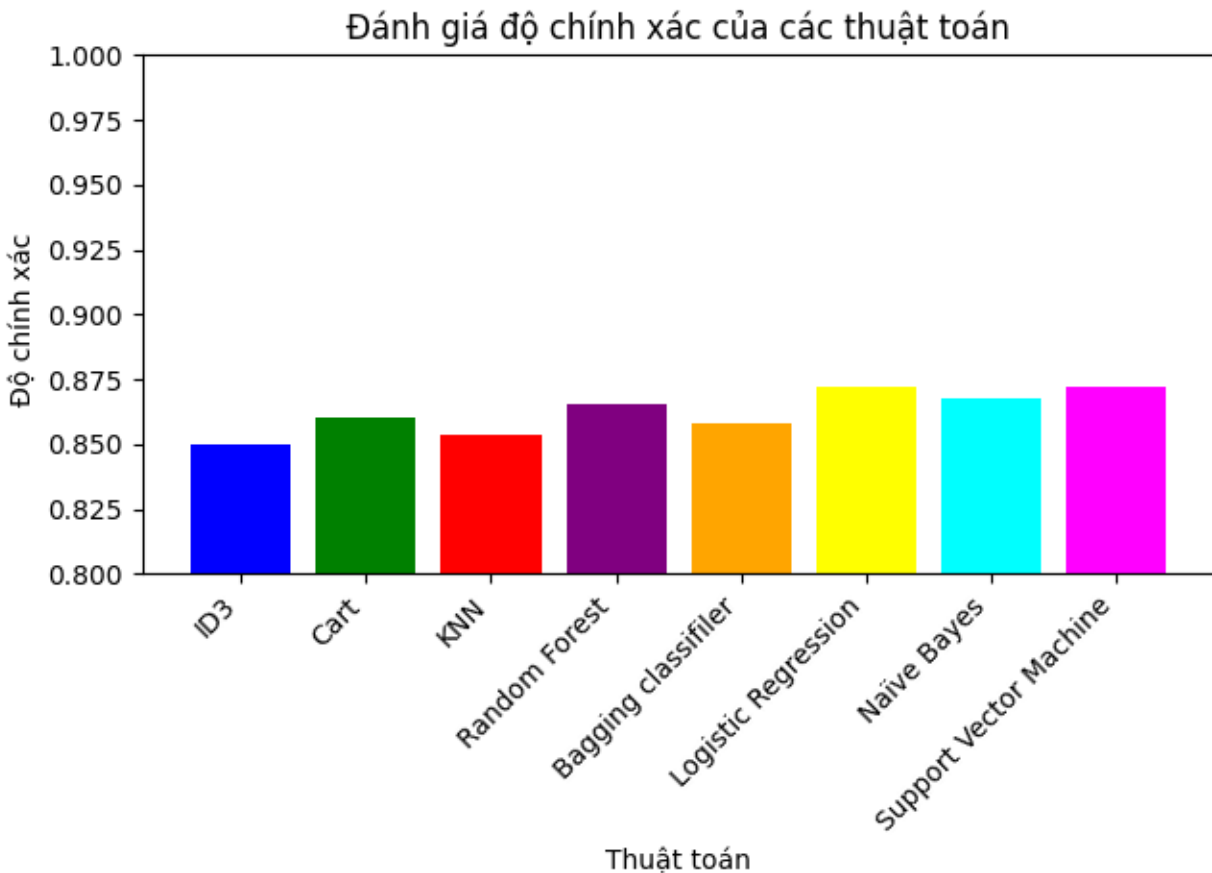
1. Runtime Evaluation



⇒ Based on the graph, we can see that the KNN algorithm has the fastest running time compared to the other algorithms due to its own Lazy Learner feature.

⇒ Random Forest takes the most time to run.

2. Compare the accuracy of algorithms



⇒ Based on the graph, we can see that the accuracy of the Logistic Regression and Support Vector Machine algorithms has the highest accuracy and the ID3 algorithm has the smallest accuracy although the algorithms are almost equal but there is a difference. small but not significant.

3. Advantages and disadvantages of the algorithm

3.1 Decision tree

❖ Advantages for the dataset:

- Aside from not requiring data normalization, Decision Tree also does not require data preprocessing, which can reduce the model's accuracy, but will result in faster runtime.
- Overall, the algorithm is effective in terms of both time and accuracy.

❖ Disadvantages for the dataset:

- The model generates many unnecessary branches, so additional processing is required to further reduce the runtime.

3.2 Random Forest

- ❖ Advantages for the dataset:
 - The runtime for various cases is fast.
 - The accuracy of the Bagging Classifier is relatively high, similar to other algorithms like Decision Tree and Random Forest.
- ❖ Disadvantages for the dataset:
 - There is a large gap between the accuracy on the test set and the training set.
 - It is computationally expensive and may not be suitable for other use cases, as the Bagging Classifier builds multiple models.

3.3 Bagging Classifier

- ❖ Advantages for the dataset:
 - The runtime for various cases is fast.
 - The accuracy of the Bagging Classifier is relatively high, similar to other algorithms like Decision Tree and Random Forest.
- ❖ Disadvantages for the dataset:
 - There is a large gap between the accuracy on the test set and the training set.
 - It is computationally expensive and may not be suitable for other use cases, as the Bagging Classifier builds multiple models.

3.4 KNN

- ❖ Advantages for the dataset:
 - KNN Model does not require extensive data preprocessing or normalization, making it straightforward to implement.
 - This simplicity helps save time in data preparation and allows for quick deployment of the model.
 - KNN often achieves high accuracy when applied to small or well-structured datasets.
- ❖ Disadvantages for the dataset:
 - KNN can face performance issues with large datasets due to its need to compute distances between data points, which can be time-consuming and memory-intensive.
 - The model is sensitive to noisy data (outliers) and can struggle with datasets containing many irrelevant features.
 - This sensitivity necessitates careful data handling and preprocessing to optimize performance.

3.5 Logistic Regression

- ❖ Advantages for the dataset:
 - The more the data is normalized and preprocessed, the faster the runtime and the higher the accuracy.
- ❖ Disadvantages for the dataset:
 - Including more independent variables that are irrelevant and unrelated to the model will diminish the quality of the model.
 - Logistic Regression is quite sensitive to outliers, which can lead to a model with low accuracy.
 - The input variables must be independent of each other to avoid multicollinearity.

3.6 Naïve Bayes

- ❖ Advantages for the dataset:
 - The Naive Bayes algorithm operates very quickly and easily predicts the class of the dataset. Since the features and labels have been converted to numerical form, the runtime is not too time-consuming. Furthermore, Naive Bayes utilizes few resources, making the model deployment fast and lightweight.
 - For cases where the data is not encoded, Naive Bayes has higher accuracy compared to other algorithms, as this model is not severely affected by other factors.
- ❖ Disadvantages for the dataset:
 - The Naive Bayes algorithm works best on independent features, so the accuracy of the model is not very high when applied to a dataset where the attributes are correlated with each other.

3.7 Support Vector Machine

- ❖ Advantages for the dataset:
 - Support Vector Machine (SVM) can effectively handle complex datasets and does not require extensive data preprocessing.
 - This model can address nonlinear classification problems by using kernel functions, which enhances the model's accuracy.
 - SVM works well with high-dimensional datasets (many features) and provides accurate and reliable classification results.
- ❖ Disadvantages for the dataset:
 - When applied to large datasets, SVM can become slow and computationally intensive, affecting execution time.
 - The model is sensitive to noisy data (outliers) and can struggle with selecting appropriate parameters, such as the type of kernel and hyperparameters, requiring careful optimization.

- Additionally, SVM does not directly provide probabilistic predictions, which can be a limitation in some real-world applications that require probability estimates.

4. Development

- *Strengths*: Proficient in data exploration and analysis (preprocessing, classification using learned algorithms), result evaluation, teamwork, and constructing model to predict the victim's ability to survive based on the use of airbags and other factors in traffic accidents.
- *Weaknesses*: Due to time constraints within a semester, the team has not yet found a way to optimize for the highest accuracy and lacks thorough proficiency in the Python language.
- *Developmental direction*: Will research algorithms to reduce columns in data preprocessing to identify which columns yield the highest accuracy for algorithms.

CHAPTER 5. ASSIGNMENT TABLE

	Nguyễn Dương Chí Tâm (21520439)	Huỳnh Mạnh Huy (21520259)	Đỗ Hiền Thảo (21520460)	Nguyễn Trương Đình Giang (21520215)
Select and discuss about the dataset	x	x	x	x
Overview the data and problem statement	x			
Describe data and attributes			x	
Handle unnecessary data, data binning		x		
Handle noisy data, export processed file				x

Handle missing value and duplicated value	x			
Implement the Decision Tree ID3 and Decision Tree Cart model	x			
Implement the Random Forest and Bagging Classifier model				x
Implement the Naïve Bayes and SVM model		x		
Implement the Logistic Regression and KNN model			x	
Runtime evaluate			x	
Compare the accuracy of algorithms		x		
Give advantages and disadvantages and algorithm development direction.				x
Edit, write report and find references	x	x	x	x

CHAPTER 6. REFERENCES

- [1] Lecture slides.
- [2] Practical guidance documents.
- [3] K-Nearest Neighbor (KNN) Algorithm for Machine Learning.

<https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>

[4] Logistic Regression, <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/#:~:text=Logistic%20regression%20is%20a%20supervised%20machine%20learning%20algorithm%20that%20accomplishes,1%2C%20or%20true%2Ffalse>.

[5] Decision tree cart model, <https://www.geeksforgeeks.org/cart-classification-and-regression-tree-in-machine-learning/>

[6] Decision tree ID3 model , <https://www.geeksforgeeks.org/sklearn-iterative-dichotomiser-3-id3-algorithms/>

