BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN ĐHQG-HCM
KHOA TOÁN – TIN HỌC
---o0o---

BTTH TUẦN 1: KHAI THÁC DỮ LIỆU

Sinh viên thực hiện: Nguyễn Công Hoài Nam

Mã số sinh viên: 21280099

Tp. Hồ Chí Minh, ngày 6 tháng 4 năm 2024

1. Làm sạch và tiền xử lý dữ liệu

Bước 1: tạo file data.csv

ID	First Name	Last Name	Age	Gender	Department	Salary	Date of Joining
1	John	Doe	25	M	Sales	50000	1/1/2020
2	Jane	Smith	30	F	Marketing	60000	6/1/2018
3	Bod	Johnson	45	M	HR	70000	9/1/2026
4	Alice	Williams	33	F	IT	80000	2/1/2017
5	James	Brown	27	M	Sales	55000	3/1/2029
6	Sarah	Lee		F	Marketing	65000	12/1/2018
7	Michael	Davis	39	M	HR		8/1/2015
8	Susan	Miller	42	F	IT	90000	11/1/2014
9	David	Wilson	28	M	Sales	60000	5/1/2020
10	Emily	Brown	35	F	Marketing	55000	4/1/2017
11	John	Doe	25	M	Sales	50000	1/1/2020
12	John	Doe	25	M	Sales	50000	1/1/2020

Bước 2: Load dữ liệu

```
#- Load dữ liệu thành DataFrame Pandas
import pandas as pd
import numpy as np
from sklearn import preprocessing

df = pd.read_csv('LAB 01/data.csv', delimiter=',')

#đặt lại index
df = df.set_index('ID')
df
```

Bước 3: Xử lý các giá trị missing

Kiểm tra giá trị null và và điền bằng mean()

```
#2. Kiểm tra các giá trị missing
print(df.isnull().sum())
# Làm đầy các giá trị missing bằng các giá trị trung bình
numeric_columns = df.select_dtypes(include=['float']).columns
df[numeric_columns] =
df[numeric_columns].fillna(df[numeric_columns].mean())
```

Bước 4: Xử lý các giá trị giống nhau

```
#3. Kiểm tra các giá trị giống nhau
df.duplicated().sum()

## Loại bỏ các dòng giống nhau
df = df.drop_duplicates()
df
```

Bước 5: Mã hóa các biến categorical

Sử dụng thuật toán Get Dummies để mã hóa các biến categorical

```
#- Mã hóa dữ liệu categorical
df = pd.get_dummies(df, columns=['Gender', 'Department'])
```

Bước 6: Xử lý dữ liệu Datetime

Chuyển cột date thành một đối tượng datetime
Tách tháng và ngày trong tuần từ cột date
Loại bỏ cột date

```
#- Xử lý dữ liệu datetime
## Chuyển cột date thành một đối tượng datetime
df['Date of Joining']=pd.to_datetime(df['Date of Joining'])

## extract month and dat of week from date column
df['month'] = df['Date of Joining'].dt.month
df['day_of_week'] = df['Date of Joining'].dt.day_name()

## Drop the original 'Date' column
df = df.drop('Date of Joining', axis=1)

df
```

Bước 7: Xử lý các giá trị ngoại lai và chuẩn hoá

```
#- xử lý các giá trị ngoại lai
#- chuẩn hóa và scale dữ liệu
df1 = df.drop(['First Name', 'Last Name', 'day_of_week'], axis=1)
array = df1.values
array
```

```
### sử dụng RobustScaler() để loại bỏ những giá trị ngoại lai
scaler = preprocessing.RobustScaler()
robust_df = scaler.fit_transform(array)
robust_df = pd.DataFrame(robust_df)
print('Su lieu duoc scale va bo gia tri ngoai lai bang Robust
Scaler\n')
robust_df
```

```
### Chuẩn hóa dữ liệu bằng phương pháp z-score (Standard)
scaler = preprocessing.StandardScaler()
standard = scaler.fit_transform(array)
standard_df = pd.DataFrame(standard, index = df.index)
print('Du lieu da duoc chuan hoa\n')
standard_df
```

```
### scale dữ liệu bằng phương pháp minmax
scaler = preprocessing.MinMaxScaler()
min_max = scaler.fit_transform(array)
min_max_df = pd.DataFrame(min_max, index = df.index)
print('Du lieu da duoc scale\n')
min_max_df
```

```
Du lieu da duoc chuan hoa
```

	0	1	2	3	4	5	6	7	8
ID									
1	-1.369782	-1.264391	-1.0	1.0	-0.5	-0.5	-0.654654	1.527525	-1.420508
2	-0.575077	-0.405854	1.0	-1.0	-0.5	-0.5	1.527525	-0.654654	-0.027853
3	1.809037	0.452683	-1.0	1.0	2.0	-0.5	-0.654654	-0.654654	0.807740
4	-0.098254	1.311220	1.0	-1.0	-0.5	2.0	-0.654654	-0.654654	-1.141977
5	-1.051900	-0.835122	-1.0	1.0	-0.5	-0.5	-0.654654	1.527525	-0.863446
6	-0.228297	0.023415	1.0	-1.0	-0.5	-0.5	1.527525	-0.654654	1.643333
7	0.855391	-0.210732	-1.0	1.0	2.0	-0.5	-0.654654	-0.654654	0.529209
8	1.332214	2.169757	1.0	-1.0	-0.5	2.0	-0.654654	-0.654654	1.364802
9	-0.892959	-0.405854	-1.0	1.0	-0.5	-0.5	-0.654654	1.527525	-0.306384
10	0.219627	-0.835122	1.0	-1.0	-0.5	-0.5	1.527525	-0.654654	-0.584915

Bước 8: rời rạc hoá dữ liệu
– 10 equi-width ranges (sử dụng mục 2.a): dùng hàm pd.cut()

```
####### 10 equi-width ranges với cột đầu tiên của standard_df
df2=standard_df.copy()
df2['equi-width_column0'] = pd.cut(x=df2[0], bins=10)
print('Roi rac hoa cot 0 bang 10 equi-width ranges:\n')
df2
```

```
Roi rac hoa cot 0 bang 10 equi-dpth ranges:
```

ID	0	1	2	3	4	5	6	7	8	equi-depth_column0
1	-1.369782	-1.264391	-1.0	1.0	-0.5	-0.5	-0.654654	1.527525	-1.420508	(-1.371, -1.084]
2	-0.575077	-0.405854	1.0	-1.0	-0.5	-0.5	1.527525	-0.654654	-0.027853	(-0.67, -0.367]
3	1.809037	0.452683	-1.0	1.0	2.0	-0.5	-0.654654	-0.654654	0.807740	(1.38, 1.809]
4	-0.098254	1.311220	1.0	-1.0	-0.5	2.0	-0.654654	-0.654654	-1.141977	(-0.163, 0.0289]
5	-1.051900	-0.835122	-1.0	1.0	-0.5	-0.5	-0.654654	1.527525	-0.863446	(-1.084, -0.925]
6	-0.228297	0.023415	1.0	-1.0	-0.5	-0.5	1.527525	-0.654654	1.643333	(-0.367, -0.163]
7	0.855391	-0.210732	-1.0	1.0	2.0	-0.5	-0.654654	-0.654654	0.529209	(0.41, 0.951]
8	1.332214	2.169757	1.0	-1.0	-0.5	2.0	-0.654654	-0.654654	1.364802	(0.951, 1.38]
9	-0.892959	-0.405854	-1.0	1.0	-0.5	-0.5	-0.654654	1.527525	-0.306384	(-0.925, -0.67]
10	0.219627	-0.835122	1.0	-1.0	-0.5	-0.5	1.527525	-0.654654	-0.584915	(0.0289, 0.41]

– 10 equi-depth ranges (sử dụng mục 2.b): dùng hàm pd.qcut()

```
####### 10 equi-depth ranges với cột đầu tiên của standard_df
df3=standard_df.copy()
df3['equi-depth_column0'] = pd.qcut(x=df3[0], q=10)
print('Roi rac hoa cot 0 bang 10 equi-dpth ranges:\n')
df3
```

```
Roi rac hoa cot 0 bang 10 equi-dpth ranges:
```

ID	0	1	2	3	4	5	6	7	8	equi-depth_column0
1	-1.369782	-1.264391	-1.0	1.0	-0.5	-0.5	-0.654654	1.527525	-1.420508	(-1.371, -1.084]
2	-0.575077	-0.405854	1.0	-1.0	-0.5	-0.5	1.527525	-0.654654	-0.027853	(-0.67, -0.367]
3	1.809037	0.452683	-1.0	1.0	2.0	-0.5	-0.654654	-0.654654	0.807740	(1.38, 1.809]
4	-0.098254	1.311220	1.0	-1.0	-0.5	2.0	-0.654654	-0.654654	-1.141977	(-0.163, 0.0289]
5	-1.051900	-0.835122	-1.0	1.0	-0.5	-0.5	-0.654654	1.527525	-0.863446	(-1.084, -0.925]
6	-0.228297	0.023415	1.0	-1.0	-0.5	-0.5	1.527525	-0.654654	1.643333	(-0.367, -0.163]
7	0.855391	-0.210732	-1.0	1.0	2.0	-0.5	-0.654654	-0.654654	0.529209	(0.41, 0.951]
8	1.332214	2.169757	1.0	-1.0	-0.5	2.0	-0.654654	-0.654654	1.364802	(0.951, 1.38]
9	-0.892959	-0.405854	-1.0	1.0	-0.5	-0.5	-0.654654	1.527525	-0.306384	(-0.925, -0.67]
10	0.219627	-0.835122	1.0	-1.0	-0.5	-0.5	1.527525	-0.654654	-0.584915	(0.0289, 0.41]

2. Bài tập vận dụng

a) Sơ lược về bộ dữ liệu

- Đây là bộ dữ liệu về rối loạn nhịp tim gồm 279 features với 206 features là định lượng, số còn lại là định tính
- Với kích thước 452 dòng và có missing values được đánh dấu là '?'
- Cột biến mục tiêu là cột cuối cùng trong bộ dữ liệu

b) Cài đặt thư viện scikit-learn và import thư viện

Em đã cài đặt thư viện này từ trước nên không phải cài đặt lại

```
!pip install scikit-learn
```
✓ 2.0s Python

```
Requirement already satisfied: scikit-learn in c:\users\hnam\appdata\local\programs\python\python312\lib\site-packages (1.4.1.post1)
Requirement already satisfied: numpy<2.0,≥1.19.5 in c:\users\hnam\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (1.
Requirement already satisfied: scipy≥1.6.0 in c:\users\hnam\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (1.13.0)
Requirement already satisfied: joblib≥1.2.0 in c:\users\hnam\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl≥2.0.0 in c:\users\hnam\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (
```

c) Import thư viện

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import KBinsDiscretizer
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
```

d) Load data

Đây là đoạn code sử dụng thư viện pandas để load dữ liệu dưới dạng DataFrame, các nội dung ngăn nhau bởi dấu " , " và file data này không có tiêu đề
Sau đó hiển thị ra df

```
df = pd.read_csv
('LAB 01/arrhythmia.data', delimiter=',',header=None)
# Chỉ hiện thị đoạn đầu của df
df.head()
```

5 dòng đầu của df:

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275	276	277	278	279
0	75	0	190	80	91	193	371	174	121	-16	...	0.0	9.0	-0.9	0.0	0.0	0.9	2.9	23.3	49.4	8
1	56	1	165	64	81	174	401	149	39	25	...	0.0	8.5	0.0	0.0	0.0	0.2	2.1	20.4	38.8	6
2	54	0	172	95	138	163	386	185	102	96	...	0.0	9.5	-2.4	0.0	0.0	0.3	3.4	12.3	49.0	10
3	55	0	175	94	100	202	380	179	143	28	...	0.0	12.2	-2.2	0.0	0.0	0.4	2.6	34.6	61.6	1
4	75	0	190	80	88	181	360	177	103	-16	...	0.0	13.1	-3.6	0.0	0.0	-0.1	3.9	25.4	62.8	7

5 rows × 280 columns

Tiếp theo là các bước làm sạch và tiền xử lý dữ liệu

e) Xử lý missing values

Vì giá trị khuyết trong bộ data này được đánh dấu bởi dấu '?' nên ta không thể dùng hàm isnull() để kiểm tra được mà cần thay thế dấu '?' này bằng giá trị NaN

```
# Chuyển '?' thành NaN, và chuyển các cột sang giá trị số
df.replace('?', np.nan, inplace=True)
```

Vì có một số feature không để ở dạng số (numeric) nên ta cần chuyển nó về dạng số

```
df = df.apply(pd.to_numeric)
```

Có thể gộp hai thứ trên với dòng lệnh dưới (tham số errors = 'coerce' có nghĩa với những giá trị không chuyển thành số được ('?) sẽ chuyển thành NaN)

```
#Tương đương với (các giá trị không phải số sẽ thành NaN)
# df = df.apply(pd.to_numeric, errors='coerce')
```

Bây giờ ta có thể kiểm tra missing values

```
# Kiểm tra giá trị khuyết
df.isnull().sum()
```

```
7        0
8        0
9        0
10       8
11      22
12       1
13     376
14       1
15       0
```

Sau khi kiểm tra ta tiến hành điền missing values với mean()

```
# Điền giá trị khuyết
df.fillna(df.mean(), inplace= True)
```

f) Xoá bỏ các hàng trùng lặp

Vì kiểm tra cho thấy data không có hàng trùng nên không cần drop

```
df.duplicated().sum()

# Ở đây không có dòng duplicate nên không cần drop

# df = df.drop_duplicates()
```

g) Scale và chuẩn hoá dữ liệu

Dưới đây là 3 cách chuẩn hoá dữ liệu: gồm Robust Scaler() scale và loại bỏ ngoại lai, Standard Scaler chuẩn hoá bằng Z-score và MinMaxScaler các data được scale về trong khoảng [0,1]

```
#Robust Scaler
scaler = RobustScaler()
robust_df = scaler.fit_transform(df)
robust_df = pd.DataFrame(robust_df)

#Standard Scaler
scaler = StandardScaler()
std = scaler.fit_transform(df)
std_df = pd.DataFrame(std)

#MinMaxScaler
scaler = MinMaxScaler()
min_max = scaler.fit_transform(df)
min_max_df = pd.DataFrame(min_max)
```

Ở đây ta chọn chuẩn hoá Standard Scaler để làm các bước tiếp theo

```
print('Du lieu chuan hoa\n')
std_df.head()
```

✓ 0.0s

```
Du lieu chuan hoa
```

	0	1	2	3	4	5	6	7	8	9	...	270	271	272	273	274	275
0	1.734439	-1.107520	0.641327	0.713814	0.135505	0.844945	0.113709	0.113809	1.201469	-1.094661	...	0.508843	-0.013839	0.278621	-0.079546	0.0	1.109553
1	0.579312	0.902918	-0.031998	-0.251644	-0.516072	0.420769	1.013301	-0.588564	-1.977064	-0.191203	...	0.508843	-0.157972	0.728573	-0.079546	0.0	-0.906889
2	0.457720	-1.107520	0.156533	1.618932	3.197915	0.175193	0.563505	0.422853	0.464980	1.373324	...	0.508843	0.130294	-0.471299	-0.079546	0.0	-0.618826
3	0.518516	-1.107520	0.237332	1.558590	0.721924	1.045871	0.383587	0.254284	2.054247	-0.125096	...	0.508843	0.908612	-0.371310	-0.079546	0.0	-0.330763
4	1.734439	-1.107520	0.641327	0.713814	-0.059968	0.577044	-0.216141	0.198094	0.503742	-1.094661	...	0.508843	1.168051	-1.071235	-0.079546	0.0	-1.771079

5 rows × 280 columns

h) Rời rạc hoá

Ta chỉ cần rời hoá các biến định lượng nên ta sẽ lọc những biến định tính ra khỏi df

```
nominal_columns = [1, 21, 22, 23, 24, 25, 26, 33, 34, 35, 36, 37, 38, 45, 46, 47, 48, 49, 50,
                   57, 58, 59, 60, 61, 62, 69, 70, 71, 72, 73, 74, 81, 82, 83, 84, 85, 86, 93,
                    94, 95, 96, 97, 98, 105, 106, 107, 108, 109, 110, 117, 118, 119, 120,
                   121, 122, 129, 130, 131, 132, 133, 134, 141, 142, 143, 144, 145, 146, 153,
                   154, 155, 156, 157, 158]

linear_columns = df.columns.difference(nominal_columns)
```

- 10 equi-width ranges

```
# Tạo bản sao cho std_df để không ảnh hưởng dữ liệu gốc
df2 = std_df.copy()
# Lấy các cột định lượng
df2 = df2[linear_columns]

for column in df2.columns:
    # Áp dụng pandas cut cho từng cột với bins = 10
    df2[column] = pd.cut(x=df2[column], bins=10)

# In ra các hàng đầu df
df2.head()
```

	0	2	3	4	5	6	7	8	9	10	...	270	271	272	273	274	275	276	277	278	279
0	(1.248, 2.221]	(0.237, 16.532]	(0.412, 0.714]	(0.0703, 0.201]	(0.51, 0.845]	(-0.00624, 0.204]	(-0.0548, 0.114]	(0.969, 4.458]	(-1.267, -0.786]	(-0.453, -0.299]	...	(-0.221, 0.509]	(-0.0715, 0.188]	(0.179, 0.329]	(-0.0805, 15.9]	NaN	(0.821, 1.11]	(1.108, 3.354]	(0.198, 0.391]	(0.839, 1.268]	(0.0271, 0.936]
1	(0.275, 0.579]	(-0.0589, -0.032]	(-0.493, -0.191]	(-0.699, -0.516]	(0.325, 0.51]	(0.623, 1.193]	(-0.729, -0.532]	(-3.4899999999999998, -0.892]	(-0.493, -0.147]	(-0.055, 0.0847]	...	(-0.221, 0.509]	(-0.273, -0.0715]	(0.329, 0.729]	(-0.0805, 15.9]	NaN	(-1.166, -0.619]	(0.476, 0.757]	(-0.0909, 0.198]	(0.458, 0.839]	(0.0271, 0.936]
2	(0.275, 0.579]	(0.13, 0.237]	(1.076, 6.507]	(0.911, 6.456]	(0.166, 0.325]	(0.414, 0.623]	(0.114, 0.502]	(0.387, 0.581]	(1.065, 2.982]	(-0.055, 0.0847]	...	(-0.221, 0.509]	(-0.0715, 0.188]	(-0.971, -0.471]	(-0.0805, 15.9]	NaN	(-1.166, -0.619]	(1.108, 3.354]	(-0.721, -0.491]	(0.839, 1.268]	(0.936, 1.39]
3	(0.275, 0.579]	(0.13, 0.237]	(1.076, 6.507]	(0.461, 0.911]	(0.845, 8.235]	(0.204, 0.414]	(0.114, 0.502]	(0.969, 4.458]	(-0.147, 0.139]	(-0.453, -0.299]	...	(-0.221, 0.509]	(0.822, 1.255]	(-0.471, -0.221]	(-0.0805, 15.9]	NaN	(-0.619, -0.331]	(0.757, 1.108]	(0.642, 1.147]	(1.268, 4.678]	(-0.655, -0.427]
4	(1.248, 2.221]	(0.237, 16.532]	(0.412, 0.714]	(-0.19, 0.0703]	(0.51, 0.845]	(-0.366, -0.204]	(0.114, 0.502]	(0.387, 0.581]	(-1.267, -0.786]	(-0.453, -0.299]	...	(-0.221, 0.509]	(0.822, 1.255]	(-13.571, -0.971]	(-0.0805, 15.9]	NaN	(-3.7889999999999997, -1.166]	(1.108, 3.354]	(0.391, 0.642]	(1.268, 4.678]	(0.0271, 0.936]

5 rows × 207 columns

- 10 equi-depth ranges
-

Với chia theo chiều sâu, các nhóm được chọn sao cho số bản ghi của mỗi nhóm là bằng nhau với số nhóm theo yêu cầu.

Trong trường hợp mà dữ liệu có quá nhiều dữ liệu trùng lặp hoặc quá ít giá trị độc nhất thì các nhóm có thể bị trùng nhau gây ảnh hưởng đến tính chất của dữ liệu nên ta phải drop những vùng này bằng tham số duplicates = ‘drop’ .

```
df3 = std_.copy()
df3 = df3[linear_columns]

for column in df3.columns:
    df3[column] = pd.qcut(x=df3[column], q = 10, duplicates='drop')

df3.head()
```

	0	2	3	4	5	6	7	8	9	10	...	270	271	272	273	274	275	276	277	278	279
0	(1.248, 2.221]	(0.237, 16.532]	(0.412, 0.714]	(0.0703, 0.201]	(0.51, 0.845]	(-0.00624, 0.204]	(-0.0548, 0.114]	(0.969, 4.458]	(-1.267, -0.786]	(-0.453, -0.299]	...	(-0.221, 0.509]	(-0.0715, 0.188]	(0.179, 0.329]	(-0.0805, 15.9]	NaN	(0.821, 1.11]	(1.108, 3.354]	(0.198, 0.391]	(0.839, 1.268]	(0.0271, 0.936]
1	(0.275, 0.579]	(-0.0589, -0.032]	(-0.493, -0.191]	(-0.699, -0.516]	(0.325, 0.51]	(0.623, 1.193]	(-0.729, -0.532]	(-3.4899999999999998, -0.892]	(-0.493, -0.147]	(-0.055, 0.0847]	...	(-0.221, 0.509]	(-0.273, -0.0715]	(0.329, 0.729]	(-0.0805, 15.9]	NaN	(-1.166, -0.619]	(0.476, 0.757]	(-0.0909, 0.198]	(0.458, 0.839]	(0.0271, 0.936]
2	(0.275, 0.579]	(0.13, 0.237]	(1.076, 6.507]	(0.911, 6.456]	(0.166, 0.325]	(0.414, 0.623]	(0.114, 0.502]	(0.387, 0.581]	(1.065, 2.982]	(-0.055, 0.0847]	...	(-0.221, 0.509]	(-0.0715, 0.188]	(-0.971, -0.471]	(-0.0805, 15.9]	NaN	(-1.166, -0.619]	(1.108, 3.354]	(-0.721, -0.491]	(0.839, 1.268]	(0.936, 1.39]
3	(0.275, 0.579]	(0.13, 0.237]	(1.076, 6.507]	(0.461, 0.911]	(0.845, 8.235]	(0.204, 0.414]	(0.114, 0.502]	(0.969, 4.458]	(-0.147, 0.139]	(-0.453, -0.299]	...	(-0.221, 0.509]	(0.822, 1.255]	(-0.471, -0.221]	(-0.0805, 15.9]	NaN	(-0.619, -0.331]	(0.757, 1.108]	(0.642, 1.147]	(1.268, 4.678]	(-0.655, -0.427]
4	(1.248, 2.221]	(0.237, 16.532]	(0.412, 0.714]	(-0.19, 0.0703]	(0.51, 0.845]	(-0.366, -0.204]	(0.114, 0.502]	(0.387, 0.581]	(-1.267, -0.786]	(-0.453, -0.299]	...	(-0.221, 0.509]	(0.822, 1.255]	(-13.571, -0.971]	(-0.0805, 15.9]	NaN	(-3.7889999999999997, -1.166]	(1.108, 3.354]	(0.391, 0.642]	(1.268, 4.678]	(0.0271, 0.936]

5 rows × 207 columns

3. PCA

a) Load data và tách X,y

Nhập dữ liệu bằng thư viện (online)

```
#Load bằng thư viện (online)

from ucimlrepo import fetch_ucirepo

# fetch dataset
musk_version_1 = fetch_ucirepo(id=74)

# data (as pandas dataframes)
X = musk_version_1.data.features
y = musk_version_1.data.targets
X = X.drop(['molecule_name', 'conformation_name'], axis=1)
```

Nhập dữ liệu bằng file

```
# Load bằng file (local)
musk_version_1 = pd.read_csv('LAB 01/clean1.data',
delimiter=',',header=None)
pd.set_option('display.max_rows', None)
df.head()

# Tách X,y
X = musk_version_1.iloc[:, 2:-1]
y = musk_version_1.iloc[:, -1]
```

b) Chuẩn hoá dữ liệu

```
scaler = StandardScaler()
X_std = scaler.fit_transform(X)
```

c) Tiến hành PCA

Thực hiện thuật toán PCA với số thành phần chính giữ lại là 2

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

X_pca = pca.fit_transform(X_std)
```

Các vector riêng và trị riêng lần lượt là

```
eigenvectors = pca.components_
print("Eigenvectors:")
print(eigenvectors)
```

```
Eigenvectors:
[[ 3.68698531e-02  6.83111424e-02  9.48053725e-02 -9.25859839e-02
   9.00473084e-04  6.09551501e-02 -1.25062362e-01  5.58131913e-03
  -1.24083914e-01  8.62403142e-02 -3.71693280e-02 -5.76714000e-02
   8.57068196e-04 -2.02699252e-03 -6.02295444e-03  5.84143220e-03
  -1.22013120e-01 -6.91416790e-03 -4.83480633e-02  3.07739083e-02
  -9.54809181e-02 -1.26712611e-01 -1.17147875e-01  1.15541364e-01
   1.21644756e-01 -1.18388035e-01 -9.07931865e-02 -7.32678677e-02
  -5.77062987e-02  1.23635183e-01  1.81526293e-02  1.07793223e-02
  -8.45717016e-02  1.13458177e-01 -1.10275730e-01  5.32722990e-02
  -3.45904361e-02 -9.89882830e-02  9.32206484e-02 -8.62779151e-02
   8.99903712e-02  1.96823063e-02  4.01366499e-02 -2.16287911e-02
  -2.49877139e-02 -3.05508969e-02 -1.71025631e-02 -1.00801521e-01
  -8.30083401e-03  7.50975451e-02 -7.18098225e-02 -1.26625307e-01
  -1.25194714e-01 -3.97046134e-02  8.82637054e-02 -7.06470907e-02
  -1.23431331e-01  2.42818647e-02  1.52772575e-03  1.15603400e-01
  -8.26278306e-02  3.55783589e-02 -3.25128309e-02  1.22809879e-01
  -9.87151885e-02  4.87277179e-02  6.81757224e-04  3.24641970e-02
   5.48237868e-02  2.58031185e-02  9.35888592e-02  1.00031688e-01
  -2.98026117e-02 -2.10980975e-02  1.82529832e-02  6.02179891e-03
  -1.24757548e-01 -9.97120897e-05 -6.16432834e-02 -3.86337461e-02
  -1.15717322e-01 -1.25266186e-01 -9.19184063e-02  4.88688475e-02
   1.22522024e-01 -1.24073488e-01  3.26324568e-02 -5.22407206e-02
  -8.79366116e-02  1.19782676e-01  4.45936532e-02  2.04434936e-02
  -1.10161609e-01  1.63200754e-02 -6.94749253e-02  3.36916767e-02
 ...
  -3.77746936e-03  1.55486774e-02  5.23883961e-03  1.07309993e-02
  -3.62512392e-02 -4.72480408e-02 -2.03576423e-02  1.69028509e-02
   5.29270171e-03  7.50248850e-04  2.12310545e-02 -2.91685897e-02
  -3.19654663e-02  1.17401001e-02]]
```

```python
eigenvalues = pca.explained_variance_
print("\nEigenvalues:")
print(eigenvalues)
```

```
Eigenvalues:
[51.88128199 23.15904633]
```