Chapter5 - Ex2: Chronic_Kidney_Disease Data Set - Full

- Cho dữ liệu chronic_kidney_disease.csv chứa thông tin của các bệnh nhân. Bộ dữ liệu này có thể được sử dụng để dự đoán bệnh thận mãn tính và nó được thu thập trong bệnh viện gần 2 tháng. ## Thông tin dữ liệu:
- Dữ liệu có thể tham khảo và download tại: https://archive.ics.uci.edu/ml/datasets/chronic_kidney_disease#### Data Information:
- age - age
- bp - blood pressure
- sg - specific gravity
- al - albumin
- su - sugar
- rbc - red blood cells
- pc - pus cell
- pcc - pus cell clumps
- ba - bacteria
- bgr - blood glucose random
- bu - blood urea
- sc - serum creatinine
- sod - sodium
- pot - potassium
- hemo - hemoglobin
- pcv - packed cell volume
- wc - white blood cell count
- rc - red blood cell count
- htn - hypertension
- dm - diabetes mellitus
- cad - coronary artery disease
- appet - appetite
- pe - pedal edema
- ane - anemia
- class - class

Yêu cầu:

- Đọc dữ liệu, tìm hiểu sơ bộ về dữ liệu
- Chọn phương pháp để chuẩn hóa dữ liệu và thực hiện việc chuẩn hóa.

In [1]:
```
import pandas as pd
import numpy as numpy
```

In [2]:
```
dataset = pd.read_csv('chronic_kidney_disease.csv', header=None,
                      names=['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba',
```

```
                                'bgr', 'bu', 'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc',
                                'rc', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane', 'class'])
print(dataset.shape)
dataset.info()
```

```
(400, 25)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 25 columns):
age      400 non-null object
bp       400 non-null object
sg       400 non-null object
al       400 non-null object
su       400 non-null object
rbc      400 non-null object
pc       400 non-null object
pcc      400 non-null object
ba       400 non-null object
bgr      400 non-null object
bu       400 non-null object
sc       400 non-null object
sod      400 non-null object
pot      400 non-null object
hemo     400 non-null object
pcv      400 non-null object
wc       400 non-null object
rc       400 non-null object
htn      400 non-null object
dm       400 non-null object
cad      400 non-null object
appet    400 non-null object
pe       400 non-null object
ane      400 non-null object
class    400 non-null object
dtypes: object(25)
memory usage: 78.2+ KB
```

In [3]:

```
dataset.head()
```

Out[3]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet
0	48	80	1.02	1	0	?	normal	notpresent	notpresent	121	...	44	7800	5.2	yes	yes	no	good
1	7	50	1.02	4	0	?	normal	notpresent	notpresent	?	...	38	6000	?	no	no	no	good
2	62	80	1.01	2	3	normal	normal	notpresent	notpresent	423	...	31	7500	?	no	yes	no	poor
3	48	70	1.01	4	0	normal	abnormal	present	notpresent	117	...	32	6700	3.9	yes	no	no	poor
4	51	80	1.01	2	0	normal	normal	notpresent	notpresent	106	...	35	7300	4.6	no	no	no	good

5 rows × 25 columns

In [4]:

```
import numpy as np
dataset = dataset.replace('?', np.nan)
```

In [5]:

```
dataset.head()
```

Out[5]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appe

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appe
0	48	80	1.02	1	0	NaN	normal	notpresent	notpresent	121	...	44	7800	5.2	yes	yes	no	good
1	7	50	1.02	4	0	NaN	normal	notpresent	notpresent	NaN	...	38	6000	NaN	no	no	no	good
2	62	80	1.01	2	3	normal	normal	notpresent	notpresent	423	...	31	7500	NaN	no	yes	no	poo
3	48	70	1.01	4	0	normal	abnormal	present	notpresent	117	...	32	6700	3.9	yes	no	no	poo
4	51	80	1.01	2	0	normal	normal	notpresent	notpresent	106	...	35	7300	4.6	no	no	no	good

5 rows × 25 columns

In [6]:
```
dataset.columns
```

Out[6]:
```
Index(['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu',
       'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad',
       'appet', 'pe', 'ane', 'class'],
      dtype='object')
```

In [7]:
```
cols = ['age', 'bp', 'sg', 'al', 'su', 'bgr', 'bu', 'sc', 'sod', 'pot', 'hemo', 'pcv', 'wc', 'rc']
dataset[cols] = dataset[cols].apply(pd.to_numeric, errors='coerce', axis=1)

# df[['col.name1', 'col.name2'...]] = df[['col.name1', 'col.name2'..]].astype('data_type')
```

In [8]:
```
dataset.head()
```

Out[8]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad
0	48.0	80.0	1.02	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44.0	7800.0	5.2	yes	yes	no
1	7.0	50.0	1.02	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38.0	6000.0	NaN	no	no	no
2	62.0	80.0	1.01	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...	31.0	7500.0	NaN	no	yes	no
3	48.0	70.0	1.01	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32.0	6700.0	3.9	yes	no	no
4	51.0	80.0	1.01	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...	35.0	7300.0	4.6	no	no	no

5 rows × 25 columns

In [9]:
```
# chuyên chuỗi số sang số
dataset.age = pd.to_numeric(dataset.age)
dataset.bp = pd.to_numeric(dataset.bp)
dataset.sg = pd.to_numeric(dataset.sg)
dataset.al = pd.to_numeric(dataset.al)
dataset.su = pd.to_numeric(dataset.su)
dataset.bgr = pd.to_numeric(dataset.bgr)
dataset.bu = pd.to_numeric(dataset.bu)
dataset.sc = pd.to_numeric(dataset.sc)
dataset.sod = pd.to_numeric(dataset.sod)
dataset.pot = pd.to_numeric(dataset.pot)
dataset.hemo = pd.to_numeric(dataset.hemo)
dataset.pcv = pd.to_numeric(dataset.pcv)
dataset.wc = pd.to_numeric(dataset.wc)
dataset.rc = pd.to_numeric(dataset.rc)
```

In [10]:
```
# dataset.info()
```

In [11]:
```
dataset.head()
```

Out[11]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad
0	48.0	80.0	1.02	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44.0	7800.0	5.2	yes	yes	no
1	7.0	50.0	1.02	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38.0	6000.0	NaN	no	no	no
2	62.0	80.0	1.01	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...	31.0	7500.0	NaN	no	yes	no
3	48.0	70.0	1.01	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32.0	6700.0	3.9	yes	no	no
4	51.0	80.0	1.01	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...	35.0	7300.0	4.6	no	no	no

5 rows × 25 columns

In [12]:
```
dataset.isna().sum()
```

Out[12]:
```
age        9
bp        12
sg        47
al        46
su        49
rbc      152
pc        65
pcc        4
ba         4
bgr       44
bu        19
sc        17
sod       87
pot       88
hemo      52
pcv       71
wc       106
rc       131
htn        2
dm         2
cad        2
appet      1
pe         1
ane        1
class      0
dtype: int64
```

Nhận xét:

- Trường hợp đơn giản nhất là xóa tất cả các dòng có na.
- Tuy nhiên, như vậy thì mất rất nhiều dữ liệu, ảnh hưởng đến kết quả.

In [13]:
```
dataset = dataset.dropna()
dataset.head()
```

Out[13]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	ca

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	ca
3	48.0	70.0	1.01	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32.0	6700.0	3.9	yes	no	n
9	53.0	90.0	1.02	2.0	0.0	abnormal	abnormal	present	notpresent	70.0	...	29.0	12100.0	3.7	yes	yes	n
11	63.0	70.0	1.01	3.0	0.0	abnormal	abnormal	present	notpresent	380.0	...	32.0	4500.0	3.8	yes	yes	n
14	68.0	80.0	1.01	3.0	2.0	normal	abnormal	present	present	157.0	...	16.0	11000.0	2.6	yes	yes	y
20	61.0	80.0	1.02	2.0	0.0	abnormal	abnormal	notpresent	notpresent	173.0	...	24.0	9200.0	3.2	yes	yes	y

5 rows × 25 columns

In [14]:
```
# Categorical boolean mask
categorical_feature_mask = dataset.dtypes==object
# filter categorical columns using mask and turn it into a list
categorical_cols = dataset.columns[categorical_feature_mask].tolist()
categorical_cols
```

Out[14]:
```
['rbc', 'pc', 'pcc', 'ba', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane', 'class']
```

In [15]:
```
X = dataset.drop(columns='class', axis=1)
```

In [16]:
```
y = dataset['class']
```

In [17]:
```
X = pd.get_dummies(data=X, columns=categorical_cols[:-1], drop_first=True)
```

In [18]:
```
X.head()
```

Out[18]:

	age	bp	sg	al	su	bgr	bu	sc	sod	pot	...	rbc_normal	pc_normal	pcc_present	ba_present	ht
3	48.0	70.0	1.01	4.0	0.0	117.0	56.0	3.8	111.0	2.5	...	1	0	1	0	
9	53.0	90.0	1.02	2.0	0.0	70.0	107.0	7.2	114.0	3.7	...	0	0	1	0	
11	63.0	70.0	1.01	3.0	0.0	380.0	60.0	2.7	131.0	4.2	...	0	0	1	0	
14	68.0	80.0	1.01	3.0	2.0	157.0	90.0	4.1	130.0	6.4	...	1	0	1	1	
20	61.0	80.0	1.02	2.0	0.0	173.0	148.0	3.9	135.0	5.2	...	0	0	0	0	

5 rows × 24 columns

In [19]:
```
# Đếm theo loại
occ = y.value_counts()
occ
```

Out[19]:
```
notckd    115
ckd        43
Name: class, dtype: int64
```

In [20]:

```
y = y.replace('notckd', 0)
y = y.replace('ckd', 1)
```

In [21]:
```
y.head()
```

Out[21]:
```
3     1
9     1
11    1
14    1
20    1
Name: class, dtype: int64
```

Nhận xét:

- Dữ liệu cũng không cân bằng => có thể dẫn đến kết quả không chính xác
- Nếu dùng Logistic Regression thì thực hiện trước với dữ liệu gốc => xem kết quả. Nếu không tốt thì thử áp dụng cân bằng dữ liệu.

Áp dụng thuật toán

In [22]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.20)
```

In [23]:
```
from sklearn.linear_model import LogisticRegression
```

In [24]:
```
model = LogisticRegression(solver='liblinear')
```

In [25]:
```
model.fit(X_train,y_train)
```

Out[25]:
```
LogisticRegression(solver='liblinear')
```

In [26]:
```
model.score(X_train,y_train)
```

Out[26]:
```
1.0
```

In [27]:
```
model.score(X_test,y_test)
```

Out[27]:
```
0.96875
```

In [28]:
```
y_pred = model.predict(X_test)
```

In [29]:
```
# Xem kết quả thống kê
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[21  0]
 [ 1 10]]
```

	precision	recall	f1-score	support
0	0.95	1.00	0.98	21
1	1.00	0.91	0.95	11
accuracy			0.97	32
macro avg	0.98	0.95	0.96	32
weighted avg	0.97	0.97	0.97	32

Nhận xét:

- Model cho kết quả tốt ở cả train và test
- Confussion matrix cũng cho thấy kết quả cao