STATS 3DA3

Project Chronic Kidney Disease Classification Challenge

Group 3
Xiangdong Wang (400335790)
Lingyun Huang (Student ID)
Jingyang Li (Student ID)

2024-04-18

```
pip install ucimlrepo
```

```
Requirement already satisfied: ucimlrepo in /Library/Frameworks/Python.framework/Versions/3.11/
Note: you may need to restart the kernel to use updated packages.
```

```
from ucimlrepo import fetch_ucirepo
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

1. Classification Problem Identification

Dataset is used from the Early Stage of Indians Chronic Kidney Disease (CKD) project, which comprises data on 250 early-stage CKD patients and 150 healthy controls.

In this assignment, machine learning (ML) techniques have been deployed to predict, diagnose, and treat chronic kidney disease (CKD).

```
## Load Dataset
data_url = 'https://archive.ics.uci.edu/static/public/336/data.csv'
df = pd.read_csv(data_url)
df.head(2)
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wbcc	rbcc	ht
0	48.0	80.0	1.02	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44.0	7800.0	5.2	ye
1	7.0	50.0	1.02	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38.0	6000.0	NaN	no

```
# fetch dataset
chronic_kidney_disease = fetch_ucirepo(id=336)
# metadata
print(chronic_kidney_disease.metadata)
```

```
{'uci_id': 336, 'name': 'Chronic Kidney Disease', 'repository_url': 'https://archive.ics.uci.e
```

```
# data (as pandas dataframes)
X = chronic_kidney_disease.data.features
y = chronic_kidney_disease.data.targets
```

```
# Features
X.head(2)
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	hemo	pcv	wbcc	r
0	48.0	80.0	1.02	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	15.4	44.0	7800.0	5
1	7.0	50.0	1.02	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	11.3	38.0	6000.0	N

```
# Target
y.head(2)
```

	class
0	ckd
1	ckd

The classification problem is determining whether a patient has early-stage CKD based on various medical measurements included in the dataset. There are two classes here: Early-stage Indian CKD patients and Healthy patients.

2. Variable Transformation

```
df.dtypes
```

```
age       float64
bp        float64
sg        float64
al        float64
```

```
su       float64
rbc       object
pc        object
pcc       object
ba        object
bgr      float64
bu       float64
sc       float64
sod      float64
pot      float64
hemo     float64
pcv      float64
wbcc     float64
rbcc     float64
htn       object
dm        object
cad       object
appet     object
pe        object
ane       object
class     object
dtype: object
```

From the dictionary `sg`, `al`, `su` are Categorical variables. `age`, `bp`, `bgr`, `bu`, `sod`, `pcv`, `wbcc` are Integer variable. `rbc`, `pc`, `pcc`, `ba`, `htn`, `dm`, `cad`, `appet`, `pe`, `ane`, `class` are Binary variables. `sc`,`pot`,`hemo`,and `rbcc` are continuous varibles. Then, we need to transform the data type.

In general, we do not need to convert categorical and binary variables. Since the classification algorithm is sensitive to the scale of the data, we choose to standardize those data under integer and continuous variable.

```python
from sklearn.preprocessing import StandardScaler

#df_scale = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
#df_scale

scale = ['age','bp','bgr','bu','sod','pcv','wbcc','sc','pot','hemo','rbcc']
scaler = StandardScaler()
df[scale] = scaler.fit_transform(df[scale])
df.head(2)
```

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv
0	-0.203139	0.258373	1.02	1.0	0.0	NaN	normal	notpresent	notpresent	-0.341498	...	0.569881
1	-2.594124	-1.936857	1.02	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	-0.098536

3. Dataset Overview

```python
df.describe
```

```
<bound method NDFrame.describe of           age        bp     sg   al   su     rbc         pc
0   -0.203139  0.258373  1.020  1.0  0.0     NaN     normal  notpresent
1   -2.594124 -1.936857  1.020  4.0  0.0     NaN     normal  notpresent
2    0.613295  0.258373  1.010  2.0  3.0  normal     normal  notpresent
3   -0.203139 -0.473370  1.005  4.0  0.0  normal   abnormal     present
4   -0.028189  0.258373  1.010  2.0  0.0  normal     normal  notpresent
..        ...       ...    ...  ...  ...     ...        ...         ...
395  0.205078  0.258373  1.020  0.0  0.0  normal     normal  notpresent
396 -0.553039 -0.473370  1.025  0.0  0.0  normal     normal  notpresent
397 -2.302541  0.258373  1.020  0.0  0.0  normal     normal  notpresent
398 -2.010957 -1.205114  1.025  0.0  0.0  normal     normal  notpresent
399  0.380028  0.258373  1.025  0.0  0.0  normal     normal  notpresent
```

```
             ba       bgr  ...       pcv      wbcc      rbcc  htn   dm  cad  \
0    notpresent -0.341498  ...  0.569881 -0.206202  0.481295  yes  yes   no
1    notpresent       NaN  ... -0.098536 -0.818559       NaN   no   no   no
2    notpresent  3.473064  ... -0.878356 -0.308261       NaN   no  yes   no
3    notpresent -0.392022  ... -0.766953 -0.580420 -0.788961  yes   no   no
4    notpresent -0.530963  ... -0.432744 -0.376301 -0.104977   no   no   no
..          ...       ...  ...       ...       ...       ...  ...  ...  ...
395  notpresent -0.101509  ...  0.904090 -0.580420  0.188159   no   no   no
396  notpresent -0.922524  ...  1.683910 -0.206202  1.458415   no   no   no
397  notpresent -0.606749  ...  1.126896 -0.614440  0.676719   no   no   no
398  notpresent -0.429915  ...  1.349701 -0.410321  1.165279   no   no   no
399  notpresent -0.215188  ...  1.572507 -0.546400  1.360703   no   no   no

    appet   pe  ane   class
0    good   no   no     ckd
1    good   no   no     ckd
2    poor   no  yes     ckd
3    poor  yes  yes     ckd
4    good   no   no     ckd
..    ...  ...  ...     ...
395  good   no   no  notckd
396  good   no   no  notckd
397  good   no   no  notckd
398  good   no   no  notckd
399  good   no   no  notckd

[400 rows x 25 columns]>
```

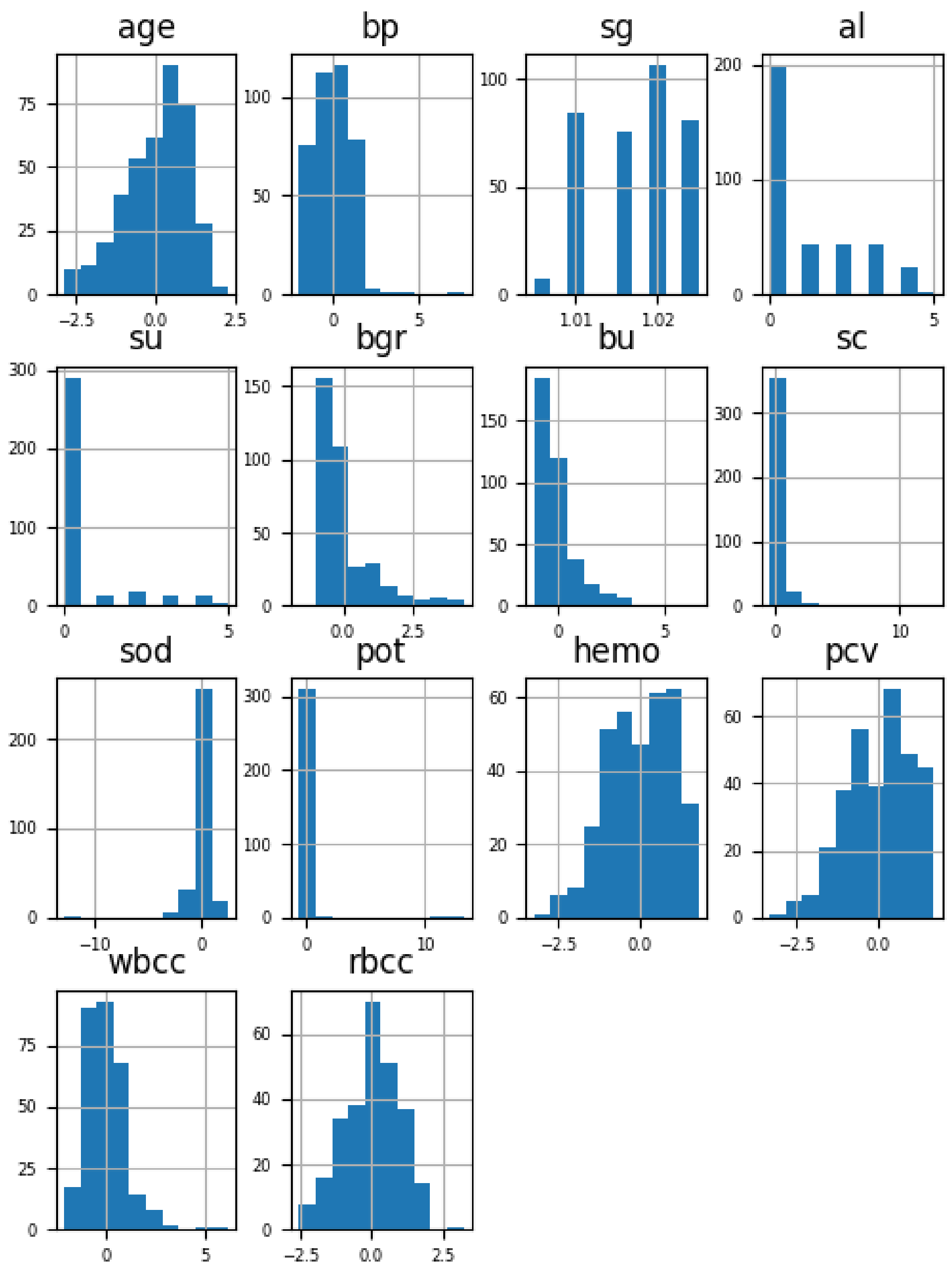
```
# Observations count
print(df.shape)
# type check
df.dtypes
```

```
(400, 25)
```

```
age       float64
bp        float64
sg        float64
al        float64
su        float64
rbc        object
pc         object
pcc        object
ba         object
bgr       float64
bu        float64
sc        float64
sod       float64
pot       float64
hemo      float64
pcv       float64
wbcc      float64
rbcc      float64
htn        object
dm         object
cad        object
appet      object
pe         object
ane        object
class      object
dtype: object
```

```
## Check the distribution of each variable
df.hist(xlabelsize=6,ylabelsize=6,figsize=(6,8))
```

```
array([[<Axes: title={'center': 'age'}>, <Axes: title={'center': 'bp'}>,
  <Axes: title={'center': 'sg'}>, <Axes: title={'center': 'al'}>],
 [<Axes: title={'center': 'su'}>, <Axes: title={'center': 'bgr'}>,
  <Axes: title={'center': 'bu'}>, <Axes: title={'center': 'sc'}>],
 [<Axes: title={'center': 'sod'}>, <Axes: title={'center': 'pot'}>,
  <Axes: title={'center': 'hemo'}>,
  <Axes: title={'center': 'pcv'}>],
 [<Axes: title={'center': 'wbcc'}>,
  <Axes: title={'center': 'rbcc'}>, <Axes: >, <Axes: >]],
dtype=object)
```


age
bp
sg
al
su
bgr
bu
sc
sod
pot
hemo
pcv
wbcc
rbcc