Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №3
по дисциплине
«Методы машинного обучения»

Выполнил:
студент группы ИУ5-22М

Лю Ченхао

Москва — 2024 г.

1. Цель лабораторной работы

изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

2. Основная часть

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data=pd.read_csv("bank-full.csv",encoding='utf-8')
data.head()
```

5 rows × 8 columns pd.DataFrame

	age	marital	education	default	balance	housing	loan	duration
0	58	married	tertiary	no	2143	yes	no	261
2	33	married	secondary	no	2	yes	yes	76
3	47	married	unknown	no	1506	yes	no	92
1	44	single	secondary	no	29	yes	no	151
4	33	single	unknown	no	1	no	no	198

In 39

```
data.info()
```

Executed at 2024.06.06 20:50:32 in 100ms

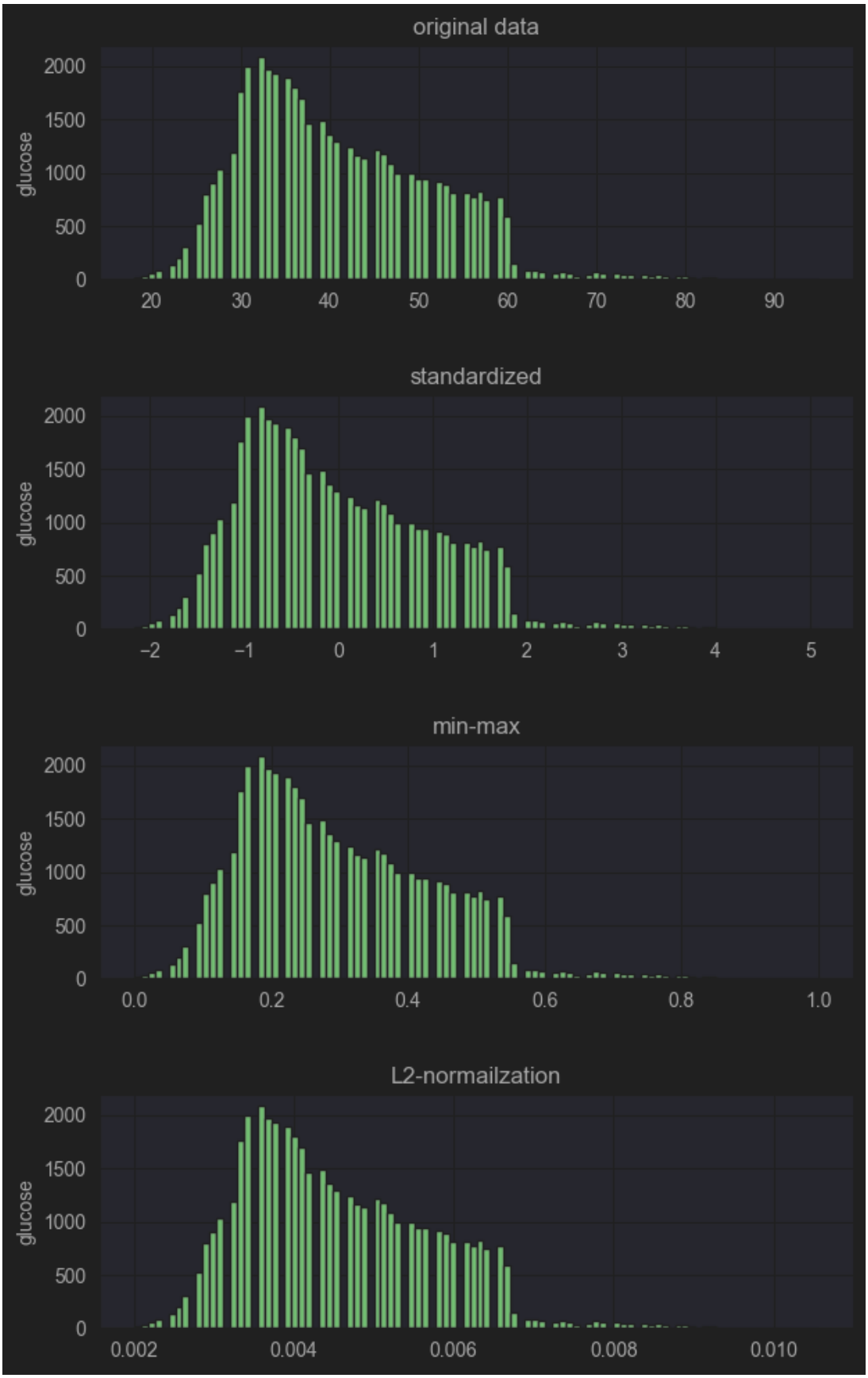
```
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   marital    45211 non-null  object
 2   education  45211 non-null  object
 3   default    45211 non-null  object
 4   balance    45211 non-null  int64
 5   housing    45211 non-null  object
 6   loan       45211 non-null  object
 7   duration   45211 non-null  int64
dtypes: int64(3), object(5)
memory usage: 2.8+ MB
```

8 rows 8 rows × 3 columns pd.DataFrame

	age	balance	duration
count	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	258.163080
std	10.618762	3044.765829	257.527812
min	18.000000	-8019.000000	0.000000
25%	33.000000	72.000000	103.000000
50%	39.000000	448.000000	180.000000
75%	48.000000	1428.000000	319.000000
max	95.000000	102127.000000	4918.000000

3. масштабирование признаков (не менее чем тремя способами)

```
data['standardized_n'] = preproc.StandardScaler().fit_transform(data[['age']])
data['minmax_n'] = preproc.minmax_scale(data[['age']])
data['l2_normalized_n'] = preproc.normalize(data[['age']], axis = 0)

fig,(ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize = (7, 12))
plt.subplots_adjust(wspace =0, hspace =0.5)

ax1.hist(data['age'], bins = 100, color = 'g')
ax1.set_xlabel('')
ax1.set_ylabel('glucose')
ax1.set_title('original data')

ax2.hist(data['standardized_n'], bins = 100, color = 'g')
ax2.set_xlabel('')
ax2.set_ylabel('glucose')
ax2.set_title('standardized')

ax3.hist(data['minmax_n'], bins = 100, color = 'g')
ax3.set_xlabel('')
ax3.set_ylabel('glucose')
ax3.set_title('min-max')

ax4.hist(data['l2_normalized_n'], bins = 100, color = 'g')
ax4.set_xlabel('')
ax4.set_ylabel('glucose')
ax4.set_title('L2-normailzation')
```


original data
glucose
2000
1500
1000
500
0
20
30
40
50
60
70
80
90
standardized
glucose
2000
1500
1000
500
0
−2
−1
0
1
2
3
4
5
min-max
glucose
2000
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0
L2-normailzation
glucose
2000
1500
1000
500
0
0.002
0.004
0.006
0.008
0.010

4. обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);

```
u = data['age'].mean()
std = data['age'].std()
error = data[np.abs(data['age'] - u) > 3*std]
data_c = data[np.abs(data['age'] - u) <= 3*std]
print(data_c.head())
print(error.head())
```

Executed at 2024.06.06 20:50:33 in 20ms

```
   age  marital  education default  balance housing loan  duration  \
0   58  married   tertiary      no     2143     yes   no       261
1   44   single  secondary      no       29     yes   no       151
2   33  married  secondary      no        2     yes  yes        76
3   47  married    unknown      no     1506     yes   no        92
4   33   single    unknown      no        1      no   no       198

   standardized_n  minmax_n  l2_normalized_n
0        1.606965  0.519481         0.006450
1        0.288529  0.337662         0.004893
2       -0.747384  0.194805         0.003670
```

5. обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);

```
from sklearn.preprocessing import LabelEncoder


gle = LabelEncoder()
genre_labels = gle.fit_transform(data['education'])
genre_mappings = {index: label for index, label in enumerate(gle.classes_)}
genre_mappings
```

Executed at 2024.06.06 21:57:28 in 22ms

```
{0: 0, 1: 1, 2: 2, 3: 3}
```

6. отбор признаков:

6.1один метод из группы методов фильтрации (filter methods);

```
#Filter
from sklearn.feature_selection import VarianceThreshold

selector = VarianceThreshold(threshold=3)
x_selected = selector.fit_transform(x)

print('Selected X Shape: ', x_selected.shape)
```

Executed at 2024.06.06 21:10:22 in 14ms

```
array([[ 67.  , 228.69,  36.6 ],
       [ 61.  , 202.21,    nan],
       [ 80.  , 105.92,  32.5 ],
       ...,
       [ 35.  ,  82.99,  30.6 ],
       [ 51.  , 166.29,  25.6 ],
       [ 44.  ,  85.28,  26.2 ]])
```

6.2один метод из группы методов обертывания (wrapper methods);

```python
#Wrapper
from imblearn.over_sampling import BorderlineSMOTE
from sklearn.feature_selection import RFE, SelectFromModel
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

features = ['age','marital',
 'education',
 'default',
 'balance',
 'housing',
 'loan',
]

label = ['duration']

X_1 = data[features]
y_1 = data[label]
X_1.loc[:, 'age'] = X_1['age'].fillna(28.74)
X_1.loc[:, 'marital'] = X_1['marital'].fillna(1)

print(X_1.head())
print(y_1.head())
print(y_1['duration'].value_counts())
```

Executed at 2024.06.06 22:02:38 in 33ms

```
from imblearn.over_sampling import SMOTE

smote = SMOTE()
x_smote, y_smote = smote.fit_resample(X_1, y_1)

from sklearn.model_selection import train_test_split
X_train,X_test, y_train,y_test=train_test_split(x_smote,y_smote,test_size=0.33,random_state=42)

from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

RFE(estimator=LogisticRegression(), n_features_to_select=2).fit_transform(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:760: DataConversionWarning:...
```

```
array([[0.        , 1.        ],
       [0.25230092, 0.74769908],
       [0.        , 1.        ],
       ...,
       [0.        , 1.        ],
       [0.        , 1.        ],
       [0.        , 1.        ]])
```

6.3один метод из группы методов вложений (embedded methods).

```
#Embedded
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression

SelectFromModel(LogisticRegression(C=0.1)).fit_transform(X_train, y_train)
```

```
array([[0.        , 2.        , 3.        ],
       [0.        , 2.24309724, 0.        ],
       [0.        , 2.        , 2.        ],
       ...,
       [0.        , 2.63624944, 2.75749963],
       [0.        , 2.        , 3.        ],
       [0.        , 2.        , 0.50776673]])
```

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей» [Электронный ресурс] // GitHub. — Режим доступа: https://github.com/ugapanyuk/ml_course/wiki/LAB_KNN.

[2] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource] // Read the Docs.. — Access mode: https://ipython.readthedocs.io/en/stable/

[3] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData.. —

Access mode: https://seaborn.pydata.org/.

[4] pandas 0.24.1 documentation [Electronic resource] // PyData.. — Access mode: http://pandas.pydata.org/pandas-docs/stable/.