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 ИУ5И-21М

Ли Лююй

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("drive/MyDrive/healthcare-dataset-stroke-data.csv",encoding='utf-8')
data.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

```
data.info()
```

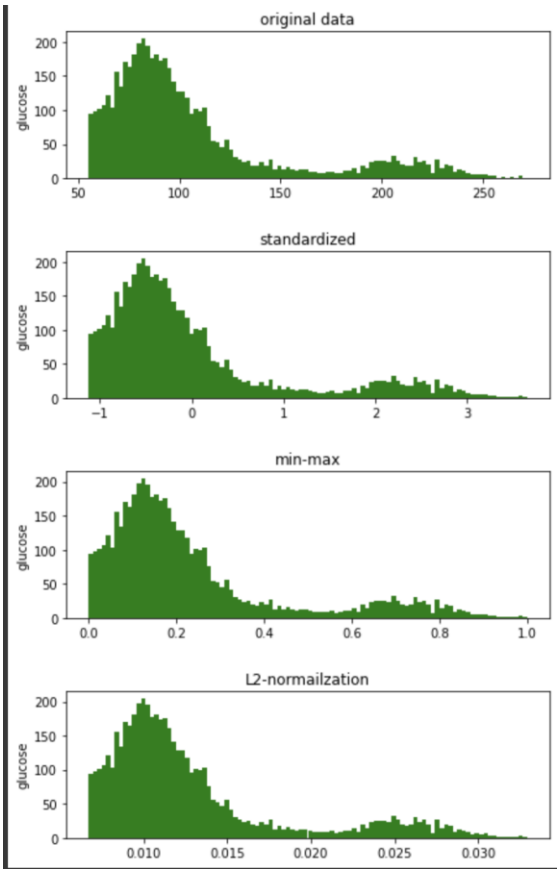
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 5110 non-null   int64  
 1   gender             5110 non-null   object 
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64  
 4   heart_disease      5110 non-null   int64  
 5   ever_married       5110 non-null   object 
 6   work_type          5110 non-null   object 
 7   Residence_type     5110 non-null   object 
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object 
 11  stroke             5110 non-null   int64  
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

```
data.describe()
```

	id	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909.000000	5110.000000
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28.893237	0.048728
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7.854067	0.215320
min	67.000000	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23.500000	0.000000
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28.100000	0.000000
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33.100000	0.000000
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

3. масштабирование признаков (не менее чем тремя способами)

```
import sklearn.preprocessing as preproc


data['standardized_n'] = preproc.StandardScaler().fit_transform(data[['avg_glucose_level']])
data['minmax_n'] = preproc.minmax_scale(data[['avg_glucose_level']])
data['l2_normalized_n'] = preproc.normalize(data[['avg_glucose_level']], 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['avg_glucose_level'], 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')

plt.show()
```



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

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

```
      id  gender   age  ...   bmi  smoking_status  stroke
0   9046       1  67.0  ...  36.6               1       1
1  51676       0  61.0  ...   NaN               2       1
2  31112       1  80.0  ...  32.5               2       1
3  60182       0  49.0  ...  34.4               3       1
4   1665       0  79.0  ...  24.0               2       1

[5 rows x 12 columns]
        id  gender   age  ...   bmi  smoking_status  stroke
33   54401       1  80.0  ...  30.5               1       1
45   19824       1  76.0  ...  33.6               2       1
122  13491       1  80.0  ...  31.7               3       1
123  44033       1  56.0  ...  35.8               2       1
135  71279       0  71.0  ...  38.7               2       1

[5 rows x 12 columns]
```

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['work_type'])
genre_mappings = {index: label for index, label in enumerate(gle.classes_)}
genre_mappings
```

```
{0: 'Govt_job',
 1: 'Never_worked',
 2: 'Private',
 3: 'Self-employed',
 4: 'children'}
```

6. отбор признаков:

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

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[0,5,9])],remainder='passthrough')
x = np.array(ct.fit_transform(x))
```

```
#Filter
from sklearn.feature_selection import VarianceThreshold

VarianceThreshold(threshold=3).fit_transform(x)
```

```
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);

```
#Wrapper
features = ['id','age',
 'hypertension',
 'heart_disease',
 'ever_married',
 'Residence_type',
 'avg_glucose_level',
 'bmi',
 'gender',
 'work_type',
 'smoking_status']

label = ['stroke']

X_1 = data[features]
y_1 = data[label]
X_1.bmi=(X_1.bmi.fillna(28.74))
X_1.gender=(X_1.gender.fillna(1))
```

```
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)
```

```
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. — 2019. — Режим доступа: https://github.com/ugapanyuk/ml_course/wiki/LAB_KNN (дата обращения: 05.04.2019).
[2] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource] // Read the Docs. — 2019. — Access mode: https://ipython.readthedocs.io/en/stable/ (online; accessed: 20.02.2019).

[3] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData. — 2018. —

Access mode: https://seaborn.pydata.org/ (online; accessed: 20.02.2019).
[4] pandas 0.24.1 documentation [Electronic resource] // PyData. — 2019. — Access mode: http://pandas.pydata.org/pandas-docs/stable/ (online; accessed: 20.02.2019).
[5] dronio. Solar Radiation Prediction [Electronic resource] // Kaggle. — 2017. — Access mode: https://www.kaggle.com/dronio/SolarEnergy (online; accessed: 18.02.2019).
[6] Chrétien M. Convert datetime.time to seconds [Electronic resource] // Stack Overflow. — 2017. — Access mode: https://stackoverflow.com/a/44823381 (online; accessed: 20.02.2019).
[7] scikit-learn 0.20.3 documentation [Electronic resource]. — 2019. — Access mode: https://scikit-learn.org/ (online; accessed: 05.04.2019).