

Уралова Е.А., ИУ5-24М, Вариант №12


✓ Задание №1

Задача №12. Для набора данных проведите нормализацию для одного (произвольного) числового признака с использованием функции "логарифм - $\text{np.log}(X)$ ".

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

```
url = "https://raw.githubusercontent.com/stedy/Machine-Learning-with-R-datasets/master/insurance.csv"
data = pd.read_csv(url)
```

```
data.head()
```



	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

Выберем числовой признак bmi (индекс массы тела) для нормализации:

```
feature_to_normalize = 'bmi'
```

Нормализация выбранного признака с использованием логарифма:

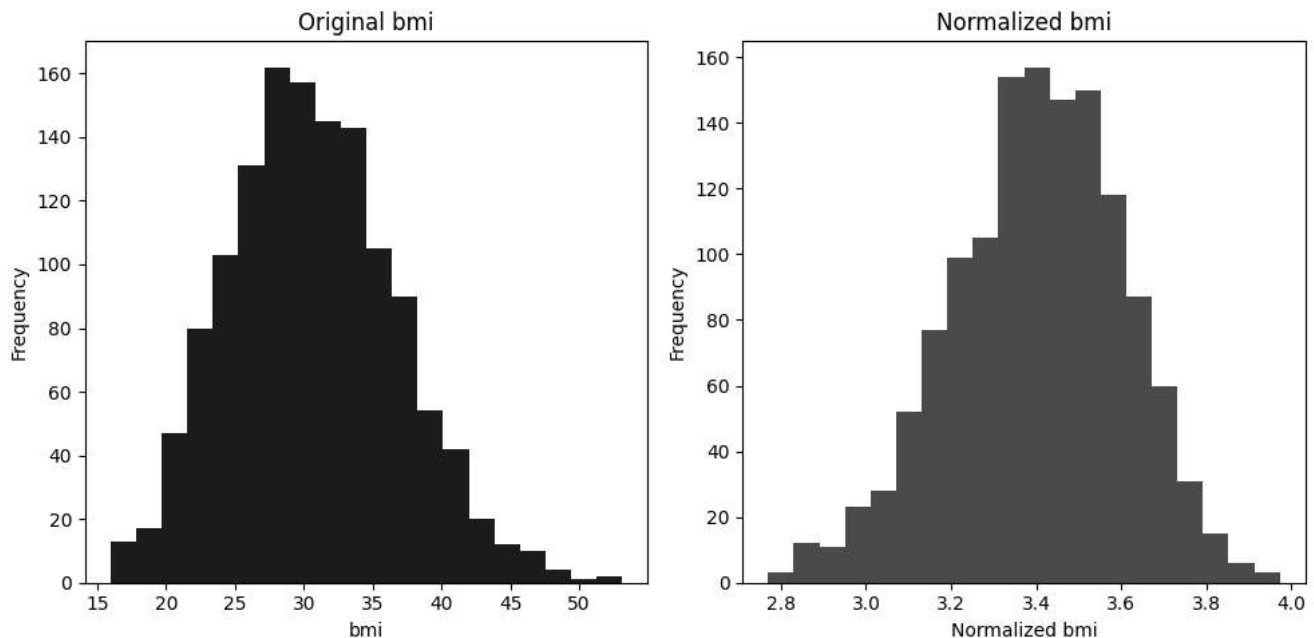
```
data['normalized_' + feature_to_normalize] = np.log(data[feature_to_normalize])
```

Визуализация исходного и нормализованного признаков:

```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.hist(data[feature_to_normalize], bins=20, color='blue')
plt.title('Original ' + feature_to_normalize)
plt.xlabel(feature_to_normalize)
plt.ylabel('Frequency')

plt.subplot(1, 2, 2)
plt.hist(data['normalized_' + feature_to_normalize], bins=20, color='green')
plt.title('Normalized ' + feature_to_normalize)
plt.xlabel('Normalized ' + feature_to_normalize)
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



✓ Задание №2

Задача №32. Для набора данных проведите процедуру отбора признаков (feature selection). Используйте метод обертывания (wrapper method), обратный алгоритм (sequential backward selection).

```
!pip install mlxtend
```

```
Requirement already satisfied: mlxtend in /usr/local/lib/python3.10/dist-packages (0.22.0)
Requirement already satisfied: scipy>=1.2.1 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.11.4)
Requirement already satisfied: numpy>=1.16.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.25.2)
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.5.3)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.2.2)
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (3.7.1)
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.3.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from mlxtend) (67.7.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (4.50.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (24.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->mlxtend) (2023.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.2->mlxtend) (3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend) (1.16.0)
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.feature_selection import RFECV
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
```

```
url = "https://raw.githubusercontent.com/stedy/Machine-Learning-with-R-datasets/master/insurance.csv"
data_1 = pd.read_csv(url)
data_1.head()
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

Определение признаков и целевой переменной:

```
X = data.drop(['charges'], axis=1)
y = data['charges']
```

Разделение данных на обучающий и тестовый наборы:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
print("Пропущенные значения в X_train:", X_train.isnull().sum())
print("Пропущенные значения в y_train:", y_train.isnull().sum())
```

```
Пропущенные значения в X_train: age          0
sex              0
bmi              0
children         0
smoker           0
region           0
normalized_bmi   0
dtype: int64
Пропущенные значения в y_train: 0
```

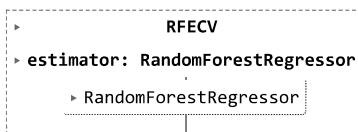
```
X_train_encoded = pd.get_dummies(X_train)
```

Определение модели (случайный лес):

```
rf = RandomForestRegressor()
```

Определение метода отбора признаков (рекурсивное исключение признаков с кросс-валидацией)

```
rfecv = RFECV(estimator=rf, step=1, cv=5, scoring='r2')
rfecv.fit(X_train_encoded, y_train)
```



```
print("Optimal number of features : %d" % rfecv.n_features_)
print("Selected features:", X_train_encoded.columns[rfecv.support_])
```

```
Optimal number of features : 9
Selected features: Index(['age', 'bmi', 'children', 'normalized_bmi', 'sex_male', 'smoker_no',
                        'smoker_yes', 'region_northeast', 'region_northwest'],
                        dtype='object')
```