РК1, ИУ5-25М Стрихар П.А.

О наборе данных

Входные переменные (на основе физико-химических тестов):

1 - фиксированная кислотность

2 - летучая кислотность

3 - лимонная кислота

4 - остаточный сахар

5 - хлориды

6 - свободный диоксид серы

7 - общий диоксид серы

8 - плотность

9 - pH

10 - сульфаты

11 - алкоголь

Выходная переменная (на основе сенсорных данных):

12 - качество (оценка между 0 и 10)

Задача №12.

Для набора данных проведите нормализацию для одного (произвольного) числового признака с использованием функции "логарифм - np.log(X)".

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

```
data = pd.read_csv("winequality-red.csv", sep=',')
data.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	

Next steps: View recommended plots

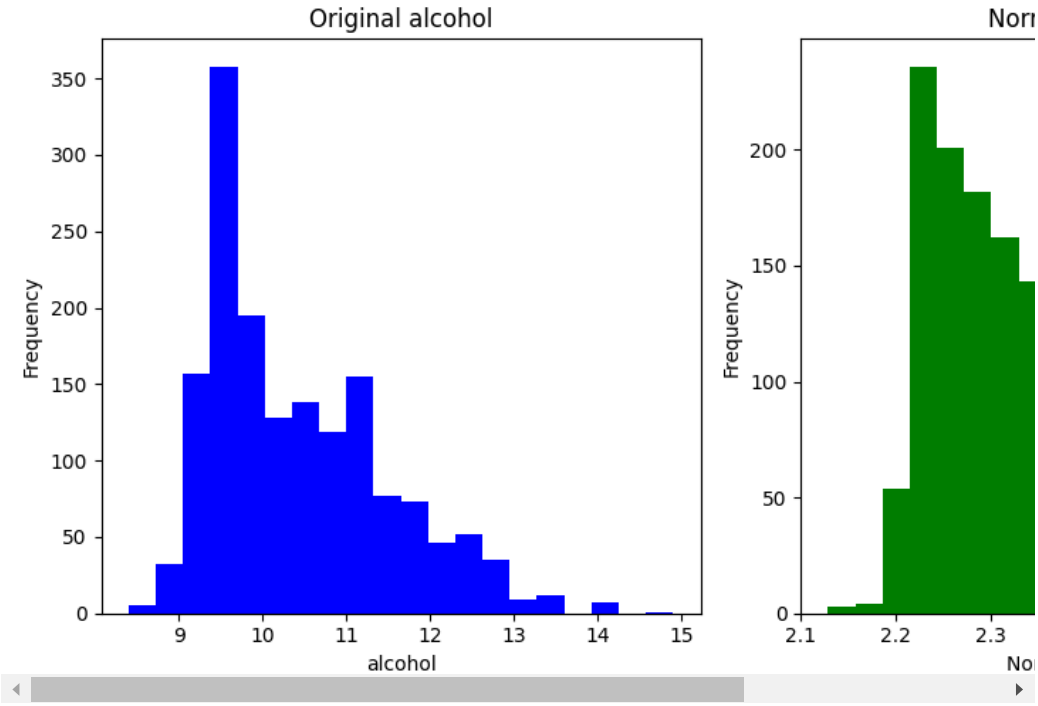
```
#Выберем числовой признак Бти (индекс массы тела) для нормализации:
feature_to_normalize = 'alcohol'
```

```
#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()
```



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) (2.0.3)
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) (1.4.0)
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) (1.2.1)
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) (4.51.0)
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: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->mlxtend) (2024.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.2->mlxtend) (3.4.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxten
```

```
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


data_1 = pd.read_csv("winequality-red.csv", sep=',')
data_1.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	

Next steps: View recommended plots

```
#Определение признаков и целевой переменной:
X = data.drop(['quality'], axis=1)
Y = data['quality']
```

```
#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: fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
normalized_alcohol      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, v=5, scoring='r2')
rfecv.fit(X_train_encoded, y_train)
```

```
RFECV
estimator: RandomForestRegressor
RandomForestRegressor()
RandomForestRegressor
```

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

```
Optimal number of features : 12
Selected features: Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'normalized_alcohol'],
      dtype='object')
```