|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное автономное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления

**ОТЧЕТ**

***ПО ЛАБОРАТОРНОЙ РАБОТЕ №2***

***ПО ДИСЦИПЛИНЕ***

***«МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ»***

***НА ТЕМУ:***

**Обработка признаков (часть 1).**

Студент \_\_ИУ5И-23М\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Фу Чэньтянь\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Ю.Е.Гапанюк\_\_**

(Подпись, дата) (И.О.Фамилия)

2025 г.

**ВВЕДЕНИЕ**

Обработка признаков (Feature Engineering) в машинном обучении - это процесс создания, преобразования и выбора наиболее релевантных признаков (features) из исходных данных для улучшения производительности и точности моделей машинного обучения. Этот процесс включает в себя: Очистку данных и удаление аномалий; Выделение признаков (features) для решения задачи, включая feature extraction и feature engineering, а также кодирование признаков; Подготовку данных.

Цель лабораторной работы: изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

**ЗАДАНИЕ**

Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.) Просьба не использовать датасет, на котором данная задача решалась в лекции.

Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:

* устранение пропусков в данных;
* кодирование категориальных признаков;
* нормализация числовых признаков.

**ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

# Часть 1. Текстовое описание набора данных

1. Устранение пропусков в данных

Этот набор данных собирает различную информацию о здоровье сна и образе жизни и используется для анализа влияния различных факторов на сон.

Он включает в себя такие данные, как пол, возраст, род занятий, продолжительность сна, качество сна, уровень физической активности, уровень стресса, категорию ИМТ, артериальное давление, частоту сердечных сокращений, количество шагов в день, а также наличие или отсутствие нарушений сна.

图形用户界面, 文本

AI 生成的内容可能不正确。

Рисунок 1: Информация о наборе данных

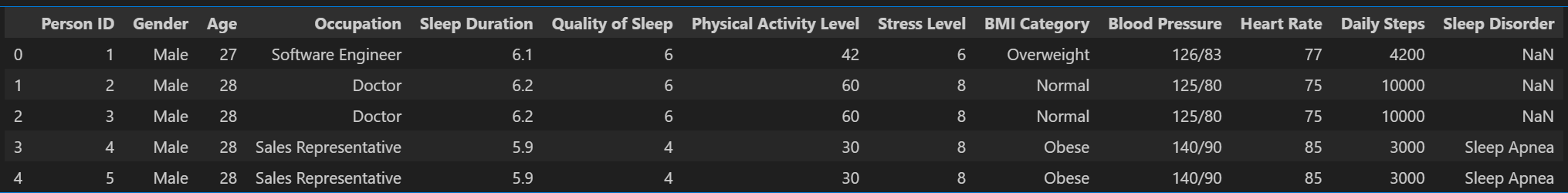


Рисунок 2: Первые 5 строк набора данных

Этот набор данных содержит категориальные и числовые признаки. Чтобы удовлетворить требование, некоторые данные удаляются из набора данных, чтобы создать недостающие данные. Создаем набор данных с недостающими данными ' Sleep\_health\_with\_missing\_values.csv' :

print(df.info())

missing\_indices = np.random.choice(df.index, size=int(len(df) \* 0.1), replace=False)

df.loc[missing\_indices, 'Physical Activity Level'] = np.nan

print(df.isnull().sum())

df.to\_csv('Sleep\_health\_with\_missing\_values.csv', index=False)

文本

AI 生成的内容可能不正确。

Рисунок 3: Пропущенные значения с использованием случайной функции

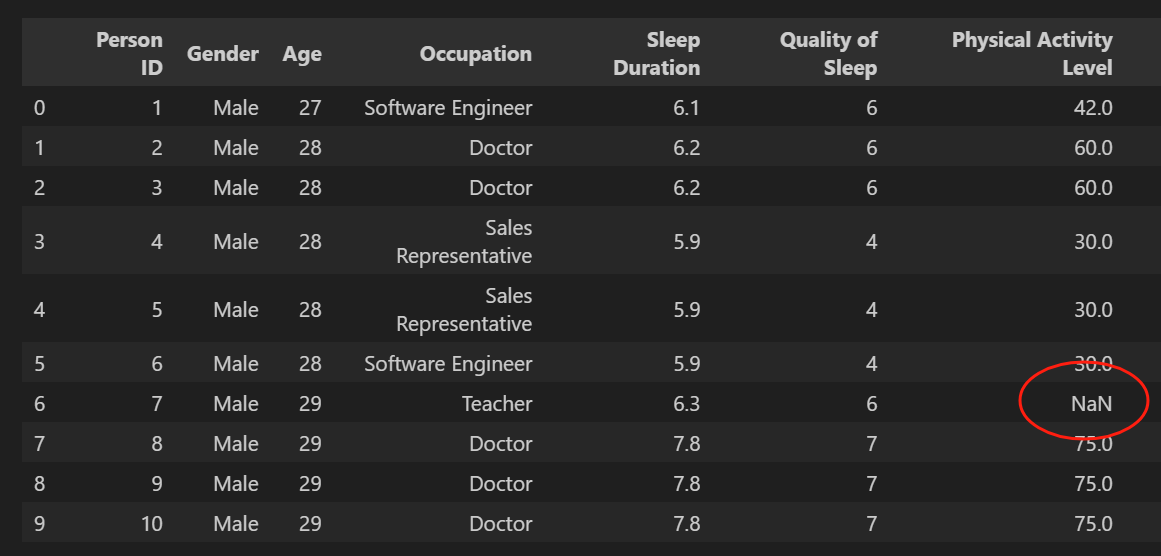


Рисунок 4: Набор данных успешно создан с отсутствующими данными.

# Часть 2. Обработка признаков

1. Устранение пропусков в данных

Если в данных есть пропуски, то большинство алгоритмов машинного обучения не будет с ними работать. Даже корреляционная матрица не будет строиться корректно.

Существуют различные способы устранения пропусков в данных, которые связаны с удалением или заполнением пропусков.

В этом наборе данных большая часть отсутствующих данных, поэтому мы удаляем столбцы с отсутствующими данными. Потому что если недостающих данных слишком много, есть риск нарушить распределение исходных данных при заполнении пробелов.

文本

AI 生成的内容可能不正确。

Рисунок 5: Наборы данных с отсутствующими столбцами данных были удалены

Если датасет большой и пропущенных данных относительно немного, то рекомендуется удалять строки, содержащие пропуски в данных. Под "немного" в идеальном случае понимается 5% от выборки. Корректируем количество недостающих данных, чтобы получить новый набор данных с недостающими данными.

df = pd.read\_csv("Sleep\_health\_and\_lifestyle\_dataset.csv")

missing\_indices = np.random.choice(df.index, size=int(len(df) \* 0.01), replace=False)

df.loc[missing\_indices, 'Age'] = np.nan

print(df.isnull().sum())

df.to\_csv('Sleep\_with\_missing\_values\_1%.csv', index=False)

文本

AI 生成的内容可能不正确。 文本

AI 生成的内容可能不正确。

Рисунок 6: Наборы данных, соответствующие количеству отсутствующих данных

Удалить строки с отсутствующими данными:

data\_cleaned\_rows = df.dropna(axis=0, how='any')

print(data\_cleaned\_rows.info())

data\_cleaned\_rows.to\_csv('Sleep\_cleaned\_rows.csv', index=False)

文本

AI 生成的内容可能不正确。

Рисунок 7: Наборы данных с отсутствующими строками данных были удалены

Заполняем недостающие значения, используя среднее, медиану или моду:

import pandas as pd

from sklearn.impute import SimpleImputer

#загрузка

data = pd.read\_csv('Sleep\_with\_missing\_values\_1%.csv')

#SimpleImputer

numeric\_features = data.select\_dtypes(include=['int64','float64']).columns

catagorical\_features = data.select\_dtypes(include=['object']).columns

#填充数值型特征

imputer\_mean = SimpleImputer(strategy='mean')

data[numeric\_features] = imputer\_mean.fit\_transform(data[numeric\_features])

#填充分类特征

imputer\_most\_fre = SimpleImputer(strategy='most\_frequent')

data[catagorical\_features] = imputer\_most\_fre.fit\_transform(data[catagorical\_features])

#保存处理后的数据

data.to\_csv('Sleep\_imputed\_mean\_mode.csv', index = False)

print(data.info())

电脑屏幕的照片上有文字

AI 生成的内容可能不正确。

Рисунок 8: Недостающие значения успешно заполнены图表, 直方图

AI 生成的内容可能不正确。 图表, 条形图

AI 生成的内容可能不正确。

Рисунок 9: График распределения необработанных и обработанных данных

Заполните недостающие значения с помощью KNNImputer:

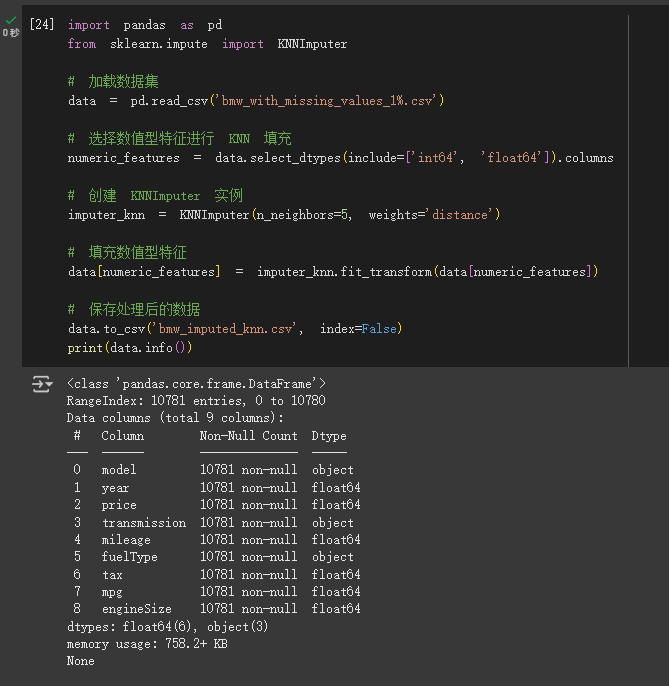


Рисунок 10: Недостающие значения успешно заполнены

图表, 直方图

AI 生成的内容可能不正确。

Рисунок 11: График распределения необработанных и обработанных данных

Используем MissForest для заполнения пропущенных значений:

from sklearn.experimental import enable\_iterative\_imputer

from sklearn.impute import IterativeImputer

from sklearn.ensemble import RandomForestRegressor

data = pd.read\_csv('Sleep\_with\_missing\_values\_1%.csv')

numeric\_features = data.select\_dtypes(include=['int64','float64']).columns

imputer\_missforest = IterativeImputer(estimator=RandomForestRegressor(n\_estimators=10,random\_state=0),max\_iter=10,random\_state=0)

data[numeric\_features] = imputer\_missforest.fit\_transform(data[numeric\_features])

data.to\_csv('Sleep\_imputed\_missforest.csv',index=False)

print(data.info())

文本

AI 生成的内容可能不正确。

Рисунок 12: Недостающие значения успешно заполнены

图表, 直方图

AI 生成的内容可能不正确。

Рисунок 13: График распределения необработанных и обработанных данных

Заполняем пропущенные значения, используя вменение крайних значений:

data = pd.read\_csv('Sleep\_with\_missing\_values\_1%.csv')

def calculate\_extreme\_value(column):

    if pd.api.types.is\_numeric\_dtype(column):

        mean = column.mean()

        std = column.std()

        return mean + 3 \* std

    else:

        return None

extreme\_values ={}

for column in data.columns:

    extreme\_value = calculate\_extreme\_value(data[column])

    if extreme\_value is not None:

        extreme\_values[column] = extreme\_value

data.fillna(value=extreme\_values, inplace=True)

data.to\_csv('Sleep\_imputed\_extreme.csv',index=False)

print(data.info())

图形用户界面, 文本

AI 生成的内容可能不正确。

Рисунок 14: Недостающие значения успешно заполнены

1. Кодирование категориальных признаков

Основываясь на этом наборе данных, выбрал два метода: Label Encoding и One-Hot Encoding. Используя восстановленный набор данных, здесь мы используем набор данных, то есть набор данных, который не содержит недостающих данных (Sleep\_cleaned\_rows.csv)

Label Encoding сопоставляет каждое значение категории с целочисленным значением:

from sklearn.preprocessing import LabelEncoder

data = pd.read\_csv('Sleep\_cleaned\_rows.csv')

# LabelEncoder

le = LabelEncoder()

# encoding для object

data['Gender\_le'] = le.fit\_transform(data['Gender'])

data['Occupation\_le'] = le.fit\_transform(data['Occupation'])

data['BMI\_le'] = le.fit\_transform(data['BMI Category'])

print(data[['Gender','Gender\_le','Occupation','Occupation\_le','BMI Category','BMI\_le']])

data.to\_csv('Sleep\_label\_encoded.csv', index=False)

文本

AI 生成的内容可能不正确。

Рисунок 15: Результаты кодирования методом Label Encoding

One-Hot Encoding преобразует каждое значение категории в двоичный вектор:

data\_onehot = pd.get\_dummies(data, columns=['Gender','Occupation','BMI Category'])

print(data\_onehot.head(5))

data\_onehot.to\_csv('Sleep\_onehot\_encoded.csv',index=False)

文本

AI 生成的内容可能不正确。

Рисунок 16: Результаты кодирования методом One-Hot Encoding

1. Нормализация числовых признаков.

Нормализация числового признака предполагает, что на основании существующего признака мы создаем новый признак, который в идеале имеет нормальное соотношение. Многие модели машинного обучения лучше работают с нормализованными источниками питания, особенно линейные модели.

* Какие функции чаще всего используются для нормализации:
* логарифм - np.log(X)
* обратная зависимость - 1 / X
* квадратный корень
* возведение в степень
* преобразование Бокса-Кокса (преобразование Бокса-Кокса)
* Преобразование Йео-Джонсона - расширение метода Бокса-Кокса для работы с произвольными крупными

Как визуально определить, является ли это нормальным:

* Гистограмма распределения.
* График квантиль-квантиль или график Q-Q.

Выбираем методы на основе этого набора данных.

Логарифмически преобразуйте числовые характеристики, чтобы приблизить их к нормальному распределению:

屏幕的截图

AI 生成的内容可能不正确。

Рисунок 17: Результат логарифмического преобразования

Преобразование Box-Cox — это широко используемый метод нормализации, особенно подходящий для данных с положительно асимметричным распределением.

文本

AI 生成的内容可能不正确。

Рисунок 18: Результат Box-Cox преобразования

Преобразование Yeo-Johnson является расширением преобразования Box-Cox и может обрабатывать нулевые и отрицательные значения.

文本

AI 生成的内容可能不正确。

Рисунок 19: Результат Yeo-Johnson преобразования

Используем гистограммы и графики Q-Q для изучения эффекта нормализации:

import matplotlib.pyplot as plt

import scipy.stats as stats

def diagnostic\_plots(df, variable):

    plt.figure(figsize=(15,6))

    # 直方图

    plt.subplot(1, 2, 1)

    df[variable].hist(bins=30)

    # Q-Q 图

    plt.subplot(1, 2, 2)

    stats.probplot(df[variable], dist="norm", plot=plt)

    plt.show()

# 检查原始数据分布

diagnostic\_plots(data, 'Age')

# 检查对数变换后的分布

diagnostic\_plots(data, 'Age\_log')

# 检查 Box-Cox 变换后的分布

diagnostic\_plots(data, 'Age\_boxcox')

# 检查 Yeo-Johnson 变换后的分布

diagnostic\_plots(data, 'Age\_yeo')

data.to\_csv('Sleep\_normalized.csv', index=False)

图表

AI 生成的内容可能不正确。

Рисунок 20: Распространение исходных данных

图表

AI 生成的内容可能不正确。

Рисунок 21: Распределение после логарифмического преобразования

图表, 直方图

AI 生成的内容可能不正确。

Рисунок 22: Распределение после Box-Cox преобразования

图表, 直方图

AI 生成的内容可能不正确。

Рисунок 23: Распределение после Yeo-Johnson преобразования

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной лабораторной работы были изучены и применены продвинутые методы предварительной обработки данных на примере набора данных с информацией о здоровье сна. Были решены три основные задачи: устранение пропусков в данных, кодирование категориальных признаков и нормализация числовых признаков. Для обработки пропусков применялись удаление строк и столбцов, а также заполнение с использованием среднего, медианы или моды. Были использованы алгоритмы KNNImputer и MissForest для более точного восстановления пропущенных значений. Категориальные признаки были закодированы с помощью Label Encoding и One-Hot Encoding, что позволило преобразовать их в формат, пригодный для машинного обучения. Числовые признаки прошли нормализацию с использованием логарифмического преобразования, Box-Cox и Yeo-Johnson методов, что значительно улучшило их распределение.

Визуальный анализ распределения данных до и после нормализации с использованием гистограмм и Q-Q графиков подтвердил эффективность примененных методов. В результате выполнения работы были получены следующие выводы: обработка пропусков критически важна для точности моделей, кодирование категориальных признаков методами Label Encoding и One-Hot Encoding позволяет эффективно их использовать в моделях машинного обучения, а нормализация числовых признаков улучшает работу линейных и других моделей. Результаты данной работы могут быть использованы для дальнейшего построения и оптимизации моделей машинного обучения на основе предобработанных данных.