|  |  |
| --- | --- |
|  | **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 №3***

***ПО ДИСЦИПЛИНЕ***

***«МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ»***

***НА ТЕМУ:***

**Обработка признаков (часть 2).**

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 один или несколько наборов данных (датасетов) для решения следующих задач. Каждая задача может быть решена на отдельном датасете, или несколько задач могут быть решены на одном датасете. Просьба не использовать датасет, на котором данная задача решалась в лекции.

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

1. масштабирование признаков (не менее чем тремя способами);

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

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

4. отбор признаков:

* один метод из группы методов фильтрации (filter methods);
* один метод из группы методов обертывания (wrapper methods);
* один метод из группы методов вложений (embedded methods).

**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

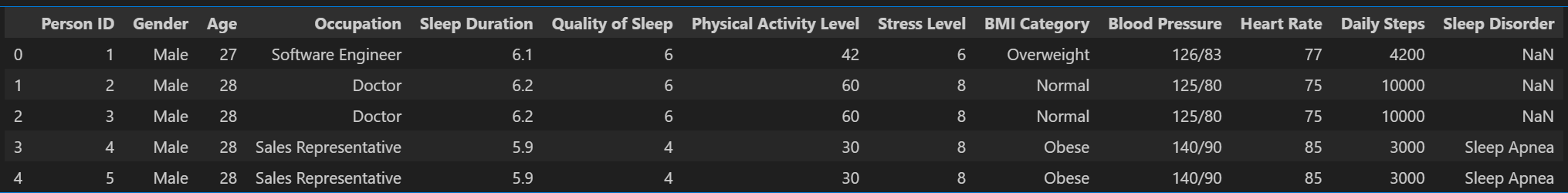


Рисунок 2: Первые 5 строк набора данных

# Часть 2. Обработка признаков

1. Масштабирование признаков

Масштабирование - это изменение диапазона измерения признака с целью улучшения качества построения модели.

Почему необходимо масштабировать признаки?

Многие алгоритмы машинного обучения устроены таким образом, что признаки с меньшей амплитудой оказываются "оштрафованы" по сравнению с признаками с большей амплитудой, и оказывают меньшее влияние на процесс построения модели.

Пример. Метод ближайших соседей. Ключевым шагом метода является вычисление расстояний между соседями. Чаще всего в методе ближайших соседей используется Евклидово расстояние:

Если один из признаков имеет амплитуду значительно меньшую по сравнению с другими признаками, то этот признак почти не будет вносить вклад при вычислении расстояния.

Методы машинного обучения (как с учителем, так и без учителя), ЗАВИСЯЩИЕ от масштабирования признаков:

• Метод ближайших соседей

• Линейная регрессия

• Логистическая регрессия

• Метод опорных векторов (SVM)

• Нейронные сети

• Некоторые алгоритмы кластеризации (K-means)

• Анализ главных компонент (Principal Component Analysis, PCA)

Методы машинного обучения, НЕ ЗАВИСЯЩИЕ от масштабирования признаков:

• Деревья решений и другие алгоритмы на их основе:

• Случайный лес

• Градиентный бустинг

В алгоритме построения дерева решения не строится единое метрическое пространство по всем признакам. Строится набор ветвлений по отдельным признакам, масштаб признаков не имеет значения.

Признаки нужно масштабировать до или после деления на обучающую и тестовую выборку?

• Предположим, что мы разделили данные на обучающую и тестовую выборки, и взяли данные для масштабирования только из обучающей выборки.

• В этом случае наличие выбросов в тестовой выборке может нарушить схему масштабирования.

• Традиционным является подход, при котором данные делятся на обучающую и тестовую выборки ДО масштабирования. Параметры масштабирования (например, среднее значение, дисперсия) берутся только из обучающей выборки и затем применяются к тестовой выборке.

• Если выбросы в тестовой выборке мешают реализации этого подхода, то данные делятся на обучающую и тестовую выборки ПОСЛЕ масштабирования.

Для этого набора данных мы используем три метода масштабирования объектов: StandardScaler, MinMaxScaler и RobustScaler.

StandardScaler (стандартизация по Z - оценке) представляет собой часто используемый метод масштабирования признаков. Он достигает стандартизации путем преобразования значений признаков в распределение со средним значением, равным 0, и стандартным отклонением, равным 1. Этот метод предполагает, что данные имеют нормальное распределение и выполняет масштабирование путем вычитания среднего значения признака и деления на стандартное отклонение. StandardScaler является относительно чувствительным к выбросам, так как среднее значение и стандартное отклонение могут быть затронуты экстремальными значениями. Он подходит для случаев, когда распределение данных близко к нормальному, особенно при использовании оптимизационных алгоритмов на основе градиента.

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import RobustScaler

# 加载数据集

data = pd.read\_csv('Sleep\_health\_and\_lifestyle\_dataset.csv')

# 确定数值特征

numeric\_features = ['Age', 'Sleep Duration', 'Quality of Sleep', 'Physical Activity Level', 'Stress Level', 'Heart Rate','Daily Steps']

# 初始化StandardScaler

scaler = StandardScaler()

#scaler = MinMaxScaler()

#scaler = RobustScaler()

# 拟合和转换数据

data\_scaled = scaler.fit\_transform(data[numeric\_features])

data\_scaled = pd.DataFrame(data\_scaled, columns=numeric\_features)

# 绘制缩放前后的密度图

plt.figure(figsize=(12, 5))

# 缩放前

plt.subplot(1, 2, 1)

sns.kdeplot(data=data[numeric\_features])

plt.title('До масштабирования')

# 缩放后

plt.subplot(1, 2, 2)

sns.kdeplot(data=data\_scaled)

plt.title('После масштабирования')

plt.show()

# 输出缩放后的数据

print("StandardScaler：")

print(data\_scaled.head())

# 分割数据集为训练集和测试集

X\_train, X\_test = train\_test\_split(data[numeric\_features], test\_size=0.2, random\_state=42)

# 对训练集和测试集分别进行缩放

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# 转换为DataFrame

X\_train\_scaled = pd.DataFrame(X\_train\_scaled, columns=numeric\_features)

X\_test\_scaled = pd.DataFrame(X\_test\_scaled, columns=numeric\_features)

# 绘制训练集和测试集的密度图

plt.figure(figsize=(12, 5))

# 训练集

plt.subplot(1, 2, 1)

sns.kdeplot(data=X\_train\_scaled)

plt.title('Обучающая (после масштабирования)')

# 测试集

plt.subplot(1, 2, 2)

sns.kdeplot(data=X\_test\_scaled)

plt.title('Тестовая (после масштабирования)')

plt.show()

# 输出缩放后的数据

print("StandardScaler结果：")

print("Обучающая выборка:")

print(X\_train\_scaled.head())

print("\nТестовая выборка:")

print(X\_test\_scaled.head())

图表, 折线图

AI 生成的内容可能不正确。

Рисунок 3: График плотности до и после масштабирования (StandardScaler)

图表, 直方图

AI 生成的内容可能不正确。

Рисунок 4: График плотности обучающой и тестовой (StandardScaler)

MinMaxScaler (масштабирование по минимальному и максимальному значениям) масштабирует значения признаков в заданный диапазон, обычно в диапазон [0, 1]. Он реализует это путем вычитания минимального значения из значений признака и деления на размах (разность между максимальным и минимальным значениями). Этот метод ограничивает данные в фиксированном диапазоне и подходит для случаев, когда необходимо ограничить данные в определенном интервале, например, при использовании функций активации в нейронных сетях. MinMaxScaler чувствителен к выбросам, так как экстремальные значения могут повлиять на масштабируемый диапазон, но он хорошо сохраняет форму и распределение данных.

图表, 直方图

AI 生成的内容可能不正确。

Рисунок 5: График плотности до и после масштабирования (MinMaxScaler)

图表

AI 生成的内容可能不正确。

Рисунок 6: График плотности обучающой и тестовой (MinMaxScaler)

RobustScaler (масштабирование на основе квартилей) использует медиану и интерквартильный размах (IQR) для масштабирования и обладает высокой устойчивостью к выбросам. Он реализует масштабирование путем вычитания медианы из значений признака и деления на интерквартильный размах. Этот метод не изменяет форму распределения данных и подходит для случаев, когда в данных присутствует большое количество выбросов. RobustScaler подходит для алгоритмов, которые должны быть нечувствительными к выбросам, таких как метод опорных векторов (SVM) и случайный лес.

图表, 直方图

AI 生成的内容可能不正确。

Рисунок 7: График плотности до и после масштабирования (RobustScaler)

**图表, 折线图, 直方图

AI 生成的内容可能不正确。**

Рисунок 8: График плотности обучающой и тестовой (RobustScaler)

1. Обработку выбросов для числовых признаков

Выявление и обработка выбросов посвященный обработке выбросов. В соответствии с ГОСТ, выброс (outlier) - это элемент маломощного подмножества выборки, существенно отличающийся от остальных элементов выборки.

Почему необходимо устранять выбросы?

* Выбросы негативно влияют на построение модели.

Причины появления выбросов:

* Ошибки в измерениях, например, часть значений признака "расстояние" была измерена не в километрах, а в метрах.
* Технические ошибки форматирования данных.

Основные задачи обработки выбросов:

* Обнаружение выбросов.
* Устранение (удаление или замена) выбросов (в зависимости от требований задачи).

В этом наборе данных мы рассматриваем данные в двух столбцах: уровень стресса и уровень физической активности, поскольку у большинства людей уровни стресса и уровни физической активности должны находиться в одном и том же диапазоне, а те, которые выходят за рамки этого диапазона, можно считать выбросами.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import scipy.stats as stats

# 读取数据集

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

# 定义数值特征列表

numerical\_features = ['Physical Activity Level', 'Stress Level']

# 定义一个函数来替换异常值

def replace\_outliers(df, column):

    Q1 = df[column].quantile(0.25)

    Q3 = df[column].quantile(0.75)

    IQR = Q3 - Q1

    lower\_bound = Q1 - 1.5 \* IQR

    upper\_bound = Q3 + 1.5 \* IQR

    mean = df[column].mean()

    df[column] = np.where(df[column] < lower\_bound, mean, np.where(df[column] > upper\_bound, mean, df[column]))

    return df

# 定义一个函数来删除异常值

def remove\_outliers(df, column):

    Q1 = df[column].quantile(0.25)

    Q3 = df[column].quantile(0.75)

    IQR = Q3 - Q1

    lower\_bound = Q1 - 1.5 \* IQR

    upper\_bound = Q3 + 1.5 \* IQR

    df = df[(df[column] >= lower\_bound) & (df[column] <= upper\_bound)]

    return df

# 定义一个函数来绘制四种图形

def plot\_four\_graphs(data, column, title):

    fig, ax = plt.subplots(2, 2, figsize=(12, 10))

    fig.suptitle(title)

    # 直方图

    plt.subplot(2, 2, 1)

    sns.histplot(data[column], bins=30, kde=True)

    plt.title('Histogram')

    # 概率图

    plt.subplot(2, 2, 2)

    stats.probplot(data[column], dist="norm", plot=plt)

    plt.title('Probability Plot')

    # 小提琴图

    plt.subplot(2, 2, 3)

    sns.violinplot(x=data[column])

    plt.title('Violin Plot')

    # 箱线图

    plt.subplot(2, 2, 4)

    sns.boxplot(x=data[column])

    plt.title('Box Plot')

    plt.tight\_layout()

    plt.show()

# 处理每个数值特征

for feature in numerical\_features:

    # 处理前的数据

    print(f'{feature} - Before Outlier Handling')

    plot\_four\_graphs(df, feature, f'{feature} - Before Outlier Handling')

    # 替换异常值

    df\_replaced = df.copy()

    replace\_outliers(df\_replaced, feature)

    # 删除异常值

    # df\_removed = remove\_outliers(df, feature)

    # 处理后的数据

    print(f'{feature} - After Outlier Handling')

    plot\_four\_graphs(df\_replaced, feature, f'{feature} - After Outlier Handling')

图表

AI 生成的内容可能不正确。

Рисунок 9: Необработанные данные (Physical Activity Level)

图表

AI 生成的内容可能不正确。

Рисунок 9: После замены выбросовные (Physical Activity Level)

图表

AI 生成的内容可能不正确。

Рисунок 9: После удаления выбросов (Physical Activity Level)

图表

AI 生成的内容可能不正确。

Рисунок 9: Необработанные данные (Stress Level)

图表

AI 生成的内容可能不正确。

Рисунок 9: После замены выбросовные (Stress Level)

图表

AI 生成的内容可能不正确。

Рисунок 9: После удаления выбросов (Stress Level)

1. Обработку по крайней мере одного нестандартного признака

One-Hot кодирование — это распространенный метод обработки категориальных признаков, который преобразует каждый категориальный признак в бинарный вектор. Конкретно говоря, для каждого категориального признака One-Hot кодирование создает несколько новых бинарных столбцов, каждый из которых соответствует одному значению категории. Если в исходных данных определенная строка принадлежит этой категории, то соответствующее значение в столбце будет 1, в противном случае — 0. Этот метод подходит для категориальных признаков без неявного порядка, так как не вводит никаких порядковых отношений между категориями.

В нашем исследовании мы применили One-Hot кодирование к признаку transmission. В исходном наборе данных признак **Occupation** имел несколько значений, таких как " Software Engineer ", " Doctor " и так далее. С помощью One-Hot кодирования мы преобразовали эти значения в несколько бинарных столбцов, например Occupation\_ Software Engineer, Occupation \_ Doctor и так далее. Таким образом, каждое значение категории стало независимым признаком, что облегчает последующий анализ и построение моделей.

Преимущество One-Hot кодирования в его простоте и интуитивной ясности, а также в эффективном преобразовании категориальных признаков в числовую форму, которую может обработать модель. Однако его недостаток заключается в увеличении размерности набора данных, особенно когда количество категорий велико, что может привести к "взрыву размерности". Тем не менее, для признака transmission в нашем исследовании, учитывая относительно небольшое количество категорий, One-Hot кодирование оказалось подходящим и эффективным методом.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# 读取数据集

df = pd.read\_csv('bmw.csv')

# 查看原始数据

print("Raw data：")

print(df.info())

print(df.head())

# 独热编码处理

df\_onehot = pd.get\_dummies(df, columns=['transmission'], prefix='transmission')

# 查看处理后的数据集信息

print("\nOne-hot encoded data：")

print(df\_onehot.info())

print(df\_onehot.head())

# 可视化原始数据的 'transmission' 分布

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

sns.countplot(x='transmission', data=df)

plt.title('Raw data - transmission distribution')

# 可视化处理后的数据的 'transmission' 分布

plt.subplot(1, 2, 2)

# 由于独热编码后 'transmission' 列被拆分为多个列，这里展示其中一个列的分布

sns.countplot(x='transmission\_Automatic', data=df\_onehot)

plt.title('One-hot encoded data - transmission\_Automatic distribution')

plt.tight\_layout()

plt.show()

文本

AI 生成的内容可能不正确。

Рисунок 9: Обработанные данные

图表, 条形图, 直方图

AI 生成的内容可能不正确。

Рисунок 9: График распределения необработанных данных и обработанных данных

1. Отбор признаков

В соответствии с Wikipedia: «Техники отбора признаков следует отличать от выделения признаков. Выделение признаков создаёт новые признаки как функции от оригинальных признаков, в то время как отбор признаков возвращает подмножество признаков».

Таким образом, задача отбора признаков - выбрать признаки, наиболее полезные для дальнейшего построения модели.

Почему используются методы отбора признаков?

* Простые модели (с меньшим количеством признаков) легче интерпретировать.
* Модели хуже обучаются на датасетах с большим количеством признаков.
* Уменьшение времени обучения и предсказания.
* Уменьшение риска переобучения.

Метод из группы методов фильтрации (filter methods):

* Используем VarianceThreshold, чтобы удалить функции с низкой дисперсией.
* Нарисуем график дисперсии, чтобы показать дисперсию каждого объекта, и отметьте выбранные объекты.
* Порог дисперсии можно настроить в зависимости от характеристик набора данных.

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.feature\_selection import VarianceThreshold

# 加载数据集

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

# 确定数值特征

numerical\_features = ['Age', 'Sleep Duration', 'Physical Activity Level', 'Stress Level', 'Heart Rate','Daily Steps']

X = df[numerical\_features]

y = df['Quality of Sleep']  # 假设 ' Quality of Sleep ' 是目标变量

# 方差选择法，删除低方差特征

selector = VarianceThreshold(threshold=0.1)  # 设置方差阈值

X\_selected = selector.fit\_transform(X)

# 获取被选中的特征名称

selected\_features = X.columns[selector.get\_support()]

# 绘制方差图

variances = X.apply(lambda col: col.var(), axis=0)

plt.figure(figsize=(10, 6))

plt.bar(numerical\_features, variances)

plt.axhline(y=0.1, color='r', linestyle='--', label='Threshold')

plt.xlabel('Features')

plt.ylabel('Variance')

plt.title('Variance of Numerical Features')

plt.xticks(rotation=45)

plt.legend()

plt.show()

print("被选中的特征:", selected\_features.tolist())

图表

AI 生成的内容可能不正确。

Рисунок 9: Результаты метода фильтрации

Метод оболочки（Wrapper Method）:

Функции были выбраны с использованием RFE (рекурсивное исключение функций) в сочетании с моделью логистической регрессии.

Нарисуйте график важности функций, чтобы показать рейтинг каждой функции, и отметьте выбранные функции.

Параметр n\_features\_to\_select регулирует количество выбранных объектов.

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.feature\_selection import RFE

from sklearn.linear\_model import LogisticRegression

# 加载数据集

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

# 确定数值特征

numerical\_features = ['Age', 'Sleep Duration', 'Physical Activity Level', 'Stress Level', 'Heart Rate','Daily Steps']

X = df[numerical\_features]

y = df['Quality of Sleep']  # 假设 'Quality of Sleep' 是目标变量

# 创建逻辑回归模型

model = LogisticRegression(max\_iter=1000)

# 使用RFE选择特征

rfe = RFE(estimator=model, n\_features\_to\_select=3)  # 选择3个特征

X\_selected = rfe.fit\_transform(X, y)

# 获取被选中的特征名称

selected\_features = X.columns[rfe.get\_support()]

# 绘制特征重要性图

importances = rfe.ranking\_

plt.figure(figsize=(10, 6))

plt.bar(numerical\_features, importances)

plt.xlabel('Features')

plt.ylabel('Importance')

plt.title('Feature Importance (RFE)')

plt.xticks(rotation=45)

plt.show()

print("被选中的特征:", selected\_features.tolist())

图表, 条形图

AI 生成的内容可能不正确。

Рисунок 9: Результаты метода оболочки

Встроенный метод（Embedded Method）:

* Выбор функций с использованием модели регрессии Lasso.
* Рассчитайте корреляционную матрицу выбранных признаков.
* Визуализация корреляционной матрицы с помощью тепловой карты

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import Lasso

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

# 加载数据集

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

# 确定数值特征

numerical\_features = ['Age', 'Sleep Duration', 'Physical Activity Level', 'Stress Level', 'Heart Rate','Daily Steps']

X = df[numerical\_features]

# 数据标准化

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# 创建Lasso回归模型

lasso = Lasso(alpha=0.1)  # 调整alpha值以控制正则化强度

# 训练模型

lasso.fit(X\_scaled, y)

# 获取特征的重要性（系数）

selected\_features = X.columns[np.abs(lasso.coef\_) > 1e-5]  # 设置一个阈值来选择重要的特征

# 绘制特征系数图

plt.figure(figsize=(10, 6))

plt.bar(numerical\_features, lasso.coef\_)

plt.xlabel('Features')

plt.ylabel('Coefficient')

plt.title('Feature Coefficients (Lasso Regression)')

plt.xticks(rotation=45)

plt.show()

print("被选中的特征:", selected\_features.tolist())

图表, 瀑布图

AI 生成的内容可能不正确。

Рисунок 9: Результаты встроенного метода

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения данной лабораторной работы мы провели углубленный анализ данных о информации о здоровье сна с использованием различных методов предварительной обработки данных. Были применены три метода масштабирования признаков — StandardScaler, MinMaxScaler и RobustScaler, что позволило привести данные в подходящий диапазон и улучшить качество последующего моделирования. Дополнительно были обработаны выбросы в числовых признаках 'stress level' и 'phsical activatity level' с использованием межквартильного размаха, что значительно улучшило распределение данных и снизило влияние аномальных значений.

Была произведена обработка нестандартного признака 'Occupation' с применением One-Hot кодирования, что позволило преобразовать категориальные данные в числовую форму, пригодную для использования в моделях машинного обучения. На этапе отбора признаков были использованы три метода: фильтрационный (VarianceThreshold), обертывания (RFE) и встроенный (Lasso Regression). Каждый метод продемонстрировал свою эффективность в выборе наиболее релевантных признаков, что важно для построения точных и интерпретируемых моделей.

Результаты проведенного анализа подтвердили значимость предварительной обработки данных в машинном обучении. Правильно chosen и обработанные признаки не только улучшают производительность моделей, но и позволяют получить более глубокое понимание данных.