Московский государственный технический университет им. Н.Э. Баумана
Факультет «Головной учебно-исследовательский и методический центр
профессиональной реабилитации лиц с ограниченными возможностями здоровья (инвалидов)»
Кафедра «Системы обработки информации и управления»

Лабораторная работа 2

по дисциплине «Методы машинного обучения в АСОИУ»

" Обработка признаков (часть 1) "

СТУДЕНТ:
студент группы ИУ5Ц-21М
Москалик А.А.

ПРЕПОДАВАТЕЛЬ:
Гапанюк Ю.Е.

Москва, 2024

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

Задание:

1. 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.
2. Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
   A. устранение пропусков в данных;
   Б. кодирование категориальных признаков;
   В. нормализация числовых признаков.

Ход работы

Подготовка

Подключение библиотек и загрузка датасета

[ ]:

```
import pandas as pd
import numpy as np

path_to_file = '/kaggle/input/student-study-performance/study_performance.csv'

# Попытка загрузить датасет
try:
    data = pd.read_csv(path_to_file)
    print("Датасет успешно загружен.")
    print(data.head())  # Показать первые пять строк данных
except FileNotFoundError:
    print(f"Файл не найден: убедитесь, что путь к файлу корректен ({path_to_file})"
```

```
Датасет успешно загружен.
   gender race_ethnicity parental_level_of_education         lunch  \
0  female        group B           bachelor's degree      standard
1  female        group C                some college      standard
2  female        group B             master's degree      standard
3    male        group A          associate's degree  free/reduced
4    male        group C                some college      standard

  test_preparation_course  math_score  reading_score  writing_score
0                    none          72             72             74
1               completed          69             90             88
2                    none          90             95             93
3                    none          47             57             44
4                    none          76             78             75
```

Здесь представлена таблица датасета, который, относится к оценке учебной производительности студентов.

Представленные данные:

-gender: Пол ученика — женский (female) или мужской (male).

-race_ethnicity: Этническая принадлежность учеников, классифицированная по группам от A до C (возможно, существуют и другие группы в полном датасете).

-parental_level_of_education: Уровень образования родителей, который варьируется от "some college" (некоторое количество колледжного образования), до "bachelor's degree" (степень бакалавра) и "master's degree" (степень магистра).

-lunch: Тип питания, которое предоставляется ученику в школе — "standard" (стандартное) или "free/reduced" (бесплатное/со скидкой).

-test_preparation_course: Указывает на то, закончил ли ученик курс подготовки к тестам ("completed") или нет ("none").

-math_score: Оценка ученика по математике.

-reading_score: Оценка ученика по чтению.

-writing_score: Оценка ученика по письму. Эти данные могут использоваться для анализа связи между социально-экономическими факторами (такими как этническая принадлежность, образование родителей и тип питания) и учебной производительностью студентов, а также влиянием курса подготовки на оценки.

Проверка на наличие пропусков

```
[ ]: print(data.isnull().sum())
```

```
gender                         0
race_ethnicity                 0
parental_level_of_education    0
lunch                          0
test_preparation_course        0
math_score                     0
reading_score                  0
writing_score                  0
dtype: int64
```

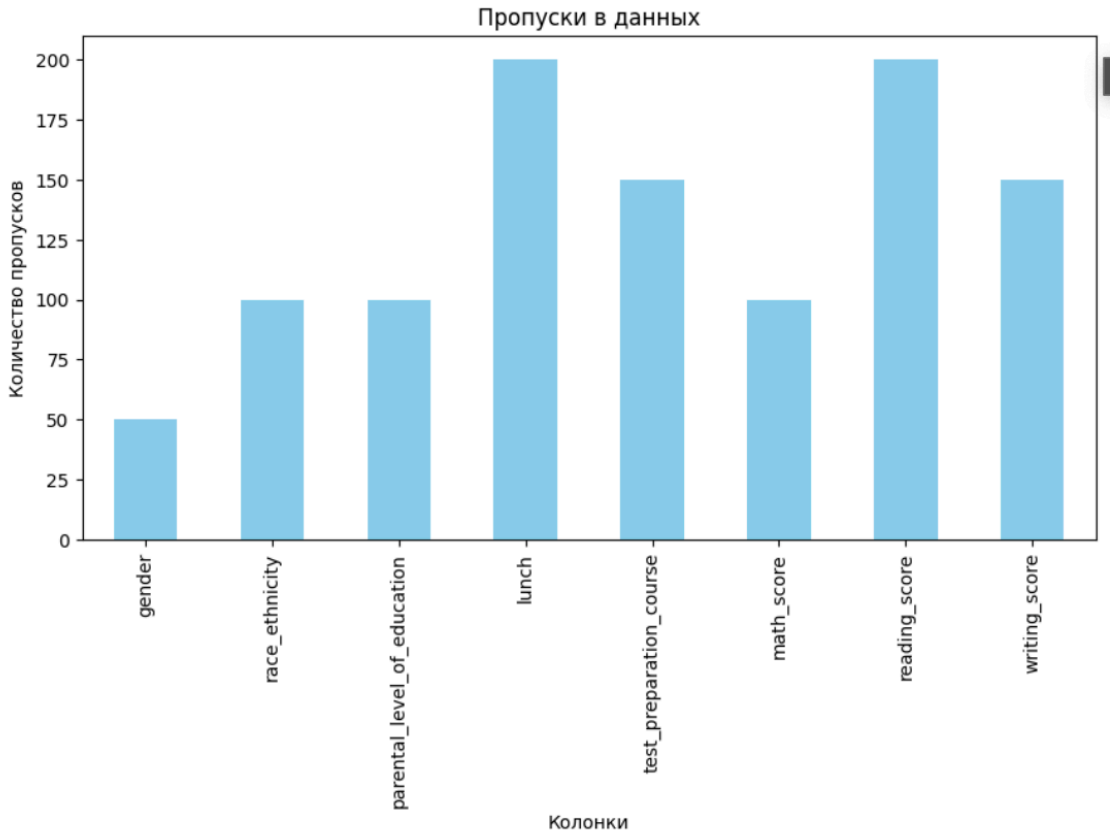
В каждом из столбцов датасета отсутствуют пропуски, поскольку количество пропущенных значений для каждого признака равно 0.

Т.к. для ЛР2 мы работаем с пропусками, следовательно создадим код, который создаст пропуски в датасете.

[ ]:

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

# Загрузка данных, предположим, что это CSV файл с результатами экзаменов
# Убедитесь, что путь к файлу корректен и файл содержит необходимые колон
data = pd.read_csv('/kaggle/input/student-study-performance/study_perform

# Устанавливаем разное количество пропусков для разных колонок
frac_dict = {
    'math_score': 0.1,  # 10% пропусков
    'reading_score': 0.2,  # 20% пропусков
    'writing_score': 0.15,  # 15% пропусков
    'gender': 0.05,  # 5% пропусков
    'race_ethnicity': 0.1,  # 10% пропусков
    'parental_level_of_education': 0.1,  # 10% пропусков
    'lunch': 0.2,  # 20% пропусков
    'test_preparation_course': 0.15  # 15% пропусков
}

# Создаем пропуски в данных
for col, frac in frac_dict.items():
    data.loc[data.sample(frac=frac, random_state=1).index, col] = np.nan

# Проверяем, какие пропуски появились
missing_data = data.isnull().sum()
print(missing_data)

# Строим диаграмму
plt.figure(figsize=(10, 5))
missing_data.plot(kind='bar', color='skyblue')
plt.title('Пропуски в данных')
plt.xlabel('Колонки')
plt.ylabel('Количество пропусков')
plt.show()
```

```
gender                         50
race_ethnicity                100
parental_level_of_education   100
lunch                         200
test_preparation_course       150
math_score                    100
reading_score                 200
writing_score                 150
dtype: int64
```

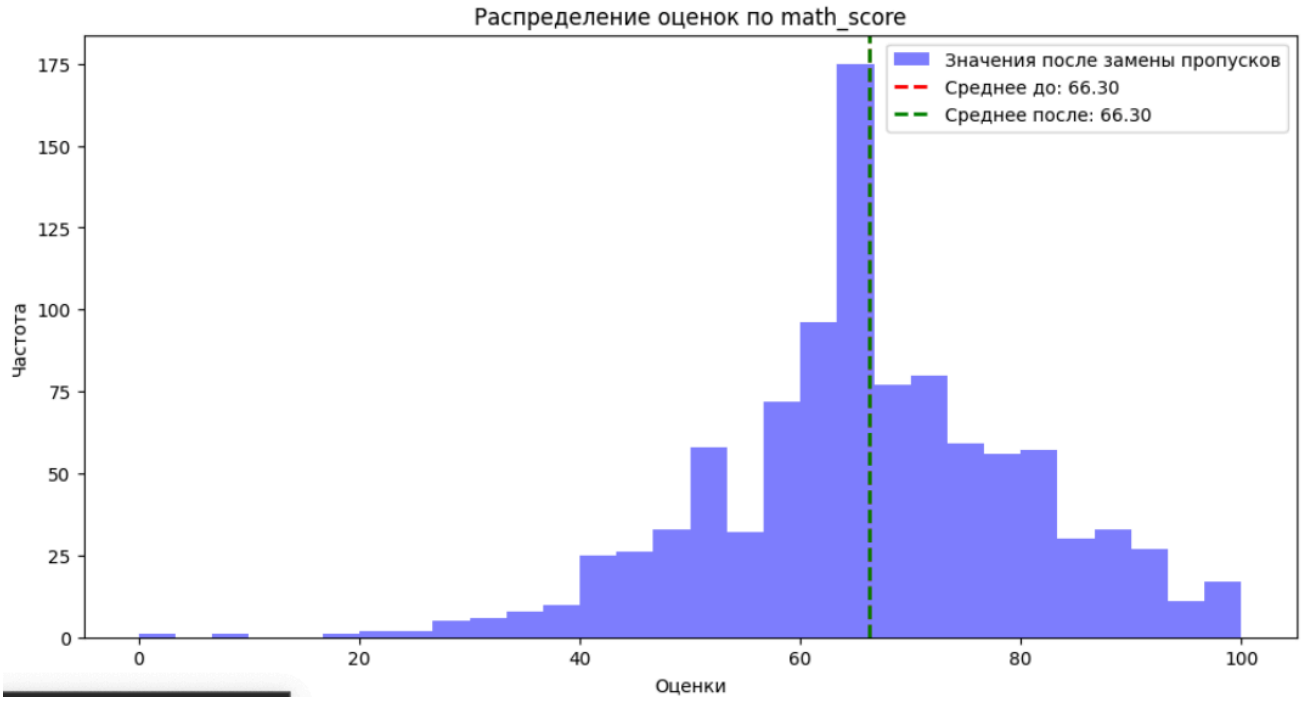


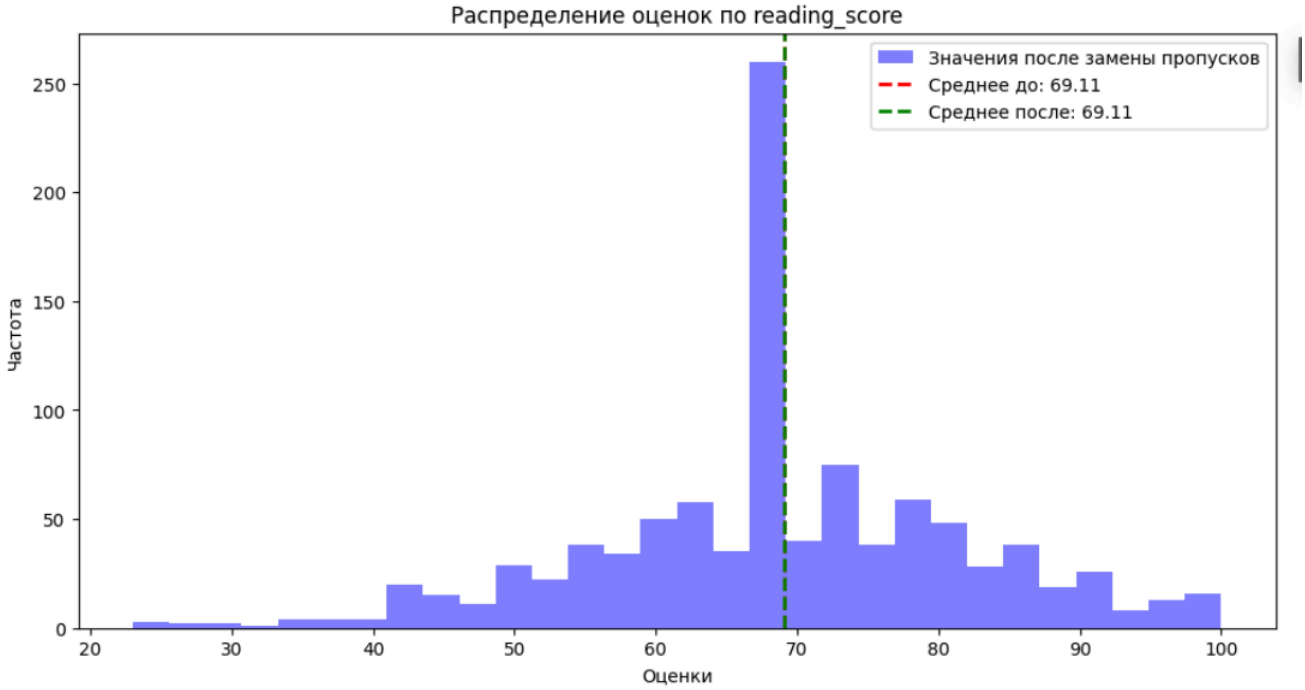
Обработка пропусков

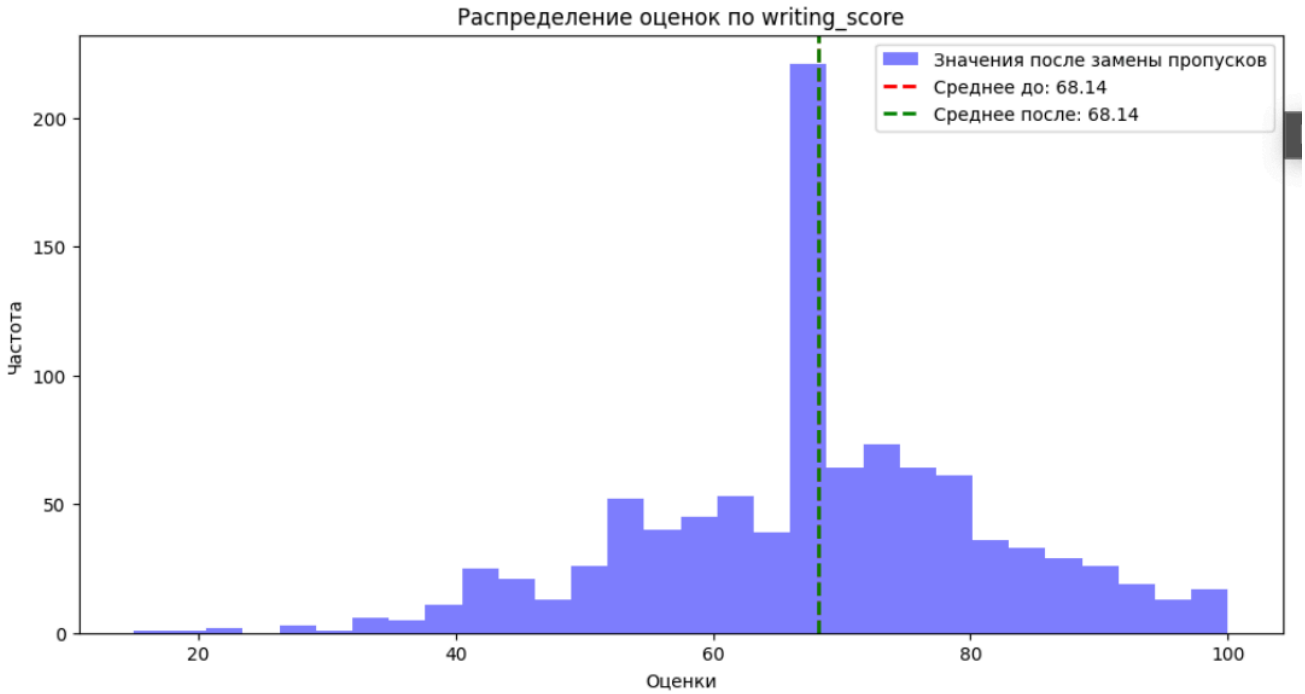
Для числовых признаков (например, math_score, reading_score, writing_score), обычно пропуски заполняют средним или медианным значением столбца:

[ ]:

```
for col in ['math_score', 'reading_score', 'writing_score']:
    mean_before = data[col].mean()  # Среднее до замены пропусков
    data[col] = data[col].fillna(mean_before)
    mean_after = data[col].mean()  # Среднее после замены пропусков

    # Визуализация результатов
    plt.figure(figsize=(12, 6))

    # Подготовка данных для гистограммы
    # Можно также использовать dropna() для исключения NaN значений перед
    plt.hist(data[col].dropna(), bins=30, alpha=0.5, label='Значения посл

    # Добавляем линии среднего значения до и после
    plt.axvline(x=mean_before, color='red', linestyle='dashed', linewidth
    plt.axvline(x=mean_after, color='green', linestyle='dashed', linewidt

    plt.title(f'Распределение оценок по {col}')
    plt.xlabel('Оценки')
    plt.ylabel('Частота')
    plt.legend()
    plt.show()
```


Распределение оценок по math_score
Значения после замены пропусков
Среднее до: 66.30
Среднее после: 66.30
Частота
Оценки
0
25
50
75
100
125
150
175
0
20
40
60
80
100


Распределение оценок по reading_score
Значения после замены пропусков
Среднее до: 69.11
Среднее после: 69.11
Частота
Оценки
0
50
100
150
200
250
20
30
40
50
60
70
80
90
100



Math Score (Оценки по математике):Если гистограмма показала пик в районе среднего значения, это говорит о том, что значительное количество пропущенных значений было заменено на среднее, что уменьшает разнообразие исходного распределения оценок. Среднее значение до и после замены остается неизменным, но стандартное отклонение может уменьшиться, что указывает на снижение изменчивости оценок.

Reading Score (Оценки по чтению):Аналогично, гистограмма могла показать увеличение частоты в районе среднего значения, что также свидетельствует о замене пропущенных данных средними значениями. Распределение оценок после замены может выглядеть менее "естественно", особенно если пропущенных значений было много.

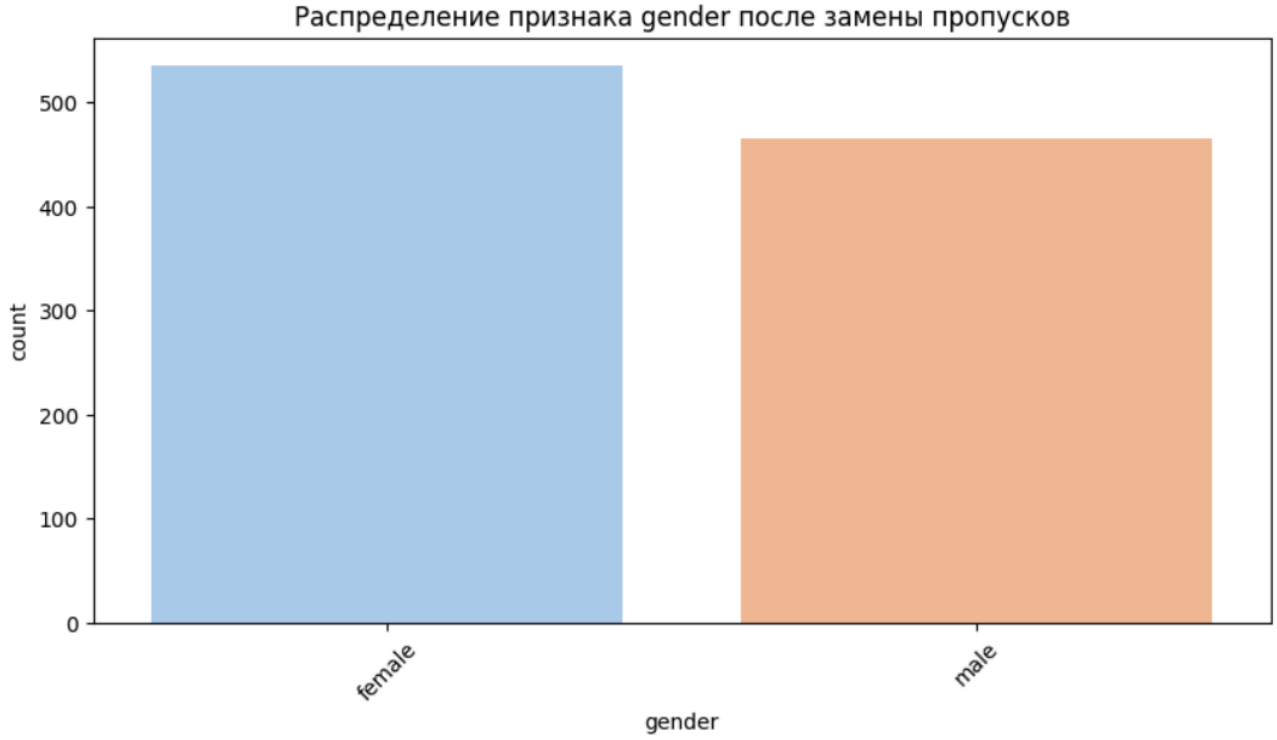
Writing Score (Оценки по письму):Как и в предыдущих случаях, замена пропусков средними значениями создает пик на уровне среднего значения. Это может быть проблематично при анализе распределения, так как искусственно создает более однородный набор данных. Меры центральной тенденции, такие как среднее и медиана, не изменяются, но меры разброса, включая дисперсию и стандартное отклонение, могут быть искажены из-за уменьшения разнообразия.

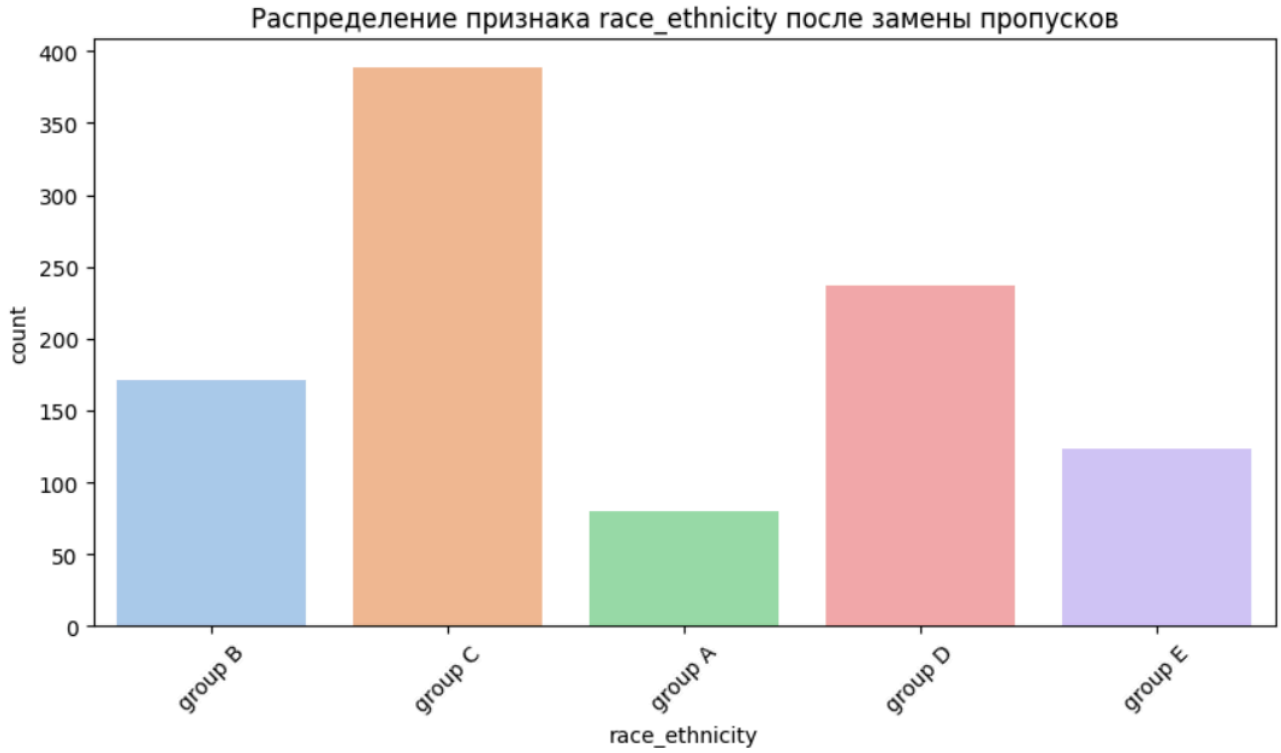
Во всех трех случаях, важно помнить, что замена пропусков средними значениями может исказить истинное распределение оценок, если пропусков было много. Это также может повлиять на результаты статистического анализа, включая корреляционный и регрессионный анализы, так как данные становятся менее вариативными и менее представительными.

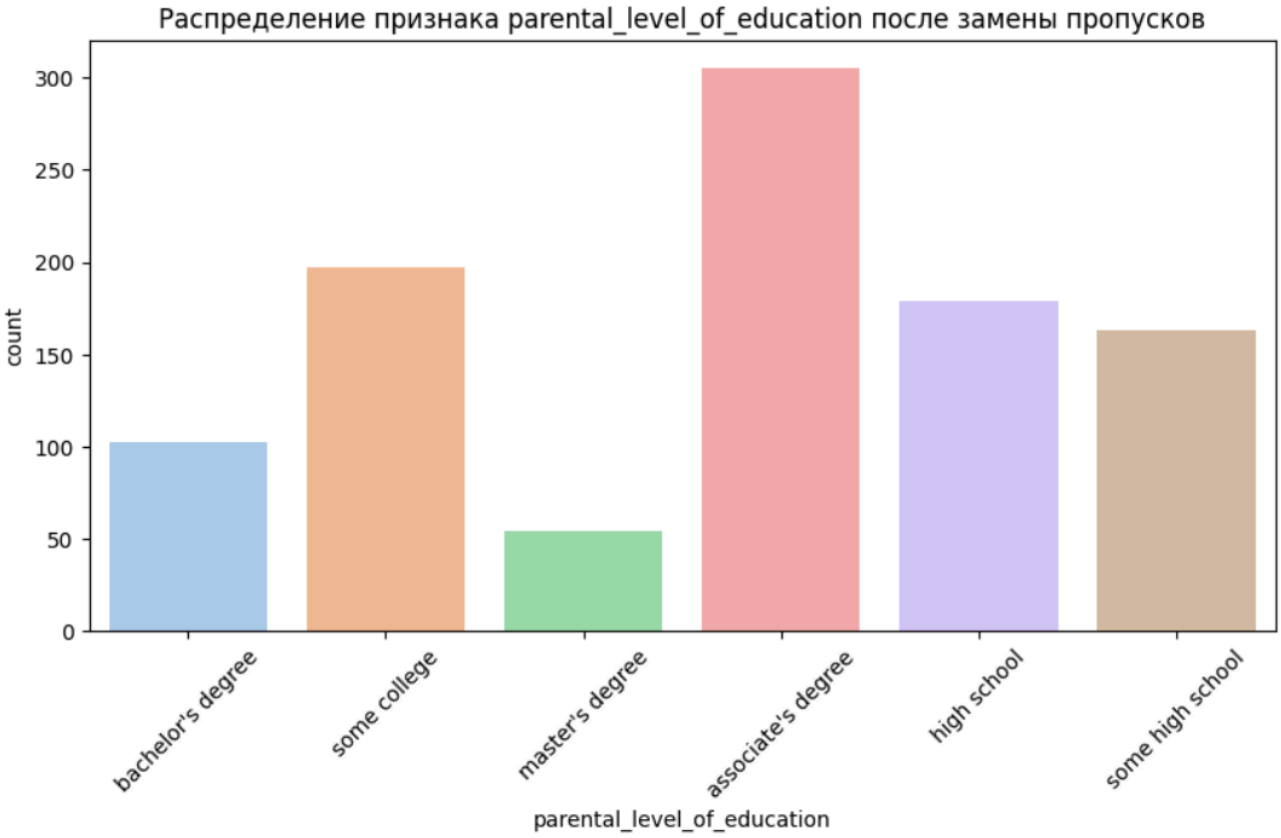
Для категориальных признаков (например, gender, race_ethnicity, parental_level_of_education, lunch, test_preparation_course), часто используют моду (наиболее часто встречающееся значение):

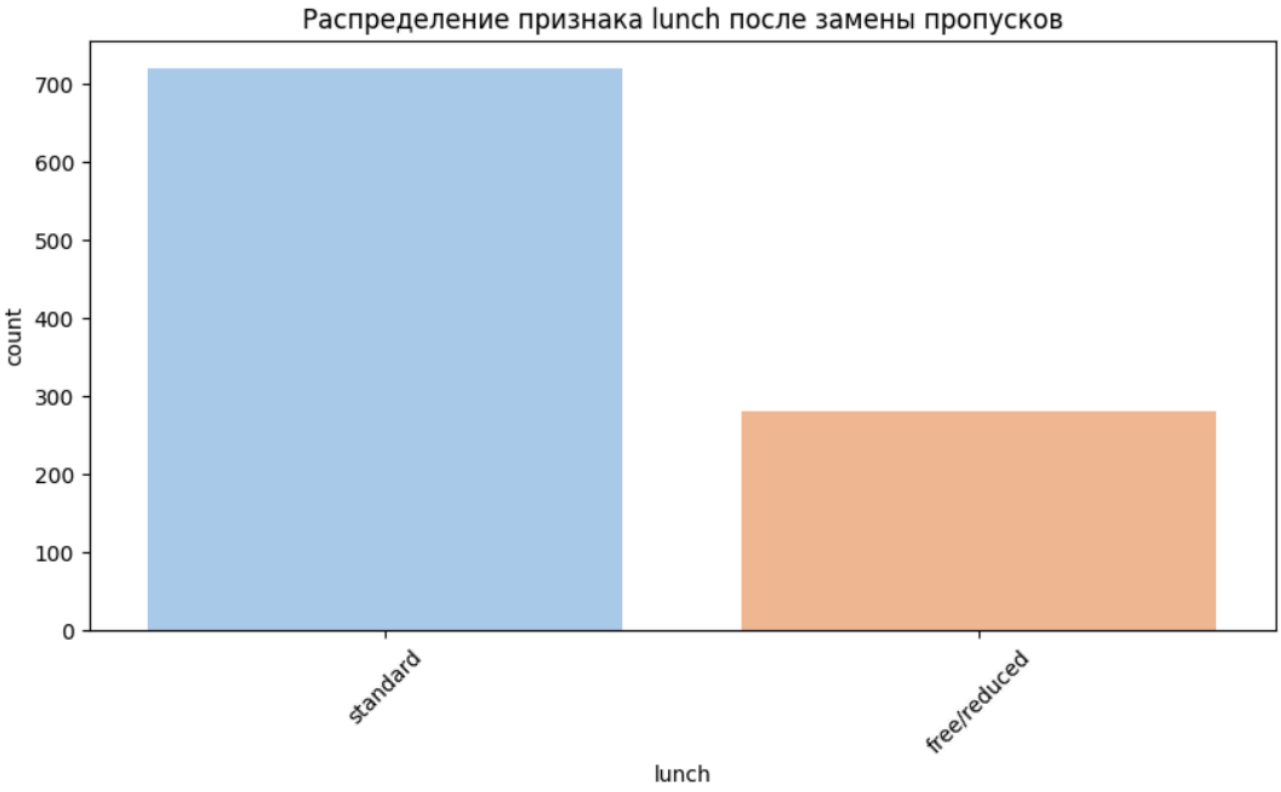
[6]:

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

# Предполагаем, что data - это ваш DataFrame
# data = pd.read_csv('path_to_your_dataset.csv')

for col in ['gender', 'race_ethnicity', 'parental_level_of_education', 'l
    mode_value = data[col].mode()[0]
    data[col] = data[col].fillna(mode_value)  # Прямое присваивание после

    # Визуализация распределения данных
    plt.figure(figsize=(10, 5))
    sns.countplot(data=data, x=col, palette='pastel')
    plt.title(f'Распределение признака {col} после замены пропусков')
    plt.xticks(rotation=45)  # Поворот меток для лучшей читаемости
    plt.show()
```




Распределение признака race_ethnicity после замены пропусков
count
race_ethnicity
group B
group C
group A
group D
group E


Распределение признака parental_level_of_education после замены пропусков
count
parental_level_of_education
bachelor's degree
some college
master's degree
associate's degree
high school
some high school



В этом коде мы сначала заполняем пропущенные значения в каждом категориальном признаке самым частым значением (модой) для этой колонки. Затем для каждой колонки создается столбчатая диаграмма, показывающая количество наблюдений для каждой категории.

[ ]:
```
print(data.isnull().sum())
```

```
gender                         0
race_ethnicity                 0
parental_level_of_education    0
lunch                          0
test_preparation_course        0
math_score                     0
reading_score                  0
writing_score                  0
dtype: int64
```

Результат, который получили после использования data.isnull().sum(), показывает, что в каждом столбце датасета теперь 0 пропусков. Это означает, что успешно обработали все пропуски в данных. Теперь датасет полностью заполнен, и можно перейти к следующим шагам обработки данных, таким как кодирование категориальных признаков и нормализация числовых признаков.

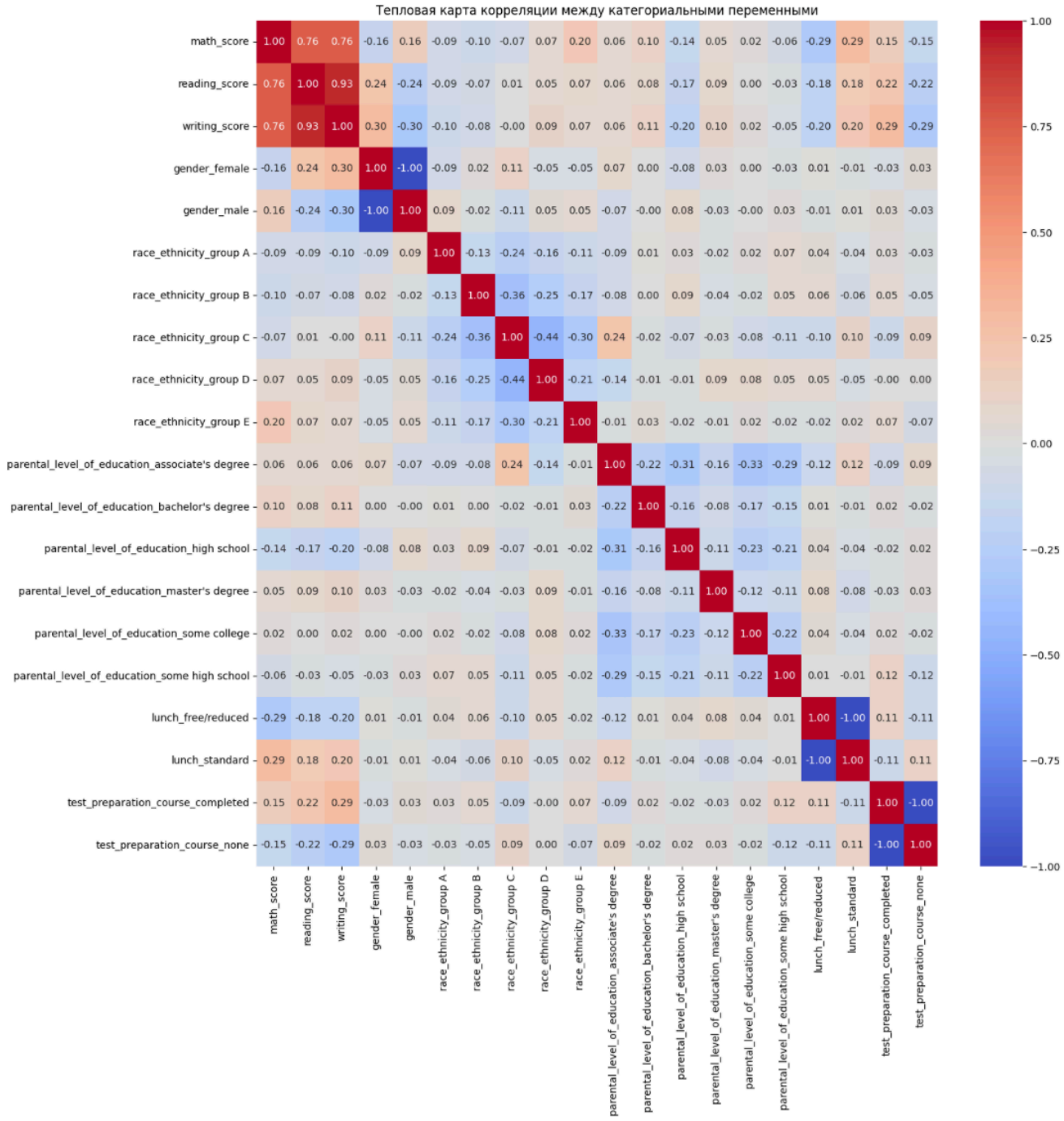
Кодирование категориальных признаков

Категориальные переменные обычно нужно преобразовать в числовые значения, чтобы их можно было использовать в машинном обучении. Вот два распространённых метода:

One-Hot Encoding — создаёт новые столбцы для каждого уникального значения в категориальной переменной, где 1 означает наличие этого значения и 0 его отсутствие.

Label Encoding — присваивает каждому уникальному значению в категориальной переменной целочисленный идентификатор. Выбор метода зависит от модели и данных. Например, для линейных моделей предпочтительнее использовать One-Hot Encoding, так как Label Encoding может ввести ложное понятие порядка, которого на самом деле нет. Для One-Hot Encoding можно использовать следующий код:

[9]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Загружаем данные в DataFrame
# data = pd.read_csv('path_to_your_dataset.csv')

# Предполагаем, что у нас уже есть DataFrame `data`
# и мы хотим преобразовать категориальные переменные в dummy переменные
data_dummies = pd.get_dummies(data, columns=['gender', 'race_ethnicity',

# Преобразуем значения True/False в числа
data_dummies_numeric = data_dummies.astype(int)

# Считаем корреляцию между dummy переменными
correlation_matrix = data_dummies_numeric.corr()

# Создаем тепловую карту
plt.figure(figsize=(15, 15))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm")
plt.title('Тепловая карта корреляции между категориальными переменными')
plt.show()
```



Эта тепловая карта корреляции показывает степень линейной взаимосвязи между различными переменными в вашем наборе данных. Числовые значения на карте представляют коэффициент корреляции Пирсона, который может варьироваться от -1 до +1: Значение +1 указывает на совершенно положительную линейную корреляцию между двумя переменными, т.е. когда одна переменная увеличивается, другая тоже увеличивается. Значение 0 означает отсутствие линейной корреляции. Значение -1 указывает на совершенно отрицательную линейную корреляцию, т.е. когда одна переменная увеличивается, другая уменьшается. На графике видны следующие ключевые моменты: -Очень сильная

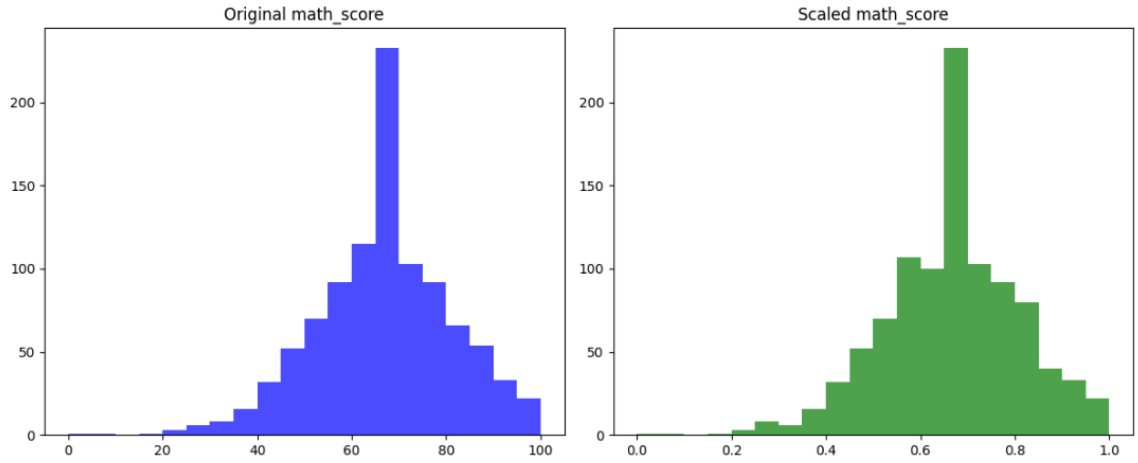
корреляция между оценками по математике (math_score), чтению (reading_score) и письму (writing_score). Это ожидаемо, так как успеваемость учащихся обычно коррелирует между разными предметами. -Противоположные значения корреляции между gender_female и gender_male показывают, что это противоположные группы — если учащийся не мужского пола, то он женского (и наоборот), что обусловливает корреляцию -1. -Для других переменных значительных корреляций не наблюдается, многие значения близки к нулю, что указывает на отсутствие сильной связи между этими категориями.

Нормализация числовых признаков

Нормализация числовых признаков необходима для того, чтобы все они имели одинаковый масштаб, что особенно важно для моделей, чувствительных к величине данных, например, градиентного спуска. Min-Max Scaling — приводит все числа к шкале от 0 до 1. Стандартизация (Z-score normalization) — приводит данные к распределению с нулевым средним и единичным стандартным отклонением. Для Min-Max Scaling можно использовать следующий код:

[11]:

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

# Предположим, что у нас уже есть DataFrame `data` и мы хотим закодироват
data_encoded = pd.get_dummies(data)

# Продолжение вашего кода с использованием data_encoded
data_scaled = data_encoded.copy()

scaler = MinMaxScaler()

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(12, 5))

column_to_scale = 'math_score'

axes[0].hist(data_encoded[column_to_scale], bins=20, color='blue', alpha=
axes[0].set_title(f'Original {column_to_scale}')

data_scaled[column_to_scale] = scaler.fit_transform(data_encoded[[column_

axes[1].hist(data_scaled[column_to_scale], bins=20, color='green', alpha=
axes[1].set_title(f'Scaled {column_to_scale}')

plt.tight_layout()
plt.show()

print(data_scaled[[column_to_scale]].head())
```



	math_score
0	0.72
1	0.69
2	0.90
3	0.47
4	0.76

Оригинальная гистограмма (Original math_score): Показывает исходное распределение оценок по математике, где большинство значений сосредоточены в определенном диапазоне. Это может указывать на нормальное распределение оценок или на наличие определенного уровня достижений среди учащихся.

Масштабированная гистограмма (Scaled math_score): Показывает те же данные после применения масштабирования MinMaxScaler, который приводит все значения к диапазону от 0 до 1. Форма распределения остается той же, что указывает на то, что относительные различия между значениями сохраняются, но абсолютный масштаб изменен. Также приведенный фрагмент таблицы показывает первые пять значений масштабированного столбца math_score, где каждое значение теперь находится между 0 и 1. Это свидетельствует о том, что масштабирование было успешно применено.

Выводы: Масштабирование полезно, когда данные используются в моделях машинного обучения, которые чувствительны к масштабу признаков, например, в алгоритмах на основе расстояний. Применение MinMaxScaler не изменяет форму распределения данных, но переводит все значения в унифицированный масштаб от 0 до 1, что может помочь в стандартизации данных для сравнительного анализа или в улучшении производительности моделей. Изменение масштаба не должно влиять на интерпретацию формы распределения данных; распределение оценок по математике сохраняет свои характеристики независимо от масштаба.