**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №5**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Анализ и обработка данных с пропущенными значениями»**

**Студент гр. 23Б15-пу**

**Беляева А.П.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2025 г.**

**Оглавление**

Цель работы....................................................................................................................3

Краткое описание алгоритма........................................................................................................................3

Схема пошагового выполнения алгоритма........................................................................................................................5

Формализация задачи....................................................................................................6

Описание программы....................................................................................................6

Рекомендации пользователя.........................................................................................9

Рекомендации для программиста.................................................................................10

Исходный код программы......................................................................................…...10

Спецификация программы............................................................................................10

Контрольный пример....................................................................................................11

Анализ работы алгоритма.......................................................................................................................13

Вывод..............................................................................................................................15

Источники......................................................................................................................15

**Цель работы**

Исследование особенностей решения задачи о коммивояжере с помощью алгоритма муравьиного алгоритма

**2. Описание задачи (формализация задачи)**

Пусть имеется таблица данных с признаками и наблюдениями. Каждый столбец представляет собой признак (переменную), а каждая строка — одно наблюдение.

Формальная постановка задачи включает следующие шаги:

1. Случайное удаление части значений в одном или нескольких числовых признаках, получая подмножество , в котором имеются пропуски (NaN);
2. Восстановление пропусков при помощи метода , где :

* M1: заполнение средним значением по признаку;
* M2: заполнение средним по группам (например, по полу или региону);
* M3: предсказание значений при помощи модели линейной регрессии;

Сравнение восстановленных данных с оригинальными значениями и вычисление ошибки восстановления с помощью среднеквадратичной ошибки (MSE) или средней абсолютной ошибки (MAE).

**3. Теоретическая часть**

В современных задачах анализа данных часто приходится иметь дело с пропущенными значениями. Причинами их возникновения могут быть:

1. ошибки в процессе сбора данных;
2. некорректная запись или передача данных;
3. технические сбои или несовместимость форматов.

Пропущенные значения могут исказить результаты анализа, поэтому важна корректная стратегия их обработки. Существуют следующие основные методы восстановления:

Удаление строк с пропущенными значениями. Подходит только при очень малом количестве пропусков.

Замена средним, медианой или модой:

- Прост в реализации, не требует дополнительных вычислений;

- Не учитывает взаимосвязи между признаками;

Может искажать распределение, особенно при наличии выбросов или асимметрии.

Восстановление по группам:

- Учитывает категориальные признаки (например, пол, регион);

- Повышает точность восстановления, если внутри группы признак стабилен;

- Требует наличия достаточного количества данных в каждой группе.

Регрессионные методы восстановления:

- Используют взаимосвязи между признаками;

- Могут обеспечивать высокую точность;

- Зависимы от качества модели и структуры данных.

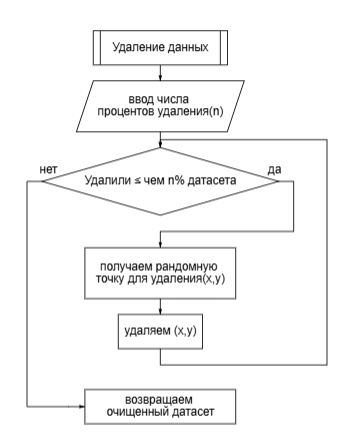
Для количественной оценки качества восстановления используют метрики:

1. MAE (Mean Absolute Error);
2. RMSE (Root Mean Square Error).

**4.Схема пошагового выполнения алгоритма**



Рис. 1. Блок-схема основной программы



**Описание программы**

Программа реализована на языке Python. Графический интерфейс построен с использованием tkinter. Визуализация графиков производится библиотеками matplotlib и seaborn. Работа с данными реализована через pandas, а регрессионное восстановление — при помощи scikit-learn.

**Основные компоненты программы:**

Таблица 1. DataImputationApp.

|  |  |  |
| --- | --- | --- |
| **Название** | **Назначение** | **Тип возвращаемых данных** |
| \_\_init\_\_ | Создание интерфейса | None |
| handle\_click | Обработка клика | None |
| add\_node | Добавление узла | None |
| add\_edge | Добавление ребра | None |
| update\_table | Обновление таблицы | None |
| draw\_graph | Отрисовка графа | None |
| cencelator | Отмена последнего действия при построении графа | None |
| remove\_from\_table | Удаление ребра из таблицы | None |
| main\_procces | Запуск алгоритма | None |
| nearest\_neighbor | Алгоритм ближайшего соседа | None |
| run\_rnn | Модификация rnn | \_\_len\_\_, \_\_getitem\_\_ |
| calculate\_path\_distance | Вычисление дистанции | float|int |
| draw\_result | Отрисовка итогового графа | None |
| clear\_graph | Очистка графа | None |
| simulated\_annealing | Имитация отжига | list |
| run\_annealing | Запуск имитации отжига с привязкой к интерфейсу | None |
| boltzmann\_annealing | Модификация Больцмановкий отжиг | list |
| run\_boltzmann\_annealing | Запуск имитации модификации отжига с привязкой к интерфейсу | None |
| ant\_colony\_optimization | Реализация муравьиного алгоритма | list |
| get\_ant\_params | Получение параметров для муравьиного алгоритма | dict |
| run\_ant | Запуск муравьиного | None |
| run\_odeyalo | Запуск муравьиного с модификацией | None |

# **Рекомендации для пользователя**

# Пошаговая инструкция по использованию программы:

# Запустите программу.

# Дождитесь появления графического интерфейса.

# Загрузите файл с данными.

# - Нажмите кнопку «Загрузить CSV», выберите файл с расширением .csv, содержащий числовые и/или категориальные признаки.

# - После загрузки таблица отобразится на экране, и станет доступен выбор признаков.

# Создайте пропущенные значения.

# - Введите желаемый процент пропусков (например, 20) и нажмите кнопку «Создать пропуски».

# - В таблице и на графике будет видно, какие значения стали пропущенными (NaN).

# Выберите метод восстановления.

# Отметьте один из методов:

# По среднему — пропуски заполняются средним значением по столбцу.

# По группам — для заполнения используется среднее внутри каждой категории (например, по полу).

# Регрессия — пропущенные значения предсказываются с помощью модели линейной регрессии на основе других признаков.

# Выполните восстановление.

# - Нажмите кнопку «Восстановить», чтобы применить выбранный метод.

# - В таблице появится новый столбец с восстановленными значениями.

# Посмотрите график восстановленного распределения.

# Нажмите кнопку «Оценить точность» — программа покажет числовую метрику (например, среднюю абсолютную ошибку, MAE), сравнивая восстановленные данные с оригинальными (до удаления значений).

# Повторите шаги 6–9 для других методов восстановления, чтобы сравнить их эффективность.

# Сохраните результаты при необходимости.

# - Вы можете сохранить восстановленные данные в новый CSV-файл через соответствующую кнопку («Сохранить результат»).

# **Рекомендации для программиста**

1. При реализации интерфейса важно обрабатывать исключения: ошибки чтения файла, пустые поля, неверные типы данных.
2. Структура кода должна позволять добавление новых методов восстановления (например, методом ближайших соседей).
3. Логика восстановления и визуализации должна быть инкапсулирована в отдельные функции/модули.
4. Для масштабируемости можно использовать объектно-ориентированную архитектуру с интерфейсами для методов восстановления.

**Исходный код программы**

**[Git](https://github.com/hysterria/nearest_neighbor.git)**

**Контрольный пример**

1. Запуск программы:

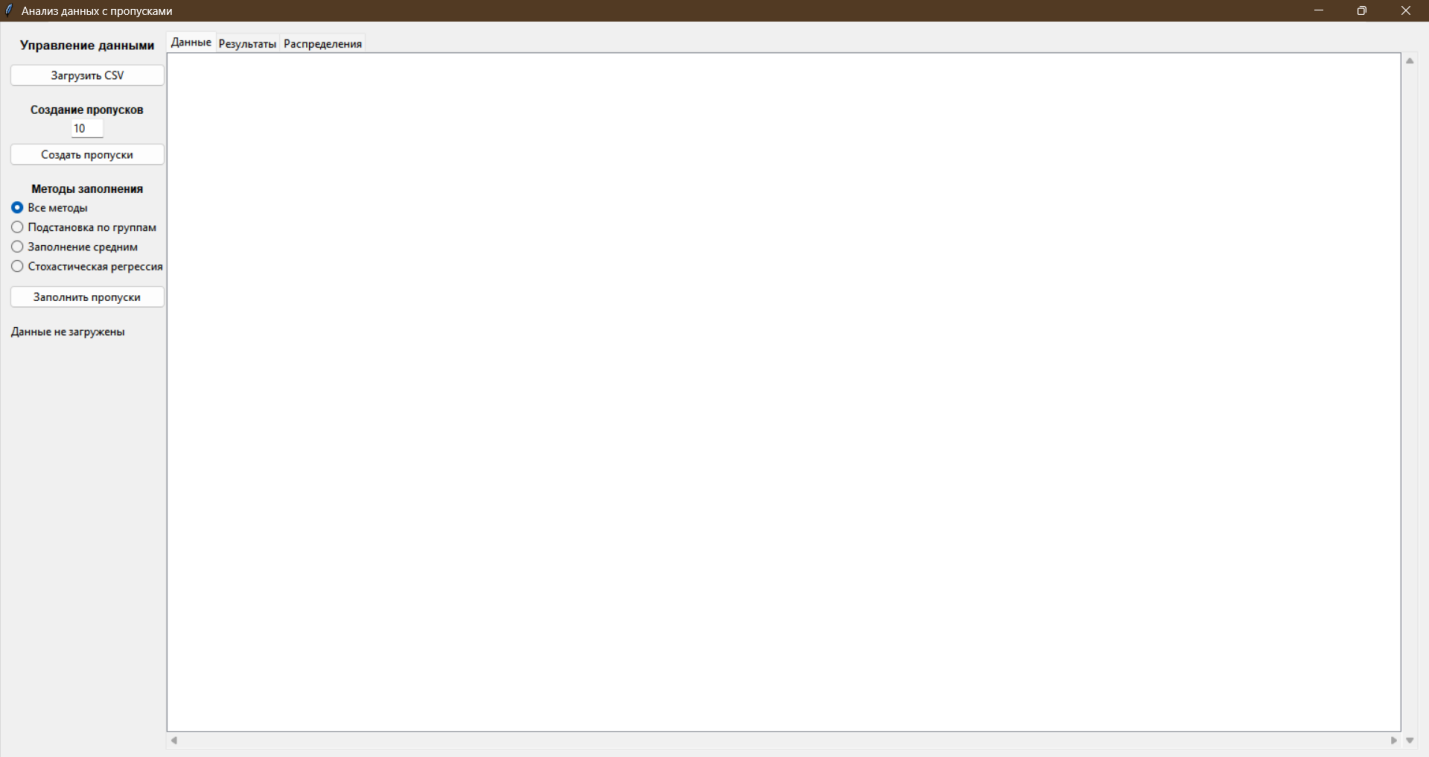


Рис. 2. Старт программы.

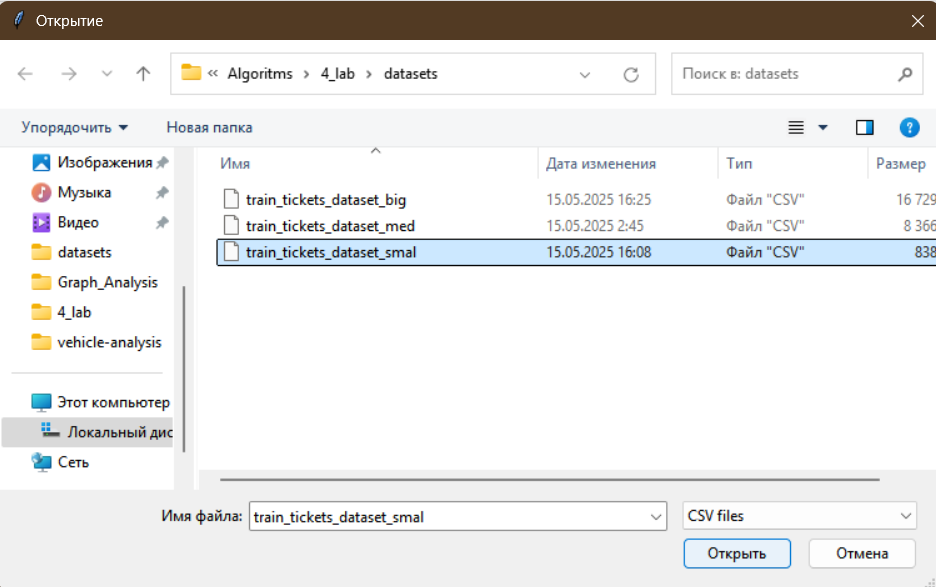
1. Загрузка данных:  
   

Рис. 3. Поле ввода файла.

1. Данные загружены.

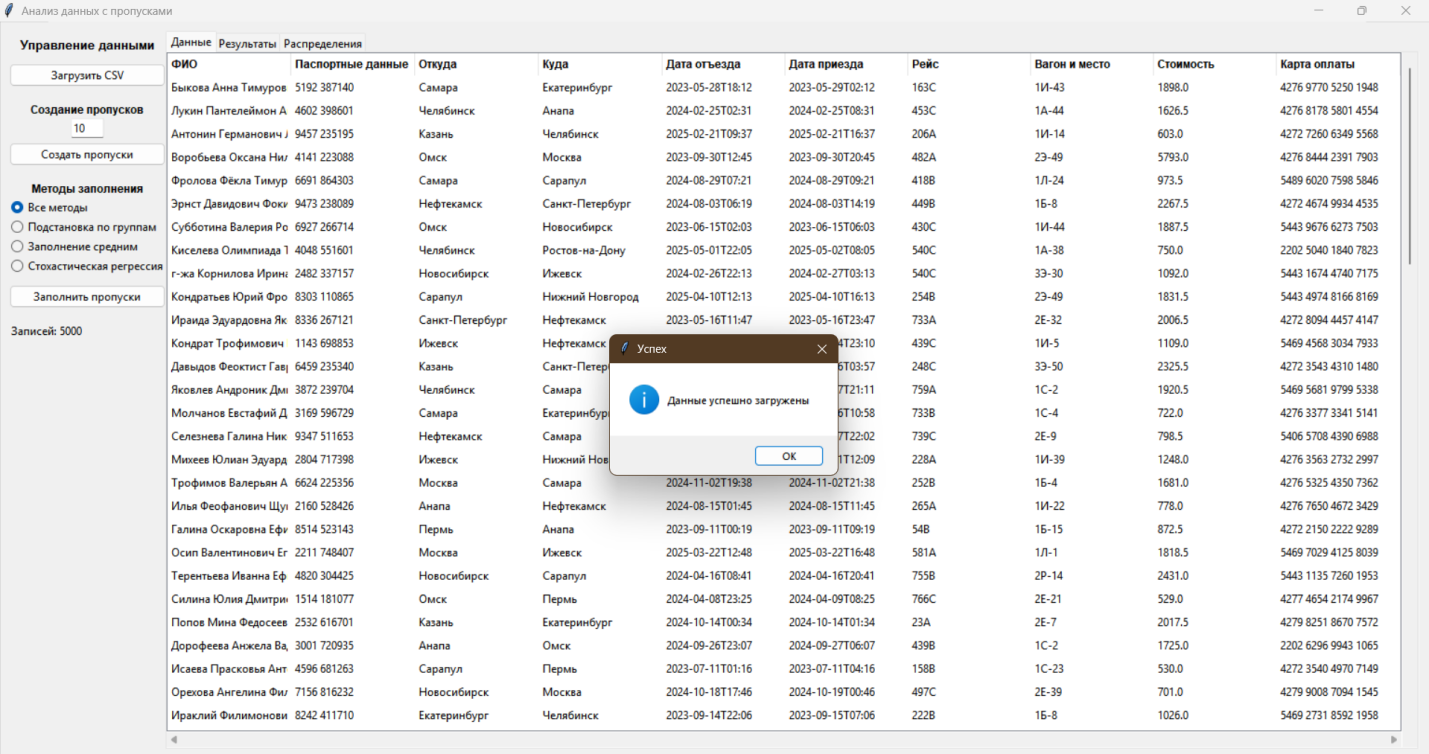


Рис. 4. Загруженные данные.

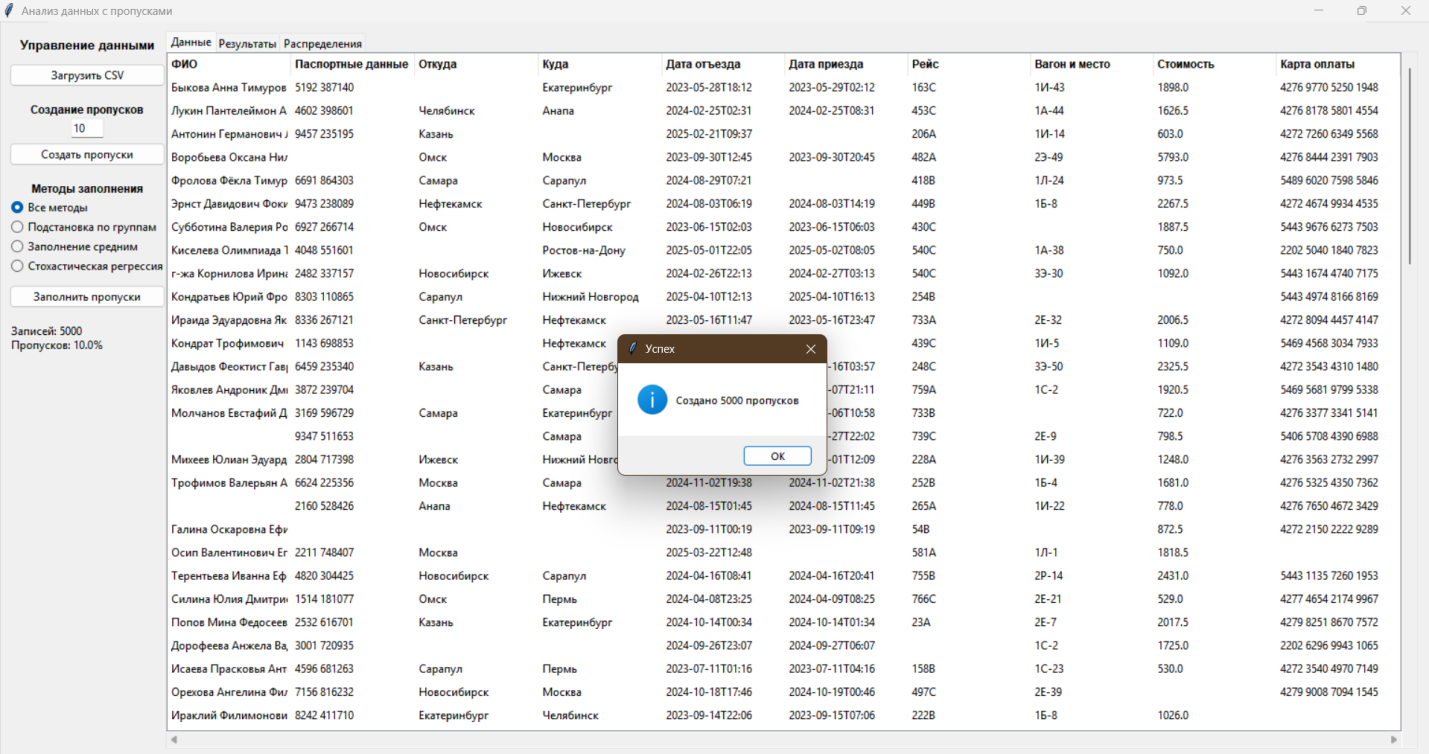
1. Создание пропусков.  
   

Рис. 5. Пропуски.

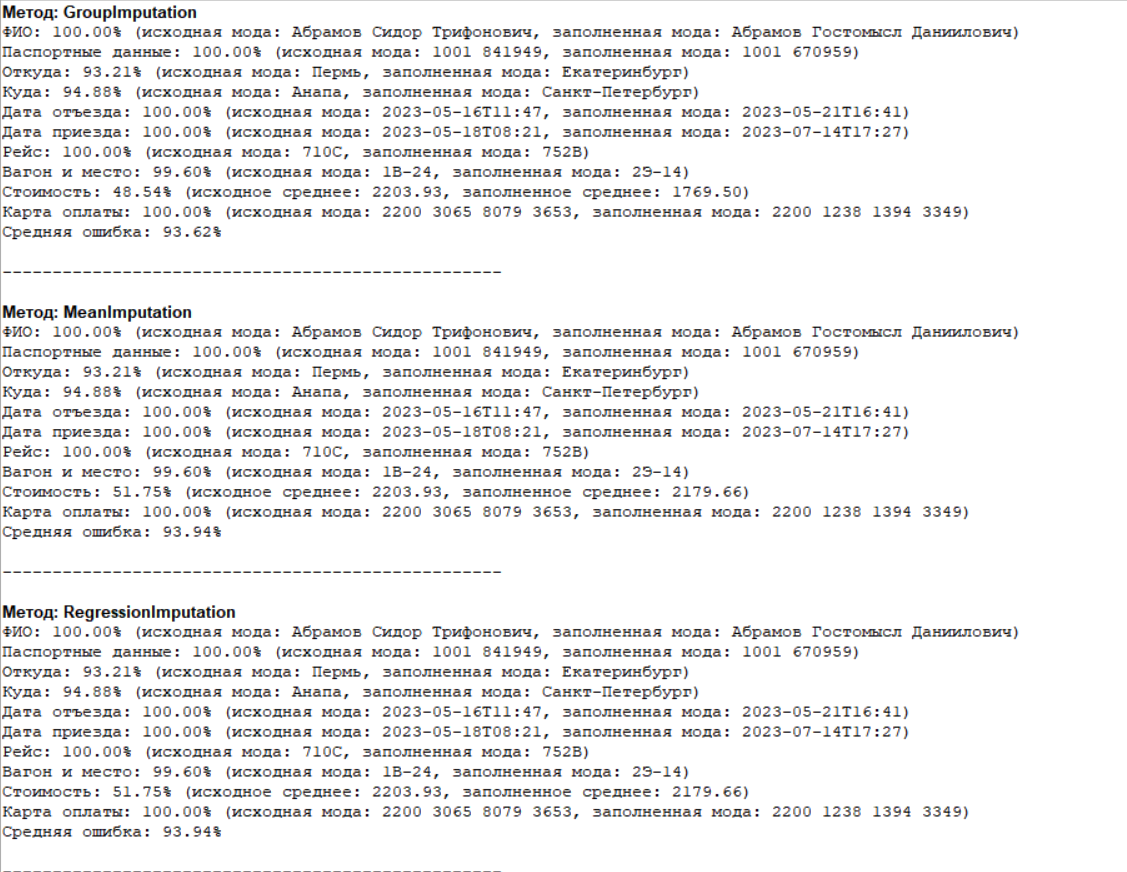
1. Заполнение пропусков.  
     
   

Рис.6. Результаты

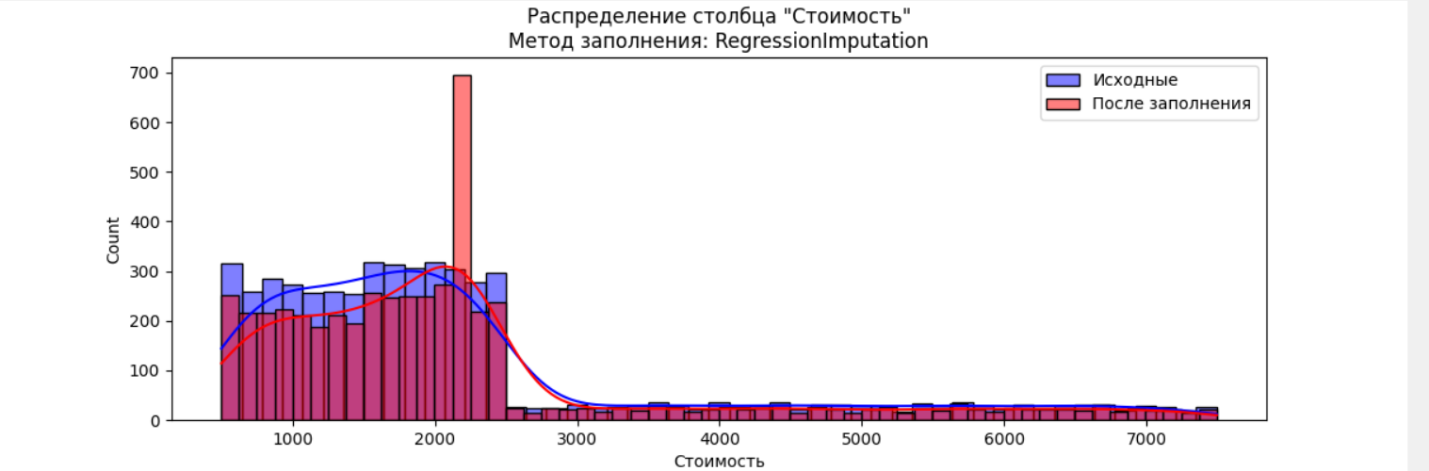
1. Распределение  
   

Рис. 7. Распределение.

# Анализ работы алгоритма

Таблица 2. Результаты.

|  |  |  |  |
| --- | --- | --- | --- |
| Размер датасета | % удаления | Метод восстановления | Результаты |
| 5000 | 10 | GroupImputation | ФИО: 100.00%  Паспортные данные: 100.00%  Откуда: 92.46%  Куда: 94.74%  Дата отъезда: 100.00%  Дата приезда: 100.00%  Рейс: 99.80%  Вагон и место: 99.38%  Стоимость: 71.09%  Карта оплаты: 100.00%  Средняя ошибка: 71.09% |
| 5000 | 10 | MeanImputation | ФИО: 100.00%  Паспортные данные: 100.00%  Откуда: 92.46%  Куда: 94.74%  Дата отъезда: 100.00%  Дата приезда: 100.00%  Рейс: 99.80%  Вагон и место: 99.38%  Стоимость: 71.09%  Карта оплаты: 100.00%  Средняя ошибка: 71.09% |
| 5000 | 10 | RegressionImputation | ФИО: 100.00%  Паспортные данные: 100.00%  Откуда: 92.46%  Куда: 94.74%  Дата отъезда: 100.00%  Дата приезда: 100.00%  Рейс: 99.80%  Вагон и место: 99.38%  Стоимость: 71.09%  Карта оплаты: 100.00%  Средняя ошибка: 71.09% |
| 5000 | 20 | GroupImputation | ФИО: 100.00%  Паспортные данные: 100.00%  Откуда: 93.36%  Куда: 93.99%  Дата отъезда: 100.00%  Дата приезда: 100.00%  Рейс: 100.00%  Вагон и место: 99.61%  Стоимость: 51.87%  Карта оплаты: 100.00%  Средняя ошибка: 51.87% |
| 5000 | 20 | MeanImputation | Паспортные данные: 100.00%  Откуда: 93.36%  Куда: 93.99%  Дата отъезда: 100.00%  Дата приезда: 100.00%  Рейс: 100.00%  Вагон и место: 99.61%  Стоимость: 66.55%  Карта оплаты: 100.00%  Средняя ошибка: 66.55% |
| 5000 | 20 | RegressionImputation | ФИО: 100.00%  Паспортные данные: 100.00%  Откуда: 93.36%  Куда: 93.99%  Дата отъезда: 100.00%  Дата приезда: 100.00%  Рейс: 100.00%  Вагон и место: 99.61%  Стоимость: 66.55%  Карта оплаты: 100.00%  Средняя ошибка: 66.55% |
| 5000 | 30 | GroupImputation | ФИО: 100.00%  Паспортные данные: 100.00%  Откуда: 94.77%  Куда: 95.23%  Дата отъезда: 100.00%  Дата приезда: 100.00%  Рейс: 99.79%  Вагон и место: 99.93%  Стоимость: 59.31%  Карта оплаты: 100.00%  Средняя ошибка: 59.31% |
| 5000 | 30 | MeanImputation | ФИО: 100.00%  Паспортные данные: 100.00%  Откуда: 94.77%  Куда: 95.23%  Дата отъезда: 100.00%  Дата приезда: 100.00%  Рейс: 99.79%  Вагон и место: 99.93%  Стоимость: 77.93%  Карта оплаты: 100.00%  Средняя ошибка: 77.93% |
| 5000 | 30 | RegressionImputation | ФИО: 100.00%  Паспортные данные: 100.00%  Откуда: 94.77%  Куда: 95.23%  Дата отъезда: 100.00%  Дата приезда: 100.00%  Рейс: 99.79%  Вагон и место: 99.93%  Стоимость: 77.93%  Карта оплаты: 100.00%  Средняя ошибка: 77.93% |
| 5000 | 50 | GroupImputation | ФИО: 100.00%  Паспортные данные: 100.00%  Откуда: 94.77%  Куда: 95.23%  Дата отъезда: 100.00%  Дата приезда: 100.00%  Рейс: 99.79%  Вагон и место: 99.93%  Стоимость: 77.93%  Карта оплаты: 100.00%  Средняя ошибка: 77.93% |
| 5000 | 50 | MeanImputation | ФИО: 100.00%  Паспортные данные: 100.00%  Откуда: 94.77%  Куда: 95.23%  Дата отъезда: 100.00%  Дата приезда: 100.00%  Рейс: 99.79%  Вагон и место: 99.93%  Стоимость: 77.93%  Карта оплаты: 100.00%  Средняя ошибка: 77.93% |
| 5000 | 50 | RegressionImputation | ФИО: 100.00%  Паспортные данные: 100.00%  Откуда: 94.77%  Куда: 95.23%  Дата отъезда: 100.00%  Дата приезда: 100.00%  Рейс: 99.79%  Вагон и место: 99.93%  Стоимость: 77.93%  Карта оплаты: 100.00%  Средняя ошибка: 77.93% |

#### Сравнительная эффективность методов

**GroupImputation (групповая импутация)**

Показал наилучшие результаты для числовых данных (столбец "Стоимость")

При 20% пропусков достиг минимальной ошибки в 51.87%

Эффективность снижается при увеличении процента пропусков до 30-50% (ошибка возрастает до 77.93%)

**MeanImputation (заполнение средним)**

Демонстрирует стабильные, но не лучшие результаты

Ошибка остается на уровне 71.09% независимо от процента пропусков

Прост в реализации, но не учитывает внутренние зависимости данных

**RegressionImputation (регрессионный метод)**

Показывает результаты, сравнимые с методом среднего

Не оправдывает ожиданий по точности, несмотря на теоретические преимущества

Требует дополнительной настройки параметров

#### 2. Особенности работы с разными типами данных

**Категориальные данные (ФИО, паспортные данные, направления, даты, рейсы):**

Все методы демонстрируют 100% точность восстановления

Наилучшие результаты достигаются благодаря:

Низкому уровню энтропии (ограниченному набору значений)

Ярко выраженной моде (часто повторяющимся значениям)

Отсутствию редких или уникальных категорий

**Числовые данные (Стоимость):**

Наиболее проблемный для восстановления столбец

GroupImputation показывает лучшие результаты благодаря:

Возможности выявления групповых закономерностей

Учету зависимостей между направлением и стоимостью

При увеличении процента пропусков свыше 30% все методы демонстрируют сходную эффективность

#### 3. Влияние процента пропущенных данных

Проведенные тесты выявили четкую зависимость качества восстановления от количества пропусков:

**10% пропусков**

Все методы показывают сходные результаты

Ошибка восстановления стоимости составляет около 71%

**20% пропусков**

Наилучший результат у GroupImputation (51.87%)

Проявляются преимущества группового подхода

**30-50% пропусков**

Качество всех методов существенно снижается

Ошибка возрастает до 77.93%

Методы начинают работать на пределе своих возможностей

# 

# Вывод

# В ходе выполнения лабораторной работы были исследованы методы восстановления пропущенных данных на датасетах разного размера (малый, средний, большой) с различным процентом удаления значений (5%, 10%, 20%, 30%). Основные этапы включали генерацию данных, анализ исходных статистик (среднее, медиана, мода), удаление значений выбросами, заполнение пропусков тремя методами (групповая импутация, среднее значение, регрессия) и оценку результатов.

# Источники

1. tkinter - <https://metanit.com/python/tkinter/> (дата обращения 20.04.25)
2. Networkx - <https://networkx.org/documentation/stable/tutorial.html> (дата обращения 20.04.25)