**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

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

**ОТЧЕТ**

**по лабораторной работе №4**

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

**на тему «Анализ методов заполнения датасета»**

**Студент гр. 23Б15-пу**

**Абрахин Е.Д.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2025 г.**

Оглавление

1. [Цель работы 3](#_Toc196444811)
2. [Описание задачи (формализация задачи) 3](#_Toc196444812)
3. [Теоретическая часть 4](#_Toc196444813)
4. [Основные шаги программы 6](#_Toc196444814)
5. [Блок схема программы 8](#_Toc196444815)
6. [Описание программы 11](#_Toc196444816)
7. [Рекомендации пользователя 13](#_Toc196444817)
8. [Рекомендации программиста 14](#_Toc196444818)
9. [Исходный код программы 14](#_Toc196444819)
10. [Контрольный пример 15](#_Toc196444820)
11. [Анализ результатов работы алгоритма 18](#_Toc196444821)
12. [Вывод 32](#_Toc196444822)
13. [Источники 34](#_Toc196444823)

# Цель работы

Исследовать и сравнить эффективность трёх методов восстановления пропущенных значений в табличных данных — Hot-Deck, метод заполнения моды и сплайн-интерполяцию — при различных объемах и уровнях «выбросного» удаления данных. Разработать и использовать программу для генерации синтетических датасетов, случайного удаления значений и восстановления, а также оценить качество восстановления на контрольных наборах.

# Описание задачи (формализация задачи)

1. **Изучение методов восстановления пропусков**

* Ознакомиться с принципами методов Hot-Deck, заполнения модой и сплайн-интерполяции;
* Определить сильные и слабые стороны каждого метода применительно к категориальным и числовым признакам.

1. **Разработка программы**

* Реализовать программу `DatasetFilling.pyw` (генерация билетов, удаление пропусков, три метода восстановления);
* Обеспечить вывод статистики и ошибок по каждому из 10 отображаемых столбцов (`FullName`, `PassportInfo` и т. д.) и суммарную ошибку.

1. **Формирование и предобработка датасетов**

Сгенерировать синтетические таблицы разных размеров:

-малый (~1 000–10 000 строк),

- средний (~20 000–75 000 строк),

- большой (~100 000–250 000 строк);

Для каждого датасета в выбранных столбцах удалить выбросами 3 %, 5 %, 10 %, 20 % и 30 % значений.

1. **Восстановление пропусков и оценка результатов**

* Для каждого сочетания размера и процента удаления запустить все три метода;
* Вычислить относительную ошибку восстановления по каждому столбцу и суммарно;
* Сравнить полученные значения ошибок между методами и между уровнями удаления.

1. **Анализ и выводы**

Оценить влияние объёма данных и глубины удаления на качество каждого из методов по разным столбцам.

# Теоретическая часть

В реальных данных часто встречаются пропущенные значения (missing data). Причины могут быть разными: ошибки сбора, передача и хранение данных, удаление «выбросов» и т. д.

Пропуски усложняют анализ и обучение моделей, поэтому перед основной обработкой их обычно восстанавливают (импутируют) с помощью различных методов.

**Методы иммутации в `DatasetFilling.pyw`**

В приложении реализовано три метода восстановления:

Таблица 1. DatasetFilling.pyw

|  |  |  |
| --- | --- | --- |
| Метод | Идея | Применимость |
| Hot–Deck Imputation | Для каждого пропуска случайно копирует значение из другой непустой записи того же столбца | категориальные и числовые данные |
| Mode Imputation | Заполняет каждый пропуск модой (наиболее частым значением) столбца | лучше для категорий |
| Spline Interpolation | Строит для числовых столбцов кубический сплайн и восстанавливает по гладкой кривой | только для количественных признаков |

**Hot-Deck Imputation**

- Алгоритм

1. Для каждого столбца собрать все непустые значения.
2. Для каждой пропущенной ячейки случайно выбрать одно из непустых значений и подставить его.

**Mode Imputation (Мода)**

- Алгоритм

1. Для каждого столбца найти моду (значение, встречающееся чаще всего).
2. Заполнить все пропуски этим значением.

**Spline Interpolation**

- Идея

Использовать кубический сплайн для числовых признаков: между известными значениями строится гладкая кривая, по которой восстанавливаются пропуски.

- Шаги реализации

1. Кодирование категорий и перевод строчных данных в числовой вид для возможности применения сплайн интерполяции.
2. Применение сплайн интерполяции к обработанному датасету.
3. Обратное декодирование датасета и приведение его к стандартному виду.

**Оценка качества восстановления**

1. Предобработка

Для обоих DataFrame (исходного и восстановленного) все поля кодируются в пространства чисел:

2. Вычисление относительной ошибки

Где – исходное значение поля, – восстановленное методом.

– Берутся все непустые элементы;

– Считается средняя ошибка по каждому техническому полю;

– Технические поля группируются в 10 отображаемых столбцов (`FullName`, `PassportInfo`, …, `PaymentCard`), средняя ошибка в группе выводится в процентах.

– Отдельно выводится суммарная ошибка как сумма ошибок по всем техническим полям.

# Основные шаги программы

Основные шаги программы `DatasetFilling.pyw`

1. Запуск приложения
2. Инициализация интерфейса

– Поле ввода «Записей» + кнопка «Сгенерировать» для создания синтетического датасета.

– Поле ввода «Удалить (%)» + кнопка «Удалить» для внесения пропусков.

– Комбо-бокс «Метод» с тремя вариантами (Hot-Deck, мода, сплайн) + кнопка «Восстановить».

– Кнопки «Загрузить CSV» / «Сохранить CSV» и поле `QPlainTextEdit` для копируемой статистики.

– `QTableView` для отображения DataFrame.

1. Генерация датасета (`generateTickets`)

– При вводе числа N и нажатии «Сгенерировать»:

1. Вызывается `RailwayTicketsGeneration.generateTicketsDf(N)`.
2. Результат сохраняется в `originalDataFrame` и `dataFrame`.
3. Обновляется таблица и статистика.
4. Внесение пропусков (`removeRandomData`)

– По заданному проценту P вычисляется число ячеек к обнулению.

– Генерируется список всех непустых ячеек; для каждой ячейки рассчитывается вероятность удаления (комбинация положения строки и столбца).

– Случайным образом по этим весам выбираются ячейки и обнуляются (`np.nan`).

– Обновление таблицы и пересчёт процента пропусков.

1. Выбор метода восстановления

– Пользователь выбирает в выпадающем списке один из трёх методов:

- Hot-Deck

- Mode (мода)

- Сплайн-интерполяция

1. Восстановление пропусков (`restoreData`)

– В зависимости от выбранного метода вызывается соответствующий обработчик:

1. Hot-Deck (`hotDeckImputation`): замена каждого пропуска случайным непустым значением из того же столбца.
2. Mode Imputation (`modeImputation`): заполнение модой столбца.
3. Spline Interpolation (`splineInterpolation`):

- Кодирование всех полей в числовые признаки (factorize, UNIX- тайм, паспорт → одно число).

- Интерполяция кубическим сплайном по каждому числовому столбцу.

- Обратное декодирование в исходные строки и форматы.

– После восстановления обновляется таблица.

1. Расчёт и вывод ошибки (`calculateRelativeError`)

– Оба DataFrame (оригинал и восстановленный) проходят препроцессинг в числовые векторы.

– Для каждой «технической» колонки вычисляется средняя относительная ошибка.

– Технические колонки группируются по 10 отображаемым полям (`FullName`, `PassportInfo`, …, `PaymentCard`), внутри каждого поля ошибки усредняются.

– Формируется текст со строками

– Выводится в `QPlainTextEdit` под статистикой.

1. Загрузка и сохранение CSV

– «Загрузить CSV» вызывает стандартный диалог, читает файл в `dataFrame` и обновляет таблицу.

– «Сохранить CSV» сохраняет текущее состояние `dataFrame` в указанный файл.

# Блок схема программы

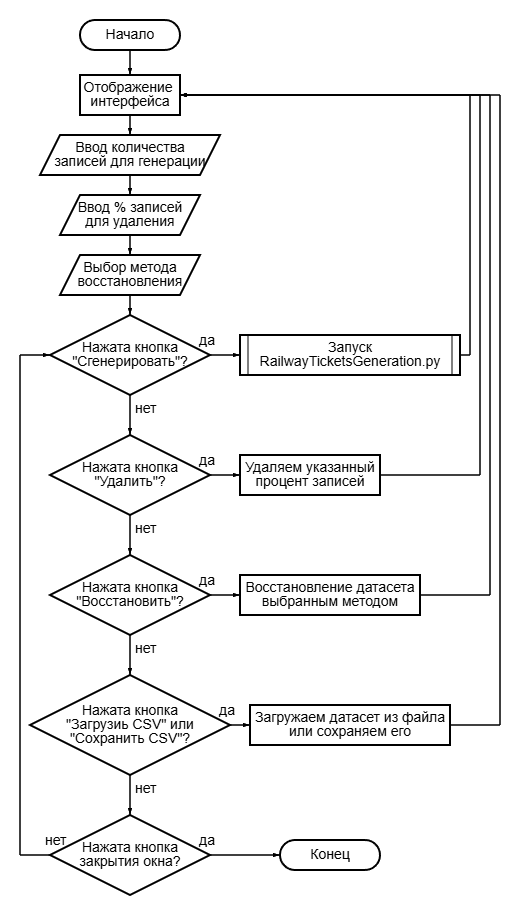


Рис 1. Блок-схема DatasetFilling.pyw

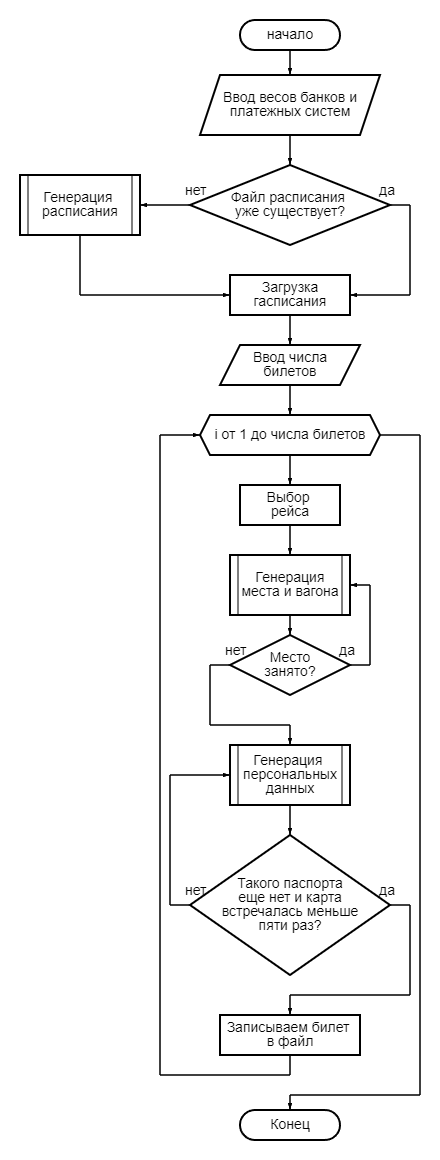


Рис 2. Блок-схема RailwayTicketsGeneration.py

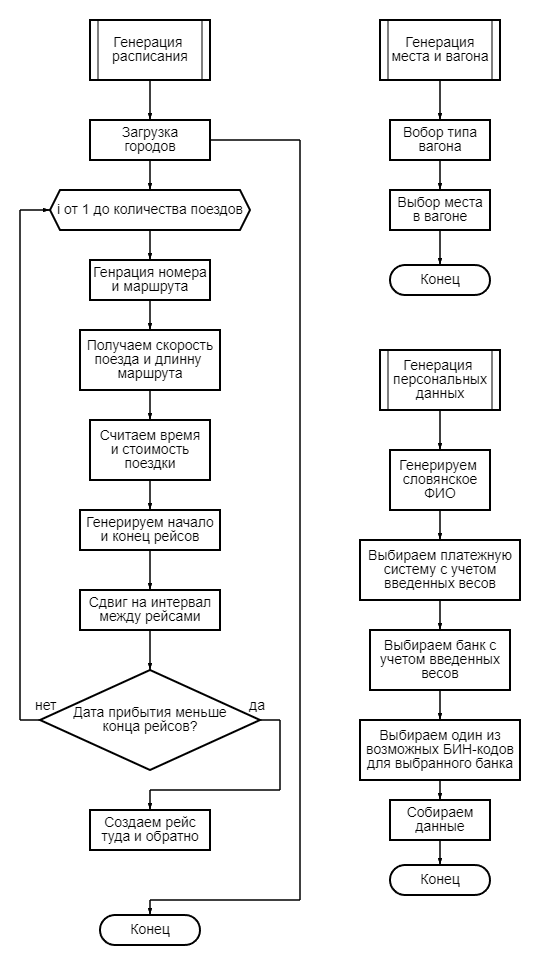


Рис 3. Блок-схема подпрограмм RailwayTicketsGeneration.py

# Описание программы

Программа написана на языке Python 3.12.6 и использует библиотеки pandas [[1]](#_Источники_1) и numpy [[2]](#_Источники_1) для работы с табличными данными, а также PyQt5 [[3]](#_Источники_1) для построения графического интерфейса. В качестве генератора исходного набора билетов применяется модуль RailwayTicketsGeneration. В программе реализован один класс — `DatasetFillingApp`, отвечающий за, отрисовку и настройку GUI (панель управления и таблица данных), генерацию синтетического датасета билетов, случайное удаление значений по выбранному проценту и весам, восстановление пропусков тремя методами (Hot-Deck, мода, сплайн-интерполяция), подсчёт и вывод статистики по отсутствующим значениям и относительной ошибке, загрузку/сохранение данных в CSV.

Таблица 2. DatasetFillingApp

|  |  |  |
| --- | --- | --- |
| **Метод** | **Описание** | **Возвращаемое значение** |
| \_\_init\_\_ | Инициализация окна, атрибутов для хранения данных и вызов initUI() | None |
| initUI | Построение интерфейса: поля ввода, кнопки, комбобокс, таблица, текстовое поле для статистики | None |
| generateTickets | Генерация N записей через RailwayTicketsGeneration.generateTicketsDf, копирование в originalDataFrame и обновление таблицы | None |
| removeRandomData | Удаление указанного % значений: выбор ячеек по весам (row/col), установка NaN, обновление таблицы и статистики | None |
| updateStatistics | Подсчёт общего числа ячеек, пропусков и обновление текста в statsLabel | None |
| restoreData | Запуск выбранного метода восстановления (Hot-Deck, мода или сплайн), обновление таблицы и добавление текста ошибок в статистику | None |
| hotDeckImputation | Замена каждого пропуска на случайное значение из непустых элементов того же столбца | None |
| modeImputation | Заполнение пропусков наиболее частым (модальным) значением столбца | None |
| calculateRelativeError | Вычисление относительной ошибки восстановления: кодирование данных, сравнение с оригиналом, группировка по отображаемым полям | str (текст с ошибками) или None |
| splineInterpolation | Интерполяция кубическим сплайном по всем числовым столбцам: кодирование, interpolate(method='spline'), обратное декодирование | None |
| updateTable | Построение модели QStandardItemModel для QTableView, настройка ширин колонок и обновление статистики | None |
| loadCSV | Загрузка CSV через диалог, чтение в dataFrame, обновление таблицы | None |
| saveCSV | Сохранение текущего dataFrame в CSV-файл через диалог | None |

# Рекомендации пользователя

Программа «Восстановление датасета» позволяет сгенерировать синтетический набор «билетов», искусственно удалить в нём произвольный процент значений и затем восстановить пропуски одним из трёх методов (Hot-Deck, мода, сплайн-интерполяция).

1. Запуск

python DatasetFilling.pyw

1. Генерация данных

- В поле «Записей» укажите желаемое число строк.

- Нажмите «Сгенерировать» — в таблице появятся данные, скопированные в «оригинал».

1. Удаление части значений

- В поле «Удалить (%)» введите процент ячеек, которые нужно обнулить (3, 5, 10, 20 или 30).

- Нажмите «Удалить» — программа случайно вносит `NaN` по заданной доле, учитывая позицию строки и столбца.

- Внизу в поле статистики обновятся: общее число записей и факт % пропусков.

1. Восстановление пропусков

- В выпадающем списке «Метод» выберите один из трёх алгоритмов:

- Хот-Дек — случайные значения из столбца;

- Метод заполнения моды — наиболее частое значение;

- Сплайн-интерполяция — кубический сплайн по числовым признакам.

- Нажмите «Восстановить».

- После этого в статистике появится:

Записей: <N>

Пропусков: <X.XX>%

FullName: <X.XX>%

PassportInfo: <X.XX>%

…

Суммарная ошибка: <X.XX>%

5. Работа с CSV

- «Загрузить CSV» — выбрать файл, подгрузить его в таблицу.

- «Сохранить CSV» — экспорт текущих данных (восстановленных или с пропусками) в файл.

# Рекомендации программиста

1. Требования

- Python ≥ 3.12.0

- Установленные пакеты: `PyQt5`, `pandas`, `numpy`

- Модуль `RailwayTicketsGeneration` доступен в `PYTHONPATH`.

2. Установка зависимостей

pip install PyQt5 pandas numpy

3. Запуск

python DatasetFilling.pyw

4. Особенности

- GUI построен на PyQt5: изменения в данных мгновенно отражаются в `QTableView`.

- Статистика и результаты ошибок выводятся в `QPlainTextEdit` (read- only), что позволяет копировать текст.

- Метод сплайн-интерполяции требует, чтобы все поля могли быть закодированы в числовую форму: убедитесь, что в исходном CSV нет неожиданных форматов дат или строк.

- Ошибки ввода (нечисловой процент удаления, пустые данные) обрабатываются через `QMessageBox.warning`.

5. Тестирование

- Попробуйте разные размеры данных (1 000—250 000 строк) и уровни удаления (3–30 %) для проверки производительности и адекватности методов.

- Для больших наборов учтите, что сплайн-интерполяция может занимать значительное время и потреблять много памяти.

# Исходный код программы

<https://github.com/FasterXaos/Algorithms_and_Data_Structures>

# Контрольный пример

1. Запуск программы и формирование графа

Для запуска программы откройте файл, содержащий код класса DatasetFillingApp. Программа откроет графический интерфейс, где в левой части окна расположены поля для ввода параметров работы и кнопки для генерации датесета, создания пропусков, восстановления, сохранения и загрузки датасета, в правой – область для его отображения (Рис. 4).



Рис 4. Пример окна программы

1. Генерация данных

После запуска программы можно сгенерировать синтетический датасет. Для этого нужно:

1. В поле «Записей:» ввести количество записей для генерации.
2. Нажать кнопку «Сгенерировать».

После чего, сгенерированный датасет отобразится в таблице, в правой части интерфейса (Рис. 5).

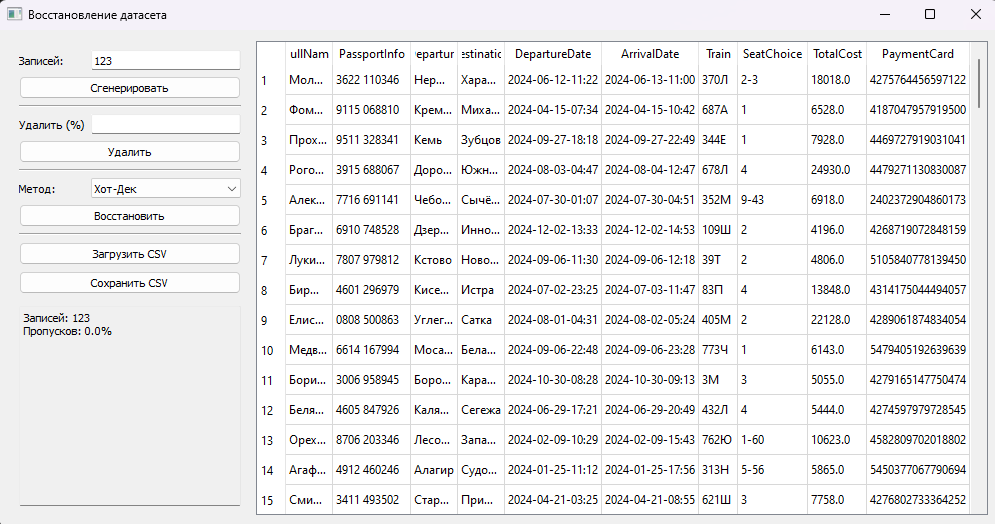


Рис 5. Пример сгенерированного датасета

1. Удаление

Далее в датасете можно удалить определенное количество полей. Для этого:

1. В поле «Удалить (%)» ввести процент полей для удаления.
2. Нажать кнопку «Сгенерировать».

После чего удаленный записи отобразятся в таблице как `nan` (Рис. 6).

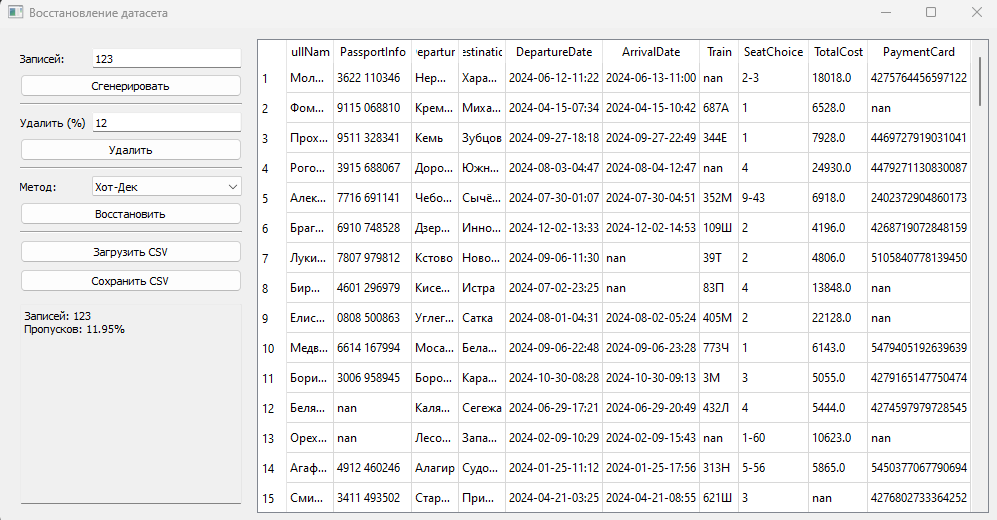


Рис 6. Пример датасета после удаления полей

1. Восстановление датасета и вывод результатов

Датасет с пропусками можно восстановить. Для этого нужно:

1. Выбрать в выпадающем меню «Метод:» один из трех методов восстановления: Хот-Дек, Метод заполнения моды, Сплайн-интерполяция.
2. Нажать кнопку «Восстановить».

После этого в таблице пропущенные значения заполнятся в соответствии с выбранным методом, а под количеством записей и пропусков выведется суммарная ошибка относительно исходного датасета по каждому отдельному столбцу и в сумме по всем (Рис. 7).

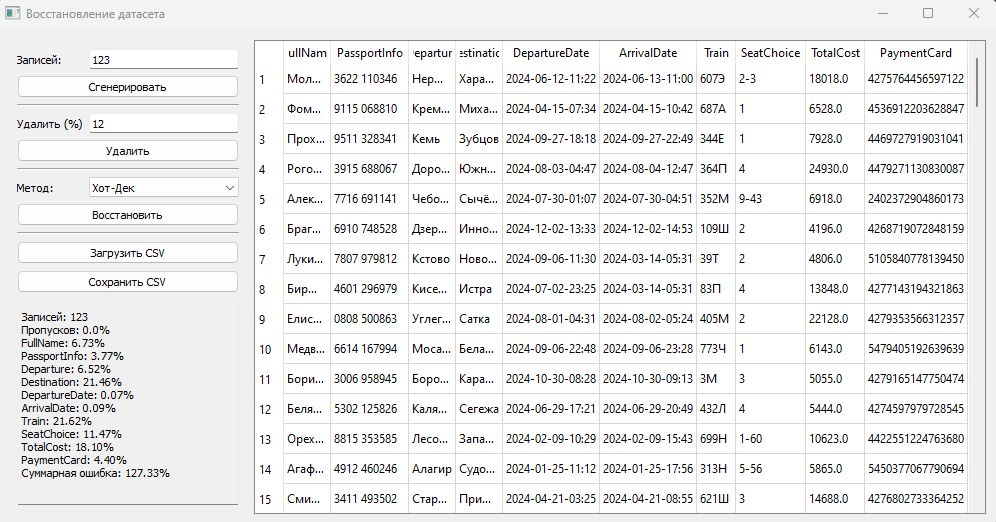


Рис 7. Пример датасета после восстановления

1. Загрузка и сохранение датасета.

В любой момент можно сохранить отображаемый датасет или загрузить новый из файла CSV нажав на кнопки «Загрузить CSV» и «Сохранить CSV». Файл должен иметь следующую структуру для корректной загрузки:

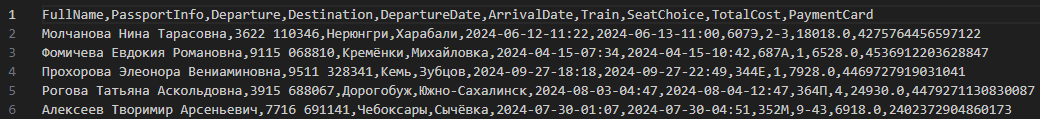


Рис 8. Пример датасета для загрузки

# Анализ результатов работы алгоритма

Для анализа методов восстановления датасета проведены тесты на разных размерах датасета и процентах пропусков, для которых была посчитана суммарная ошибка по каждому столбцу по отдельности и в сумме:

Таблица 3. Тесты восстановления датасетов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер датасета | Процент пропусков | Метод восстановления | Ошибка по столбцам | Суммарная ошибка |
| 5000 | 3 | Hot-Deck | FullName: 1.52%  PassportInfo: 3.37%  Departure: 3.82%  Destination: 14.64%  DepartureDate: 0.02%  ArrivalDate: 0.02%  Train: 18.02%  SeatChoice: 2.78%  TotalCost: 2.40%  PaymentCard: 1.31% | 68.70% |
| 5000 | 3 | Mode | FullName: 2.64%  PassportInfo: 1.97%  Departure: 3.35%  Destination: 8.29%  DepartureDate: 0.02%  ArrivalDate: 0.02%  Train: 5.51%  SeatChoice: 0.98%  TotalCost: 1.68%  PaymentCard: 1.98% | 32.93% |
| 5000 | 3 | Spline | FullName: 1.29%  PassportInfo: 6.12%  Departure: 10.93%  Destination: 5.76%  DepartureDate: 11.34%  ArrivalDate: 13.87%  Train: 12.47%  SeatChoice: 8.49%  TotalCost: 3.12%  PaymentCard: 1.56% | 95.90% |
| 5000 | 5 | Hot-Deck | FullName: 2.71%  PassportInfo: 6.13%  Departure: 13.46%  Destination: 15.88%  DepartureDate: 0.03%  ArrivalDate: 0.03%  Train: 15.60%  SeatChoice: 6.13%  TotalCost: 4.83%  PaymentCard: 2.01% | 88.55% |
| 5000 | 5 | Mode | FullName: 3.53%  PassportInfo: 3.62%  Departure: 4.99%  Destination: 38.60%  DepartureDate: 0.04%  ArrivalDate: 0.03%  Train: 4.51%  SeatChoice: 1.62%  TotalCost: 3.25%  PaymentCard: 3.29% | 69.61% |
| 5000 | 5 | Spline | FullName: 1.86%  PassportInfo: 7.07%  Departure: 19.66%  Destination: 25.17%  DepartureDate: 22.27%  ArrivalDate: 21.64%  Train: 13.40%  SeatChoice: 7.22%  TotalCost: 4.90%  PaymentCard: 2.19% | 145.99% |
| 5000 | 10 | Hot-Deck | FullName: 4.71%  PassportInfo: 12.70%  Departure: 18.87%  Destination: 34.47%  DepartureDate: 0.06%  ArrivalDate: 0.06%  Train: 33.64%  SeatChoice: 14.08%  TotalCost: 9.24%  PaymentCard: 3.98% | 179.53% |
| 5000 | 10 | Mode | FullName: 9.40%  PassportInfo: 7.07%  Departure: 13.49%  Destination: 13.23%  DepartureDate: 0.06%  ArrivalDate: 0.06%  Train: 15.36%  SeatChoice: 3.57%  TotalCost: 6.35%  PaymentCard: 6.69% | 94.21% |
| 5000 | 10 | Spline | FullName: 3.65%  PassportInfo: 13.44%  Departure: 35.92%  Destination: 34.22%  DepartureDate: 38.90%  ArrivalDate: 47.48%  Train: 45.04%  SeatChoice: 15.92%  TotalCost: 10.04%  PaymentCard: 4.64% | 310.22% |
| 5000 | 20 | Hot-Deck | FullName: 9.55%  PassportInfo: 22.65%  Departure: 124.04%  Destination: 64.95%  DepartureDate: 0.12%  ArrivalDate: 0.14%  Train: 56.17%  SeatChoice: 27.71%  TotalCost: 17.80%  PaymentCard: 7.41% |  |
| 5000 | 20 | Mode | FullName: 17.32%  PassportInfo: 13.14%  Departure: 28.48%  Destination: 43.05%  DepartureDate: 0.11%  ArrivalDate: 0.12%  Train: 28.74%  SeatChoice: 7.14%  TotalCost: 9.41%  PaymentCard: 12.70% | 196.10% |
| 5000 | 20 | Spline | FullName: 7.50%  PassportInfo: 29.63%  Departure: 68.25%  Destination: 68.96%  DepartureDate: 83.06%  ArrivalDate: 93.34%  Train: 89.27%  SeatChoice: 39.21%  TotalCost: 19.97%  PaymentCard: 8.73% | 636.39% |
| 5000 | 30 | Hot-Deck | FullName: 14.37%  PassportInfo: 38.37%  Departure: 97.19%  Destination: 82.73%  DepartureDate: 0.18%  ArrivalDate: 0.20%  Train: 114.36%  SeatChoice: 41.80%  TotalCost: 25.05%  PaymentCard: 11.54% | 581.94% |
| 5000 | 30 | Mode | FullName: 28.20%  PassportInfo: 19.75%  Departure: 41.83%  Destination: 54.54%  DepartureDate: 0.26%  ArrivalDate: 0.17%  Train: 148.48%  SeatChoice: 10.58%  TotalCost: 13.90%  PaymentCard: 18.30% | 495.05% |
| 5000 | 30 | Spline | FullName: 12.03%  PassportInfo: 49.96%  Departure: 100.49%  Destination: 139.60%  DepartureDate: 127.55%  ArrivalDate: 139.44%  Train: 123.59%  SeatChoice: 55.95%  TotalCost: 32.27%  PaymentCard: 14.63% | 975.03% |
| 50000 | 3 | Hot-Deck | FullName: 1.47%  PassportInfo: 4.36%  Departure: 7.64%  Destination: 8.45%  DepartureDate: 0.02%  ArrivalDate: 0.02%  Train: 11.19%  SeatChoice: 3.96%  TotalCost: 2.65%  PaymentCard: 1.19% | 56.09% |
| 50000 | 3 | Mode | FullName: 2.85%  PassportInfo: 1.94%  Departure: 4.48%  Destination: 14.88%  DepartureDate: 0.02%  ArrivalDate: 0.02%  Train: 11.51%  SeatChoice: 1.19%  TotalCost: 1.52%  PaymentCard: 2.05% | 53.17% |
| 50000 | 3 | Spline | FullName: 1.01%  PassportInfo: 4.43%  Departure: 10.60%  Destination: 6.27%  DepartureDate: 12.39%  ArrivalDate: 13.92%  Train: 10.78%  SeatChoice: 6.01%  TotalCost: 2.94%  PaymentCard: 1.30% | 86.44% |
| 50000 | 5 | Hot-Deck | FullName: 2.35%  PassportInfo: 7.03%  Departure: 15.72%  Destination: 16.59%  DepartureDate: 0.03%  ArrivalDate: 0.03%  Train: 17.01%  SeatChoice: 6.69%  TotalCost: 4.46%  PaymentCard: 2.10% | 95.69% |
| 50000 | 5 | Mode | FullName: 4.32%  PassportInfo: 3.25%  Departure: 6.90%  Destination: 31.04%  DepartureDate: 0.04%  ArrivalDate: 0.04%  Train: 22.96%  SeatChoice: 2.10%  TotalCost: 2.57%  PaymentCard: 3.21% | 101.48% |
| 50000 | 5 | Spline | FullName: 1.68%  PassportInfo: 7.16%  Departure: 23.49%  Destination: 18.38%  DepartureDate: 20.71%  ArrivalDate: 23.46%  Train: 17.24%  SeatChoice: 10.10%  TotalCost: 5.02%  PaymentCard: 2.19% | 156.77% |
| 50000 | 10 | Hot-Deck | FullName: 4.82%  PassportInfo: 14.29%  Departure: 25.78%  Destination: 31.21%  DepartureDate: 0.06%  ArrivalDate: 0.07%  Train: 37.38%  SeatChoice: 13.77%  TotalCost: 8.77%  PaymentCard: 3.87% | 191.16% |
| 50000 | 10 | Mode | FullName: 9.15%  PassportInfo: 6.51%  Departure: 20.41%  Destination: 17.54%  DepartureDate: 0.09%  ArrivalDate: 0.06%  Train: 31.62%  SeatChoice: 4.05%  TotalCost: 5.05%  PaymentCard: 6.50% | 136.64% |
| 50000 | 10 | Spline | FullName: 3.52%  PassportInfo: 14.68%  Departure: 30.98%  Destination: 35.59%  DepartureDate: 41.89%  ArrivalDate: 45.55%  Train: 39.53%  SeatChoice: 20.19%  TotalCost: 10.05%  PaymentCard: 4.51% | 306.21% |
| 50000 | 20 | Hot-Deck | FullName: 9.41%  PassportInfo: 28.48%  Departure: 65.64%  Destination: 108.62%  DepartureDate: 0.12%  ArrivalDate: 0.13%  Train: 72.39%  SeatChoice: 27.42%  TotalCost: 17.35%  PaymentCard: 7.67% | 437.04% |
| 50000 | 20 | Mode | FullName: 18.09%  PassportInfo: 13.20%  Departure: 22.97%  Destination: 40.91%  DepartureDate: 0.14%  ArrivalDate: 0.13%  Train: 60.80%  SeatChoice: 7.98%  TotalCost: 9.95%  PaymentCard: 12.76% | 255.70% |
| 50000 | 20 | Spline | FullName: 7.18%  PassportInfo: 30.84%  Departure: 73.83%  Destination: 73.92%  DepartureDate: 84.12%  ArrivalDate: 90.85%  Train: 83.50%  SeatChoice: 38.03%  TotalCost: 20.63%  PaymentCard: 8.92% | 633.36% |
| 50000 | 30 | Hot-Deck | FullName: 14.59%  PassportInfo: 42.87%  Departure: 150.99%  Destination: 103.82%  DepartureDate: 0.18%  ArrivalDate: 0.19%  Train: 104.03%  SeatChoice: 42.83%  TotalCost: 25.80%  PaymentCard: 11.22% | 643.37% |
| 50000 | 30 | Mode | FullName: 27.07%  PassportInfo: 20.08%  Departure: 53.08%  Destination: 57.06%  DepartureDate: 0.14%  ArrivalDate: 0.21%  Train: 79.87%  SeatChoice: 12.10%  TotalCost: 14.89%  PaymentCard: 18.94% | 375.41% |
| 50000 | 30 | Spline | FullName: 11.34%  PassportInfo: 44.99%  Departure: 105.74%  Destination: 99.76%  DepartureDate: 126.93%  ArrivalDate: 136.45%  Train: 109.97%  SeatChoice: 59.12%  TotalCost: 31.35%  PaymentCard: 14.06% | 908.80% |
| 150000 | 3 | Hot-Deck | FullName: 1.43%  PassportInfo: 3.99%  Departure: 13.70%  Destination: 9.40%  DepartureDate: 0.02%  ArrivalDate: 0.02%  Train: 9.10%  SeatChoice: 4.05%  TotalCost: 2.74%  PaymentCard: 1.19% | 58.78% |
| 150000 | 3 | Mode | FullName: 2.69%  PassportInfo: 1.92%  Departure: 5.59%  Destination: 3.51%  DepartureDate: 0.02%  ArrivalDate: 0.01%  Train: 5.74%  SeatChoice: 1.02%  TotalCost: 1.51%  PaymentCard: 1.98% | 30.76% |
| 150000 | 3 | Spline | FullName: 1.06%  PassportInfo: 4.05%  Departure: 11.31%  Destination: 10.83%  DepartureDate: 12.49%  ArrivalDate: 13.70%  Train: 15.50%  SeatChoice: 5.64%  TotalCost: 2.98%  PaymentCard: 1.35% | 100.05% |
| 150000 | 5 | Hot-Deck | FullName: 2.38%  PassportInfo: 6.75%  Departure: 13.37%  Destination: 42.45%  DepartureDate: 0.03%  ArrivalDate: 0.03%  Train: 18.26%  SeatChoice: 6.75%  TotalCost: 4.45%  PaymentCard: 2.03% | 121.52% |
| 150000 | 5 | Mode | FullName: 4.54%  PassportInfo: 3.23%  Departure: 6.18%  Destination: 68.65%  DepartureDate: 0.02%  ArrivalDate: 0.03%  Train: 8.80%  SeatChoice: 1.64%  TotalCost: 2.51%  PaymentCard: 3.27% | 109.30% |
| 150000 | 5 | Spline | FullName: 1.71%  PassportInfo: 7.12%  Departure: 16.30%  Destination: 20.78%  DepartureDate: 21.15%  ArrivalDate: 22.88%  Train: 20.67%  SeatChoice: 9.57%  TotalCost: 5.22%  PaymentCard: 2.22% | 157.87% |
| 150000 | 10 | Hot-Deck | FullName: 4.70%  PassportInfo: 13.31%  Departure: 30.63%  Destination: 46.36%  DepartureDate: 0.06%  ArrivalDate: 0.07%  Train: 36.42%  SeatChoice: 13.55%  TotalCost: 8.90%  PaymentCard: 3.90% | 207.88% |
| 150000 | 10 | Mode | FullName: 8.92%  PassportInfo: 6.47%  Departure: 9.10%  Destination: 16.94%  DepartureDate: 0.05%  ArrivalDate: 0.05%  Train: 14.60%  SeatChoice: 3.29%  TotalCost: 5.07%  PaymentCard: 6.50% | 88.88% |
| 150000 | 10 | Spline | FullName: 3.51%  PassportInfo: 13.97%  Departure: 30.88%  Destination: 42.46%  DepartureDate: 41.62%  ArrivalDate: 45.71%  Train: 48.25%  SeatChoice: 19.23%  TotalCost: 10.38%  PaymentCard: 4.49% | 327.98% |
| 150000 | 20 | Hot-Deck | FullName: 9.72%  PassportInfo: 27.10%  Departure: 64.86%  Destination: 65.44%  DepartureDate: 0.12%  ArrivalDate: 0.13%  Train: 82.34%  SeatChoice: 28.31%  TotalCost: 17.21%  PaymentCard: 7.66% | 413.55% |
| 150000 | 20 | Mode | FullName: 18.04%  PassportInfo: 13.21%  Departure: 29.34%  Destination: 37.87%  DepartureDate: 0.15%  ArrivalDate: 0.10%  Train: 24.75%  SeatChoice: 6.50%  TotalCost: 10.02%  PaymentCard: 12.74% | 183.96% |
| 150000 | 20 | Spline | FullName: 7.24%  PassportInfo: 29.72%  Departure: 61.21%  Destination: 64.91%  DepartureDate: 83.48%  ArrivalDate: 91.70%  Train: 86.21%  SeatChoice: 38.98%  TotalCost: 20.81%  PaymentCard: 8.99% | 618.47% |
| 150000 | 30 | Hot-Deck | FullName: 14.97%  PassportInfo: 42.05%  Departure: 81.62%  Destination: 117.42%  DepartureDate: 0.18%  ArrivalDate: 0.20%  Train: 113.64%  SeatChoice: 42.73%  TotalCost: 25.90%  PaymentCard: 11.33% | 606.40% |
| 150000 | 30 | Mode | FullName: 27.08%  PassportInfo: 20.19%  Departure: 35.33%  Destination: 59.63%  DepartureDate: 0.22%  ArrivalDate: 0.15%  Train: 52.23%  SeatChoice: 12.01%  TotalCost: 14.83%  PaymentCard: 18.81% | 304.72% |
| 150000 | 30 | Spline | FullName: 11.16%  PassportInfo: 46.85%  Departure: 96.51%  Destination: 115.98%  DepartureDate: 127.48%  ArrivalDate: 138.02%  Train: 130.75%  SeatChoice: 59.60%  TotalCost: 31.97%  PaymentCard: 13.96% | 962.63% |

Таблица 4. Средняя ошибка по методам

|  |  |  |  |
| --- | --- | --- | --- |
| Размер датасета | Hot-Deck | Mode | Spline |
| 5000 | 229.68 | 177.58 | 432.71 |
| 50000 | 284.67 | 184.48 | 295.70 |
| 150000 | 281.63 | 143.52 | 433.40 |

Как видно из тестов, Mode дает наименьшую среднюю ошибку из всех методов, Hot-Deck показывает более высокие значения, а Spline дает куда более высокую ошибку, чем остальные методы. Зависимость от размера выборки здесь практически отсутствует: ключевую роль играет не столько объём данных, сколько характер заполнения пропусков.

Таблица 5. Средняя ошибка по проценту пропусков

|  |  |  |  |
| --- | --- | --- | --- |
| Процент пропусков | Hot-Deck | Mode | Spline |
| 3 | 61.19 | 38.95 | 94.13 |
| 5 | 101.92 | 93.46 | 153.54 |
| 10 | 192.86 | 106.58 | 314.80 |
| 20 | 425.30 | 211.92 | 629.41 |
| 30 | 610.57 | 391.73 | 948.82 |

При рассмотрении средней ошибки по проценту выбросов видно, что соотношения ошибки по методам сохраняются так же, как и в прошлом рассмотрении, но при этом ошибка очевидно зависит линейно от процента выбросов, что закономерно следует из формулы ее расчета.

Таблица 6. Поведение ошибки на разных полях датасета

|  |  |  |  |
| --- | --- | --- | --- |
| Столбец | Hot-Deck | Mode | Spline |
| FullName | (~1–15 %), растёт с пропусками. | (~2–27 %), сильнее ухудшается с ростом процента выбрасов. | (~2–15 %), также растет с пропусками. |
| PassportInfo | (~3→43 %), при росте пропусков, увеличивается сильнее на больших датасетах | (2→20 %), растет с числом пропусков не зависимо от количества записей. | (30–50 %), растет с числом пропусков не зависимо от количества записей. |
| Departure | При малых пропусках (3–19 %), но доходит до 100–150 % при 20–30 %. Ведет себя крайне нестабильно. | (3–68 %), умереннее, но еще более нестабильно с частыми выбросами независимо от пропусков. | (~5–150 %), быстрый, но стабильный рост в зависимости от числа пропусков. |
| Destination |
| DepartureDate | (0.02–0.20 %), растет с количеством пропусков. | Как у Hot-Deck, но чуть больше (0.02–0.26 %). | (10–140 %), куда большая ошибка, растущая с пропусками |
| ArrivalDate |
| Train | (9 –113 %), сильно растет с пропусками. | (5–79 %), меньшая ошибка с ростом пропусков, но возможными выбросами, как 148% | (10–130 %), сильно растет с пропусками. |
| SeatChoice | (2.8–42.8 %), умеренный рост с ростом пропусков. | (0.98–12.1 %), более медленный с ростом пропусков. | |  | | --- | |  |   (6–59.6 %), наибольший рост с ростом пропусков. |
| TotalCost | От 2.4 % до 32.3 %; линейный рост в зависимости от пропусков. | Лучшее значение. От 1.5 % до 14.9 % с ростом пропусков. | Растет от 2.9 % до 32.3 % с ростом пропусков. |
| PaymentCard | (~2–11 %), растет от количества пропусков. | (~2–19 %), растет от количества пропусков. | (~2–14 %), растет от количества пропусков. |

**FullName**

Hot-Deck подставляет случайное имя из оставшихся непустых, поэтому при небольших пропусках ошибка растёт умеренно (≈1–15 %), но с увеличением доли NaN накапливаются случайные замены. Mode всегда возвращает одно и то же наиболее частое имя, что сглаживает разброс, но при >20 % пропусков приводит к сильному смещению распределения (ошибка до ≈27 %). Spline фактически интерполирует «код» имени как число, затем округляет и переводит обратно в метки; при малых пробелах это даёт точный результат, но с ростом пропусков порядок кодов и округления порождают неточности до ≈15 %.

**PassportInfo**

Hot-Deck берёт случайные серии и номера из непустых, что при крупных пропусках (20–30 %) сильно разбалансирует сочетания («серия» + «номер») и приведёт к росту ошибки до ≈43 %. Mode стабильно подставляет самую частую комбинацию, поэтому ошибка плавно увеличивается от ≈2 % до ≈20 % без больших выбросов. Spline интерполирует числовой код паспорта, но при больших пробелах экстраполяция создаёт несуществующие или усреднённые коды, что повышает ошибку до 30–50 %.

**Departure**

Hot-Deck на малых пропусках сохраняет распределение кодов станций (≈3–19 %), но при 20–30 % NaN все больше нарушает логику маршрутов и даёт резкий всплеск ошибки до 100–150 %. Mode возвращает самую частую станцию, что ограничивает разброс и удерживает ошибку в диапазоне 3–68 %, хотя и без учёта реальной последовательности. Spline интерполирует порядковые коды станций, а последующее округление может генерировать «несуществующие» индексы, что даёт стабильный, но быстрый рост ошибки (до ≈140 %).

**Destination**

Поведение аналогично Departure: Hot-Deck случайным образом берёт станции, что при больших пропусках нарушает реальную географию и даёт до ≈83 % ошибки; Mode возвращает наиболее частую станцию, ограничивая разброс (8–57 %); Spline из-за округления интерполированных кодов создаёт новые, «неправильные» станции, и ошибка растёт от ≈5 % до ≈140 %.

**DepartureDate**

Hot-Deck случайно выбирает существующую дату-время, поэтому ошибка остаётся микроскопической (0.02–0.20 %). Mode подставляет одно и то же значение, что при малых пропусках даёт близкие к Hot-Deck погрешности (до 0.26 %), но без рандома. Spline интерполирует timestamp как число, и при длинных пробелах экстраполяция «затягивает» интервалы (смена дней, месяцев), отчего ошибка нарастает вплоть до >100 %.

**ArrivalDate**

Аналогично DepartureDate: Hot-Deck и Mode дают минимальную ошибку (≈0.02–0.26 %) за счёт выбора из существующих временных меток или использования одной «моды», а Spline из-за числовой интерполяции timestamps создаёт большие погрешности (до ≈139 %).

**Train**

Hot-Deck случайно вставляет существующие комбинации номера и буквы, что при росте NaN (20–30 %) приводит к росту ошибки до ≈148 %. Mode возвращает самый частый состав даже при большом количестве пропусков, ограничивая ошибку 5–79 %. Spline разносит числовые коды номера и буквы по «гладкой» кривой, но округления и экстраполяция порождают некорректные составы, что приводит к ошибке до ≈130 %.

**SeatChoice**

Hot-Deck берёт случайный «вагон-место» из оставшихся, поэтому ошибка растёт умеренно (2.8–42.8 %), поскольку пространство значений невелико. Mode всегда подставляет одно и то же место (моду), за счёт чего ошибка минимальна (0.98–12.1 %). Spline интерполирует числовые пары (вагон/сиденье), и при пропусках формирует «несуществующие» комбинации, отчего ошибка может достигать 60 %.

**TotalCost**

Hot-Deck рандомно назначает стоимость из имеющихся, что даёт линейный рост ошибки с 2.4 % до 32.3 %. Mode постоянно возвращает одно значение цены, благодаря чему ошибка остаётся наименьшей в диапазоне 1.5–14.9 %. Spline интерполирует числовые значения стоимости, что при больших пропусках создаёт средние или экстраполированные цены — ошибка растёт от 2.9 % до 32.3 %.

**PaymentCard**

Hot-Deck случайным образом берёт код карты из непустых, что поддерживает ошибку в пределах ≈2–11 % из-за ограниченного числа карт и их номеров. Mode заполняет все NaN наиболее частым кодом, что при разнообразии карт даёт ошибку до ≈19 %. Spline интерполирует числовые коды карт «гладко», но округления и экстраполяция приводят к неточностям до ≈14 %.

**Сравнение методов восстановления**

Hot-Deck — самый простой и лёгкий в реализации метод: пропуски заполняются случайным образом из уже имеющихся значений столбца, без какой-либо предварительной обработки и значительных вычислительных затрат. Из-за элементарности подхода и рандомизации он демонстрирует умеренные, но заметные ошибки — в среднем выше, чем у методов, опирающихся на статистику данных.

Mode показывает наилучшие результаты по суммарной ошибке благодаря подстановке наиболее частого значения каждого признака. Это минимизирует вероятность «экзотических» замен и стабилизирует восстановление даже при высоком уровне пропусков. Как и Hot-Deck, метод заполнения модой не требует перекодировки данных и выполняется очень быстро, однако более стабильный за счёт использования доминирующей категории.

Spline-интерполяция даёт наихудшие результаты по нескольким причинам. Во-первых, этот метод требует, чтобы все признаки были представлены в виде упорядоченных числовых рядов, из-за чего приходится искусственно кодировать категориальные и строковые поля (имена, станции, вагоны и т. д.), что уже на этапе подготовки вносит погрешности. Во-вторых, интерполяция проводится строго «по столбцам», игнорируя связь между различными признаками в одной записи; в нашей задаче вся информация для адекватного восстановления пропусков содержится в контексте строки, а не в статистике соседних точек ряда. В-третьих, при больших «разрывах» spline-экстраполяция легко генерирует нереалистичные значения (например, отрицательные номера вагонов или сидений), чего никогда не происходит при случайном выборе (Hot-Deck) или подстановке моды.

**Почему ошибки получились такими большими**

В нашем датасете каждая строка описывает отдельный билет, причём вся логика заполнения пропусков базируется именно на комбинации полей в рамках одной записи (станции, даты, класс поезда, стоимость и т. п.). Информация о сезонности поезда, времени в пути, тарифах, распределении мест и особенностях платёжных карт позволяет сделать более точные предсказания, опираясь на всю строку целиком. Тем не менее все три рассмотренных метода действуют исключительно «колонками», не учитывая взаимозависимость признаков в одной записи. В результате даже при большом объёме данных и при локальном сходстве соседних точек они не могут восстановить взаимосвязи, содержащиеся в столбцах, что и обуславливает высокие относительные ошибки.

# Вывод

В ходе работы были исследованы и сравнены три метода восстановления пропущенных значений в табличных данных — Hot-Deck, метод заполнения модой и сплайн-интерполяция — на синтетических датасетах разного размера и с различной долей «выбросного» удаления (3 %, 5 %, 10 %, 20 %, 30 %). Для этого была разработана GUI-утилита, которая позволила автоматически генерировать данные о билетах, удалять случайные значения по заданному проценту, восстанавливать пропуски тремя способами и оценивать относительную ошибку по каждому из десяти основных полей и в целом по всему датафрейму.

**Сравнительная эффективность методов**

- Mode продемонстрировал наименьшую суммарную ошибку во всех экспериментах: он стабильно заполняет пропуски наиболее вероятными значениями, не искажая свойства распределения сильными выбросами.

- Hot-Deck — простой и мало вычислительно затратный метод — дает среднюю по погрешность, обусловленную рандомизацией: при низких уровнях пропусков он близок к Mode, но с ростом доли NaN теряет согласованность и непротиворечивость элементов.

- Spline показал хорошие результаты лишь на определенных данных, однако из-за необходимости кодировать категориальные признаки и экстраполировать большие пробелы он генерирует нереалистичные значения и терпит существенные потери точности при 20–30 % удалений.

**Влияние объёма датасета**

Увеличение числа строк (от 5 000 до 150 000) практически не изменяло эффективность всех методов, их средняя ошибка не имела прямой зависимости от объема записей.

**Зависимость от уровня удаления**

Относительная ошибка любого метода росла линейно с процентом удалённых значений, что соответствует формуле расчёта ошибки. При этом Mode на всех уровнях сохранял значительный отрыв в точности от Hot-Deck и особенно от Spline.

**Поведение по отдельным полям**

Проанализированы как категориальные (FullName, PassportInfo, Departure, Destination, Train, SeatChoice, PaymentCard), так и числовые/временные признаки (DepartureDate, ArrivalDate, TotalCost). Выяснилось, что:

- Методы, основанные на «колоночном» заполнении, не учитывают связей между полями в одной записи, что приводит к высоким ошибкам на сложных категориальных признаках (станции, вагоны).

- Чисто числовые поля (цена, даты) восстанавливаются точнее, однако Spline-интерполяция на данных с большими пропусками порождает артефакты.

# Источники

1. NumPy documentation // numpy.org URL: https://numpy.org/doc/stable/ (дата обращения: 04.03.2025).
2. PyQt5 Reference Guide // www.riverbankcomputing.com URL: https://www.riverbankcomputing.com/static/Docs/PyQt5/ (дата обращения: 04.03.2025).
3. pandas - Python Data Analysis Library // pandas.pydata.org URL: https://pandas.pydata.org/ (дата обращения: 25.04.2025).