Тема 5. Предобработка и очистка больших данных

Лекция 5.1. Качество данных: виды проблем и их последствия

По различным исследованиям, от 60 до 80% времени специалиста по данным занимает работа с качеством данных — поиск, исправление, согласование. Некачественные данные приводят к неверным моделям, ошибочным бизнес-решениям и потере доверия к аналитике. Согласно Gartner, плохое качество данных обходится организациям в среднем в 12,9 миллиона долларов в год.

Классификация проблем качества данных

— Пропущенные значения (NULL, NaN, пустые строки): возникают из-за ошибок ввода, технических сбоев, необязательных полей форм, объединения таблиц с разными схемами. Важно различать случайные пропуски (MCAR) и систематические (MNAR), так как стратегия обработки зависит от механизма пропуска.

— Дубликаты: одна сущность представлена несколькими записями. Возникают при слиянии источников данных, повторных загрузках ETL, отсутствии уникальных ограничений. При агрегации дубликаты искажают счётчики и суммы.

— Выбросы: значения, сильно отклоняющиеся от характерного диапазона. Могут быть ошибочными (опечатка «возраст 999 лет») или реальными (рекордная транзакция). Критично различать эти случаи перед удалением.

— Несогласованность форматов: одна и та же дата записана как «01.05.2024», «2024-05-01», «1 мая 2024». Имя: «Иванов И.И.», «Иванов Иван», «Ivan Ivanov». Объединение таких данных даёт ложные группы.

— Нарушения ограничений целостности: отрицательный возраст, сумма заказа меньше нуля, ссылка на несуществующий внешний ключ.

— Устаревшие данные: адреса, должности, телефоны устаревают. Без регулярного обновления справочников аналитика теряет актуальность.

Лекция 5.2. Обработка пропущенных значений

Механизмы пропуска данных

— MCAR (Missing Completely At Random): пропуск не зависит ни от пропущенного значения, ни от других переменных. Пример: случайный сбой датчика. Можно безопасно удалять строки, если доля мала.

— MAR (Missing At Random): пропуск зависит от других наблюдаемых переменных, но не от самого пропущенного значения. Пример: пациенты старшего возраста реже указывают вес. Импутация на основе других переменных корректна.

— MNAR (Missing Not At Random): пропуск зависит от самого пропущенного значения. Пример: люди с очень высоким доходом не указывают его. Ни простое удаление, ни стандартная импутация не дают несмещённых оценок.

Стратегии обработки

— Удаление строк: применимо при MCAR и доле пропусков менее 5%. Риск: потеря данных снижает мощность статистических тестов.

— Удаление столбцов: оправдано, если столбец содержит более 70–80% пропусков и имеет мало предсказательной силы.

— Импутация средним или медианой: быстро, но сужает распределение признака и занижает дисперсию. Медиана предпочтительнее при наличии выбросов.

— Импутация по группе: заменять пропущенное значение медианой или модой соответствующей группы (например, средняя зарплата по отрасли и городу).

— MICE (Multiple Imputation by Chained Equations): итерационный метод, моделирующий каждый признак с пропусками как функцию остальных признаков. Более точный, но вычислительно дорогой.

— KNN-импутация: значение заменяется взвешенным средним K ближайших по остальным признакам соседей. Хорошо сохраняет локальную структуру данных.

— В Spark: df.fillna({'col': value}), df.dropna(thresh=N), Imputer из pyspark.ml.feature.

Лекция 5.3. Обнаружение и обработка выбросов

Статистические методы обнаружения

— Метод межквартильного размаха (IQR): нижняя граница = Q1 - 1,5*IQR, верхняя = Q3 + 1,5*IQR. Прост, интерпретируем, не требует предположений о распределении. Стандарт для одномерных данных.

— Z-score: выброс при |z| > 3. Предполагает нормальное распределение. Чувствителен к самим выбросам (они смещают среднее и std). Modified Z-score использует медиану и MAD — устойчивее.

— Процентильный метод: удалять значения ниже p-го и выше (100-p)-го перцентиля. Например, 1-й и 99-й перцентили.

Алгоритмические методы обнаружения

— Isolation Forest: алгоритм на основе случайных деревьев. Аномальные точки изолируются меньшим числом разбиений, чем нормальные. Эффективен для многомерных данных.

— Local Outlier Factor (LOF): сравнивает локальную плотность точки с плотностью её соседей. Выявляет выбросы относительно локального контекста — полезно при данных с разной плотностью.

— DBSCAN: кластеризация по плотности. Точки, не принадлежащие ни одному кластеру, считаются шумом (выбросами).

Стратегии обработки выбросов

— Удаление: только при явной ошибке. Удаление реальных рекордных значений ведёт к потере информации.

— Winsorization (усечение): замена значений ниже p-го перцентиля на p-й перцентиль, выше (100-p)-го — на (100-p)-й. Сохраняет размер выборки.

— Логарифмическое преобразование: log(x+1) сжимает правый хвост распределения, уменьшая влияние крупных значений.

— Отдельная обработка: выбросы выделяются в отдельную страту и анализируются независимо от основной выборки.

Лекция 5.4. Дедупликация данных

При слиянии нескольких источников данных один реальный объект может быть представлен несколькими записями с небольшими отличиями в написании. Дедупликация миллиарда записей — нетривиальная вычислительная задача: наивное попарное сравнение требует 5*10^17 сравнений.

Точная дедупликация

— df.dropDuplicates(['col1', 'col2']): удаление строк с полным совпадением по указанным столбцам.

— Хэширование записи: вычислить MD5 или SHA256 по всем полям, удалить строки с совпадающим хэшем.

— Window + row_number(): пронумеровать строки в группе (по ключу), оставить только строку с номером 1 — позволяет выбирать «наилучшую» запись из дублей.

Нечёткая дедупликация

— Расстояние Левенштейна: минимальное число операций вставки, удаления и замены символов для преобразования одной строки в другую. «Иванов» и «Иаванов» — расстояние 1.

— Jaccard Similarity на n-граммах: разбить строку на n-граммы, вычислить долю совпадающих. Устойчива к перестановкам слов.

— Phonetic algorithms (Soundex, Metaphone): кодировать имена фонетически. «Смит» и «Smith» дают одинаковый код.

— MinHash + LSH (Locality Sensitive Hashing): приближённый поиск похожих документов в больших коллекциях за сублинейное время. Реализован в Spark MLlib.

Лекция 5.5. Масштабирование и кодирование признаков

Алгоритмы, использующие понятие расстояния (KNN, SVM, K-Means, PCA, нейронные сети), чувствительны к масштабу признаков. Признак «доход» с диапазоном 10 000–500 000 будет доминировать над признаком «возраст» с диапазоном 18–90. Масштабирование приводит признаки к сопоставимому масштабу.

— Min-Max Normalization: x' = (x - min) / (max - min). Результат в [0, 1]. Чувствителен к выбросам: один выброс сжимает все нормальные значения в узкий диапазон.

— Z-score Standardization: x' = (x - mean) / std. Результат: среднее 0, std 1. Устойчивее к выбросам. Стандарт для большинства линейных алгоритмов.

— Robust Scaler: x' = (x - median) / IQR. Наиболее устойчив к выбросам. Рекомендуется при загрязнённых данных.

— Max Abs Scaler: x' = x / max(|x|). Результат в [-1, 1]. Сохраняет разрежённость (sparse matrices) — важно при работе с текстовыми признаками.

Кодирование категориальных переменных

— Label Encoding: категории заменяются целыми числами. Вносит ложный порядок: «кошка»=0, «собака»=1, «птица»=2 подразумевает, что собака «больше» кошки. Приемлемо только для деревьев решений.

— One-Hot Encoding: каждая категория становится отдельным бинарным столбцом. При 1000 категориях создаёт 1000 столбцов — проблема высокой кардинальности.

— Binary Encoding: категория кодируется бинарным числом — log2(N) столбцов вместо N. Компромисс между Label и OHE.

— Target Encoding: категория заменяется средним значением целевой переменной в этой категории. Высокая предсказательная сила, но риск утечки данных — необходима кросс-валидация.

— В Spark MLlib: StringIndexer (Label Encoding) затем OneHotEncoder. VectorAssembler объединяет все признаки в вектор.

Лекция 5.6. ETL-конвейеры и оркестрация

ETL (Extract, Transform, Load) — стандартный процесс интеграции данных из источников в аналитическое хранилище. В Big Data контексте ETL-конвейеры обрабатывают терабайты данных, должны быть отказоустойчивыми, воспроизводимыми и мониторируемыми.

Spark Pipeline API

Pipeline в Spark MLlib — цепочка трансформеров (Transformer) и оценщиков (Estimator). Трансформер применяет преобразование к DataFrame (StringIndexer, OneHotEncoder, VectorAssembler). Оценщик обучается на данных и порождает трансформер (StandardScaler, IDF). Pipeline сохраняется и загружается как единый объект, что гарантирует применение одинаковых преобразований при обучении и эксплуатации

модели.

Инструменты оркестрации

— Apache Airflow: описание ETL-задач в виде DAG на Python. Планирование по расписанию (cron), зависимости между задачами, повторный запуск при сбое, веб-интерфейс мониторинга. Стандарт де-факто для оркестрации в Big Data.

— Apache NiFi: визуальный инструмент для построения потоков данных. Более 300 готовых процессоров для чтения, трансформации и записи данных. Удобен для нетехнических пользователей.

— dbt (Data Build Tool): SQL-ориентированный инструмент трансформации данных в хранилище. Версионирование через Git, документирование, тесты качества данных — встроены.

— Great Expectations: библиотека валидации данных. Определение ожидаемых свойств (expectations) и автоматическая проверка при каждом запуске пайплайна.