DOI https://doi.org/10.46845/2541-8254-2026-3(55)-67-67
УДК 004.6: 336

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ИНСТРУМЕНТОВ ДЛЯ ОБРАБОТКИ ГИБРИДНЫХ ДАННЫХ В ЗАДАЧАХ ФИНАНСОВОЙ АНАЛИТИКИ

А.С. Котюргина, студент,
email: kotyurgina.as@mail.ru
ФГБОУ ВО «Калининградский государственный технический университет»

Е.Ю. Скоробогатых, канд. пед. наук, доц.,
email: elena.skorobogatykh@klgtu.ru
ФГБОУ ВО «Калининградский государственный технический университет»

Проведён сравнительный анализ четырёх инструментов обработки гибридных данных (Pandas, Polars, PySpark, Dask) для задач финансовой аналитики. Гибридные данные включают числовые показатели, категориальные признаки, текстовые поля, пропуски, временные ряды в формате JSON и эмбеддинги изображений. Предложен оригинальный синтетический датасет «Калининград 2026» объёмом 70 000 записей в формате Parquet, калиброванный по реальным статистическим данным Калининградской области. На основе экспериментальных сравнений производительности и масштабируемости инструментов на различных типах данных сформулированы практические рекомендации по выбору оптимального инструментария. Установлено, что критерий выбора должен учитывать не только объём данных, но и их типологическую структуру.

Ключевые слова: *гибридные данные, финансовая аналитика, Pandas, Polars, PySpark, Dask, синтетический датасет, выбор инструмента.*

ВВЕДЕНИЕ

Финансовая аналитика в современных условиях требует обработки больших объёмов данных, сочетающих числовые показатели, текстовые поля, категориальные признаки и временные метки. Такое сочетание структурированных и неструктурированных данных принято называть «гибридными» [1]. Задачи кредитного скоринга нуждаются в эффективной обработке гибридной информации, поскольку стандартные финансовые показатели часто дополняются текстовыми описаниями, комментариями и другими слабоструктурированными данными.

Ряд исследований [2, 3] показывает, что использование методов машинного обучения и интеллектуального анализа данных на гибридных наборах позволяет повысить точность прогнозирования дефолтов по сравнению с традиционными статистическими моделями. Однако в российской практике внедрение таких подходов осложняется фрагментацией данных между различными источниками - банками, государственными органами, операторами связи и маркетплейсами. Как отмечает Р. Ахтямов, «в отличие от Китая, где мега-платформы видят практически всю экономическую активность пользователя, в России информация распределена между банками, госуслугами, операторами связи, маркетплейсами и управляющими компаниями» [4, С. 1].

С инструментальной точки зрения обработка гибридных данных объёмом от нескольких миллионов строк требует выбора эффективных библиотек. В экосистеме Python стандартной является Pandas, однако она обладает недостатками: высоким потреблением памяти и низкой скоростью агрегаций на больших данных [6]. В качестве альтернативных решений рассматриваются библиотека Polars [7], реализованная на языке Rust с использованием колоночного формата Apache Arrow и механизма ленивых вычислений, PySpark, предназначенный для распределенной обработки данных на кластерах [8], а также фреймворк Dask [9], обеспечивающий параллелизацию Pandas-операций с сохранением API. Их сравнительная эффективность применительно к реальным гибридным банковским данным в российском контексте систематически не изучалась.

Исследовательский пробел заключается в отсутствии сравнительных работ, оценивающих производительность и удобство использования указанных инструментов при обработке гибридных транзакционных данных, содержащих как структурированные, так и неструктурированные компоненты, в условиях ограниченных вычислительных ресурсов (одна машина).

ЦЕЛЬ, ЗАДАЧИ, ОБЪЕКТ И ПРЕДМЕТ ИССЛЕДОВАНИЯ

Цель исследования: на основе анализа инструментов Pandas, Polars, PySpark и Dask, в соответствии с разработанными критериями сравнения, проведенной экспериментальной оценки их производительности на примере задачи очистки и объединения транзакций, сформулировать рекомендации для выбора наиболее оптимального инструмента для решения задач финансовой аналитики.

Для достижения поставленной цели решены следующие задачи:

1) выполнен обзор существующих инструментов обработки табличных данных (Pandas, Polars, PySpark, Dask) и выявлены их архитектурные особенности;
2) сформулированы критерии сравнения, учитывающие специфику гибридных финансовых данных (наличие текстовых полей, пропусков, категориальных признаков, вложенных JSON-структур и эмбеддингов изображений;
3) разработан синтетический датасет, калиброванный по реальным статистическим данным Калининградской области, и реализован сквозной пайплайн обработки (очистка текста, заполнение пропусков, скользящее среднее);
4) проведены экспериментальные замеры производительности и потребления памяти для каждого инструмента;
5) на основе полученных результатов сформулированы рекомендации по выбору инструмента в зависимости от объёма и типа гибридности данных.

Объект исследования - гибридные данные в задачах финансовой аналитики (в данном случае, это банковские транзакции, содержащие числовые, текстовые и категориальные поля, а также вложенные временные ряды и эмбеддинги изображений).

Предмет исследования - способы и методы выполнения операций очистки текста, заполнения пропусков, и вычисления скользящего среднего при работе с гибридными данными в библиотеках Pandas, Polars PySpark и Dask.

МЕТОДЫ ИССЛЕДОВАНИЯ

1. Анализ типовых особенностей гибридных данных в задачах финансовой аналитики.

Под гибридными данными в контексте финансовой аналитики понимаются наборы, одновременно содержащие в себе структурированные данные и неструктурированные. Такая структура характерна для банковских транзакций, логов платёжных систем и данных мониторинга финансовых операций. [10]

Структурированные данные - это информация, организованная в заранее определённом формате с чёткими типами данных, фиксированными полями и связями между элементами.

Они обычно хранятся в реляционных базах данных, электронных таблицах или других системах с фиксированной структурой. Примеры структурированных данных в финансовой сфере: банковские транзакции, кредитные истории, балансы счетов, данные клиентов (возраст, доход), номера счетов, графики погашения кредита, процентные ставки и кредитные рейтинги. [11]

Неструктурированные данные - это информация, которая не имеет предопределённой структуры или организации. Они могут быть представлены в виде текста, изображений, видео, аудиофайлов, данных из социальных сетей и т. д. [10]. Примеры неструктурированных данных в финансах: финансовые новости и отчёты компаний, посты в социальных сетях, звонки в колл-центр, электронные письма клиентов, стенограммы чатов, отсканированные документы (формы KYC, контракты), записи голосовых сообщений от службы поддержки.

Для данного исследования из всего многообразия гибридных данных были выбраны те виды, которые наиболее часто встречаются в реальных задачах финансового анализа и при этом создают разнородную нагрузку на инструменты обработки.

В качестве *структурированных данных* использованы банковские транзакции, включающие числовые показатели (суммы операций), категориальные признаки (названия банков, типы транзакций, коды валют) и временные метки.

В качестве *неструктурированных данных* выбраны: текстовые комментарии с аномалиями («назначение платежа»), 6-месячные JSON-ряды и 64-мерные эмбеддинги изображений (подписи, фото документов). Такое сочетание позволяет оценить работу инструментов в условиях, приближенных к реальным финансовым системам.

2. Аналитический обзор инструментов для обработки гибридных данных.

Pandas является фундаментом финансового анализа на языке Python. Его преимущества включают обширную экосистему, интуитивно понятный API и большое количество доступных учебных материалов [5]. Основными ограничениями является высокое потребление оперативной памяти и низкая производительность на объёмах данных, превышающих ~10 млн строк. Данное число строк, как и другие упомянутые в этом разделе, рассматривается как условный порог, используемый в контексте только структурированных данных в формате CSV-файла; этот порог может варьироваться в зависимости от версий библиотек, конфигурации кластеров, сложности операций и характеристик данных.

Polars представляет собой современную библиотеку, реализованную на языке Rust. Ключевые особенности: использование колоночного формата Apache Arrow, механизм ленивых вычислений (Lazy API), автоматическая параллелизация операций [6, 7]. Эти особенности показывают неплохой выигрыш в производительности по сравнению с Pandas при выполнении операций группировки и агрегации на наборах данных объёмом от 1 до ~100 млн строк.

PySpark является компонентом экосистемы Apache Spark и предназначен для распределённой обработки больших данных на кластерах [8]. Основным преимуществом PySpark является масштабируемость - возможность обработки терабайтных объёмов данных путём добавления вычислительных узлов. Однако для объёмов до ~10 млн строк CSV-файла накладные расходы на инициализацию JVM и преобразование данных делают PySpark неэффективным по сравнению с Polars и Pandas. Кроме того, PySpark отличается высокой сложностью и требует написания значительного объёма программного кода.

3. Методология эксперимента

Для проведения экспериментального сравнения был разработан синтетический датасет «Калининград 2026», содержащий 70000 записей в формате Parquet.

Почему именно формат Parquet, а не часто встречающийся формат CSV-файла?

Во-первых, Parquet является колоночно-ориентированным форматом, что позволяет читать с диска только необходимые столбцы, снижая нагрузку на устройство. Во-вторых, Parquet поддерживает эффективное сжатие, особенно важное для текстовых полей и JSON-рядов. В-

третьих, все четыре библиотеки имеют нативную поддержку Parquet, обеспечивая воспроизводимость эксперимента [12].

Выбор объёма в 70000 строк обусловлен тем, что на этом размере Pandas начинает замедляться, Polars сохраняет скорость, Dask демонстрирует промежуточные результаты, а PySpark страдает от высоких накладных расходов (как известно по структурированным данным). Параметры генерации были калиброваны на основе статистических показателей Калининградстата за январь-февраль 2026 года (отраслевая структура) [13], БИК и ИНН действительно существующих филиалов банка в области [14]. Генерация сумм транзакций, доля выбросов и пропусков являются авторскими допущениями, так как целью работы является сравнение производительности инструментов, а не экономическое моделирование. Структура сгенерированного датасета (с переносами колонок) представлен на рис. 1.

```
      inn_client  loan_amount  loan_term              loan_purpose  \
0     0000104377       1359.0         24                     Сырьё
1     0000106864     147378.0         12  Модернизация производства
2  0000362170144    1090844.0         24    Расширение ассортимента
3  0000371198502       4850.0         12           Закупка товаров
4     0000396930      46658.0         18           Закупка товаров

   has_collateral  estimated_income  total_amount  avg_amount  txn_count  \
0               0              1852        377.78      377.78          1
1               0            109302      22751.54    22751.54          1
2               1           1382835     386424.93   386424.93          1
3               1              4776       2010.21     2010.21          1
4               1             82402      34050.07    34050.07          1

   dti_ratio  ...       main_bank        monthly_amounts_json  \
0   0.733405  ...      Альфа-Банк     [0, 0, 0, 0, 0, 377.78]
1   1.348344  ...             ВТБ   [0, 0, 0, 0, 0, 22751.54]
2   0.788846  ...        Сбербанк  [0, 0, 0, 0, 0, 386424.93]
3   1.015282  ...             ВТБ    [0, 0, 0, 0, 0, 2010.21]
4   0.566217  ...  Россельхозбанк   [0, 0, 0, 0, 0, 34050.07]

   monthly_txn_counts_json                          credit_application_text  \
0       [0, 0, 0, 0, 0, 1]  Производственное предприятие "БалтМеханика" – ...
1       [0, 0, 0, 0, 0, 1]  Производственное предприятие "БалтМеханика" – ...
2       [0, 0, 0, 0, 0, 1]  Компания ООО "Торговый Дом" занимается оптовой...
...
 26  log_estimated_income     69284 non-null  float64
dtypes: float64(11), int64(6), str(10)
memory usage: 265.4 MB
```

Рисунок 1 – Структура датасета (банковские транзакции)

Для оценки производительности была сформулирована задача финансовой аналитики на основе структуры описанного датасета, включающая:

- очистку текстового поля comment от аномальных символов (цифры, знаки препинания, мусор);
- заполнение пропусков в поле amount медианным значением по группам bank_name – стандартная задача в скоринге и анализе кредитоспособности;
- извлечение номера договора из текстового поля comment с помощью регулярного выражения №(\d+) (задача извлечения структурированной сущности из неструктурированного текста) – реальная задача обработки логов банков;
- разбор JSON-рядов monthly_amounts_json и monthly_txn_counts_json с вычислением среднемесячного оборота и волатильности для каждого клиента (работа с вложенными полуструктурированными данными) – ситуация при работе с API банков/бирж;
- вычисление скользящего среднего суммы транзакций за последние 100 операций для каждого банка с сохранением хронологического порядка – ключевой индикатор для антифрод-систем.

Замеры проводились на ноутбуке с 16 ГБ RAM, 8-ядерным CPU (Intel i7–1260P). Для каждого инструмента запускался сквозной пайплайн 10 раз, чтобы исключить случайность в замерах, бралась медиана, так как она чувствительна к выбросам в данных. Память измерялась с помощью memory_profiler и через UI Spark (PySpark).

Практическая реализация экспериментальной части приведена в открытой папке на Яндекс Диск с прикреплённым Jupiter Notebook, которую можно посмотреть по ссылке [12].

Конкретно для PySpark запускалась версия Java 17 во избежание конфликтов версий для работы программного кода.

4. Обоснование выбора критериев для сравнения инструментов

Критерии, описанные ниже, исходят из особенностей гибридных финансовых данных, которые были выбраны для этого исследования:

- производительность — время выполнения группировок и агрегаций. Чем быстрее инструмент выполняет такие операции, тем меньше аналитик тратит времени на ожидание результатов и тем оперативнее могут быть приняты управленческие решения;

- потребление памяти — гибридные данные (текст + числа) увеличивают расход RAM, особенно в Pandas;

- работа с пропусками — в финансах пропуски часты (незаполненные комментарии, суммы), важна скорость и удобство fillna/dropna, важно не только быстро, но и удобно выполнять операции, так как способ обработки пропусков может различаться в зависимости от поставленной задачи;

- поддержка текстовой обработки — ключевой признак гибридности (очистка комментариев, извлечение сущностей); скорость выполнения регулярных выражений и работа со строками существенно различаются между инструментами и влияют на общую производительность запусков кода;

- аудируемость кода: под аудируемостью кода понимается его читаемость, понятность и пригодность для проверки (верификации) другим специалистом. В финансовой сфере код часто проходит внутренний аудит, требования к которому прописаны в нормативных актах. Код должен быть понятен не только его автору, но и другим.

РЕЗУЛЬТАТЫ И ВЫВОДЫ

Результаты эксперимента приведены в таблице 1.

Таблица 1 – сравнительная таблица Pandas, Polars, PySpark и Dask

Критерии	Pandas	Polars	PySpark	Dask
Время выполнения (медиана, в сек.)	15,47	0,51	6,57 из-за JVM	86,40
Память (ГБ)	0,03	0,00	0,001	0,04
Работа с пропусками	Интуитивно, .fillna()	Лениво (т.е. оптимизируется автоматически)	Требует явных join/then	Требует явного merge
Текстовая обработка	Циклы Python (медленно)	Векторизованная (Rust) - быстро	Умеренно (JVM)	Параллелизация Python-функций
Аудируемость кода	Отличная	Хорошая	Низкая	Хорошая

В дополнении к таблице 1 построена тепловая карта сравнения библиотек по задачам для большей наглядности результатов (рис. 2). Исходные метрики имеют разные единицы измерения, поэтому была проведено их нормирование к единой шкале (0, 1) с помощью минимаксного преобразования. Для времени выполнения и потребления памяти библиотекой использовалось инвертированное нормализованное значение, где 1 соответствует минимальному (т.е. наилучшему) показателю, а 0 – максимальному (наихудшему).

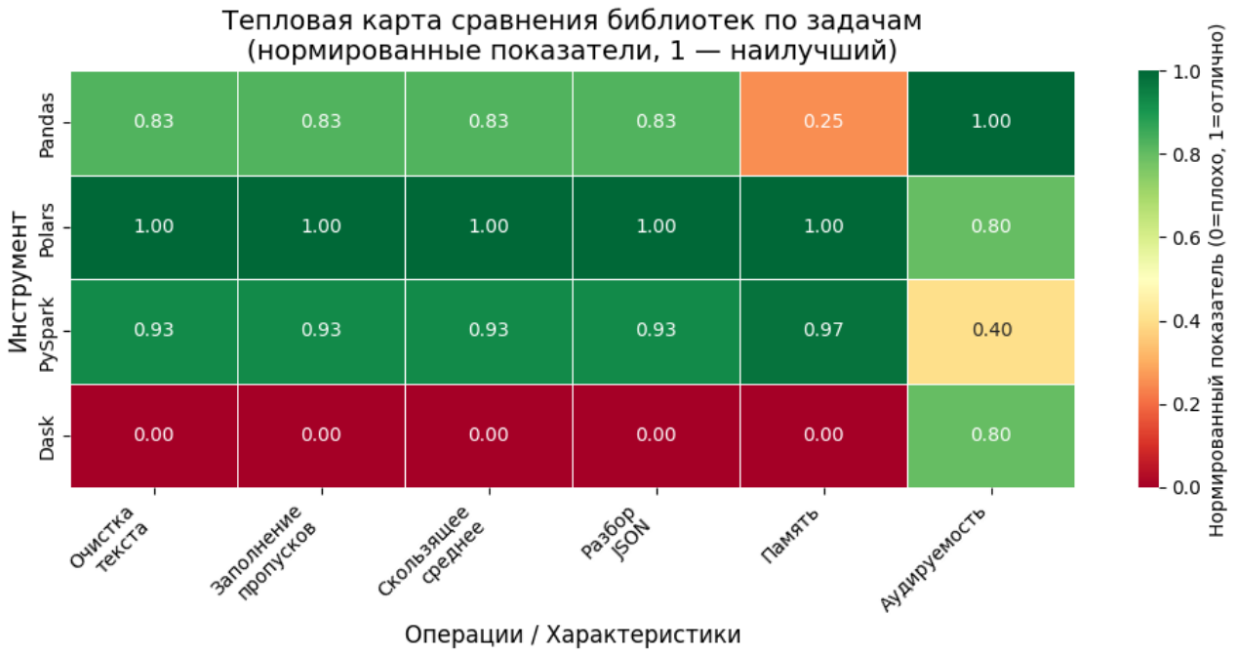

Рисунок 2 – Тепловая карта сравнения Pandas, Polars, PySpark и Dask

Замеры выполнены автором с помощью кода [15]. Время для PySpark включает создание сессии, загрузку и преобразования. Память для PySpark можно проверить только через кеширование после запуска кода, перейдя по ссылке http://localhost:4040 (обычные внутренние методы Python не реализуются для этой библиотеки). Во вкладке Executors выведена сводная таблица, где во вкладке Storage Memory можно посмотреть использование памяти. Просмотр использования ресурсов в PySpark представлен на рис. 3.

Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size on Disk
Disk Memory Deserialized 1x Replicated	8	4,00%	1067.5 KiB	0.0 B

Рисунок 3 – Результаты замеров памяти для библиотеки PySpark

Распределение времени в дополнение к результатам по каждому инструменту изображено на рис. 4.

Рисунок 4 – Распределение времени между Pandas, Polars, PySpark и Dask

Анализ результатов эксперимента.

Неоднородность финансовых данных по-разному нагружает каждый инструмент и должна учитываться при их выборе.

По объёму данных. При объёме менее 100 тыс. строк и приоритете простоты разработки целесообразно использовать Pandas (15,47 сек из табл.). Для данных объёмом до 1 млн строк рекомендуется Polars, который обеспечивает максимальную производительность. При работе с массивами данных более 100 млн строк предпочтительным является PySpark, однако на малых объёмах (70 тыс. строк) он показывает результат (6,57 сек), сопоставимый с Pandas. Dask не рекомендуется для малых объёмов данных из-за высоких накладных расходов (86,40 сек) и нестабильности времени выполнения (разброс от 29 до 131 сек).

При высокой доле текстовых полей. Polars демонстрирует наибольшую производительность (0,51 сек) благодаря векторизованной обработке на Rust. Pandas уступает Polars (15,47 сек) из-за интерпретируемых Python-циклов. PySpark показывает приемлемую скорость (6,57 сек), однако сопровождается значительным объёмом программного кода и сложностью реализации регулярных выражений. Dask на текстовых операциях страдает от высоких накладных расходов на сериализацию, что не выгодно для такого рода данных.

При наличии *пропусков, требующих заполнения на основе группировок*, синтаксис Pandas и Polars более интуитивен и позволяет реализовать операцию в одну-две строки. PySpark требует явного выполнения соединений (join) и использования условных конструкций, что увеличивает сложность реализации. Dask также требует явного merge и демонстрирует худшую производительность.

В случае *категориальных признаков с высокой кардинальностью* (например, тысячи банков-контрагентов) Polars и PySpark обрабатывают данные эффективнее Pandas. Однако эксперимент показал, что на малом объёме данных Pandas (15,47 сек) уступает Polars (0,51 сек) и PySpark (6,57 сек), что подтверждает преимущество векторизованных и распределённых решений.

Дополнительные возможности. Polars поддерживает потоковый режим обработки (streaming), позволяющий выполнять вычисления над данными, превышающими доступный объём оперативной памяти. Dask, несмотря на высокие накладные расходы, обеспечивает распределённые вычисления на кластере. PySpark остаётся единственным инструментом для распределённых вычислений на кластерах из десятков и сотен узлов.

Таким образом, выбор инструмента определяется не только объемом строк, но и требуемой стабильностью выполнения, сложностью реализации и структурой данных: долей текстовых полей, характером пропусков и кардинальностью категориальных признаков.

ЗАКЛЮЧЕНИЕ

В ходе проведённого исследования были решены все поставленные задачи: выполнен обзор архитектурных особенностей Pandas, Polars, PySpark и Dask; сформулированы критерии сравнения, учитывающие специфику гибридных финансовых данных, разработан синтетический датасет и проведены экспериментальные замеры производительности и потребления памяти. На основании проведённого анализа сформулированы рекомендации по выбору инструмента обработки гибридных данных в зависимости от характеристик задачи.

Полученные результаты подтвердили, что каждый из рассмотренных инструментов имеет свою область эффективного применения. Polars является абсолютным лидером по производительности (0,51 сек) благодаря реализации на Rust, векторизованной обработке текста и ленивым вычислениям, что делает его предпочтительным выбором для гибридных данных объёмом до 1 млн строк. Pandas (15,47 сек) сохраняет актуальность для малых объёмов данных и задач с высокими требованиями к аудируемости кода благодаря интуитивному синтаксису. PySpark (6,57 сек) показал неожиданно высокую производительность на малом объёме данных, однако сложность реализации и накладные расходы на инициализацию JVM ограничивают его применение случаями наличия готовых кластеров или объёмов данных более 100 млн строк. Dask (86,40 сек) продемонстрировал наихудшую производительность и низкую стабильность времени выполнения (разброс от 29 до 131 сек), что делает его нерекомендуемым для гибридных финансовых данных малого и среднего объёма.

Результаты исследования могут быть использованы при проектировании систем финансовой аналитики для обоснованного выбора инструмента под гибридные данные с учётом не только объёма, но и требуемой стабильности, сложности реализации и структуры данных.

СПИСОК ЛИТЕРАТУРЫ

1. Определение набора данных: [Электронный ресурс] / coder-castrov.vercel.app. — URL: https://coder-castrov.vercel.app/blogs/opredelenie-nabora-dannyh-a303f4ce-1d4 (дата обращения: 12.03.2026).
2. Ломакин Д.А. Оптимизация кредитного скоринга с использованием ансамблевых моделей машинного обучения [Электронный ресурс] / НИУ ВШЭ. — 2026. — URL: https://www.hse.ru/edu/vkr/1160980008 (дата обращения: 13.07.2026).
3. Нигматзянов Т.А. Анализ и обработка слабоструктурированных данных для последующего обучения моделей, применяемых в процессе кредитного скоринга [Электронный ресурс] / НИУ ВШЭ. — 2025. — URL: https://www.hse.ru/edu/vkr/1046121092 (дата обращения: 01.05.2026).
4. Ахтямов Р. Кредитный скоринг. Китайский опыт и российские регуляторные барьеры [Электронный ресурс] / PLUSworld.ru. — 19 апреля 2026. — URL: https://plusworld.ru/articles/71474/ (дата обращения: 01.05.2026).
5. Pandas Documentation: pandas: powerful Python data analysis toolkit [Электронный ресурс] / pandas.pydata.org. — URL: https://pandas.pydata.org/docs/ (дата обращения: 14.03.2026).
6. Что такое Polars: ускоряем обработку данных ленивыми вычислениями [Электронный ресурс] / Skillfactory media. — 6 марта 2026. — URL: https://blog.skillfactory.ru/chto-takoe-polars/ (дата обращения: 12.04.2026).
7. Polars documentation : Blazingly fast DataFrames in Rust & Python [Электронный ресурс] / pola.rs. — URL: https://pola.rs/ (дата обращения: 23.03.2026).
8. PySpark documentation [Электронный ресурс] / Apache Spark. — URL: https://spark.apache.org/docs/latest/api/python/index.html (дата обращения: 15.04.2026).

9. Dask Documentation: Scalable analytics in Python [Электронный ресурс] / dask.org. — URL: https://dask.org/ (дата обращения: 30.04.2026).
10. Технологии обработки структурированной и неструктурированной информации в задачах экономики и управления [Электронный ресурс] / Уральский федеральный университет. — URL: https://elar.urfu.ru/bitstream/10995/145122/1/978-5-91256-754-4_2025_011.pdf (дата обращения: 28.04.2026).
11. Типы данных в финансах [Электронный ресурс] / mosregdata.ru. — URL: https://mosregdata.ru/article/tipy-dannyh-financy (дата обращения: 28.04.2026).
12. What is Apache Parquet? [Электронный ресурс] / https://www.ibm.com/. —URL: https://www.ibm.com/think/topics/parquet?regionCode=us&languageCode=en&contactmodule=true&cm-history=us-en (дата обращения: 14.03.2026)
13. Территориальный орган Федеральной службы государственной статистики по Калининградской области: Доклад «Социально-экономическое положение Калининградской области» [Электронный ресурс] / 39.rosstat.gov.ru. — URL: https://39.rosstat.gov.ru/statistical_compilations?print (дата обращения: 17.03.2026).
14. Банки Калининграда (БИК, ИНН, количество отделений) [Электронный ресурс] / banklab.ru. — URL: https://banklab.ru/ (дата обращения: 17.03.2026).
15. Практическая реализация программной части эксперимента [Электронный ресурс] / disk.yandex.ru. — URL: https://disk.yandex.ru/d/5tqrtiNBd7Woaw (дата обращения: 28.04.2026).

COMPARATIVE ANALYSIS OF TOOLS FOR HYBRID DATA PROCESSING IN FINANCIAL ANALYTICS TASKS

A.S. Kotyurgina, Student,
email: kotyurgina.as@mail.ru
Kaliningrad State Technical University

E.Yu. Skorobogatykh, PhD, Associate Professor,
email: elena.skorobogatykh@klgtu.ru
Kaliningrad State Technical University

A comparative analysis of four hybrid data processing tools (Pandas, Polars, PySpark, Dask) for financial analytics tasks has been conducted. Hybrid data here includes numerical indicators, categorical features, text fields, missing values, time series in JSON format, and image embeddings. An original synthetic dataset, "Kaliningrad 2026," with 70,000 records in Parquet format, calibrated against real statistical data from the Kaliningrad region, has been proposed. Based on experimental comparisons of tool performance and scalability across different data types, practical recommendations for selecting the optimal toolset have been formulated. It has been established that the selection criterion should take into account not only the volume of data but also its typological structure.

Keywords: hybrid data, financial analytics, Pandas, Polars, PySpark, Dask, synthetic dataset, tool selection.