Оригинальная статья / Original article

УДК 004.622

https://doi.org/10.21869/2223-1560-2024-28-4-67-85

Эффективность ETL-процесса для предиктивной аналитики

А. В. Олейникова [1] ✉, И. О. Бондарева [1], А. А. Олейников [1]

[1] Астраханский государственный технический университет
ул. Татищева, д.16, г. Астрахань 414056, Российская Федерация

✉ e-mail: a.oleynikova.astu@mail.ru

Резюме

Цель исследования. *В настоящей работе исследуется эффективность различных методов обработки пропущенных значений в датафреймах применительно к задачам предобработки данных в рамках предиктивной аналитики. В качестве тестовых данных используются три открытых датасета, которые содержат информацию о характеристиках зданий, метеорологических условиях и энергопотреблении. Цель исследования состоит в выявлении наиболее эффективного метода для предобработки данных в процессе ETL для решения задач предиктивной аналитики.*

Методы. *В работе происходит объединение датафреймов из каждого датасета и анализ стандартных методов модуля Pandas, высокоуровневой библиотеки языка Python, таких как прямое присваивание, использование индексаторов, а также метод fillna с словарем. Кроме того, разработан модуль на языке Cython, C-подобном языке программирования, для оптимизации процесса заполнения пропущенных значений, произведена оценка производительности каждого метода.*

Результаты. *Результаты демонстрируют, что прямое присваивание является наиболее эффективным методом с точки зрения производительности в Pandas. Применение Cython, хотя теоретически и способно ускорить вычисления, в данном случае показало значительное снижение производительности из-за накладных расходов на преобразование данных и взаимодействие между Python и Cython. Профилирование кода подтвердило, что местом с недостаточной производительностью являются операции Pandas, а не выполнение Cython кода.*

Выводы. *Таким образом, для большинства задач ETL рекомендуется использовать оптимизированные методы Pandas, а Cython следует применять только в случаях критической необходимости повышения производительности и при тщательной оптимизации кода для минимизации накладных расходов, так как написание кода, аналогичного Pandas, потребует значительных ресурсов, в том числе и для его оптимизации, что в большинстве случаев является избыточным.*

Ключевые слова: *предиктивная аналитика; пропущенные значения; профилирование; предобработка данных; pandas; cython; etl.*

Конфликт интересов: *Авторы декларируют отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.*

Для цитирования: Олейникова А. В., Бондарева И. О., Олейников А. А. Эффективность ETL-процесса для предиктивной аналитики // Известия Юго-Западного государственного университета. 2024; 28(4): 67-85. https://doi.org/10.21869/2223-1560-2024-28-4-67-85.

Поступила в редакцию 30.10.2024 *Подписана в печать 21.11.2024* *Опубликована 10.12.2024*

ETL Process Efficiency for Predictive Analytics

Alla V. Oleynikova [1] ✉, Irina O. Bondareva [1], Aleksandr A. Oleynikov [1]

[1] Astrakhan State Technical University

16 Tatishcheva str., Astrakhan 414056, Russian Federation

✉ e-mail: a.oleynikova.astu@mail.ru

Abstract

Purpose of research. *This paper investigates the effectiveness of different missing value handling methods in dataframes for data preprocessing tasks in predictive analytics. Three open datasets containing information on building characteristics, meteorological conditions, and energy consumption are used as test data. The goal of the study is to identify the most effective method for data preprocessing in the ETL process for solving predictive analytics problems.*

Methods*. The paper combines dataframes from each dataset and analyzes standard methods of the Pandas module, a high-level library of the Python language, such as direct assignment, the use of indexers, and the fillna method with a dictionary. In addition, a module in Cython, a C-like programming language, is developed to optimize the process of filling missing values, and the performance of each method is evaluated.*

Results*. The results demonstrate that direct assignment is the most effective method in terms of performance in Pandas. Using Cython, although theoretically capable of speeding up calculations, in this case showed a significant decrease in performance due to the overhead of data transformation and interaction between Python and Cython. Code profiling confirmed that the place with insufficient performance is Pandas operations, not Cython code execution.*

Conclusion*. Thus, for most ETL tasks, it is recommended to use optimized Pandas methods, and Cython should be used only in cases of critical need for performance improvement and with careful optimization of the code to minimize overhead, since writing code similar to Pandas will require significant resources, including for its optimization, which in most cases is redundant.*

Keywords: *predictive analytics; missing values; profiling; data preprocessing; pandas; cython; etl.*

Conflict of interest. *The authors declare the absence of obvious and potential conflicts of interest related to the publication of this article.*

For citation: Oleynikova A. V., Bondareva I. O., Oleynikov A. A. ETL Process Efficiency for Predictive Analytics // *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the Southwest State University*. 2024; 28(4): 67-85 (In Russ.). https://doi.org/10.21869/2223-1560-2024-28-4-67-85.

Received 30.10.2024 *Accepted 21.11.2024* *Published 10.12.2024*

***

Введение

В процессе решения задачи предиктивной аналитики, ключевым этапом является ETL-процесс, включающий извлечение, трансформацию, загрузку, направленный на подготовку данных для последующего анализа и моделирования. Особую актуальность приобретает проблема обработки пропущенных значений, поскольку даже незначительные пробелы

в данных могут существенно повлиять на точность и достоверность результатов моделирования [1-5, 6-8].

В качестве инструментария для проведения ETL процесса и обработки пропущенных значений, в текущих условиях, наиболее целесообразно использовать приложения, не привязанные к какой-либо платформе. Кроссплатформенное приложение Jupyter Notebook, работающее на базе языка программирования Python, может функционировать как на базе операционных систем (ОС) windows, так и под управлением unix подобных ОС linux. Преимуществом Jupyter Notebook, в отличие от no code платформ, является его гибкость и расширяемость, обусловленная богатым набором специализированных библиотек. В частности, библиотека Pandas предоставляет широкие возможности для управления данными и обработкой пропущенных значений [9, 6, 10-12].

Материалы и методы

В качестве основы для анализа использовались три открытых датасета ASHRAE: building_metadata.csv.gz, weather_train.csv.gz и train.0.0.csv.gz с датафреймами buildings, weather и energy_0. Эти датасеты содержат информацию о характеристиках зданий, метеорологических условиях и показаниях счетчиков потребления энергии (рис. 1).

Графическое представление датафрейма energy_0 представляет собой временной ряд (рис. 2).

В исходных датафреймах присутствуют несколько столбцов, в которых пропущенные (NaN) значения необходимо заменить на логически правильные (табл. 1) [13,14,15].

	building_id	meter	timestamp	meter_reading
0	0	0	2016-01-01 00:00:00	0.0
1	0	0	2016-01-01 01:00:00	0.0
2	0	0	2016-01-01 02:00:00	0.0
3	0	0	2016-01-01 03:00:00	0.0
4	0	0	2016-01-01 04:00:00	0.0

Рис. 1. Заголовок датафрейма energy_0

Fig. 1. Dataframe header energy_0

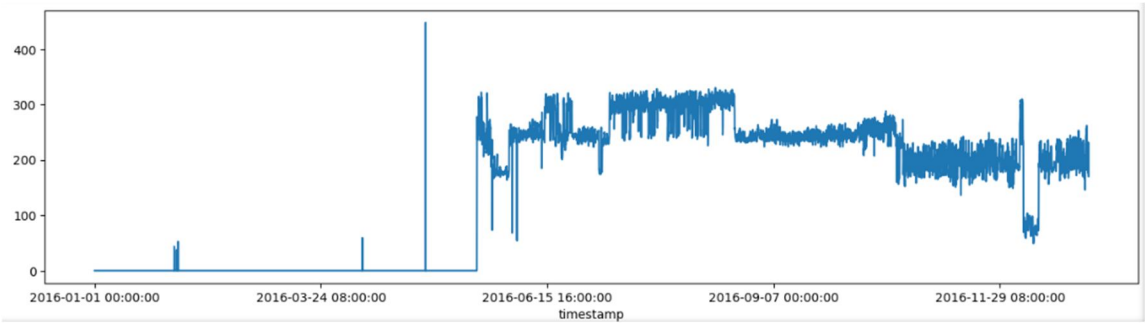

Рис. 2. График датафрейма energy_0 по осям timestamp и meter_reading

Fig. 2. Graph of the energy_0 dataframe on the timestamp and meter_reading axes

Таблица 1. Правила замены NaN значений в датафрейме

Table 1. Rules for replacing NaN values in a dataframe

Название столбца / Column Name	Правило замены / Replacement Rule	Значение / Value
air_temperature	NaN - 0	Температура
cloud_coverage	NaN - 0	Облачность
dew_temperature	NaN - 0	Точка росы
precip_depth_1_hr	NaN - 0	Осадки
sea_level_pressure	среднее	Атмосферное давление
wind_direction	среднее	Направление ветра

Правила замены руководствуются логикой и основываются на предположении, что пропущенное значение может означать отсутствие измеряемого параметра [2, 16, 17]. При этом: Температура – 0 градусов Цельсия, реальная температура, которая может быть зафиксирована; Облачность – 0% облачности – ясное небо; Точка росы – может быть близка к 0°C при определенных условиях; Осадки – 0 мм осадков означает отсутствие дождя.

Эти правила будут использованы при заполнении пропущенных значений в объединенном датафрейме energy_0, который был получен путем объединения данных датафреймов energy_0, buildings, weather (рис. 3) [12-14, 18-20].

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8784 entries, 0 to 8783
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   timestamp           8784 non-null   object
 1   site_id             8784 non-null   int64
 2   building_id         8784 non-null   int64
 3   meter               8784 non-null   int64
 4   meter_reading       8784 non-null   float64
 5   primary_use         8784 non-null   object
 6   square_feet         8784 non-null   int64
 7   year_built          8784 non-null   float64
 8   floor_count         0 non-null      float64
 9   air_temperature     8784 non-null   float64
 10  cloud_coverage      8784 non-null   float64
 11  dew_temperature     8784 non-null   float64
 12  precip_depth_1_hr   8784 non-null   float64
 13  sea_level_pressure  8784 non-null   float64
 14  wind_direction      8784 non-null   float64
 15  wind_speed          8784 non-null   float64
dtypes: float64(10), int64(4), object(2)
memory usage: 1.1+ MB
```

Рис. 3. Вывод метаданных объединенного датафрейма energy_0

Fig. 3. Outputting metadata for the merged energy_0 dataframe

После загрузки датафреймов и их объединения необходимо провести анализ набора данных на пропущенные значения. Для этого используется функционал библиотеки Pandas. С ее помощью осуществим поиск и анализ столбцов с пропущенными значениями в датафрейме energy_0, а в качестве примера возьмем столбец “объем осадков в миллиметрах в час” (precip_depth_1_hr). В начале запускается цикл, осуществляющий итерацию по столбцам в датафрейме energy_0. Подсчет пропущенных значений происходит в цикле. Здесь создается логический ряд, указывающий, является ли каждое значение в столбце пустым (True) или нет (False). Далее происходит суммирование всех значений True в логическом ряду, фактически подсчитывается количество пустых значений в конкретном столбце. Результат сохраняется в переменной и с помощью условия if energy_nulls > 0 проверяет, превышает ли количество пропущенных значений (energy_nulls) в текущем столбце нуль, есть ли пропущенные значения. Если в столбце есть пропущенные значения, выводится имя столбца, а затем количество пропущенных значений, которые он содержит (табл. 2 – 3, рис.4).

Выведены будут только те строки из исходного датафрейма energy_0, где в столбце "precip_depth_1_hr" были найдены пропущенные значения (NaN).

На следующем этапе происходит замена пропусков в соответствии с ранее установленными правилами (см. табл. 1). Для этого применяется метод библиотеки Pandas .fillna() с аргументом inplace=True. Аргумент inplace=True указывает, что изменения нужно внести непосредственно в исходный датафрейм energy_0. Если бы inplace=True не был указан, то метод .fillna() вернул бы новый датафрейм с внесенными изменениями, а исходный датафрейм energy_0 остался бы без изменений.

Таблица 2. Исходный датафрейм (energy_0)

Table 2. Original dataframe (energy_0)

…	…	precip_depth_1_hr	…	…
…	…	2.5	…	…
…	…	NaN	…	…
…	…	1.8	…	…
…	…	NaN	…	…
…	…	0.7	…	…

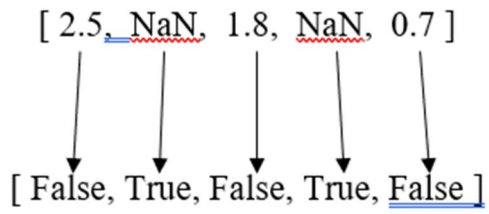

Рис. 4. Выделение столбца "precip_depth_1_hr" и поиск пропущенных значений (isnull())

Fig. 4. Selecting the column "precip_depth_1_hr" and searching for missing values (isnull())

Таблица 3. Фильтрация строк датафрейма по полученным True/False

Table 3. Filtering dataframe rows by received True/False

…	…	precip_depth_1_hr	…	…
…	…	NaN	…	…
…	…	NaN	…	…

```
C:\Users\vector06c\AppData\Local\Temp\ipykernel_21400\1682503974.py:1: FutureWarning: A value is trying to be set on
a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which
we are setting values always behaves as a copy.
```

Рис. 5. Предупреждение о недопустимости применения inplace=True в цепочках методов

Fig. 5. Warning about not using inplace=True in method chains

Тем не менее при применении указанного метода и аргумента результаты работы библиотеки Pandas могут содержать ошибочные данные. Косвенно об этом утверждает предупреждение, содержащее информацию о том, что имеет место попытка изменения не исходного датафрейма, а его копии, что вызвано применением цепочки методов (рис 5).

Библиотека Pandas не всегда может гарантировать, что изменения, сделанные с помощью inplace=True, будут применены непосредственно к исходному датафрему, если перед этим была цепочка методов. Это связано с тем, что некоторые операции в Pandas создают копии датафреймов, а не изменяют исходный объект.

После проведения анализа наиболее подходящих для текущей задачи, математических аппаратов, выбор был сделан в пользу предикатов и аксиом (ПА). ПА позволяют формализовать датафреймы и inplace=True, в условиях, когда изучение строения библиотеки Pandas и ее механизмов избыточно. Кроме ПА были рассмотрены такие математические аппараты как реляционная алгебра на мультимножествах, теория переходов состояний, и монады.

Выбор в пользу предикатов и аксиом был обусловлен балансом между формальностью и практичностью, что сделало возможным частично формализовать поведение датафреймов и inplace=True.

Чтобы проиллюстрировать проблему inplace=True необходимо выйти за установленные пределы размеров публикации, однако с помощью формального аппарата предикатов и аксиом покажем методы, позволяющие обойти её.

Введем следующие предикаты:

IS_COPY(df,df′), (1)

где df и df′ – аргументы, представляющие датафрейм.

Предикат (1) истинен (True), если df является точной копией df′ и ложный (False) в противном случае.

IS_VIEW(df,df′). (2)

Предикат (2) истинен (True), если df_1 является представлением (view)df′, т.е. ссылается на те же данные в памяти, но возможно с другими метаданными (например, отличающимися набором или порядком столбцов).

HAS_VALUE(df,i,C,v), (3)

где i – номер строки; C – номер столбца; v– значение.

Предикат (3) истинен (True), если ячейка (i,C) в df содержит значение v.

IS_NAN(v). (4)

Предикат (4) истинен (True), если значение v не является числом.

IS_DATAFRAME(df). (5)

Предикат (5) истинен (True), если df является объектом типа датафрейм.

HAS_COLUMN(df,C). (6)

Предикат (6) истинен (True), если df имеет столбец с именем *C*.

Ниже представлены аксиомы, формализующие методы для получения корректных данных:

1. Прямое присваивание. Новый датафрейм может создаваться в зависимости от сложности исполняемых инструкций. Если создается потом обратно присваивается energy_0.

Аксиома 1 (Прямое присваивание столбца):

$$\forall df\in\{DataFrame\}\forall C\in\{ColumnName\}\forall f\in\{Function\}$$
$$(\ (IS\_DATAFRAME(df)\wedge HAS\_COLUMN(df,C)\wedge$$
$$\wedge(df:=\{MODIFY\_COLUMN(df,C,f):DataFrame\}\ ))\longrightarrow$$
$$\longrightarrow(IS\_DATAFRAME(df)\wedge HAS\_COLUMN(df,C)\wedge$$
$$\wedge\forall i\in\{RowIndex\}\exists v\in\{Value\}(HAS\_VALUE(df,i,C,v)\wedge$$
$$\wedge v=f(df\_old\{i,C\}\ ))\ ) \quad (7)$$

Обоснование аксиомы 1 (7):

Пусть df – произвольный объект датафрейм; *C* – произвольное имя столбца; *f* – функция, применяемая к каждому столбцу *C*. MODIFY_COLUMN(df,C,f)– функция, которая принимает (df), имя столбца (*C*) и функцию (f) и возвращает новый датафрейм, в котором значения столбца *C* модифицированы функцией *f*. Обозначение {...:DataFrame} указывает, что возвращаемое значение имеет тип датафрейм. Если df является датафреймом IS_DATAFRAME(df), df содержит столбец с именем C HAS_COLUMN(df,C), то присваиваем df результат функции MODIFY_COLUMN(df,C,f), который является новым датафреймом (df:= ={MODIFY_COLUMN(df,C,f): DataFrame}). Тогда df (после присваивания) также будет датафреймом IS_DATAFRAME(df) и df будет содержать столбец CHAS_COLUMN(df,C). Для каждой строки i в ∀i ∈ {RowIndex} существует значение v(∃v∈{Value}) такое, что верно утверждение HAS_VALUE(df,i,C,v), то есть ячейка в строке i и столбце C DataFrame df содержит значение v. Выполняется равенство v=f(df_old{i,C}, то есть значение v равно результату применения функции f к значению, которое было в ячейке i столбца C исходного датафрейма df_old.

Вывод: **Аксиома 1** формально утверждает, что функция MODIFY_COLUMN, примененная к df с столбцом C и функцией *f*, возвращает новый датафрейм, который сохраняет свою структуру.

2. Использование индексаторов объектов DataFrame в Pandas для доступа по меткам .loc или доступ по позициям .iloc. Новый датафрейм не создается.

Применение индексаторов целесообразно, если присутствует многоуровневая индексация. То есть, если у ваше-

го датафрейма есть многоуровневый индекс (MultiIndex), .loc позволяет выбирать данные по меткам на разных уровнях индекса. Используются булевы маски. Это создает более предсказуемое поведение, так как .loc работает с исходным датафреймом, а не создаёт копии.

Аксиома 2 (Модификация с помощью индексатора):

$$\forall df\in\{DataFrame\}\forall C\in\{ColumnName\}\forall i\in\{RowIndex\}\exists v\in\{Value\}$$
$$(\ (IS\_DATAFRAME(df)\wedge HAS\_COLUMN(df,C)\wedge$$
$$\wedge df.loc:,C:=v')\longrightarrow(HAS\_VALUE(df,i,C,v')\ \ ) \quad (8)$$

Обоснование аксиомы 2 (8):

Пусть df– произвольный объект датафрейм; C – произвольное имя столбца; i – произвольный индекс строки, а v'– произвольное новое значение. Если df является датафреймом IS_DATAFRAME(df), и df содержит столбец с именем C HAS_COLUMN(df,C). Присваиваем значение v' (которое существует во множестве Value) всем ячейкам в столбце C DataFramedfc помощью конструкции df.loc:,C:=v'. Тогда, для любого индекса строки i в df$\forall$i$\in${RowIndex} верно утверждение HAS_VALUE(df,i,C,v').

Вывод: **Аксиома 2** формально описывает поведение операции присваивания значения с помощью loc:,C в Pandas. Она утверждает, что после присваивания нового значенияv'всем ячейкам в столбце C, каждая ячейка в этом столбце (для любой строки) будет гарантированно содержать это новое значение.

Пояснение:

Кванторы $\forall$ (для любого) перед df, C и i означают, что аксиома верна для любых датафреймов, столбцов и индексов строк. Квантор $\exists$ (существует) перед v'означает, что мы предполагаем существование какого-то значения v', которое будет присвоено. df.loc:,C:=v' – это не стандартная математическая нотация, а адаптация синтаксиса Pandas для формального описания. Данная аксиома формализует конкретное поведение Pandas и может не иметь аналогов в других языках или библиотеках.

3. Метод fillna() с dict. Можно использовать метод fillna() со словарем, чтобы заполнить пропуски в нескольких столбцах за раз. Новый датафрейм создается, но потом обратно присваивается energy_0.

Данный способ заполнения пропущенных значений в нескольких столбцах Pandas считается наиболее эффективным из-за своей краткости.

Аксиома 3 (Заполнение пропусков с помощью словаря):

$$\forall df\in DataFrame\forall D\in\{Dict[ColumnName,Value]\ \ \}$$
$$(IS\_DATAFRAME(df)\wedge(df:=df.fillna(D))\ \ )\longrightarrow(IS\_DATAFRAME(df)\wedge$$
$$\wedge\forall C\in D.keys(\ \ )\forall i\in\{RowIndex\}$$
$$(IS\_NAN(df(\ \ [i,C]\ \ ))\longrightarrow HAS\_VALUE(df,i,C,D[C])\ \ )\wedge$$
$$\wedge(\neg IS\_NAN(df(\ \ [i,C]\ \ ))\longrightarrow HAS\_VALUE(df,i,C,dfi,C)\ \ )) \quad (9)$$

Обоснование аксиомы 3 (9):

Пусть df– произвольный объект датафрейм; *C* – произвольное имя столбца; *i* – произвольный индекс строки; df.fillna(D) – Функция Pandas, которая возвращает новый датафрейм, где значения NaN в столбцах, указанных в ключах словаря D, заменены соответствующими значениями из D', где D' – это копия словаря D.df:=– Оператор присваивания, обновляет df результатом выражения справа. Для любого df и любого словаря D, где ключи – это имена столбцов, а значения – заполнители ∀df∈DataFrame∀D∈{Dict[ColumnName, Value] }. Если df является датафреймом и мы применим к нему fillna(D) ((IS_DATAFRAME(df) ∧ ∧(df:=df.fillna(D))) тогда верно следующее (⟶)df останется датафреймом (IS_DATAFRAME(df)). Для каждого ключа *C*, присутствующего в словаре *D*, и для каждого индекса строки *i*(∀C∈D.keys( )∀i∈{RowIndex}). Если в исходном df значение в ячейке (i,C) было NaN (IS_NAN(df(i,C)), то в новом df эта ячейка будет содержать значение из словаря *D*, соответствующее столбцу *C* (D[C]). (⟶ HAS_VALUE(df,i,C,DC])). Если в исходном df значение в ячейке (i,C) не было NaN ((¬IS_NAN(df([i,C])), то в новом df эта ячейка сохранит свое исходное значение (⟶ HAS_VALUE(df,i,C,dfi,C])).

Вывод: **Аксиома 3** отражает поведение fillna(D) в Pandas, подчеркивая, что исходный словарь не изменяется, а для заполнения используется его копия.

4. Функция на Cython. Cython — это язык программирования, позволяющий писать код на Python с использованием статических типов, что дает возможность компилировать этот код в C-код, работающий гораздо быстрее интерпретируемого Python. Cython легко интегрируется с Pandas, позволяя использовать преимущества обоих инструментов.

Аксиома 4 (Модификация с помощью индексатора):

$$\begin{gathered}\forall df\in\{DataFrame\}\forall C\in\{ColumnName\}\forall vals\in\{ValueArray\}\\ (IS\_DATAFRAME(df)\wedge HAS\_COLUMN(df,C)\wedge\\ \wedge(df:=CYTHON\_FILLNA(df,C,vals)\ )\longrightarrow\\ \longrightarrow(IS\_DATAFRAME(df)\wedge HAS\_COLUMN(df,C)\wedge\\ \wedge\forall c\in C,\forall i\in\{RowIndex\},\exists v\in\{Value\}\\ (IS\_NAN(df\_oldi,c)\rightarrow HAS\_VALUE(df,i,c,vals[IndexOf(c,C)\ ])\ )\wedge\\ \wedge(\neg IS\_NAN(df\_oldi,c)\rightarrow HAS\_VALUE(df,i,c,df\_oldi,c)\ )\end{gathered}\qquad(10)$$

Обоснование аксиомы 4 (10):

Пусть df– произвольный объект датафрейм; *C* – произвольное имя столбца; *i*– произвольный индекс строки; *vals* – массив значений для замены NaN; ValueArray – массив значений для заполнения NaN, индексированный по порядку столбцов в *C*; CYTHON_FILLNA(df,C,vals) – функция Cython, которая заполняет значения NaN в столбцах *C* датафрейм df значе-

ниями из массива vals; IndexOf(c,C) – функция, которая возвращает индекс столбца c в множестве столбцов C; df_old – исходный датафрейм до применения CYTHON_FILLNA.

– Если df является датафреймом IS_DATAFRAME(df), df содержит все столбцы из множества C HAS_COLUMN(df, C), присваиваем df результат выполнения функции CYTHON_FILLNA(df,C,vals), которая возвращает новый датафрейм с заполненными NaN значениями. Тогда результирующий df также является датафреймом IS_DATAFRAME(df), df попрежнему содержит все столбцы из множества C в HAS_COLUMN(df,C), для каждого столбца c из C и для каждого индекса строки i:

– Если значение в ячейке i,c исходного df_old было NaN в IS_NAN(df_oldi,c, то в новом df эта ячейка будет содержать значение из массива vals, соответствующее столбцу c в HAS_VALUE(df,i,c,valsIndexOf(c,C)]).

– Если значение в ячейке i, c исходного df_old не было NaN IS_NAN(df_oldi,c]), то в новом df эта ячейка сохранит свое исходное значение HAS_VALUE(df,i,c, df_oldi,c]).

Вывод: **Аксиома 11** гарантирует, что структура датафрейма (тип и набор столбцов) сохраняется, пропущенные значения в указанных столбцах заменяются значениями из заданного массива, остальные значения остаются неизменными.

Результаты и их обсуждение

Тестовый стенд Intel совместимая платформа, пакет программного обеспечения Anaconda с Jupiter. Чтобы проверить скорость выполнения каждого из способов, можно воспользоваться инструментом %timeit в Jupyter Notebook. Код выполнит 7 запусков, каждый из которых будет состоять из числа циклов 10, 100, 1000 и 10000. Необходимо учитывать, что fillna() со словарем работает с датафреймом целиком, а не с отдельными столбцами (рис.6-9, табл. 4).

```
111 µs ± 2.67 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
108 µs ± 5.76 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
96.6 µs ± 17.7 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

Рис. 6. Результаты тестирования способа inplace=True

Fig. 6. Results of testing the inplace=True method

```
42.7 µs ± 1.51 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
43.2 µs ± 593 ns per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
44.6 µs ± 1.38 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

Рис. 7. Результаты тестирования способа с прямым присваиванием

Fig. 7. Test results of the direct assignment method

```
84.9 µs ± 448 ns per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
84 µs ± 378 ns per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
84.5 µs ± 529 ns per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

Рис. 8. Результаты тестирования способа с применением индексаторов

Fig. 8. Results of testing the method using indexers

```
933 µs ± 10.9 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

Рис. 9. Результаты тестирования способа с применением словаря

Fig. 9. Results of testing the method using a dictionary

Новый модуль на Cython выдает более низкие показатели производительности, видно, что они превосходят нативный код Pandas (табл. 4) по времени исполнения.

Скомпилируем и вызовем функцию из модуля Cython с включенным профилированием (рис. 10).

Результаты профилирования показывают, что большая часть времени выполнения кода тратится на операции, связанные с Pandas, а не на сам Cython код. Отсутствие информации о Cython функциях в результатах профилирования является основанием для вывода, что местом, где высоки накладные расходы, являются операции Pandas, а не Cython. Cython код выполняется настолько быстро, что профилировщик не замечает его вклада в общее время выполнения.

```
1053 function calls (1048 primitive calls) in 0.003 seconds

Ordered by: internal time
List reduced from 214 to 10 due to restriction <10>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     3    0.000    0.000    0.001    0.000 C:\Users\vector06c\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:2276(_merge_blocks)
     3    0.000    0.000    0.000    0.000 C:\Users\vector06c\anaconda3\Lib\site-packages\numpy\core\shape_base.py:219(vstack)
     8    0.000    0.000    0.000    0.000 {method 'copy' of 'numpy.ndarray' objects}
     6    0.000    0.000    0.000    0.000 C:\Users\vector06c\anaconda3\Lib\site-packages\pandas\core\internals\blocks.py:790(copy)
     9    0.000    0.000    0.001    0.000 C:\Users\vector06c\anaconda3\Lib\site-packages\pandas\core\frame.py:4062(__getitem__)
     6    0.000    0.000    0.000    0.000 C:\Users\vector06c\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1012(iget)
     2    0.000    0.000    0.000    0.000 {built-in method builtins.compile}
    24    0.000    0.000    0.000    0.000 C:\Users\vector06c\anaconda3\Lib\site-packages\pandas\core\generic.py:6301(__setattr__)
     1    0.000    0.000    0.001    0.001 C:\Users\vector06c\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1782(_consolidate_inplace)
     1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Рис. 10. Отчет функции профилирования

Fig. 10. Profiling function report

Прямое присваивание (df['col'] = df['col'].fillna(value)) – самый быстрый метод. Это наиболее эффективный способ заполнения пропущенных значений в Pandas, так как он напрямую изменяет столбец DataFrame. inplace=True: Хоть и показывает производительность, близкую к прямому присваиванию, его использование не рекомендуется из-за потенциальных проблем с непредсказуемым поведением, особенно в цепочках методов. Индексаторы (.loc): Этот метод медленнее прямого присваивания, так как он выполняет дополнительные проверки и операции, связанные с индексацией (рис.11).

Таблица 4. Сводные данные измерения производительности способов заполнения пропусков в Pandas

Table 4. Summary of performance measurements of gap filling methods in Pandas

Способ / Method	Количество циклов / Number of cycles	air_temperature (мкс)	Стд. откл. air_temperature (мкс)	cloud_coverage **(мкс)**	Стд. откл. cloud_coverage (мкс)	dew_temperature **(мкс)**	Стд. откл. dew_temperature (мкс)
inplace=True	10000	111	2.67	108	5,76	96,6	17,7
	1000	108	11.7	96,3	23	97	19,6
	100	129	7.71	81,6	34,3	45	1,67
	10	161	53.8	158	42	133	8,25
Прямое присваивание	10000	42.7	1.51	43,2	0,593	44,6	1,38
	1000	109	4.41	108	4,22	104	2,04
	100	112	5.17	108	4,36	110	25,8
	10	156	40.1	159	46,6	136	27,5
Индексаторы	10000	84.9	0.448	84	0,378	84,5	0,529
	1000	183	11.2	177	18,2	155	41,3
	100	213	29.4	192	5,36	187	4,77
	10	249	39.9	222	31	207	10,7
fillna() с словарем	10000	933	0	933	0	933	0
	1000	1900	29.7	1900	29.7	1900	29.7
	100	1820	126	1820	126	1820	126
	10	1840	199	1840	199	1840	199
Cython	10000	68100	5190	68100	5190	68100	5190
	1000	52100	1290	52100	1290	52100	1290
	100	49700	1290	49700	1290	49700	1290
	10	48700	12800	48700	12800	48700	12800

Вертикальные Т-образные линии, отходящие вверх и вниз от каждой точки на графике, созданном с помощью Matplotlib, представляют собой доверительный интервал или стандартное отклонение для каждой точки данных.

Модуль на Cython демонстрирует значительно более высокое время выполнения, чем fillna() со словарем, и все остальные методы Pandas. Это противоречит ожиданиям, так как Cython обычно используется для ускорения кода

Python. В данном случае, накладные расходы на преобразование данных между Pandas и Cython, а также вызов функций NumPy из Cython, нивелируют потенциальный выигрыш в производительности. fillna() с словарем, хоть и не самый быстрый метод Pandas, работает эффективнее реализации на Cython (рис. 12).

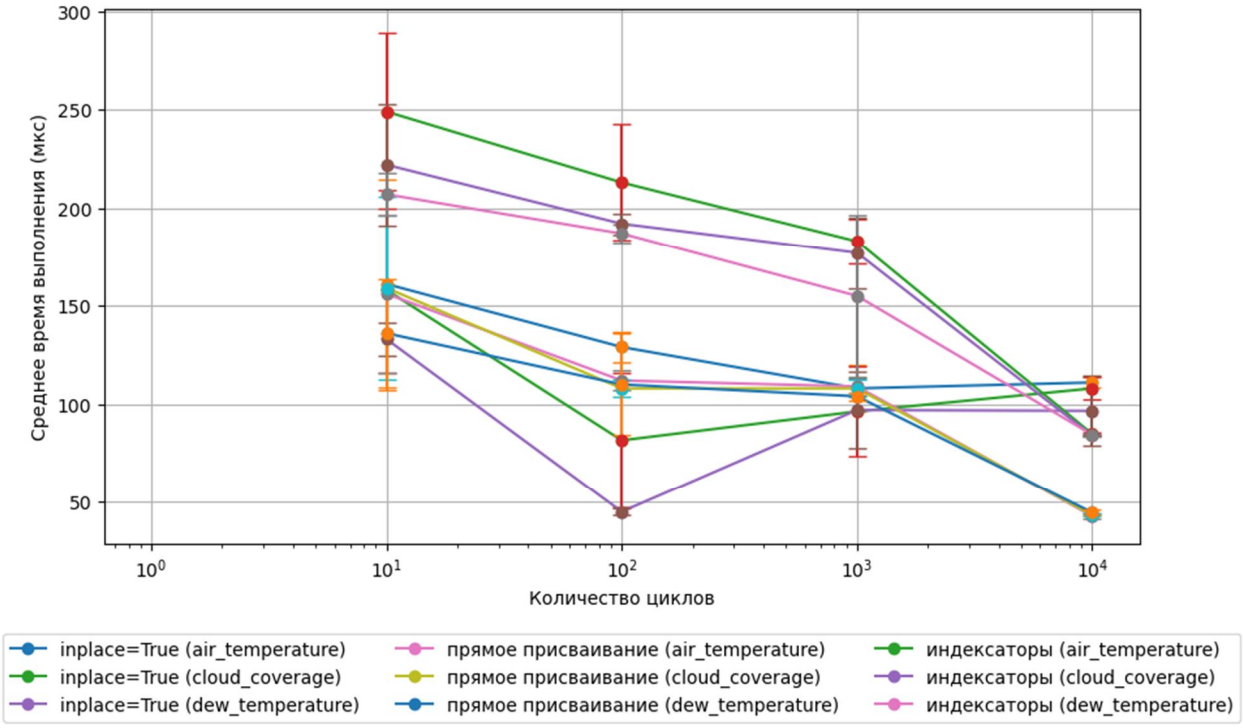

Рис. 11. Производительность основных методов Pandas

Fig. 11. Performance of the main Pandas methods

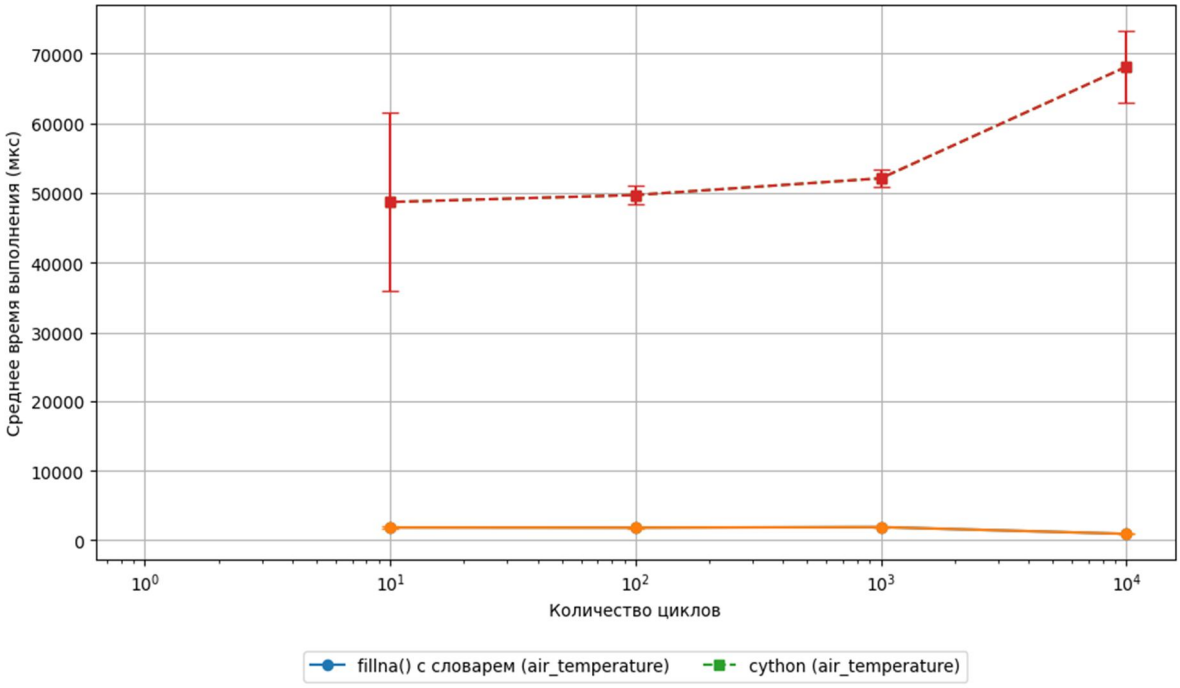

Рис. 12. Производительность Cython и fillna() с словарем

Fig. 12. Performance of Cython and fillna() with dictionary

Проведенный анализ производительности различных методов заполнения пропущенных значений в Pandas DataFrame показывает, что наиболее эффективным методом является прямое присваивание (df["column"] = df["column"].fillna(value)). Этот метод демонстрирует наименьшее среднее время выполнения одного цикла (~43 мкс) с минимальным стандартным отклонением (~1.5 мкс) для всех исследуемых столбцов (рис. 13).

В сравнении с методами Pandas, модуль, написанный на Cython, в значительной степени отстает по производительности (рис. 14).

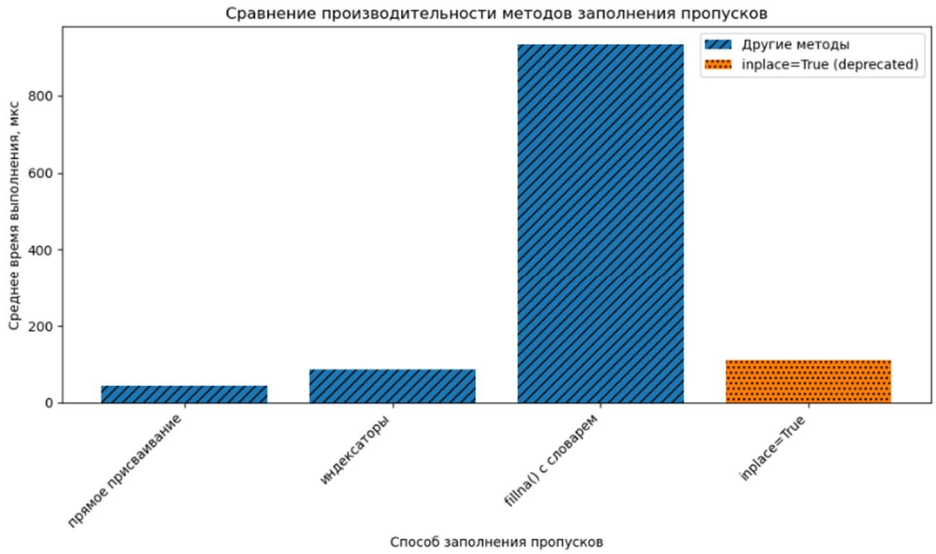

Рис. 13. Столбчатая диаграмма производительности способов замены пропущенных значений в Pandas

Fig. 13. Column chart showing the performance of missing value replacement methods in Pandas

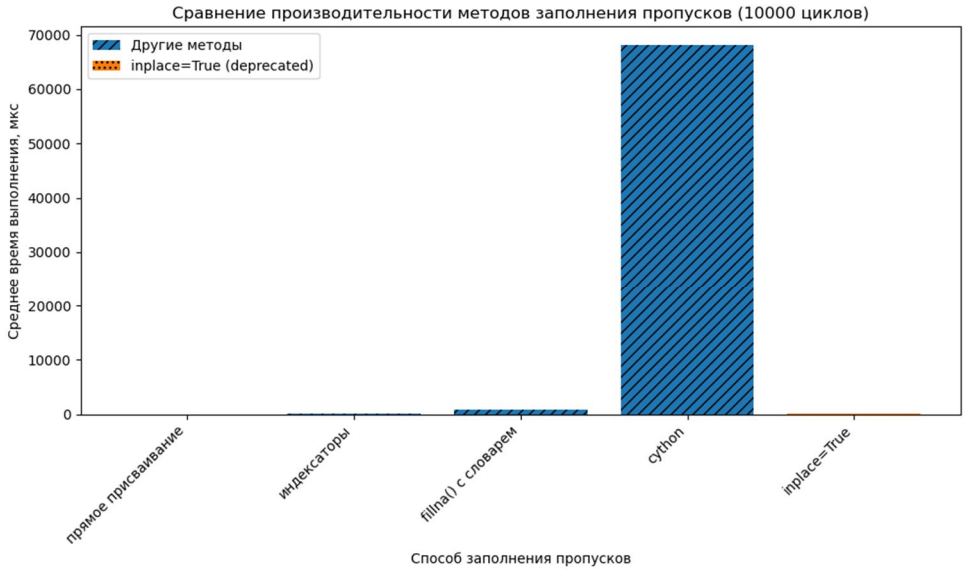

Рис. 14. Столбчатая диаграмма производительности способов замены пропущенных значений в Pandas в сравнении с модулем Cython

Fig. 14. Bar chart showing the performance of missing value replacement methods in Pandas compared to the Cython module

Выводы

Использование inplace=True хоть и показало сравнимую скорость (~100 мкс), но настоятельно не рекомендуется из-за его функциональных ограничений, и удаления в будущих версиях Pandas.

Применение индексаторов (.loc, .iloc) демонстрирует более низкую производительность (~84 мкс) по сравнению с прямым присваиванием. Это объясняется дополнительными проверками и операциями, которые Pandas выполняет при работе с индексаторами.

Метод fillna() со словарем, хоть и является более лаконичным при заполнении нескольких столбцов, демонстрирует наихудшую производительность (~933 мкс). Это объясняется созданием нового DataFrame с заполненными значениями и последующим присваиванием его исходной переменной.

Метод, реализованный в модуле на Cython, имеет на порядок большее время исполнения, чем все предыдущие (~68100 мкс). Как показало применение профилирования, код на Cython выполняется достаточно быстро, но ввиду обмена данными между модулем и Pandas растут накладные расходы и время работы увеличивается.

Список источников

1. Гончар А. А. Использование предиктивной аналитики для повышения эффективности бизнеса // Актуальные исследования. 2023. № 50-4(180). С. 22-46.

2. Громов Н. Д., Платошин А. И. Сравнительный анализ средств и платформ для автоматизации ETL-процессов в современных хранилищах данных // Международный журнал гуманитарных и естественных наук. 2023. № 11-4(86). С. 46-48. DOI 10.24412/2500-1000-2023-11-4-46-48.

3. Дрянкова Д. А. Визуализация данных с помощью библиотек Pandas и Matplotlib для языка программирования Python // Дневник науки. 2023. № 6(78). DOI 10.51691/2541-8327_2023_6_10.

4. Дьяконов Н. А., Логунова О. С. Системы управления технологическим процессом на основе предиктивной аналитики: проектирование // Электротехнические системы и комплексы. 2021. № 1(50). С. 58-64. DOI 10.18503/2311-8318-2021-1(50)-58-64.

5. Ильичев В. Ю., Юрик Е. А. Анализ массивов данных с использованием библиотеки Pandas для Python // Научное обозрение. Технические науки. 2020. № 4. С. 41-45.

6. Лескова В. Ю., Соловьев В. А. Анализ методов ETL // Наука и образование: актуальные исследования и разработки: сборник статей III Всероссийской научно-практической конференции, Чита, 29–30 апреля 2020 года. Чита: Забайкальский государственный университет, 2020. С. 36-40.

7. Носырева А. А., Абрамов В. И. Предиктивная аналитика - основа для цифровой трансформации компаний // Актуальные проблемы экономики, учета, аудита и анализа

в современных условиях: сборник научных статей Международной научно-практической конференции. Курск, 28–29 апреля 2021 года. Курск: Курский государственный университет, 2021. С. 179-182.

8. Соломонов А. А. Оптимизация ETL-процессов для больших данных // Вестник науки. 2024. Т. 3, № 9(78). С. 390-396.

9. Кисляков А. Н. Отбор признаков для использования в моделях предиктивной аналитики внешнеэкономической деятельности регионов // Прикладная математика и вопросы управления. 2022. № 1. С. 176-195. DOI 10.15593/2499-9873/2022.1.09.

10. Судариков Г. В., Ашмаров И. А. Использование библиотеки Pandas для анализа данных // Мир образования - образование в мире. 2023. № 1(89). С. 184-188. DOI 10.51944/20738536_2023_1_184.

11. Терешина В. В. Применение систем предиктивной аналитики и предикативного моделирования // Инновационное развитие экономики. 2022. № 5(71). С. 243-246. DOI 10.51832/2223798420225243.

12. Терентьева В. С., Логинова И. М., Эшелиоглу Р. И. Работа с датами в pandas // Научные исследования молодых ученых: материалы I Международной научно-практической конференции, посвященной памяти д.э.н., профессора Л.М. Рабиновича. Казань, 25–26 февраля 2022 года. Казань: Казанский государственный аграрный университет, 2022. Т. 2. С. 285-291.

13. Using SAP Predictive Analytics to Analyze Individual Student Profiles in LMS Moodle / A. N. Ambrajei, N. M. Golovin, A. V. Valyukhova, N. A. Rybakova // Communications in Computer and Information Science. 2022. Vol. 1539. P. 66-77. DOI 10.1007/978-3-030-95494-9_6.

14. Bushuev S. Application of AI for monitoring and optimizing IT infrastructure: economic prospects for implementing predictive analytics in enterprise operations // International Journal of Humanities and Natural Sciences. 2024. No. 8-3(95). P. 125-129. DOI 10.24412/2500-1000-2024-8-3-125-129.

15. Comparative Analysis of ETL Tools in Big Data Analytics / A. Qaiser, M. U. Farooq, S. M. Nabeel Mustafa, N. Abrar // Pakistan Journal of Engineering and Technology. 2023. Vol. 6, no. 1. P. 7-12. DOI 10.51846/vol6iss1pp7-12.

16. Singh, M. M. Extraction Transformation and Loading (ETL) of Data Using ETL Tools // International Journal for Research in Applied Science and Engineering Technology. 2022. Vol. 10, no. 6. P. 4415-4420. DOI 10.22214/ijraset.2022.44939.

17. Prepare and analyze taxation data using the Python Pandas library / M. Vagizov, A. Potapov, K. Konzhgoladze, et al. // IOP Conference Series: Earth and Environmental Science: 6, Politics, Industry, Science, Education. St. Petersburg; 2021. P. 1-8. DOI 10.1088/1755-1315/876/1/012078.

18. Godé C., Brion S. The affordance-actualization process of predictive analytics: Towards a configurational framework of a predictive policing system // Technological Forecasting and Social Change. 2024. Vol. 204. 123452 P. DOI 10.1016/j.techfore.2024.123452.

19. Identification of Critical States of Technological Processes Based on Predictive Analytics Methods / S. M. Kovalev, I. A. Olgeizer, A. V. Sukhanov, K. I. Kornienko // Automation and Remote Control. 2023. Vol. 84, no. 4. C. 424-433. DOI 10.1134/S0005117923040100.

20. Software Solution for the Implementation of a Predictive Analytics System for Investment Instruments / Н. А. Мамедова, О. В. Староверова, А. М. Епифанов [et al.] // WSEAS Transactions on Systems and Control. 2023. Vol. 18. P. 18-25. DOI 10.37394/23203.2022.18.2.

References

1. Gonchar A. A. Using predictive analytics to improve business efficiency. *Aktual`ny`e issledovaniya = Current research.* 2023; (50-4): 22-46 (In Russ.).

2. Gromov N. D. Comparative analysis of tools and platforms for automating ETL processes in modern data warehouses. *Mezhdunarodnyi zhurnal gumanitarnykh i estestvennykh nauk = International journal of humanitarian and natural sciences.* 2023; 11-4: 46-48. DOI 10.24412/2500-1000-2023-11-4-46-48 (In Russ.).

3. Dryankova D. A. Data visualization using Pandas and Matplotlib libraries for the Python programming language. *Dnevnik nauki = Science Diary.* 2023; 6. (In Russ.). DOI 10.51691 / 2541-8327_2023_6_10

4. Dyakonov N. A., Logunova O. S. Process control systems based on predictive analytics: design. *Elektrotekhnicheskie sistemy i kompleksy = Electrical systems and complexes.* 2021; (1): 58-64. (In Russ.). DOI 10.18503 / 2311-8318-2021-1 (50) -58-64

5. Ilyichev V. Yu., Yurik E. A. Analysis of data arrays using the Pandas library for Python. *Nauchnoe obozrenie. Texnicheskie nauki. = Scientific Review. Technical sciences.* 2020; (4): 41-45 (In Russ.).

6. Leskova V. Yu., Solov'ev V. A. Analysis of ETL methods. *In: Science and education: current research and development. Collection of articles of the III All-Russian scientific and practical conference, Chita, April 29-30, 2020. Chita: Zabaikal'skii gosudarstvennyi universitet;* 2020. P. 36-40 (In Russ.).

7. Nosyreva A. A., Abramov V. I. Predictive analytics - the basis for the digital transformation of companies. *In: Current problems of economics, accounting, auditing and analysis in modern conditions. Collection of scientific articles of the International Scientific and Practical Conference, Kursk, 28–29 April 2021.* Kursk: Kursk State University; 2021. P. 179-182 (In Russ.).

8. Solomonov A. A. Optimization of ETL processes for big data. *Vestnik nauki = Bulletin of Science.* 2024; 3(9): 390-396 (In Russ.).

9. Kislyakov A. N. Selection of features for use in predictive analytics models of foreign economic activity of regions. *Prikladnaya matematika i voprosy upravleniya = Applied Mathematics and Management Issues.* 2022; (1): 176-195. (In Russ.). DOI 10.15593/2499-9873/2022.1.09.

10. Sudarikov G. V., Ashmarov I. A. Using the Pandas library for data analysis. *Mir obrazovaniya - obrazovanie v mire = The world of education - education in the world.* 2023; (1): 184-188. (In Russ.). DOI 10.51944/20738536_2023_1_184

11. Tereshina V. V. Application of predictive analytics and predictive modeling systems. *Innovatsionnoe razvitie ekonomiki = Innovative development of the economy.* 2022; (5): 243-246. (In Russ.). DOI 10.51832/2223798420225243.

12. Terentyeva V. S., Loginova I. M., Eshelioglu R. I. Working with dates in pandas. In: *Scientific research of young scientists: Proceedings of the I International scientific and practical conference dedicated to the memory of Doctor of Economics, Professor L. M. Rabinovich, Kazan, February 25-26, 2022. Kazan: Kazanskii gosudarstvennyi agrarnyi universitet*. 2022; 2. P. 285-291 (In Russ.).

13. Ambrajei A. N., Golovin N. M., Valyukhova A. V., Rybakova N. A. Using SAP Predictive Analytics to Analyze Individual Student Profiles in LMS Moodle. *Communications in Computer and Information Science*, 2022; 1539: 66-77. DOI 10.1007/978-3-030-95494-9_6.

14. Bushuev S. Application of AI for monitoring and optimizing IT infrastructure: economic prospects for implementing predictive analytics in enterprise operations. *International Journal of Humanities and Natural Sciences.* 2024: (8-3): 125-129. DOI 10.24412/2500-1000-2024-8-3-125-129.

15. Qaiser A., Farooq M. U., Nabeel Mustafa S. M., Abrar N. Comparative Analysis of ETL Tools in Big Data Analytics. *Pakistan Journal of Engineering and Technology*. 2023; 6 (1): 7-12. DOI 10.51846/vol6iss1pp7-12.

16. Singh M. M. Extraction Transformation and Loading (ETL) of Data Using ETL Tools. *International Journal for Research in Applied Science and Engineering Technology*. 2022; 10(6): 4415-4420. DOI 10.22214/ijraset.2022.44939.

17. Vagizov M., Potapov A., Konzhgoladze K., et al. Prepare and analyze taxation data using the Python Pandas library. *IOP Conference Series: Earth and Environmental Science: 6, Politics, Industry, Science, Education*. St. Petersburg, May 26–28, 2021, St. Petersburg; 2021. P. 1-8. DOI 10.1088/1755-1315/876/1/012078.

18. Godé C., Brion S. The affordance-actualization process of predictive analytics: Towards a configurational framework of a predictive policing system. *Technological Forecasting and Social Change*. 2024; 204:123452. DOI 10.1016/j.techfore.2024.123452.

19. Kovalev S. M., Olgeizer I. A., Sukhanov A. V., Kornienko K. I. Identification of Critical States of Technological Processes Based on Predictive Analytics Methods. *Automation and Remote Control*. 2023; 84 (4): 424-433. DOI 10.1134/S0005117923040100.

20. Mamedova N. A., Staroverova O. V., Epifanov A. M., et al. Software Solution for the Implementation of a Predictive Analytics System for Investment Instruments. *WSEAS Transactions on Systems and Control*. 2023; 18: 18-25. DOI 10.37394/23203.2022.18.2.

Информация об авторах / Information about the Authors

Олейникова Алла Владимировна, аспирант, кафедра "Прикладная информатика", Астраханский государственный технический университет, г. Астрахань, Российская Федерация, e-mail: a.oleynikova.astu@mail.ru

Alla V. Oleynikova, Post-Graduate Stugent, Applied Informatics Department, Astrakhan State Technical University, Astrakhan, Russian Federation, e-mail: a.oleynikova.astu@mail.ru

Бондарева Ирина Олеговна, кандидат технических наук, доцент, завкафедрой "Прикладная информатика", Астраханский государственный технический университет, г. Астрахань, Российская Федерация, e-mail: orange8@mail.ru

Irina O. Bondareva, Cand. of Sci. (Engineering), Associate Professor, Head of Applied Informatics Department, Astrakhan State Technical University, Astrakhan, Russian Federation, e-mail: orange8@mail.ru

Олейников Александр Александрович, кандидат технических наук, доцент, кафедра "Прикладная информатика" Астраханский государственный технический университет, г. Астрахань, Российская Федерация, e-mail: ale15101338@yandex.ru

Aleksandr A. Oleynikov, Cand. of Sci. (Engineering), Associate Professor, Applied Informatics Department, Astrakhan State Technical University, Astrakhan, Russian Federation, e-mail: ale15101338@yandex.ru