Министерство науки и высшего образования Российской Федерации
Волжский филиал федерального государственного автономного образовательного учреждения высшего образования
«Волгоградский государственный университет»

Кафедра математики, информатики и естественных наук

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА НА ТЕМУ

«АНАЛИЗ ДАННЫХ ДАТЧИКОВ ЭНЕРГЕТИЧЕСКОГО ОБОРУДОВАНИЯ»

по направлению подготовки
01.03.02 Прикладная математика и информатика
профиль «Математическое и информационное обеспечение экономической деятельности»

ВЫПОЛНИЛ:
студент гр. ПМИ-201
Кереселидзе Давид Кобаевич
_____

НАУЧНЫЙ РУКОВОДИТЕЛЬ:
зав. кафедрой МИиЕН
канд. физ.-мат. наук, доцент
Полковников Александр Александрович
_____

РАБОТА ДОПУЩЕНА К ЗАЩИТЕ:
зав. кафедрой МИиЕН
канд. физ.-мат. наук, доцент
Полковников Александр Александрович
_____
«3» июня 2024 г.
(протокол № 6 заседания кафедры)

Волжский - 2024

Министерство науки и высшего образования Российской Федерации
Волжский филиал федерального государственного автономного образовательного учреждения высшего образования
«Волгоградский государственный университет»
Кафедра математики, информатики и естественных наук

«Утверждаю»
Заведующий кафедрой МИиЕН
канд. физ.-мат. наук, доцент
Полковников Александр Александрович
_____
«___» _____ 2023 г.

ЗАДАНИЕ
На выполнение выпускной квалификационной работы бакалавра
Студента Кереселидзе Давида Кобаевича группы ПМИ-201

Тема. Анализ данных датчиков энергетического оборудования
Цель. Разработка моделей прогнозирования и оптимизации работы энергетического оборудования. Оценка качества моделей.
Основные задачи:

1) Анализ временных рядов данных температуры трансформаторов.
2) Построение моделей прогнозирования температуры на основе статистических методов и машинного обучения;
3) Выявление закономерностей и тенденций изменения температуры;
4) Выявление аномальных участков;
5) Оценка качества построенных моделей и выбор наилучшей модели для прогнозирования температуры.

Основные этапы выполнения работы:

1) Изучение методов обработки и анализа временных рядов.
2) Обработка данных, применение статистических методов анализа.
3) Применение методов машинного обучения для анализа временных рядов.
4) Анализ полученных результатов, сравнение моделей.

Рекомендуемая литература:

1. Марджанов, А. Введение в статистический анализ данных. — СПб.: БХВ-Петербург, 2010
2. Лукишн, Л. Л. Анализ временных рядов: учеб. пособие. — М.: ИНФРА-М, 2002.
3. Трофимов, И.В., Петрушин, В.И., Попов, Л.В. Анализ данных в энергетике. — М.: Энергоатомиздат, 1987.

Дата выдачи 01.09.2022г.
Срок выполнения 05.06.2023г.
Руководитель _____ канд. физ.-мат. наук, доцент Полковников А.А.
Задание принял _____Кереселидзе Д.К.

Содержание

Введение **4**

Глава 1. Основные компоненты и характеристики силового трансформатора **6**

1.1. Принцип работы силового трансформатора 7

1.2. Преобразование напряжения 8

1.3. Значение и применение силовых трансформаторов . . . 8

1.4. Способы охлаждения трансформаторов 9

1.5. Режимы работы трансформатора 9

Глава 2. Модели анализа и прогнозирования временных рядов 11

2.1. Понятие временного ряда 11

2.2. Компоненты временного ряда 11

2.3. Методы анализа временных рядов 12

Глава 3. Реализация моделей для анализа данных 16

3.1. Ознакомление с данными. Предобработка. Выбор моделей. 16

3.2. Использование модели CatBoost для прогнозирования временных рядов. 23

3.3. Модель Prophet. 27

3.3. Модель SARIMA. 30

3.4. Модель LSTM. 32

Заключение. **36**

Введение

С развитием технологий и увеличением объемов энергопотребления актуальность эффективного управления и мониторинга состояния энергетического оборудования значительно возросла [11, 5]. В условиях растущих требований к надежности энергоснабжения особое внимание уделяется прогнозированию работы и состояния трансформаторов, которые играют ключевую роль в поддержании энергоснабжения. Прогнозирование температуры внутри трансформатора является одним из важных аспектов этой задачи, поскольку высокая температура может привести к сбоям и повреждениям оборудования, что, в конечном итоге, приводит к перебоям в энергоснабжении [13, 3].

В данной работе рассматривается проблема прогнозирования нагрузок трансформатора с целью более точного контроля и предотвращения возможных аварийных ситуаций. Анализируются данные, собранные с датчиков, установленных на трансформаторе в жилом районе Волгограда. Эти данные включают информацию о времени, мощности, температуре снаружи и внутри трансформатора.

Целью данного исследования является построение моделей прогнозирования мощности трансформатора, которые могут помочь операторам сети в эффективном управлении оборудованием и предотвращении возможных неполадок. Для достижения этой цели будет проведен анализ данных, выбраны подходящие модели и методы прогнозирования, а затем проанализированы результаты их применения.

Современные методы машинного обучения и анализа данных предоставляют широкий спектр инструментов для решения задачи прогнозирования [6]. Важной частью данной работы является выбор наиболее эффективных алгоритмов и их адаптация под конкретные условия эксплуатации трансформаторов. Методы, такие как регрессионный анализ, нейронные сети, временные ряды и другие подходы, будут рассмотрены и оценены в контексте их применимости и точности предсказаний [7].

В рамках данной квалификационной работы выполнен анализ данных датчиков трансформатора в г. Волгограде. Были проанализированы временные ряды, содержащие технические данные с датчиков трансформа-

тора. Была найдена зависимость температуры внутри трансформатора от мощности, а также построены модели прогнозирования для временного ряда мощности на трансформаторе.

Результаты данного исследования могут быть применены не только в Волгограде, но и в других регионах с аналогичными условиями эксплуатации. Таким образом, разработанные методы и модели могут стать основой для создания универсальных решений по мониторингу и управлению энергетическим оборудованием на уровне всей страны.

Глава 1. Основные компоненты и характеристики силового трансформатора

Силовой трансформатор является одним из ключевых элементов в системах передачи и распределения электроэнергии. Он преобразует напряжение и ток для оптимальной передачи энергии на большие расстояния и обеспечения стабильной работы конечных потребителей. Основные компоненты и характеристики силового трансформатора включают:

1. Обмотки: Силовой трансформатор имеет две или более обмотки провода:

    - Первичная обмотка: Это катушка провода, к которой подключается высоковольтный источник электричества.
    - Вторичная обмотка: Это другая катушка провода, от которой электричество подается к потребителям.

2. Магнитопровод: Магнитопровод представляет собой ферромагнитный материал (например, сердечник), который служит для направления магнитного потока от одной обмотки к другой. Качество магнитопровода влияет на эффективность трансформатора. Сердечник помогает передавать магнитное поле от первичной обмотки ко вторичной.

3. Охлаждение: Силовые трансформаторы могут быть охлаждаемыми различными способами, например, естественным воздушным охлаждением, принудительным воздушным охлаждением, масляным охлаждением. Это важно для поддержания нормальной температуры работы трансформатора и предотвращения перегрева.

4. Железопорошковые или масляные герметичные баки: Их применение может обеспечивать защиту от атмосферных воздействий и минимизировать воздействие окружающей среды на трансформатор.

5. Реле и защитные устройства: Для обеспечения безопасной и стабильной работы трансформатора устанавливаются различные реле и защитные устройства, которые могут отключить трансформатор в случае возникновения неполадок.

6. Изоляция: Изоляция обмоток и других элементов трансформатора предотвращает короткое замыкание и обеспечивает безопасную эксплуатацию.

Рис. 1. Компоненты силового трансформатора.

1.1. Принцип работы силового трансформатора

Силовые трансформаторы преобразуют напряжение с помощью электромагнитной индукции, которая происходит следующим образом:

1. Подача напряжения на первичную обмотку: Когда трансформатор подключен к источнику переменного тока, по первичной обмотке начинает протекать переменный ток.

2. Создание магнитного поля: Переменный ток в первичной обмотке создает переменное магнитное поле вокруг обмотки.

3. Передача магнитного поля через сердечник: Это магнитное поле передается через сердечник трансформатора к вторичной обмотке.

4. Индукция напряжения во вторичной обмотке: Переменное магнитное поле индуцирует (создает) напряжение во вторичной обмотке.

5. Подача электричества к потребителям: Это индуцированное напряжение используется для подачи электричества к домам, предприятиям и другим потребителям.

1.2. Преобразование напряжения

Силовые трансформаторы могут быть понижающими или повышающими:

- Понижающий трансформатор: Если на первичной обмотке больше витков провода, чем на вторичной, трансформатор понижает напряжение (например, с 11000 Вольт до 220 Вольт для бытового использования).

- Повышающий трансформатор: Если на первичной обмотке меньше витков провода, чем на вторичной, трансформатор повышает напряжение (например, с 220 Вольт до 11000 Вольт для передачи на большие расстояния).

1.3. Значение и применение силовых трансформаторов

Силовые трансформаторы играют ключевую роль в электроснабжении:

- Передача электроэнергии: Они повышают напряжение для передачи электроэнергии на большие расстояния, что снижает потери энергии.

- Распределение электроэнергии: Они понижают напряжение до безопасного уровня перед подачей к домам и предприятиям.

Пример: Если на первичной обмотке 1000 витков, а на вторичной 100 витков, то при подаче 10000 Вольт на первичную обмотку, на вторичной будет 1000 Вольт, потому что витков в 10 раз меньше.

1.4. Способы охлаждения трансформаторов

Для предотвращения перегрева трансформаторы охлаждаются разными способами:

1. Естественное воздушное охлаждение (AN): Используется в маломощных трансформаторах.

2. Принудительное воздушное охлаждение (AF): Используется вентиляторами для более эффективного охлаждения.

3. Масляное охлаждение (ONAN, ONAF): Трансформатор погружен в масло, которое отводит тепло от обмоток и сердечника. Масло циркулирует естественным образом или с помощью насосов.

1.5. Режимы работы трансформатора

Силовой трансформатор может работать в трех режимах:

1. В состоянии холостого хода: Когда цепь вторичной обмотки разомкнута и ток не проходит.

2. В режиме нагрузки: Когда к второй обмотке подключена номинальная нагрузка, и по обмоткам циркулирует рабочий ток.

3. В короткозамкнутом режиме: Когда вторичная обмотка замкнута накоротко, и вся мощность концентрируется в обмотках, что используется для определения потерь и нагревания проводов.

С развитием технологий в силовых трансформаторах происходят значительные изменения. Новые материалы для магнитопровода, улучшенные

системы охлаждения, а также внедрение цифровых технологий для мониторинга и управления позволяют создавать более эффективные и надежные устройства. Современные требования к энергоэффективности, устойчивости к перегрузкам и снижению потерь стимулируют исследования в области силовых трансформаторов. Развитие новых материалов, улучшенные конструкции и инновационные методы управления обещают привести к созданию более совершенных и эффективных силовых трансформаторов.

Глава 2. Модели анализа и прогнозирования временных рядов

2.1. Понятие временного ряда

Временной ряд представляет собой последовательность значений некоторой переменной, наблюдаемой через равные промежутки времени. Примеры временных рядов включают ежедневные температуры, ежемесячные продажи товаров, ежегодные уровни производства и так далее. Временные ряды обладают важным свойством — автокорреляцией, то есть значением переменной в один момент времени может зависеть от её значений в предыдущие моменты времени.

Анализ временных рядов включает в себя изучение их структуры, выявление основных компонентов (таких как тренд, сезонность и остатки), а также построение моделей для прогнозирования будущих значений на основе исторических данных.

2.2. Компоненты временного ряда

Временные ряды часто рассматриваются как состоящие из нескольких основных компонентов:

- Тренд: Долгосрочное направление движения временного ряда. Например, рост или спад продаж компании за несколько лет.
- Сезонность: Регулярные колебания, повторяющиеся с определённой периодичностью, например, ежемесячные или ежегодные.
- Циклы: Колебания, не имеющие фиксированной периодичности, но характеризующиеся определённой регулярностью, например, экономические циклы.
- Случайные колебания (ошибки или шум): Непредсказуемые и случайные колебания, которые не могут быть объяснены трендом или сезонностью.

2.3. Методы анализа временных рядов

Анализ временных рядов включает в себя широкий спектр методов, предназначенных для изучения структурных характеристик данных и построения моделей, позволяющих прогнозировать будущие значения. Эти методы можно разделить на несколько категорий: методы сглаживания, авторегрессионные модели, современные методы машинного обучения и комбинированные подходы.

Методы сглаживания

Методы сглаживания применяются для устранения шумов и выделения основных тенденций в данных. Они позволяют упростить временные ряды, делая их более пригодными для дальнейшего анализа и моделирования.

Метод скользящего среднего включает вычисление среднего значения набора последовательных точек данных и его использование в качестве представления исходной последовательности. Скользящее среднее помогает сгладить колебания и выявить основные тренды. Математически скользящее среднее можно записать следующим образом:

$$\hat{y}_t = \frac{1}{n} \sum_{i=0}^{n-1} y_{t-i},$$

где $\hat{y}_t$ — сглаженное значение в момент времени t, n — размер окна, y_{t-i} — наблюдаемое значение временного ряда в момент времени $t - i$.

В экспоненциальном сглаживании более свежие данные имеют больший вес в расчетах, чем старые. Формула простого экспоненциального сглаживания выглядит так:

$$\hat{y}_t = \alpha y_t + (1 - \alpha)\hat{y}_{t-1},$$

где α — коэффициент сглаживания ($0 < \alpha < 1$), $\hat{y}_t$ — сглаженное значение в момент времени t, y_t — наблюдаемое значение в момент времени t.

Автокорреляционные модели

Модель ARIMA (Автокорреляционная интегрированная модель скользящего среднего) объединяет три компонента:

Авторегрессия (AR) использует зависимость между текущим значением временного ряда и его предыдущими значениями. Формально авторегрессионный процесс порядка p (AR(p)) записывается как:

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t,$$

где $\phi_1, \phi_2, \ldots, \phi_p$ — параметры модели, ε_t — белый шум.

Интеграция (I) применяется для устранения нестационарности путем взятием разностей некоторого порядка от исходного временного ряда. Если d — порядок интегрирования (порядок разностей исходного временного ряда), то интегрированный процесс порядка d (I(d)) можно записать как:

$$y'_t = \Delta^d y_t,$$

где Δ — оператор разности.

Скользящее среднее (MA) учитывает зависимость между текущим значением временного ряда и предыдущими ошибками прогноза. Скользящее среднее порядка q (MA(q)) описывается уравнением:

$$y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q},$$

где $\theta_1, \theta_2, \ldots, \theta_q$ — параметры модели, ε_t — белый шум.

Модель ARIMA объединяет эти компоненты в одну модель:

$$y_t = \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q},$$

SARIMA (Сезонная ARIMA) расширяет ARIMA за счет включения сезонных компонентов, что позволяет учитывать регулярные сезонные колебания в данных.

Формально SARIMA(p,d,q)(P,D,Q)[s] определяется как:

$$\Phi_P(B^s)\phi_p(B)\Delta^d\Delta_s^D y_t = \Theta_Q(B^s)\theta_q(B)\varepsilon_t,$$

где Φ_P и Θ_Q — сезонные авторегрессионные и скользящие средние полиномы, ϕ_p и θ_q — обычные авторегрессионные и скользящие средние полиномы, B — оператор сдвига.

Современные методы машинного обучения

Современные методы машинного обучения предлагают широкий спектр инструментов для анализа временных рядов, которые могут обрабатывать сложные и нелинейные зависимости в данных.

Prophet, разработанный Facebook, использует аддитивную модель, учитывающую тренды, сезонность и праздничные эффекты [19]. Этот инструмент особенно полезен для данных с пропусками и большими объемами. Аддитивная модель Prophet выглядит так:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t,$$

где $g(t)$ — модель тренда, $s(t)$ — сезонная компонента, $h(t)$ — праздничные эффекты, ε_t — ошибка.

CatBoost, алгоритм градиентного бустинга от компании Yandex, способен эффективно работать с временными рядами. CatBoost обрабатывает категориальные переменные и использует кросс-валидацию для предотвращения переобучения.

Рекуррентные нейронные сети (RNN) и их вариации (LSTM, GRU) учитывают последовательную природу временных рядов и могут захватывать долгосрочные зависимости. Модель LSTM (Long Short-Term Memory) состоит из ячеек, каждая из которых имеет вход, забывание и выходные ворота, что позволяет сохранять долгосрочную информацию:

$$\begin{aligned}
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i), \\
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f), \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o), \\
C_t &= f_t * C_{t-1} + i_t * \tanh(W_C[h_{t-1}, x_t] + b_C), \\
h_t &= o_t * \tanh(C_t),
\end{aligned}$$

где i_t, f_t, o_t — входные, забывчивые и выходные ворота соответственно, C_t — состояние ячейки, h_t — скрытое состояние, σ — сигмоидная функция активации, tanh — гиперболический тангенс, W и b — веса и смещения.

Комбинированные подходы

Комбинированные подходы используют преимущества различных методов для улучшения качества прогнозирования.

Гибридные модели сочетают статистические методы, такие как ARIMA, с моделями машинного обучения, например, нейронными сетями. Это позволяет улавливать как линейные, так и нелинейные зависимости в данных.

Энсемблирование объединяет прогнозы нескольких моделей для получения более точного и надежного результата. Примеры включают bagging, boosting и stacking.

Эти методы и модели позволяют комплексно подходить к анализу временных рядов, учитывая как структурные компоненты, так и нелинейные зависимости, что существенно повышает точность прогнозирования и качество управления временными рядами.

Глава 3. Реализация моделей для анализа данных

3.1. Ознакомление с данными. Предобработка. Выбор моделей.

Для выбора моделей нужно сначала проанализировать исходные данные. Для этого загрузим их в Python с помощью библиотеки pandas [8].

```
data2022 = pd.read_excel('data2022.xlsx')
data2023 = pd.read_excel('data2023.xlsx')

data = pd.concat([data2022, data2023], ignore_index=True)
data
```

	время	мощность	т снаружи	т внутри
0	01.01.2022 01:00	3965.5664	2.9	22.727832
1	01.01.2022 02:00	4119.6080	2.9	23.498040
2	01.01.2022 03:00	3778.6208	2.9	21.793104
3	01.01.2022 04:00	3324.4416	2.9	19.522208
4	01.01.2022 05:00	2745.3968	2.9	16.626984
...	...	...	...	...
17514	31.12.2023 19:00	4968.8928	-6.2	18.644464
17515	31.12.2023 20:00	4412.5168	-6.1	15.962584
17516	31.12.2023 21:00	4921.7376	-6.1	18.508688
17517	31.12.2023 22:00	4232.2240	-6.1	15.061120
17518	31.12.2023 23:00	3333.9776	-6.0	10.669888

17519 rows × 4 columns

Рис. 2. Компоненты силового трансформатора.

- *время* — Дата и время измерений.
- *мощность* — Мощность, потребляемая трансформатором, измеренная в киловаттах (кВт). Пороговое значение мощности составляет 15000 кВт. Если мощность превышает этот порог, это может указывать на перегрузку трансформатора.
- *т снаружи* — Температура наружного воздуха, измеренная в градусах Цельсия (°C).
- *т внутри* — Температура внутри трансформатора, измеренная в градусах Цельсия (°C). Пороговое значение внутренней температуры составляет 80°C. Превышение этого порога может привести к повреждению оборудования и перебоям в энергоснабжении.

Далее следует проверить, есть ли в данных NaN — пробелы в данных. Такие пробелы могут привести к некачественной работе многих моделей, а модели, работающие с временными рядами, могут не запуститься вовсе. Воспользуемся методом .info() библиотеки pandas для класса DataFrame [15]:

```
data.info()
```
✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8759 entries, 0 to 8758
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   время      8759 non-null   object
 1   мощность   8759 non-null   float64
 2   т снаружи  8759 non-null   float64
 3   т внутри   8759 non-null   float64
dtypes: float64(3), object(1)
memory usage: 273.8+ KB
```

Рис. 3. Вызов метода .info()

Как видно из вывода, данные не содержат пропусков (null). Теперь можно построить график и оценить их визуально. Для этого использован метод plot [16].

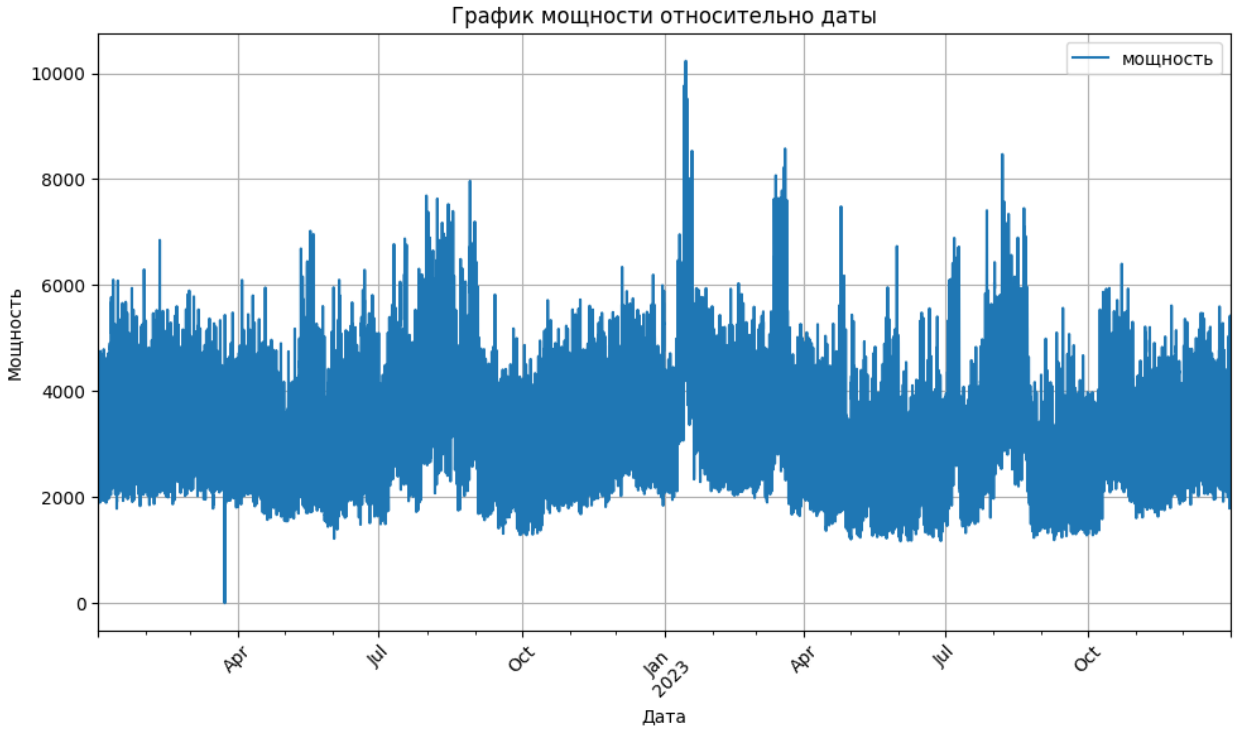

Рис. 4. График мощности за два года.

Из-за высокой плотности данных их сложно анализировать визуально. Чтобы определить сезонность и тренд, можно воспользоваться библиотекой Prophet [4] и методом ".plot_components()". Чтобы это сделать, необходимо сначала обучить модель.

```
from prophet import Prophet

# Преобразуем данные в формат, понятный Prophet
data_prophet = data.rename(columns={'время': 'ds', 'мощность': 'y'})

# Создаем и обучаем модель Prophet
model = Prophet()
model.fit(data_prophet)
```

Рис. 5. Код для обучения модели.

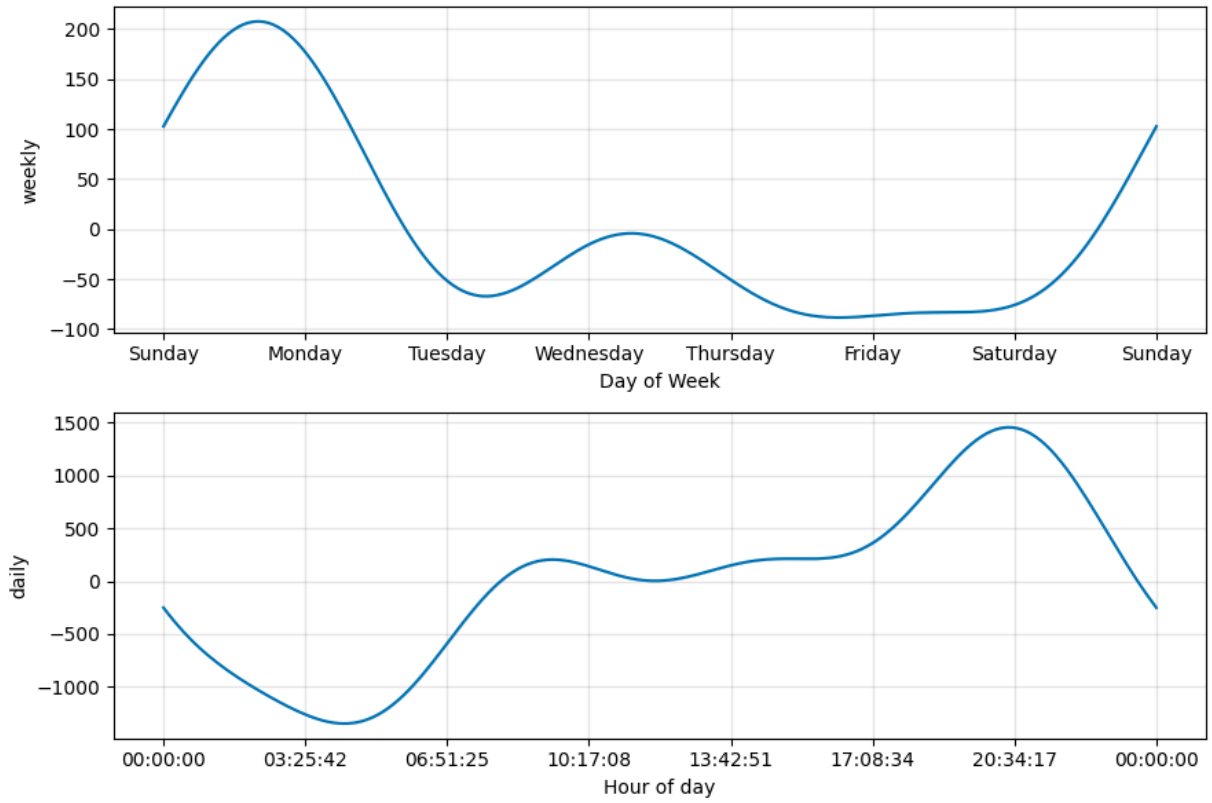

Рис. 6. Дневная и недельная сезонности.

Следует также оценить сезонность по месяцам. Для этого рассчитаем среднее значение для каждого из них.

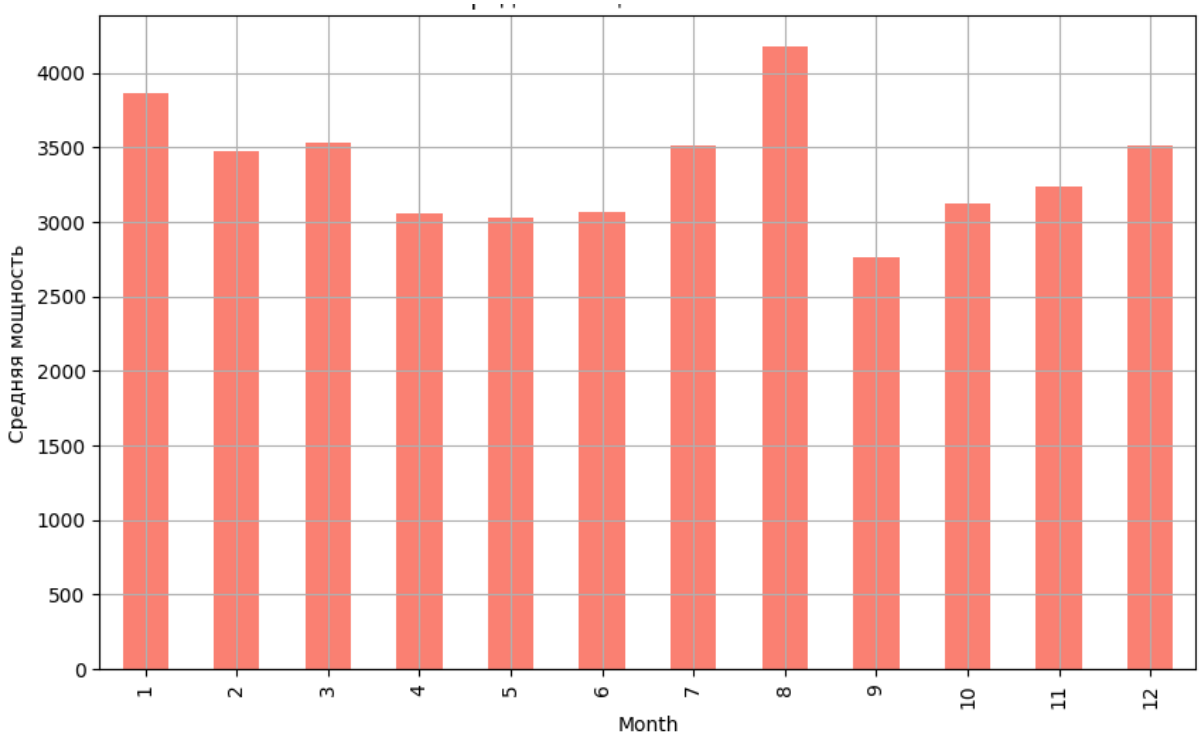

Рис. 7. Средняя мощность по месяцам.

Анализ данных выявил наличие сезонности. Существуют методы работы с временными рядами, учитывающие сезонность, такие как Prophet и SARIMA [2]. Также заметна категоризация временных признаков (например, в воскресенье мощность выше чем в любой другой день недели, а в августе — больше, чем в любом другом месяце), что можно использовать в таких моделях, как CatBoost [9].

Чтобы более глубоко понять структуру временного ряда и выявить внутренние зависимости, было решено провести анализ автокорреляции (ACF) и частичной автокорреляции (PACF). Эти методы позволяют оценить, насколько текущие значения временного ряда зависят от его прошлых значений, что является критически важным для построения моделей, учитывающих временные зависимости.

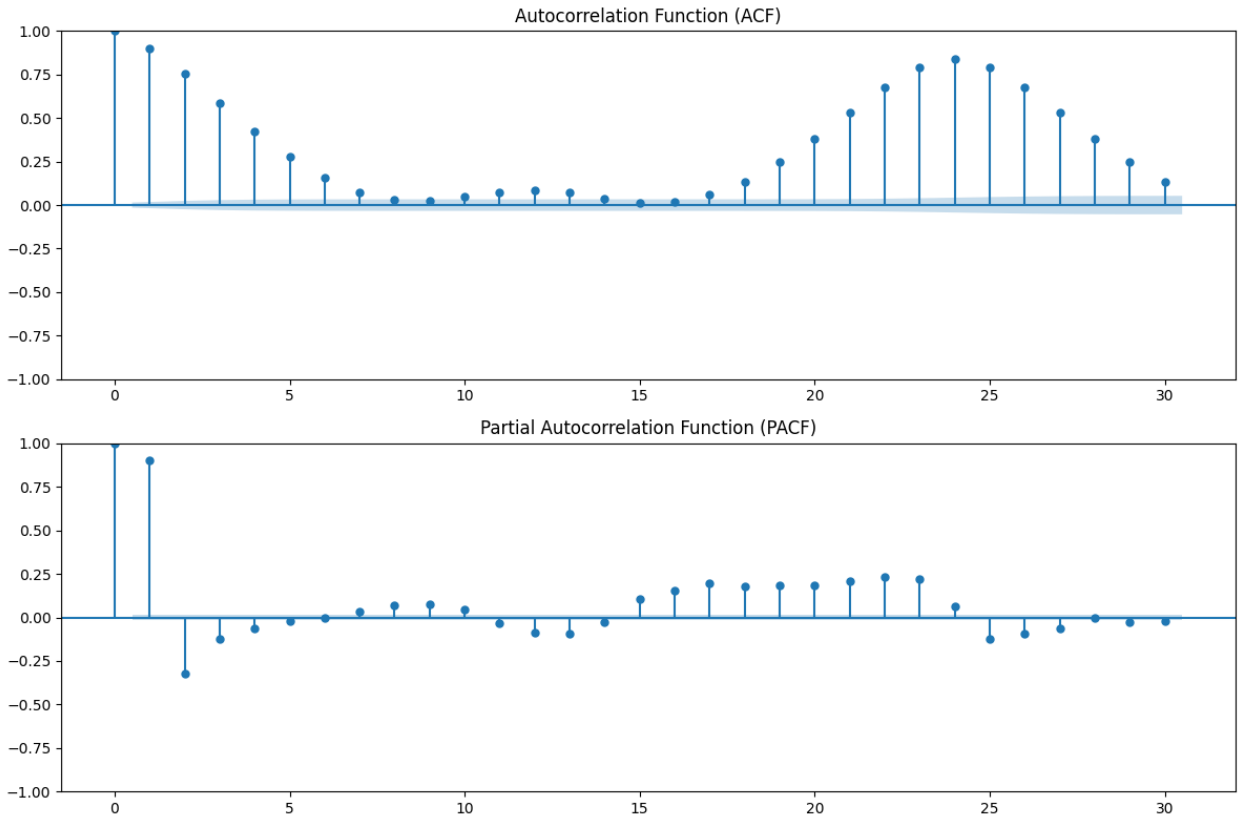

Рис. 8. Графики автокорреляции (ACF) и частичной автокорреляции (PACF).

На рисунке 8 представлены графики функций автокорреляции (ACF) и частичной автокорреляции (PACF) для анализируемого временного ряда. Давайте рассмотрим каждый график более подробно.

График автокорреляции (ACF) показывает, насколько значения временного ряда связаны с его прошлыми значениями на различных временных интервалах (лагах). Горизонтальная ось этого графика отображает количество шагов задержки (лагов), на которые текущие значения временного ряда отстают от предыдущих значений. Вертикальная ось отображает коэффициент автокорреляции, который измеряет степень связи между значением временного ряда и его предыдущими значениями на определенных лагах.

Из графика ACF видно, что на низких лагах (1-4) наблюдаются сильные положительные корреляции, что указывает на значительную зависимость

текущих значений ряда от недавних прошлых значений. Это означает, что в данных присутствуют выраженные краткосрочные зависимости.

Кроме того, на графике ACF наблюдаются значительные пики около лагов 20-26, что может указывать на наличие сезонных эффектов с периодом около 24 шагов, что логично вписывается в суточную сезонную активность человека.

График частичной автокорреляции (PACF) показывает, насколько текущие значения временного ряда зависят от его прошлых значений на определенных лагах, исключая влияние промежуточных значений. Горизонтальная ось графика PACF также отображает количество шагов задержки, на которые текущие значения временного ряда отстают от предыдущих. Вертикальная ось отображает коэффициент частичной автокорреляции, который измеряет чистую корреляцию между текущими значениями и их прошлыми значениями, исключая влияние всех промежуточных лагов.

На графике PACF можно заметить значительную положительную частичную автокорреляцию на первом лаге, что свидетельствует о сильной прямой зависимости текущих значений от предыдущих. Однако на последующих лагах частичная автокорреляция быстро уменьшается и становится незначительной. Это указывает на то, что после учета первых нескольких лагов дальнейшие значения имеют слабое влияние. Это характерно для временных рядов с короткими зависимостями, где основные влияния наблюдаются на начальных лагах.

Анализ ACF и PACF помогает глубже понять временные зависимости и сезонные компоненты в данных. Например, теперь можно быть точно уверенным в дневной сезонности данных, что поможет нам в дальнейшем.

Прежде чем строить модели для прогнозирования временных рядов, стоит учесть связь между данными. Например, полезно будет знать температуру внутри трансформатора. Имея данные о мощности трансформатора и температуре снаружи, можно определить наличие зависимости.

Предположим линейную зависимость между данными. Для проверки предположения воспользуемся моделью линейной регрессии библиотеки sklearn [10]. В качестве x будут данные мощности и температуры снару-

жи трансформатора, в качестве y - данные температуры внутри. Обучим модель на 80% выборки, оставив 20% под валидацию (проверку).

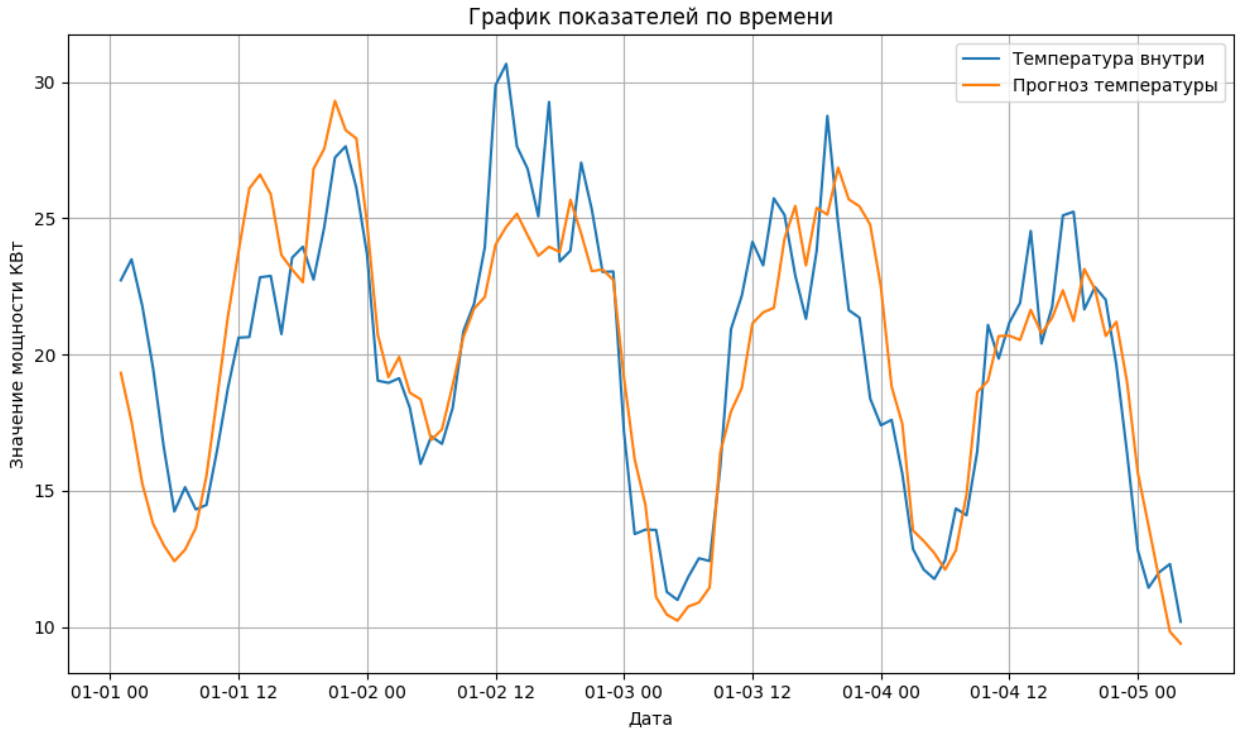

Рис. 9. График температуры внутри трансформатора и предсказанной.

Ошибка по метрике MAPE (Mean Absolute Percentage Error) [1] составила 2%, что подтверждает наличие зависимости.

Для обеспечения стабильной работы трансформатора необходимо иметь прогнозы нагрузки по мощности и температуре. Так как температура внутри трансформатора сильно зависит от внешней температуры и мощности и прогнозируется с высокой точностью, задачу предсказания температуры внутри трансформатора можно свести к задаче предсказания мощности. Это будет продемонстрировано в последующих главах.

3.2. Использование модели CatBoost для прогнозирования временных рядов.

Модель CatBoost успешно используется в качестве метода для прогнозирования временных рядов самыми крупными компаниями, такими как

Тиньков [12] и Яндекс [14]. Идея использования модели леса, уделяющей внимание категориальным признакам, состоит в том, что она позволяет эффективно обрабатывать большие объемы данных с разнообразными типами переменных, учитывая сложные и нелинейные зависимости. CatBoost автоматически обрабатывает категориальные признаки и использует алгоритмы градиентного бустинга для повышения точности прогнозов.

Категориальные признаки, такие как день недели, месяц или праздничные дни, могут существенно влиять на анализ временных рядов. Например, в данных о потреблении энергии можно выявить закономерности, связанные с повышенным потреблением в определенные дни недели или месяцы. CatBoost эффективно выявляет эти закономерности и улучшает качество прогноза за счет использования категориальных данных.

Для начала работы с CatBoost данные формата DataTime [17] необходимо декомпозировать, выделив важные категории: час, день, день недели, месяц, год.

DayOfWeek	Year	Month	Day	Hour
6	2022	1	1	1
6	2022	1	1	2
6	2022	1	1	3
6	2022	1	1	4
6	2022	1	1	5
...	...	...	...	...
7	2023	12	31	19
7	2023	12	31	20
7	2023	12	31	21
7	2023	12	31	22
7	2023	12	31	23

Рис. 10. Новые колонки с категориальными признаками.

Далее требуется определиться с методом обучения. Можно обучить модель только на данных за 2022 год и предсказывать значения мощности в 2023 году, однако такой вариант, скорее всего, покажет плохие результаты, так как не будет учитывать новые данные. По сути, это будет всё равно, что взять некоторую комбинацию усреднённых значений по часам и дням за прошлый год. Поэтому было решено обучить на 80% первых записей, куда войдёт и 2023 год. Кроме временных признаков, о которых было сказано выше, новые признаки не создавались. Произведём обучение модели. Параметры будем искать с помощью метода перебора grid_search [18].

Дополнительно хотелось бы отметить, что для модели CatBoost не принципиально использование номализации или логарифмизации данных. Этим данная модель отличается от аддитивных моделей (prophet, SARIMA), для которых логарифмирование данных временного ряда даёт большую прибавку в точности, а также от нейросетей (например, LSTM), для которых также важна нормализация.

Рис. 11. Предсказанные данные по мощности моделью CatBoost.

Результаты такого прогноза можно считать достаточными для такого большого интервала прогноза. Однако ошибка прогнозирования на ближнем горизонте (неделя) слишком высокая.

Метрики ошибок для тестовой выборки (на которой обучение не проходило): MAE: 760, MAPE: 17.6%.

Возможно, для прогнозирования на ближнем горизонте стоит добавить новые признаки. Добавим признаки lag_n_h (англ. задержка n часов назад) - значения мощности на трансформаторе n часов назад. В качестве n возьмём ряд 8, 16, ... 96. Такие данные улучшат точность модели, однако горизонт её предсказаний по сути снизится до 8 часов вперёд.

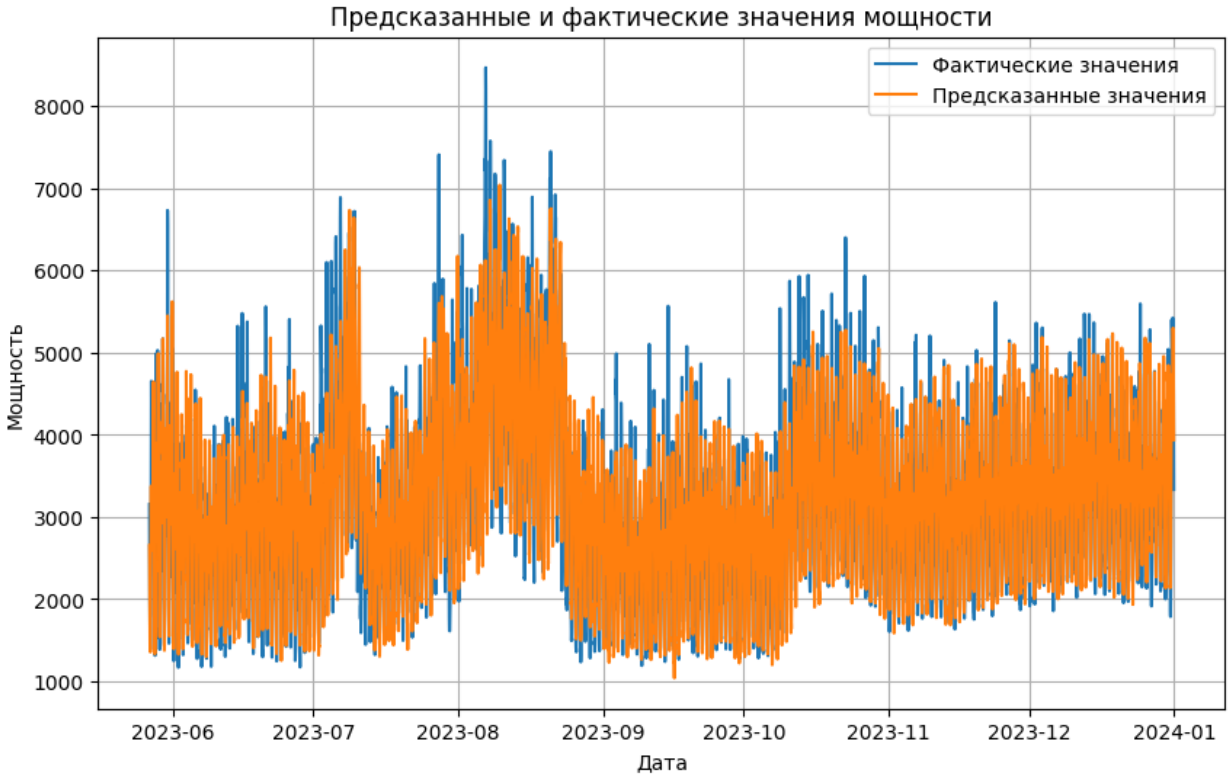

Рис. 12. Модель CatBoost с добавлением lag'ов.

Легко заметны улучшения точности.

Метрики ошибок: MAE: 520, MAPE: 12.2%.

Таким образом, с помощью модели CatBoost были обучены две модели. Одна из них может предоставить прогноз на больший промежуток времени, однако с меньшей точностью. Вторая способна точнее предсказывать мощность на трансформаторе в ближайшие часы.

3.3. Модель Prophet.

Модель Prophet широко используется в различных отраслях для прогнозирования временных рядов, таких как финансы, маркетинг и энергетика. Крупные компании, такие как Airbnb и Lyft, успешно применяют Prophet для решения задач, связанных с анализом и прогнозированием временных рядов.

Основная идея использования модели Prophet заключается в автоматическом учете сложных сезонных паттернов и трендов, что делает её особенно полезной для данных с несколькими уровнями сезонности. Prophet использует аддитивную модель, в которой прогнозируемое значение временного ряда складывается из нескольких компонент: трендовой, сезонной и компонент праздников. Это позволяет включать в анализ различные влияния на данные и улучшать точность прогнозов.

Модель Prophet предоставляет простой и понятный способ настройки параметров, что позволяет легко адаптировать модель под конкретные условия и задачи. Кроме того, Prophet устойчива к пропущенным данным и выбросам, что делает её надежным инструментом для реальных данных, часто содержащих эти аномалии.

Для использования модели Prophet в данной работе необходимо импортировать одноименную библиотеку, а также привести данные временных маркеров в необходимый формат: "год:месяц:день час:минута:секунда". Прогнозируемый столбец должен быть назван "y", а столбец с датой и временем — "ds".

Модель Prophet содержит большое количество настраиваемых параметров. Эти методы и настройки позволяют гибко адаптировать модель Prophet к различным типам данных и задачам, что делает её мощным инструментом для прогнозирования временных рядов, поэтому стоит ознакомиться с наиболее важными из них.

- *growth* (Тип тренда):
  - *linear*: Линейный тренд.
  - *logistic*: Логистический тренд, используется, если данные имеют верхний предел (cap).

- *changepoints* (Список дат изменений тренда): Список дат, в которых предполагаются изменения тренда. Если не указано, Prophet автоматически выбирает точки изменений.

- *n_changepoints* (Количество точек изменений тренда): Количество точек изменений тренда, которые модель пытается найти автоматически. Значение по умолчанию — 25.

- *changepoint_prior_scale* (Регуляризация изменений тренда): Параметр регуляризации для контроля гибкости тренда в точках изменений. Большее значение позволяет модели быть более гибкой, меньшее значение делает модель более устойчивой. Значение по умолчанию — 0.05.

- *seasonality_mode* (Режим сезонности):

  - *additive*: Аддитивная сезонность (по умолчанию).
  - *multiplicative*: Мультипликативная сезонность.

- *seasonality_prior_scale* (Регуляризация сезонности): Параметр регуляризации для контроля гибкости сезонных компонент. Значение по умолчанию — 10.

- *yearly_seasonality* (Годовая сезонность): Включение или выключение годовой сезонности. По умолчанию значение ‘’auto’‘, и модель сама решает включать или нет. Также можно явно задать ‘True‘ или ‘False‘.

- *weekly_seasonality* (Еженедельная сезонность): Включение или выключение еженедельной сезонности. По умолчанию значение ‘’auto’‘, и модель сама решает включать или нет. Также можно явно задать ‘True‘ или ‘False‘.

- *daily_seasonality* (Ежедневная сезонность): Включение или выключение ежедневной сезонности. По умолчанию значение ‘’auto’‘, и модель сама решает включать или нет. Также можно явно задать ‘True‘ или ‘False‘.

- *add_seasonality* (Пользовательская сезонность): Добавление пользовательской сезонности. Можно задать периодичность и количество гармоник.

- *holidays* (Праздники): DataFrame, содержащий информацию о праздниках и специальных днях. Prophet позволяет учитывать влияние таких дней на временной ряд.

- *holidays_prior_scale* (Регуляризация праздников): Параметр регуляризации для контроля гибкости компоненты праздников. Значение по умолчанию — 10.

Для нахождения наилучшего набора параметров был использован метод поиска по сетке (Grid Search). В этом процессе модель обучается на различных комбинациях гиперпараметров, и затем каждая комбинация оценивается с помощью метрик качества. Такой подход позволяет автоматически подобрать наиболее подходящие параметры модели для достижения наилучших результатов.

Для улучшения точности прогнозирования временного ряда, данные были преобразованы с использованием логарифмирования. Этот подход позволяет уменьшить влияние больших колебаний и стабилизировать вариативность ряда. После логарифмирования данные и прогнозы были преобразованы обратно к исходному масштабу с помощью экспоненцирования. Применение логарифмирования значительно снизило ошибку прогноза, обеспечив более точные и стабильные результаты.

Кросс-валидация также является важным инструментом для оценки качества модели Prophet. Кросс-валидация позволяет оценить, насколько хорошо модель будет работать на новых данных, и помогает избежать переобучения.

После обучения модель подобрала лучшие параметры и готова к использованию. Обучим модель на данных за 800 часов и проведем прогноз на следующие 196 часов.

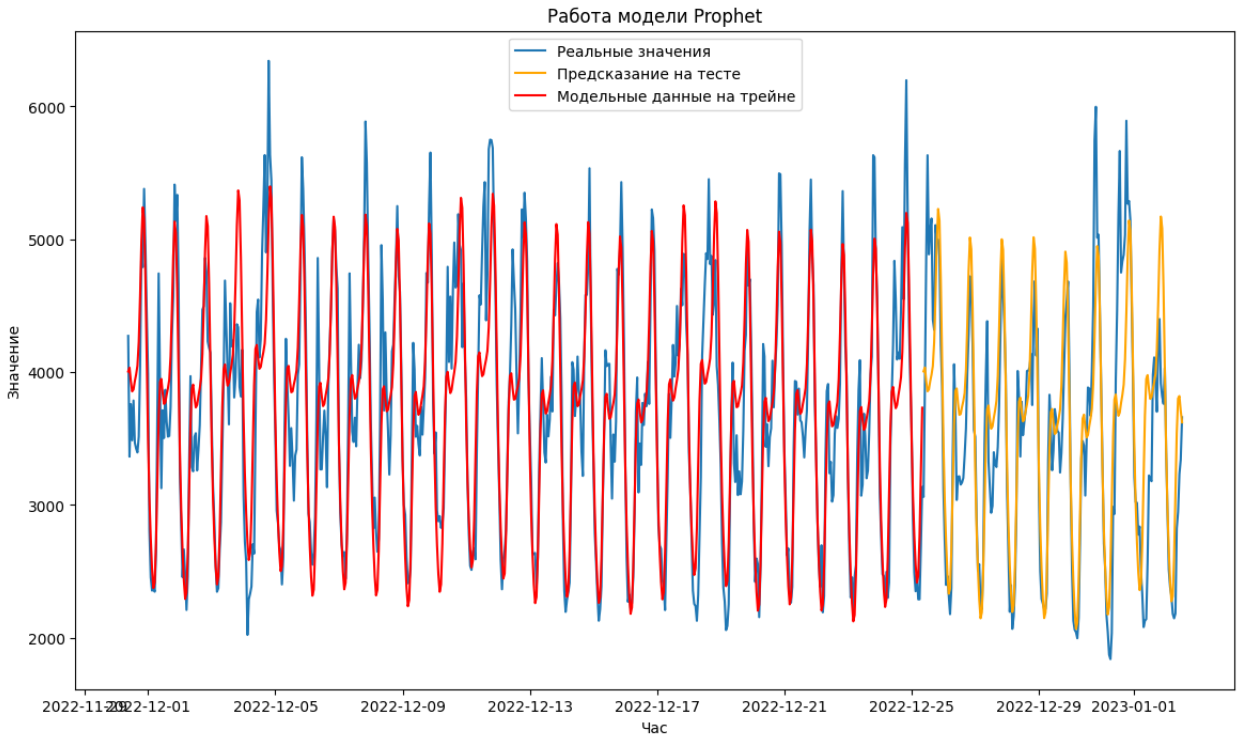

Рис. 13. Результат прогноза модели на 196 часов вперёд.

Визуально модель справляется с прогнозом достаточно хорошо. Для проверки качества работы запустим модель на ста разных участках нашего временного ряда (продолжительностью более 17 тысяч часов), обучая на отрезках 800 часов, делая прогноз на 196 часов вперёд.

После выполнения программы среднее значение метрик ошибок: MAE: 480, MAPE: 11.3%. Этот результат немного лучше результата модели CatBoost, однако, в отличие от неё, модель Prophet способна предсказывать значения мощность на более длинные интервалы без значительного ущерба точности.

3.3. Модель SARIMA.

Модель SARIMA (Seasonal AutoRegressive Integrated Moving Average) широко используется для прогнозирования временных рядов, особенно когда данные имеют выраженные сезонные компоненты. SARIMA объединяет методы авторегрессии (AR), интегрированной (I) части для учета трен-

да, скользящего среднего (MA) и сезонных компонент (S) для улучшения точности прогнозов.

Основная идея модели SARIMA заключается в использовании истории временного ряда для прогнозирования его будущих значений, учитывая как краткосрочные, так и долгосрочные зависимости и сезонные паттерны. Это делает модель SARIMA полезной для анализа данных с регулярными сезонными колебаниями.

Для использования модели SARIMA в данной работе необходимо импортировать библиотеку statsmodels и привести данные к необходимому формату. Временной ряд должен быть стационарным, поэтому перед применением модели SARIMA часто требуется провести преобразования, такие как дифференцирование.

Модель SARIMA имеет следующие параметры, которые необходимо настроить:

- p (порядок авторегрессии): Количество лагов в модели AR.
- d (порядок дифференцирования): Количество применённых дифференцирований для обеспечения стационарности.
- teq (порядок скользящего среднего): Количество лагов в модели MA.
- P (порядок сезонной авторегрессии): Количество лагов в сезонной части модели AR.
- D (порядок сезонного дифференцирования): Количество применённых сезонных дифференцирований.
- Q (порядок сезонного скользящего среднего): Количество лагов в сезонной части модели MA.
- m (длина сезонного периода): Количество наблюдений в каждом сезоне.

Для нахождения наилучшего набора параметров снова используется метод поиска по сетке (Grid Search). И, как и с моделью Prophet, прологарифмируем данные.

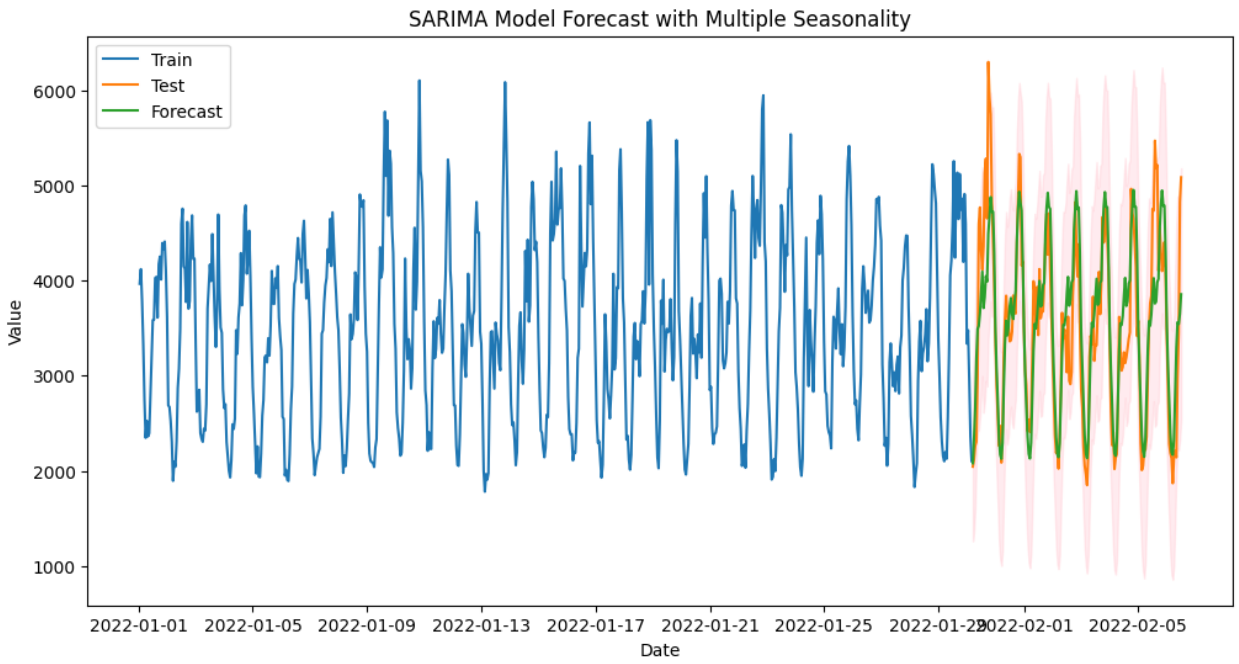

Рис. 14. Результат прогноза модели на 196 часов вперёд.

Модель SARIMA показывает хорошие результаты. Однако обучение и прогнозирование занимают 16 минут, что в 30 раз дольше, чем у модели Prophet. Для проверки качества работы запустим модель на десяти разных участках нашего временного ряда, обучая на отрезках в 800 часов и делая прогноз на 196 часов вперёд, как и у модели Prophet (так как обе модели работают непосредственно с временными рядами, тест будет одинаковым).

После выполнения программы средние значения метрик ошибок составили: MAE: 160, MAPE: 3.9%. Эти результаты намного лучше, чем у модели Prophet, и значительно превосходят результаты модели CatBoost. Модель SARIMA демонстрирует высокую точность прогнозирования и стабильность на длительных интервалах времени, что делает её эффективным инструментом для анализа временных рядов с сезонными паттернами. Однако время её работы значительно больше, чем у предыдущих моделей.

3.4. Модель LSTM.

Модель LSTM (Long Short-Term Memory) — это тип рекуррентной нейронной сети (RNN), разработанный для эффективного моделирования временных рядов и последовательных данных. В LSTM используются ячейки

памяти, которые позволяют сохранять важную информацию на длительные периоды времени, что делает её особенно полезной для работы с данными, содержащими долгосрочные зависимости.

```
model = Sequential()

model.add(Bidirectional(LSTM(100, return_sequences=True), input_shape=(time_step, 1)))
model.add(Dropout(0.3))

model.add(LSTM(100, return_sequences=True))
model.add(Dropout(0.3))

model.add(LSTM(50, return_sequences=False))
model.add(Dropout(0.3))

model.add(Dense(forecast_horizon))


model.compile(optimizer='adam', loss='mean_squared_error')
```

Рис. 15. Структура LSTM модели.

Для улучшения точности прогнозирования временных рядов в данной работе применялась следующая архитектура LSTM:

- Преобразование данных: Временной ряд был преобразован с использованием нормализации, что позволило привести данные к единому масштабу и уменьшить влияние экстремальных значений. Для этого использовался метод Min-Max Scaling, который преобразовал данные в диапазон от 0 до 1. После обучения и прогнозирования значения данных и прогнозов были обратимо преобразованы к исходному масштабу.

- Архитектура модели: Модель включала несколько слоёв LSTM и Dropout. Первый слой был двунаправленным LSTM с 100 нейронами, за ним следовал обычный LSTM с 100 нейронами, затем ещё один LSTM с 50 нейронами. Между слоями LSTM были добавлены слои Dropout для предотвращения переобучения.

- Настройка параметров: Для обучения модели использовался оптимизатор Adam и функция потерь MSE. Параметры модели, такие как размер окна временного ряда (time_step = 48) и количество прогнозируемых шагов вперёд (forecast_horizon = 12), были выбраны на основе экспериментальных исследований. Это значит, что модель на основании данных за двое суток будет предсказывать мощность на трансформаторе на 12 часов вперёд.

- Обучение модели: Модель обучалась с использованием исторических данных, захватывая как краткосрочные, так и долгосрочные зависимости. Валидация модели проводилась на отдельном тестовом наборе данных для оценки её производительности.

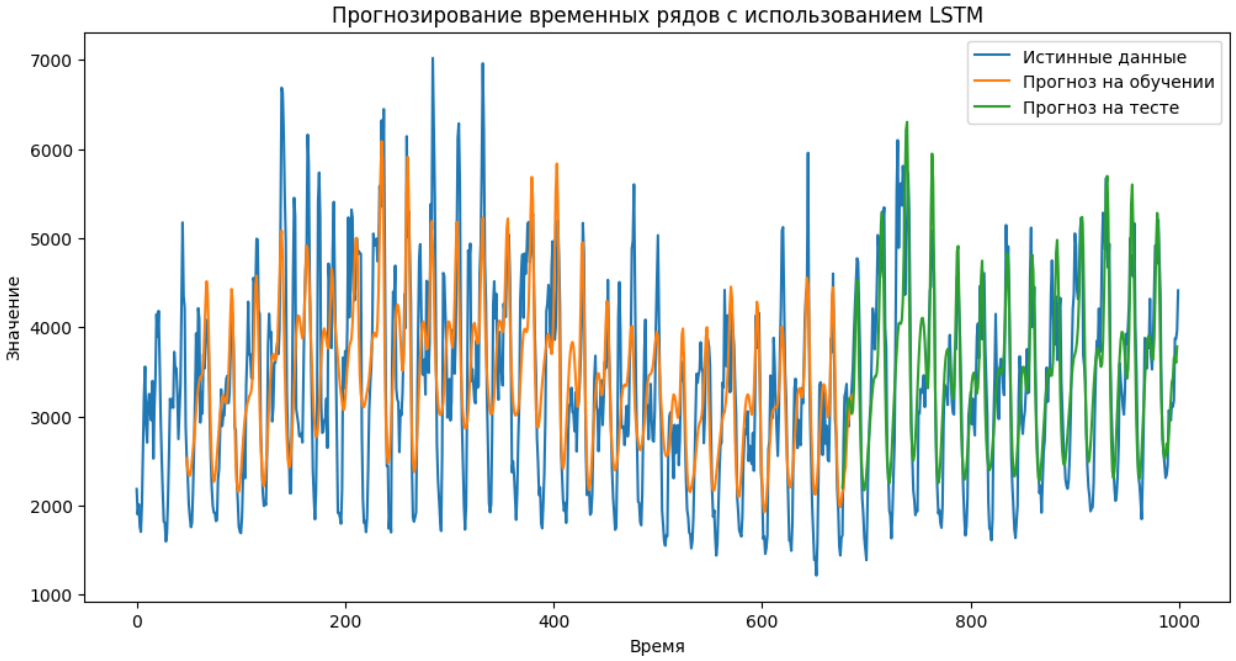

Рис. 16. Результат прогноза модели LSTM.

На рис. 16 представлен результат прогноза модели LSTM. График демонстрирует, как модель предсказывает значение на один шаг вперёд после каждого временного окна, что в данном случае эквивалентно прогнозу на час вперёд.

```
30/30 [==============================] - 2s 56ms/step
Ошибка RMSE для нового набора данных (на каждом шаге прогноза):
Шаг 1: 660.6288612611189
Шаг 2: 672.1675066194834
Шаг 3: 684.855782106009
Шаг 4: 658.7477890091047
Шаг 5: 661.6361920038845
Шаг 6: 650.8165936471648
Шаг 7: 648.1651596396791
Шаг 8: 649.2748757587514
Шаг 9: 642.1945545794009
Шаг 10: 652.4712416887004
Шаг 11: 692.4786767138296
Шаг 12: 707.1782172428516
```

Рис. 17. Значение метрики MAE для предсказаний на каждом шаге.

Модель LSTM демонстрирует среднюю ошибку MAPE около 7%. Этот результат превосходит модели Prophet и CatBoost, но всё же уступает модели SARIMA. Хотя время обучения модели LSTM сопоставимо с SARIMA, она показывает ограниченные возможности для долгосрочного прогнозирования в текущей настройке. Увеличение числа слоёв и расширение окна может потенциально улучшить точность, но это значительно увеличит время работы модели. Учитывая высокую эффективность и точность модели SARIMA, дальнейшее усложнение модели LSTM в данном контексте не представляется необходимым.

Заключение.

В данной работе была решена задача мониторинга и прогнозирования состояния силовых трансформаторов, что является важным аспектом в условиях роста энергопотребления и необходимости повышения надежности энергоснабжения. Анализ был проведен на примере данных, собранных с датчиков трансформатора, установленных в Волгограде.

Было рассмотрено несколько подходов к анализу и прогнозированию временных рядов. В частности, использованы модели CatBoost, Prophet, SARIMA, а также нейросетевая модель LSTM. Каждая из этих моделей имеет свои преимущества и недостатки, что позволило комплексно подойти к решению поставленной задачи.

Модель CatBoost продемонстрировала высокую точность прогнозов при коротких интервалах, учитывая категориальные признаки и сложные нелинейные зависимости. Prophet показал стабильные результаты при прогнозировании на длительные периоды времени, эффективно обрабатывая сезонные и трендовые компоненты данных. SARIMA, в свою очередь, обеспечила наивысшую точность среди всех рассмотренных моделей, но потребовала больше времени на обучение и прогнозирование. Модель LSTM проявила себя как мощный инструмент для анализа временных рядов с длительными зависимостями, обеспечивая конкурентоспособную точность и демонстрируя свою силу в учёте долгосрочных паттернов данных, хотя и с более продолжительным временем обучения.

Результаты показали, что точность прогнозов можно существенно повысить, комбинируя различные методы анализа данных и учитывая влияние внешних факторов, таких как температурные условия. Применение разработанных моделей позволяет эффективно управлять нагрузкой на трансформаторы и предотвращать аварийные ситуации, что особенно важно для стабильной работы энергосистем.

Разработанные методы и модели могут быть использованы не только в Волгограде, но и в других регионах с аналогичными условиями эксплуатации. Это открывает возможности для создания универсальных решений по мониторингу и управлению энергетическим оборудованием, способных улучшить надежность и эффективность энергоснабжения.

Основные этапы данной работы были опубликованы в "XXIX Межвузовской научно-практической конференции студентов и молодых ученых, посвященной 70-летию г. Волжского"и благосклонно приняты членами жюри, что позволило работе занять 3 место на конференции в своём направлении.

Таким образом, проведенное исследование подтвердило значимость применения современных методов анализа данных и машинного обучения для прогнозирования работы силовых трансформаторов. Внедрение этих технологий может существенно повысить надежность и эффективность управления энергетическим оборудованием в условиях современных энергосистем.

Литература

[1] Армстронг Дж. Скотт, Коллопи Фред. Mean Absolute Percentage Error (MAPE). International Journal of Forecasting, 1992. URL: https://www.sciencedirect.com/science/article/abs/pii/016920709290012J

[2] Бокс Джордж, Дженкинс Гвилим, Рейнзел Грегори. Прогнозирование временных рядов с использованием модели SARIMA. Time Series Analysis: Forecasting and Control, 2015. URL: https://www.wiley.com/en-us/Time+Series+Analysis%3A+Forecasting+and+Control%2C+5th+Edition-p-9781118675021

[3] Зачем проводить цифровизацию объектов электроэнергетики: от управления жизненным циклом оборудования до мониторинга параметров систем. // Технологии, инжиниринг, инновации. — 2023. — №4. — С. 60-67.

[4] Тейлор Шон Дж., Летем Бенджамин. Прогнозирование временных рядов с помощью библиотеки Prophet. Journal of Statistical Software, 2018.

[5] Мониторинг промышленного оборудования. // CDC. — 2023. — №5. — С. 30-36.

[6] Современные методы машинного обучения и технология OCR для автоматизации обработки документов. // КиберЛенинка. — 2023. — №1. — С. 12-18.

[7] Горбунова Е.Б., Синютин Е.С., Синютин С.А. Нейросетевой подход к прогнозированию потребления энергоресурсов в городской среде // ИВД. 2018. №4 (51). URL: https://cyberleninka.ru/article/n/neyrosetevoy-podhod-k-prognozirovaniyu-potrebleniya-energoresursov-v-gorodskoy-srede.

[8] Маккинни Уэс. Pandas: мощный инструмент для анализа данных на Python. Open source library, 2010.

[9] Прокхоренкова Людмила, Гусев Глеб, Воробьев Александр, Дорогуш Анна Вероника, Гулин Андрей. CatBoost: unbiased boosting with

categorical features. Advances in Neural Information Processing Systems, 2018. URL: https://catboost.ai/

[10] Педрегоса Фабиан и др. Scikit-learn: машинное обучение на Python. Journal of Machine Learning Research, 2011. URL: https://scikit-learn.org/stable/

[11] Применение нейросетей для автоматизации работы электрических подстанций и электрических сетей. // APNI. — 2023. — №2. — С. 45-52.

[12] Тинькофф. Technology Radar: Core. 2023. URL: https://radar.tinkoff.ru/android/core/

[13] Энергоменеджмент как инструмент управления качеством ресурсосбережения в жилищно-коммунальном хозяйстве. // Фундаментальные исследования. — 2023. — №3. — С. 50-56.

[14] Яндекс. CatBoost: от 0 до 1.0.0. 2021. URL: https://events.yandex.ru/events/catboost/02-10-2021

[15] Pandas Development Team. Документация по Pandas: метод info. 2021. URL: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.info.html

[16] Pandas Development Team. Документация по Pandas: метод plot. 2021. URL: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.html

[17] Python Software Foundation. Документация по модулю datetime в Python. 2021. URL: https://docs.python.org/3/library/datetime.html

[18] Scikit-learn Development Team. Grid search для настройки гиперпараметров. 2021. URL: https://scikit-learn.org/stable/modules/grid_search.html

[19] Sean J. Taylor, Benjamin Letham Forecasting at Scale // THE AMERICAN STATISTICIAN. — 2018. — №1. — С. 37-45.

```
1
2  import pandas as pd
3  import numpy as np
4  import matplotlib.pyplot as plt
5  from pmdarima import auto_arima
6  from statsmodels.tsa.statespace.sarimax import SARIMAX
7  from sklearn.metrics import mean_squared_error
8  from tqdm import tqdm
9  import warnings
10
11
12
13 warnings.filterwarnings("ignore")
14
15
16
17 file_path = 'data2022.xlsx'
18
19 data2022 = pd.read_excel(file_path, index_col='время', parse_dates=True)
20
21
22 data = data2022.drop(['т снаружи', 'т внутри'], axis=1)
23 data.index = pd.to_datetime(data.index, format="%d.%m.%Y %H:%M")
24
25
26
27 data['час'] = data.index.hour
28 data['день_недели'] = data.index.dayofweek
29 data
30
31
32 # Создание переменных сезонности
33 data['час'] = data.index.hour
34 data['день_недели'] = data.index.dayofweek
35
36 data['мощность'] = np.log(data['мощность'] + 1)
37
38 data = data[:int(len(data) * 0.1)]
39
40
41 # Разделение данных на обучающую и тестовую выборки
42 train_size = int(len(data) * 0.8)
43 train, test = data[:train_size], data[train_size:]
44
45
```

```
data
from statsmodels.tsa.stattools import adfuller
d = 0
# Проверка на стационарность
result = adfuller(data['мощность'])
print('ADF Statistic:', result[0])
print('p-value:', result[1])

# Если p-value > 0.05, данные нестационарны и требуется дифференцирование
if result[1] > 0.05:
    data_diff = data['мощность'].diff().dropna()
    result_diff = adfuller(data_diff)
    print('ADF Statistic (diff):', result_diff[0])
    print('p-value (diff):', result_diff[1])

    if result_diff[1] > 0.05:
        data_diff = data_diff.diff().dropna()
        result_diff = adfuller(data_diff)
        print('ADF Statistic (diff2):', result_diff[0])
        print('p-value (diff2):', result_diff[1])
        d = 2 if result_diff[1] < 0.05 else 1
else:
    d = 1
print(f'd = {d}')
seasonal_period = 24
D = 0
# Проверка на сезонную стационарность
data_seasonal_diff = data['мощность'].diff(seasonal_period).dropna()
result_seasonal_diff = adfuller(data_seasonal_diff)
print('ADF Statistic (seasonal diff):', result_seasonal_diff[0])
print('p-value (seasonal diff):', result_seasonal_diff[1])

# Если p-value > 0.05, данные имеют сезонную нестационарность и требуется
    сезонное дифференцирование
if result_seasonal_diff[1] > 0.05:
    D = 1
else:
    D = 0


D

# Лучшие параметры модели
best_order = (12, 1, 3)           # Параметры ARIMA
best_seasonal_order = (2, 1, 1, 24)  # Параметры сезонной части (SARIMA)

# Обучение модели с лучшими параметрами
print("Обучение модели...")
```

```
with tqdm(total=1, desc="Model Training") as pbar:
    model = SARIMAX(train['мощность'], order=best_order, seasonal_order=best_seasonal_order,
                    )
    results = model.fit(disp=False)
    pbar.update(1)

# Прогнозирование на тестовом наборе данных
print("Прогнозирование...")
with tqdm(total=1, desc="Forecasting") as pbar:
    forecast = results.get_forecast(steps=len(test))
    forecast_mean = forecast.predicted_mean
    forecast_conf_int = forecast.conf_int()
    pbar.update(1)

# Оценка модели
mse = mean_squared_error(test['мощность'], forecast_mean)
print(f'Mean Squared Error: {mse}')


# Визуализация
plt.figure(figsize=(12, 6))

# График обучающих данных
plt.plot(train.index, train['мощность'], label='Train')

# График тестовых данных
plt.plot(test.index, test['мощность'], label='Test')

# График предсказанных значений
plt.plot(test.index, forecast_mean, label='Forecast')

#plt.plot(test.index, [(forecast_mean[i] - i*20)*1.2**(1+i/24)  for i in range(len(forecast_mean))], label='Forecast')

# График доверительных интервалов
plt.fill_between(test.index, forecast_conf_int.iloc[:, 0], forecast_conf_int.iloc[:, 1], color='pink', alpha=0.3)

plt.title('SARIMA Model Forecast with Multiple Seasonality')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.show()

np.mean((forecast_mean.values - test['мощность'].values)/test['мощность'].values)
```

```
np.mean((np.exp(forecast_mean.values) - np.exp(test['мощность'].values))/
    np.exp(test['мощность'].values))
# Визуализация
plt.figure(figsize=(12, 6))

# График обучающих данных
plt.plot(train.index, np.exp(train['мощность']), label='Train')

# График тестовых данных
plt.plot(test.index, np.exp(test['мощность']), label='Test')

# График предсказанных значений
plt.plot(test.index, np.exp(forecast_mean), label='Forecast')

#plt.plot(test.index, [(forecast_mean[i] - i*20)*1.2**(1+i/24)  for i in
    range(len(forecast_mean))], label='Forecast')

# График доверительных интервалов
plt.fill_between(test.index, forecast_conf_int.iloc[:, 0],
    forecast_conf_int.iloc[:, 1], color='pink', alpha=0.3)

plt.title('SARIMA Model Forecast with Multiple Seasonality')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.show()
```

```
import pandas as pd
from datetime import datetime
from prophet import Prophet
import matplotlib.pyplot as plt
import numpy as np
import shap
from collections import namedtuple
data2022 = pd.read_excel('data2022.xlsx')
data2023 = pd.read_excel('data2023.xlsx')

data = pd.concat([data2022, data2023], ignore_index=True)
data
data['время'] = data['время'].apply(lambda x: pd.to_datetime(x, format="%d.%m.%Y %H:%M"))
data['время'] = pd.to_datetime(data['время'], format="%d.%m.%Y %H:%M:%S").dt.strftime('%m.%d.%Y %H:%M')
data['время'] = pd.to_datetime(data['время'])
data.info()
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Загрузка примера временного ряда
# Используем временные ряды, представленные в Pandas DataFrame
df = data

# Выбираем интересующий нас столбец
ts = df['мощность']

# Построение ACF и PACF графиков
fig, axes = plt.subplots(2, 1, figsize=(12, 8))

# ACF
plot_acf(ts, lags=30, ax=axes[0])
axes[0].set_title('Autocorrelation Function (ACF)')

# PACF
plot_pacf(ts, lags=30, ax=axes[1])
axes[1].set_title('Partial Autocorrelation Function (PACF)')

plt.tight_layout()
plt.show()

from statsmodels.tsa.stattools import adfuller

```

```
result = adfuller(ts)
print(f"ADF Statistic: {result[0]}")
print(f"p-value: {result[1]}")

import numpy as np

threshold = 3  # например, выбросы за пределами 3 стандартных отклонений
mean = np.mean(ts)
std = np.std(ts)

outliers = ts[(ts > mean + threshold * std) | (ts < mean - threshold * std)]
print(outliers)

ts_smooth = ts.rolling(window=12).mean()
ts_smooth.plot()









data_prophet = data.rename(columns={'время': 'ds', 'мощность': 'y'})
data_prophet = data_prophet[['ds','y']]
data_log = data_prophet.copy()
data_log['y'] = np.log(data_prophet['y'] + 1)  # +1 чтобы избежать логарифмирования нулей

data_log
data_prophet = data_log.copy()
data_prophet
# Создаем и обучаем модель Prophet
start = 6000
lenght = 1200
period = 196
m = Prophet(changepoint_prior_scale=0.01)
m.fit(data_prophet[start:start+lenght])

future = m.make_future_dataframe(periods=period, freq='H')
forecast = m.predict(future)
m.plot(forecast)
forecast['yhat']
plt.figure(figsize=(14, 8))
plt.plot(data_prophet[start:start+lenght+period]['ds'], data_prophet[start:start+lenght+period]['y'], label='Реальные значения')
plt.plot(forecast[lenght:lenght+period]['ds'], forecast[lenght:lenght+period]['yhat'], label='Предсказание на тесте', color='orange')
```

```
plt.plot(forecast[:lenght]['ds'], forecast[:lenght]['yhat'], label='Модельные данные на трейне', color='red')

# Добавляем подписи и легенду
plt.xlabel('Час')
plt.ylabel('Значение')
plt.title('Работа модели Prophet')
plt.legend()

# Отображаем график
plt.show()
forecast['yhat_orig'] = np.exp(forecast['yhat']) - 1
np.mean(abs(forecast['yhat_orig'][:lenght].values - np.exp(data_prophet[start:start+lenght]['y'].values))/np.exp(data_prophet[start:start+lenght]['y'].values))
np.mean(abs(forecast['yhat_orig'][lenght:].values - np.exp(data_prophet[start+lenght:start+lenght+period]['y'].values))/np.exp((data_prophet[start+lenght:start+lenght+period]['y'].values)))
plt.figure(figsize=(14, 8))
plt.plot(data_prophet[start:start+lenght+period]['ds'], np.exp(data_prophet[start:start+lenght+period]['y']), label='Реальные значения')
plt.plot(forecast[lenght:lenght+period]['ds'], np.exp(forecast[lenght:lenght+period]['yhat']), label='Предсказание на тесте', color='orange')
plt.plot(forecast[:lenght]['ds'], np.exp(forecast[:lenght]['yhat']), label='Модельные данные на трейне', color='red')

# Добавляем подписи и легенду
plt.xlabel('Час')
plt.ylabel('Значение')
plt.title('Работа модели Prophet')
plt.legend()

# Отображаем график
plt.show()
np.mean(abs(forecast['yhat'][:lenght].values - data_prophet[start:start+lenght]['y'].values)/data_prophet[start:start+lenght]['y'].values)
np.mean(abs(forecast['yhat'][lenght:].values - data_prophet[start+lenght:start+lenght+period]['y'].values)/(data_prophet[start+lenght:start+lenght+period]['y'].values))
np.mean(abs(forecast['yhat'][:lenght].values - data_prophet[start:start+lenght]['y'].values))


testErrMax = 0
trainErrMax = 0
trainErr = 0
```

```
testErr = 0

# Создаем и обучаем модель Prophet
for i in range(20):
    start = 48*i
    lenght = 600
    period = 196
    m = Prophet(changepoint_prior_scale=0.01)
    m.fit(data_prophet[start:start+lenght])

    future = m.make_future_dataframe(periods=period, freq='H')
    forecast = m.predict(future)
    forecast['yhat_orig'] = np.exp(forecast['yhat']) - 1


    trainErr += np.mean(abs(forecast['yhat_orig'][:lenght].values - np.exp
    (data_prophet[start:start+lenght]['y'].values))/np.exp(data_prophet[
    start:start+lenght]['y'].values))
    trainErrMax = max(trainErrMax, np.max(abs(forecast['yhat'][:lenght].
    values - data_prophet[start:start+lenght]['y'].values)/data_prophet[
    start:start+lenght]['y'].values))

    testErrMax = max(testErrMax, np.max(abs(forecast['yhat'][lenght:].
    values - data_prophet[start+lenght:start+lenght+period]['y'].values)/(
    data_prophet[start+lenght:start+lenght+period]['y'].values)))
    testErr += np.mean(abs(forecast['yhat_orig'][lenght:].values - np.exp(
    data_prophet[start+lenght:start+lenght+period]['y'].values))/np.exp((
    data_prophet[start+lenght:start+lenght+period]['y'].values)))
trainErrMax, testErrMax, trainErr/20, testErr/20,
```

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Bidirectional
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error

data2022 = pd.read_excel('data2022.xlsx')
data2023 = pd.read_excel('data2023.xlsx')
data = pd.concat([data2022, data2023], ignore_index=True)

start = 3000
length = 1000

df = pd.DataFrame(data['мощность'][start:start + length])

scaler = MinMaxScaler()
data_normalized = scaler.fit_transform(df)

def create_dataset(data, time_step=1, forecast_horizon=1):
    X, y = [], []
    for i in range(len(data) - time_step - forecast_horizon):
        X.append(data[i:(i + time_step), 0])
        y.append(data[(i + time_step):(i + time_step + forecast_horizon),
    0])
    return np.array(X), np.array(y)

time_step = 48
forecast_horizon = 12

X, y = create_dataset(data_normalized, time_step, forecast_horizon)
X = X.reshape(X.shape[0], X.shape[1], 1)

train_size = int(len(X) * 0.67)
test_size = len(X) - train_size
X_train, X_test = X[0:train_size], X[train_size:len(X)]
y_train, y_test = y[0:train_size], y[train_size:len(y)]

model = Sequential()
model.add(Bidirectional(LSTM(100, return_sequences=True), input_shape=(
    time_step, 1)))
model.add(Dropout(0.3))
model.add(LSTM(100, return_sequences=True))
```

```
model.add(Dropout(0.3))
model.add(LSTM(50, return_sequences=False))
model.add(Dropout(0.3))
model.add(Dense(forecast_horizon))
model.compile(optimizer='adam', loss='mean_squared_error')

early_stop = EarlyStopping(monitor='val_loss', patience=10,
    restore_best_weights=True)
history = model.fit(X_train, y_train, validation_data=(X_test, y_test),
    epochs=50, batch_size=64, callbacks=[early_stop], verbose=1)

train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)

plt.figure(figsize=(12, 6))
plt.plot(scaler.inverse_transform(data_normalized), label='Истинные данные
    ')

train_predict_plot = np.empty_like(data_normalized)
train_predict_plot[:, :] = np.nan
for i in range(len(train_predict)):
    train_predict_plot[i + time_step : i + time_step + forecast_horizon,
    0] = train_predict[i, :]

test_predict_plot = np.empty_like(data_normalized)
test_predict_plot[:, :] = np.nan
test_start = len(train_predict) + time_step
for i in range(len(test_predict)):
    test_predict_plot[test_start + i : test_start + i + forecast_horizon,
    0] = test_predict[i, :]

plt.plot(train_predict_plot, label='Прогноз на обучении')
plt.plot(test_predict_plot, label='Прогноз на тесте')
plt.legend()
plt.xlabel('Время')
plt.ylabel('Значение')
plt.title('Прогнозирование временных рядов с использованием LSTM')
plt.show()

def calculate_rmse(y_true, y_pred):
    return np.sqrt(mean_squared_error(y_true, y_pred))

train_rmse_list = []
for i in range(forecast_horizon):
```

```
    true_values = scaler.inverse_transform(data_normalized[time_step + i :
    len(train_predict) + time_step + i, :])
    predicted_values = train_predict[:, i].reshape(-1, 1)
    rmse = calculate_rmse(true_values[:len(predicted_values)],
   predicted_values)
    train_rmse_list.append(rmse)

test_rmse_list = []
test_start = len(train_predict) + time_step
for i in range(forecast_horizon):
    true_values = scaler.inverse_transform(data_normalized[test_start + i
   : test_start + len(test_predict) + i, :])
    predicted_values = test_predict[:, i].reshape(-1, 1)
    rmse = calculate_rmse(true_values[:len(predicted_values)],
   predicted_values)
    test_rmse_list.append(rmse)

print("Ошибка RMSE для обучающего набора данных (на каждом шаге прогноза):
   ")
for i, rmse in enumerate(train_rmse_list):
    print(f"Шаг {i + 1}: {rmse}")

print("\nОшибка RMSE для тестового набора данных (на каждом шаге прогноза)
   :")
for i, rmse in enumerate(test_rmse_list):
    print(f"Шаг {i + 1}: {rmse}")

new_df = pd.DataFrame(data['мощность'][start + length:start + length +
   length])
new_data = new_df.values
new_data_normalized = scaler.transform(new_data)

def create_new_dataset(data, time_step=1):
    X = []
    for i in range(len(data) - time_step - forecast_horizon + 1):
        X.append(data[i:(i + time_step), 0])
    return np.array(X)

X_new = create_new_dataset(new_data_normalized, time_step)
X_new = X_new.reshape(X_new.shape[0], X_new.shape[1], 1)

new_predict = model.predict(X_new)
new_predict = scaler.inverse_transform(new_predict)

y_true_new = []
for i in range(len(new_data) - time_step - forecast_horizon + 1):
```

```
125     y_true_new.append(new_data[(i + time_step):(i + time_step +
    forecast_horizon), 0])
126
127 y_true_new = np.array(y_true_new)
128
129 new_rmse_list = []
130 for i in range(forecast_horizon):
131     true_values = y_true_new[:, i]
132     predicted_values = new_predict[:, i]
133     assert len(true_values) == len(predicted_values), f"Inconsistent
    lengths: true_values={len(true_values)}, predicted_values={len(
    predicted_values)}"
134     rmse = calculate_rmse(true_values, predicted_values)
135     new_rmse_list.append(rmse)
136
137 print("Ошибка RMSE для нового набора данных (на каждом шаге прогноза):")
138 for i, rmse in enumerate(new_rmse_list):
139     print(f"Шаг {i + 1}: {rmse}")
140
141 plt.figure(figsize=(12, 6))
142 plt.plot(new_data, label='Истинные данные')
143 for i in range(1):
144     new_predict_plot = np.empty_like(new_data)
145     new_predict_plot[:, :] = np.nan
146     for j in range(len(new_predict)):
147         new_predict_plot[j + time_step + i, 0] = new_predict[j, i]
148     plt.plot(new_predict_plot, label=f'Прогноз на новых данных (шаг {i +
    1})')
149
150 plt.legend()
151 plt.xlabel('Время')
152 plt.ylabel('Значение')
153 plt.title('Прогнозирование временных рядов на новых данных с использование
    м LSTM')
154 plt.show()
```

```
import pandas as pd
from datetime import datetime
from prophet import Prophet
import matplotlib.pyplot as plt
import numpy as np
import shap
from collections import namedtuple
data2022 = pd.read_excel('data2022.xlsx')
data2023 = pd.read_excel('data2023.xlsx')

data = pd.concat([data2022, data2023], ignore_index=True)
data
for i in range(1, 13):
    data[f'lag{i*8}h'] = np.append(data['мощность'].values[24:i*8 + 24],
   data['мощность'].values[0:-1*8*i])
data['время'] = data['время'].apply(lambda x: pd.to_datetime(x, format="%d
   .%m.%Y %H:%M"))
data['время'] = pd.to_datetime(data['время'], format="%d.%m.%Y %H:%M:%S").
   dt.strftime('%m.%d.%Y %H:%M')
data['время'] = pd.to_datetime(data['время'])
data.info()

data['DayOfWeek'] = '-'
data['Year'] = '-'
data['Month'] = '-'
data['Day'] = '-'
data['Hour'] = '-'

for i, d in enumerate(data.values):
    time_obj = data.loc[i, 'время']

    year = time_obj.strftime("%Y")
    month = time_obj.strftime("%m")
    day = time_obj.strftime("%d")
    hour = time_obj.strftime("%H")
    weekday = time_obj.isoweekday()

    data.loc[i, 'Year'] = int(year)
    data.loc[i, 'Month'] = int(month)
    data.loc[i, 'Day'] = int(day)
    data.loc[i, 'Hour'] = int(hour)
    data.loc[i, 'DayOfWeek'] = int(weekday)


data
```

```
# Преобразуем данные в формат, понятный Prophet
data_prophet = data.rename(columns={'время': 'ds', 'мощность': 'y'})

# Создаем и обучаем модель Prophet
model = Prophet()
model.fit(data_prophet)

# Создаем DataFrame для прогнозирования
future = model.make_future_dataframe(periods=30)  # Прогноз на 365 дней

# Получаем прогноз
forecast = model.predict(future)

# Выводим результаты прогноза
print(forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail())

fig2 = model.plot_components(forecast,  uncertainty=True)
data.plot(x='время', y='мощность', linestyle='-',  marker=None, figsize
    =(10, 6))
plt.title('График мощности относительно даты')
plt.xlabel('Дата')
plt.ylabel('Мощность')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
from catboost import CatBoostRegressor
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error
import numpy as np

X = data.drop(['мощность', 'время', 'т внутри'], axis=1)

y = data['мощность']


#X_train = data.loc[data['Year'] == 2022].drop(['мощность', 'время', 'т вн
    утри'], axis=1)
#X_test = data.loc[data['Year'] == 2023].drop(['мощность', 'время', 'т вну
    три'], axis=1)
#y_train = data.loc[data['Year'] == 2022]['мощность']
#y_test = data.loc[data['Year'] == 2023]['мощность']

split_index = int(len(data) * 0.7)
```

```
X_train = data.iloc[:split_index].drop(['мощность', 'время', 'т внутри'],
    axis=1)
X_test = data.iloc[split_index:].drop(['мощность', 'время', 'т внутри'],
    axis=1)
y_train = data.iloc[:split_index]['мощность']
y_test = data.iloc[split_index:]['мощность']

model = CatBoostRegressor(iterations=1000, learning_rate=0.1, depth=6,
    loss_function='RMSE', random_state=42)

cv_scores = cross_val_score(model, X_train, y_train, cv=5, scoring='
    neg_mean_squared_error')
rmse_scores = np.sqrt(-cv_scores)

model.fit(X_train, y_train, verbose=100)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

print('Cross-Validation RMSE:', rmse_scores)
print('Mean Cross-Validation RMSE:', rmse_scores.mean())
print('Test RMSE:', rmse)
shap.initjs()
explainer = shap.TreeExplainer(model)
shap_values = explainer(X_test.iloc[:2000, :])
shap.plots.beeswarm(shap_values)

# Сортируем индексы, чтобы графики совпадали
y_test_sorted = y_test.sort_index()
y_pred_sorted = pd.Series(y_pred, index=y_test.index).sort_index()

# Построение графика
plt.figure(figsize=(10, 6))
plt.plot(data.iloc[split_index:]['время'], y_test_sorted, label='Фактическ
    ие значения')
plt.plot(data.iloc[split_index:]['время'], y_pred_sorted, label='Предсказа
    нные значения')
plt.xlabel('Дата')
plt.ylabel('Мощность')
plt.title('Предсказанные и фактические значения мощности')
plt.legend()
plt.grid(True)
plt.show()
dfToSave = pd.DataFrame(y_pred_sorted.values, index = data.iloc[
    split_index:]['время'])
```

```
dfToSave.to_csv('predict_catboost.csv')
import pandas as pd
import matplotlib.pyplot as plt

# Предположим, у вас есть DataFrame data с колонками "мощность", "неделя" и "Month"

# Группируем данные по дням недели и Montham, и находим среднее значение мощности
mean_power_by_week = data.groupby('DayOfWeek')['мощность'].mean()
mean_power_by_month = data.groupby('Month')['мощность'].mean()

# Построение гистограммы средних значений мощности по дням недели
plt.figure(figsize=(10, 6))
mean_power_by_week.plot(kind='bar', color='skyblue')
plt.xlabel('DayOfWeek')
plt.ylabel('Средняя мощность')
plt.title('Средняя мощность по дням недели')
plt.grid(True)
plt.show()

# Построение гистограммы средних значений мощности по Montham
plt.figure(figsize=(10, 6))
mean_power_by_month.plot(kind='bar', color='salmon')
plt.xlabel('месяц')
plt.ylabel('Средняя мощность')
plt.title('Средняя мощность по месяцам')
plt.grid(True)
plt.show()
dataPredicted = data.copy()
dataPredicted.loc[y_pred_sorted.index.intersection(dataPredicted.index), 'pred'] = y_pred_sorted.values
dataPredicted['diff'] = dataPredicted['pred'] - dataPredicted['мощность']
dataPredicted_cleaned = dataPredicted.dropna(how='any')

dataPredicted_cleaned

mean_values_by_weekday = dataPredicted_cleaned.groupby('DayOfWeek')[['мощность', 'pred']].mean()
mean_values_by_month = dataPredicted_cleaned.groupby('Month')[['мощность', 'pred']].mean()

# Построение гистограммы средних значений мощности и предсказанных значений по дням недели
plt.figure(figsize=(10, 6))
mean_values_by_weekday.plot(kind='bar')
plt.xlabel('DayOfWeek')
```

```
plt.ylabel('Среднее значение')
plt.title('Средние значения мощности и предсказанных значений по дням недели')
plt.grid(True)
plt.xticks(range(7), ['Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб', 'Вс'])
plt.show()

# Построение гистограммы средних значений мощности и предсказанных значений по Montham
plt.figure(figsize=(10, 6))
mean_values_by_month.plot(kind='bar')
plt.xlabel('Month')
plt.ylabel('Среднее значение')
plt.title('Средние значения мощности и предсказанных значений по месяцамам')
plt.grid(True)
plt.show()

errorDF = (dataPredicted_cleaned['pred'] - dataPredicted_cleaned['мощность'])/dataPredicted_cleaned['мощность']
abs(errorDF).mean(), abs(errorDF).max()
dataPredicted.iloc[abs(errorDF).nlargest(30).index]
len(errorDF.loc[errorDF > 0.1]), len(errorDF.loc[errorDF > 0.2]), len(errorDF.loc[errorDF > 0.4]), len(errorDF.loc[errorDF > 0.6])
len(errorDF.loc[errorDF < -0.1]), len(errorDF.loc[errorDF < -0.2]), len(errorDF.loc[errorDF < -0.4]), len(errorDF.loc[errorDF < -0.6])
```