Geekbrains

**Применение искусственного интеллекта и нейронных сетей для улучшения обслуживания авиационных двигателей в эксплуатации**

Программа: Искусственный интеллект

Специализация: Разработчик

Смирнов Михаил Владимирович

г. Рыбинск

2024

# Содержание

[Введение 3](#_Toc1040202500)

[1 Теоретическая часть 5](#_Toc622727371)

[1.1 Традиционные методы диагностики и прогноза отказов ТРД 7](#_Toc1302036214)

[1.2 Современные подходы и алгоритмы в предсказательной аналитике 8](#_Toc1150677036)

[1.3 Применение искусственного интеллекта и нейронных сетей 8](#_Toc1338209008)

[1.4 Искусственный интеллект и технология цифрового двойника 10](#_Toc412445715)

[Принципы построения цифрового двойника 11](#_Toc1692381358)

[2 Практическая часть 11](#_Toc1684395893)

[2.1 Методология решения задачи 12](#_Toc18840052)

[2.2 Описание исходных данных 12](#_Toc260918348)

[2.3 Построение модели 13](#_Toc1854019840)

[Описание: 15](#_Toc188295449)

[Описание: 16](#_Toc120659610)

[Описание: 18](#_Toc1326960174)

[Описание: 20](#_Toc685683319)

[Итог: 21](#_Toc198885930)

[Описание: 22](#_Toc1032199338)

[Описание: 24](#_Toc2034163313)

[Результаты: 27](#_Toc1247075625)

[2.4 Результаты модели 27](#_Toc1599301015)

[Анализ результатов 28](#_Toc1421389201)

[Вывод 28](#_Toc1636094220)

[Заключение 28](#_Toc1358340742)

[Список используемой литературы 28](#_Toc443718311)

[Приложения 28](#_Toc1965196384)

# Введение

Авиационные двигатели — это сердце любого самолета, от их надежности и эффективности зависит безопасность полетов и экономическая целесообразность эксплуатации воздушных судов. Двигатели должны обеспечивать высокую степень надежности, чтобы минимизировать риск отказов в воздухе, которые могут привести к авиационным происшествиям. Случаи отказа двигателя в воздухе, например, крушение самолета Ил-76 в Ивановской области 12.03.2024 [[[1]](#endnote-1)] подчеркивают критическую важность поддержания двигателя в исправном состоянии. Надежность авиационных двигателей также напрямую влияет на экономику авиационных компаний, так как незапланированные ремонтные работы и простои приводят к значительным финансовым потерям.

Существующие методы технического обслуживания авиационных двигателей часто основываются на регулярных осмотрах и профилактических ремонтах. Эти методы имеют ряд недостатков:

1. Ограниченная прогностическая способность: Текущие методы диагностики и технического обслуживания часто не могут точно предсказать время и место возникновения отказа, что приводит к излишним проверкам или, наоборот, к пропуску критических проблем.

2. Высокие затраты: Профилактическое обслуживание может включать в себя замены компонентов, которые еще не достигли конца своего жизненного цикла, что ведет к лишним расходам.

3. Риски пропуска критических дефектов: Визуальные осмотры и стандартные методы диагностики могут не выявить скрытые дефекты, которые могут привести к отказу двигателя.

Искусственный интеллект (ИИ) и нейронные сети представляют собой мощные инструменты, способные значительно улучшить процессы диагностики и прогнозирования технического состояния авиационных двигателей. Основные преимущества их применения включают:

1. Повышенная точность прогнозирования: Нейронные сети способны анализировать большие объемы данных, собираемых с датчиков, и выявлять сложные закономерности, которые могут ускользнуть от традиционных методов. Это позволяет с высокой точностью предсказывать момент возникновения отказа и оптимизировать графики технического обслуживания.

2. Раннее обнаружение потенциальных отказов: ИИ-алгоритмы могут анализировать тенденции и аномалии в данных о работе двигателей, что позволяет обнаруживать потенциальные проблемы задолго до их фактического возникновения.

3. Оптимизация затрат: Прогностическое обслуживание, основанное на данных, позволяет значительно снизить расходы на техническое обслуживание, заменяя компоненты только по мере необходимости и снижая количество незапланированных простоев.

Ожидается, что разработанная модель на основе нейронной сети значительно улучшит точность прогнозирования отказов авиационных двигателей по сравнению с традиционными методами. Это позволит авиакомпаниям снижать операционные затраты за счет оптимизации технического обслуживания и увеличения надежности воздушного транспорта.

Таким образом, дипломный проект направлен на практическое применение передовых технологий ИИ в авиационной отрасли, что способствует повышению безопасности полетов и экономической эффективности авиакомпаний.

**Тема проекта**: Применение искусственного интеллекта и нейронных сетей для улучшения обслуживания авиационных двигателей в эксплуатации.

**Цель:** Основной целью данного дипломного проекта является применение искусственного интеллекта и нейронных сетей для улучшения эффективности обслуживания авиационных двигателей в условиях эксплуатации.

**Какую проблему решает:** Повышение точности прогнозирования отказов двигателей с целью оптимизации планирования технического обслуживания.

**Задачи:**

1. Выполнить анализ существующих подходов и методов, исследовать литературу и практические примеры применения ИИ и нейронных сетей в авиационной отрасли для предсказания отказов авиационных двигателей.

Собрать и подготовить необходимые данные о работе авиационных двигателей, включая параметры работы, износ компонентов, историю отказов и ремонтов.

1. Разработать и обучить модель глубокой нейронной сети для прогнозирования номера цикла отказа авиационного двигателя на основе собранных данных.
2. Применить разработанную модель для оптимизации расписания технического обслуживания авиационных двигателей с целью минимизации незапланированных простоев и оптимизации затрат.

**Инструменты:** Jupyter Notebook, Python 3, Numpy, Scikit-learn, Git

**Состав команды**: Смирнов Михаил Владимирович (Разработчик)

# 1 Теоретическая часть

Авиационные турбореактивные двигатели (ТРД) являются основными источниками тяги для большинства современных реактивных самолетов. Их конструкция и принцип работы обеспечивают высокую эффективность и мощность, необходимые для выполнения длительных и скоростных полетов.

Авиационный турбореактивный двигатель состоит из нескольких ключевых компонентов, каждый из которых выполняет определенные функции, обеспечивающие эффективную работу двигателя в целом [[[2]](#endnote-2)]:

1. Вентилятор:

- Функция: Основной компонент на входе двигателя, вентилятор всасывает большой объем воздуха и ускоряет его. Большая часть этого воздуха проходит через внешний контур двигателя (байпасный поток), создавая значительную долю тяги.

- Конструкция: Состоит из нескольких лопаток, закрепленных на вращающемся валу. Лопатки изготавливаются из легких и прочных материалов, таких как титан и композиты.

2. Компрессор:

- Функция: Сжимает входящий воздух до высокого давления перед его подачей в камеру сгорания.

- Конструкция: Включает несколько ступеней, каждая из которых состоит из вращающихся и статичных лопаток. Существует два типа компрессоров: осевые и центробежные, причем осевые компрессоры чаще применяются в современных ТРД.

3. Камера сгорания:

- Функция: Место, где сжатый воздух смешивается с топливом и происходит горение, создавая высокоэнергетические газы.

- Конструкция: Включает топливные форсунки, свечи зажигания и горелки. Камеры сгорания бывают различных типов, включая трубчатые, кольцевые и трубчато-кольцевые.

4. Турбина:

- Функция: Использует энергию горячих газов из камеры сгорания для привода вентилятора и компрессора.

- Конструкция: Состоит из нескольких ступеней лопаток, аналогичных компрессору, но работающих в обратном направлении, извлекая энергию из потока горячих газов.

5. Выхлопная система:

- Функция: Направляет отработанные газы из турбины наружу, создавая реактивную тягу.

- Конструкция: Включает сопло, которое может быть регулируемым для управления потоком газов и оптимизации тяги в различных режимах полета.

Принцип работы авиационного турбореактивного двигателя основывается на преобразовании энергии сгорания топлива в механическую работу и реактивную тягу. Работу двигателя можно разбить на следующие этапы:

1. Всасывание: Вентилятор всасывает воздух и ускоряет его. Часть воздуха поступает в компрессор, а часть проходит через байпасный контур.

2. Сжатие: Компрессор сжимает воздух, увеличивая его давление и температуру. Этот процесс требует значительной энергии, которую обеспечивает турбина.

3. Сгорание: Сжатый воздух поступает в камеру сгорания, где смешивается с топливом и сгорает. В результате образуются горячие газы с высоким давлением и температурой.

4. Работа турбины: Горячие газы проходят через турбину, приводя её лопатки в движение. Турбина извлекает часть энергии газов для привода вентилятора и компрессора.

5. Выход газов и создание тяги: Отработанные газы выходят через сопло, создавая реактивную тягу, которая движет самолет вперед.

Авиационные турбореактивные двигатели представляют собой сложные и высокотехнологичные устройства, обеспечивающие эффективную тягу и надежность в эксплуатации. Понимание их структуры и принципов работы позволяет более эффективно разрабатывать и внедрять технологии для улучшения технического обслуживания и повышения безопасности полетов.

## 1.1 Традиционные методы диагностики и прогноза отказов ТРД

На протяжении многих лет авиационные компании используют различные методы диагностики и технического обслуживания авиационных двигателей. Традиционные методы включают в себя регулярные осмотры, профилактическое обслуживание и замену компонентов по расписанию. [[[3]](#endnote-3)] Основным недостатком этих подходов является их реактивный характер и ограниченная прогностическая способность. Например, анализ вибраций и использование ультразвуковых технологий часто применяются для выявления механических проблем, таких как износ подшипников и трещины в компонентах. Тем не менее, эти методы могут не обнаружить скрытые дефекты или точно предсказать время возникновения отказа [[[4]](#endnote-4)].

Статистические методы играют важную роль в оценке технического состояния ГТД. Они основаны на обработке и анализе измеряемых параметров двигателей, полученных в ходе стендовых испытаний и эксплуатации. Эти методы позволяют выявлять тенденции изменения параметров и прогнозировать возможные отклонения от нормы. Особое внимание уделяется анализу параметров вибрации, что помогает своевременно обнаруживать потенциальные проблемы в работе двигателя [[[5]](#endnote-5)].

Физические модели используются для более глубокого понимания процессов, происходящих в двигателе. Они позволяют моделировать различные режимы работы и прогнозировать поведение двигателя в различных условиях.

Комбинированные подходы объединяют преимущества статистических методов и физических моделей. Они включают в себя использование математических моделей и диагностических матриц для контроля технического состояния узлов проточной части ГТД. Кроме того, применяются методы оценки выработки ресурса ГТД по эквивалентной наработке в эксплуатации и контроль параметров по результатам наземного опробования и параметрической полётной информации.

Анализ существующих методов диагностики и прогнозирования отказов авиационных двигателей показывает, что традиционные подходы имеют значительные ограничения. Применение искусственного интеллекта и нейронных сетей предоставляет новые возможности для повышения точности прогнозов и оптимизации технического обслуживания. Современные исследования демонстрируют, что методы глубокого обучения, такие как DNN, RNN и CNN, обладают высоким потенциалом для улучшения надежности и эффективности авиационных двигателей. Внедрение этих технологий способствует повышению безопасности полетов и снижению эксплуатационных затрат авиакомпаний.

## 1.2 Современные подходы и алгоритмы в предсказательной аналитике

Современные подходы в предсказательной аналитике основаны на использовании методов машинного обучения и глубоких нейронных сетей. Основные алгоритмы включают в себя:

1. Глубокие нейронные сети (DNN): Используются для анализа больших объемов данных и выявления сложных зависимостей между различными параметрами работы двигателя (Hinton et al., 2012).

2. Рекуррентные нейронные сети (RNN): Особенно эффективны для анализа временных рядов данных и прогнозирования динамики изменения параметров двигателя (Hochreiter & Schmidhuber, 1997).

3. Сверточные нейронные сети (CNN): Применяются для анализа данных, представляющих собой многомерные массивы, такие как спектры вибраций или тепловые изображения (LeCun et al., 2015).

4. Подходы ансамблевого обучения: Включают использование нескольких моделей и их объединение для повышения точности прогнозирования (Breiman, 2001).

## 1.3 Применение искусственного интеллекта и нейронных сетей

Искусственный интеллект (ИИ) и нейронные сети в последние годы привлекли значительное внимание как эффективные инструменты для улучшения диагностики и прогнозирования отказов авиационных двигателей.

Нейронные сети обладают рядом уникальных преимуществ, которые делают их эффективными в задачах прогнозирования:

Универсальность: Нейронные сети являются универсальными аппроксиматорами, способными моделировать любую непрерывную функцию с приемлемой точностью.

Самообучение: Они представляют собой самообучающиеся модели, работа которых практически не требует вмешательства пользователя.

Нелинейность: Нейронные сети эффективно решают задачи классификации даже при отсутствии линейной разделимости классов.

Адаптивность: Они способны адаптироваться к сложным и динамичным условиям, обеспечивая быстрое и точное решение задач разной сложности.

Для задач прогнозирования используются различные архитектуры нейронных сетей:

Сети прямого распространения: Наиболее часто используемая архитектура для классификации. На входные нейроны подаются значения признаков классифицируемого объекта, а на выходе формируется метка или числовой код класса.

Многослойные персептроны: Часто применяются для решения задач классификации и прогнозирования.

Рекуррентные нейронные сети: Эффективны для анализа временных рядов и прогнозирования последовательностей.

В авиационной отрасли нейронные сети находят широкое применение:

Проектирование: Использование нейронных сетей позволяет повысить эффективность проектирования изделий авиационной техники, повысить точность расчётов и безопасность [[[6]](#endnote-6)]

Техническое обслуживание: Нейросети анализируют данные от датчиков и сенсоров летательных аппаратов, предоставляя информацию о состоянии оборудования и потребности в техническом обслуживании.

Системы связи: Применяются для распознавания и синтеза речи в системах связи между пилотами и диспетчерами.

Навигация: Используются для распознавания и обработки изображений в системах навигации и контроля за полётом.

Xu и соавторы (2018) продемонстрировали, что глубокие нейронные сети могут анализировать большие объемы данных о вибрациях двигателей и с высокой точностью предсказывать отказ подшипников. [[[7]](#endnote-7)] В исследовании была разработана модель, которая обучалась на исторических данных о вибрациях и показала значительное улучшение по сравнению с традиционными методами.

Zhang et al. (2020) [[[8]](#endnote-8)] провели исследование, в котором использовали нейронные сети для прогнозирования оставшегося ресурса (RUL) авиационных двигателей на основе данных о температуре и давлении. Их результаты показали, что такие модели могут точно предсказывать время до отказа, что позволяет оптимизировать планирование технического обслуживания и снижать эксплуатационные расходы. [[[9]](#endnote-9)]

В работе Li et al. (2019) [[[10]](#endnote-10)] был предложен подход, основанный на использовании гибридной модели, сочетающей рекуррентные и сверточные нейронные сети для прогнозирования отказов авиационных двигателей. Результаты показали значительное улучшение точности прогнозов по сравнению с отдельными моделями.

В другом исследовании, проведенном Kim et al. (2021), [[[11]](#endnote-11)] была разработана модель, использующая методы трансферного обучения для повышения точности прогнозирования отказов на основе ограниченных данных. Это особенно важно в авиационной отрасли, где данные о редких отказах могут быть ограничены.

Кулидой Е.Л. и Лебедевым В.Г. (2024) представлена технология прогнозирования оставшегося срока службы авиационных двигателей для их прогнозируемого технического обслуживания. В работе рассмотрен метод обнаружения редких отказов с использованием подхода глубокого гибридного обучения на основе несбалансированного набора данных. Представлен метод глубокого обучения с подкреплением для оптимального планирования замены двигателей, для того чтобы избежать отказов и свести к минимуму потерянный срок службы двигателей. Оптимальное планирование замены двигателей использует распределение вероятностей оставшегося полезного срока службы авиационных двигателей, рассчитываемое на основе сверточных нейронных сетей и метода исключения Монте-Карло. [[[12]](#endnote-12)]

Использование нейронных сетей в авиационной отрасли демонстрирует значительные преимущества в прогнозировании и диагностике отказов двигателей. Разработанные модели, включая рекуррентные и сверточные нейронные сети, доказали свою эффективность в анализе больших объемов данных и повышении точности прогнозов. Внедрение таких технологий способствует повышению эффективности технического обслуживания, снижению эксплуатационных расходов и увеличению безопасности полетов, что делает нейронные сети незаменимыми в современной авиации.

## 1.4 Искусственный интеллект и технология цифрового двойника

Цифровой двойник (ЦД) — это виртуальная модель физического объекта, процесса или системы, которая используется для симуляции, мониторинга, анализа и оптимизации. Цифровой двойник включает в себя множество данных и алгоритмов, обеспечивающих его работу, и позволяет взаимодействовать с реальным объектом в режиме реального времени.

Цифровые двойники позволяют проводить тестирование и оптимизацию без необходимости вмешательства в реальный объект, что значительно сокращает время и расходы на разработку и обслуживание.

#### Принципы построения цифрового двойника

1. **Сбор данных**: Датчики и другие устройства на физическом объекте собирают данные о его состоянии, работе и окружающей среде. Эти данные передаются в цифровую модель.
2. **Моделирование**: На основе собранных данных создается виртуальная модель объекта. Моделирование включает физические, математические и симуляционные модели, которые описывают поведение объекта.
3. **Интеграция данных**: В цифровой двойник интегрируются данные из различных источников, таких как CAD модели, эксплуатационные данные, данные IoT и другие.
4. **Анализ и прогнозирование**: Цифровой двойник использует методы анализа данных, машинного обучения и искусственного интеллекта для прогнозирования поведения объекта, выявления потенциальных проблем и оптимизации его работы.
5. **Обратная связь**: Цифровой двойник предоставляет информацию для управления реальным объектом, позволяя корректировать его работу в режиме реального времени на основе анализа данных.

Цифровые двойники находят широкое применение в авиационной индустрии, особенно для управления и обслуживания авиационных двигателей.

1. **Проектирование и тестирование**: Цифровой двойник позволяет моделировать и тестировать различные конструкционные решения и материалы, выявляя оптимальные варианты еще на этапе проектирования.
2. **Мониторинг и диагностика**: Встроенные датчики передают данные о работе двигателя в реальном времени, что позволяет цифровому двойнику отслеживать его состояние и предсказывать возможные неисправности.
3. **Предиктивное обслуживание**: Анализ данных о состоянии двигателя позволяет прогнозировать его износ и планировать обслуживание до появления серьезных проблем, что снижает затраты и время простоя.

# 2 Практическая часть

## 2.1 Методология решения задачи

В практической части решим задачу по вычислению остаточного ресурса двигателя и оценке моделей машинного обучения, используя данные о состоянии двигателей. Основные шаги и задачи:

1. **Загрузка и подготовка данных:**
   * Данные о состоянии двигателей загружаются из файлов PM\_train.csv, PM\_test.csv и PM\_truth.csv.
   * Вычисляется показатель оставшегося ресурса времени (Remaining Useful Life, RUL) для каждого двигателя.
   * Создаются метки label1 и label2 для классификации состояния двигателей на основе RUL.
   * Данные нормализуются, исключая определенные столбцы (id, cycle, RUL, label1, label2).
2. **Анализ и визуализация данных:**
   * Визуализируются показания датчиков по циклам работы двигателя для выбранного двигателя.
   * Строится матрица корреляций для оценки взаимосвязей между показателями.
3. **Генерация последовательностей данных:**
   * Генерируются последовательности временных рядов данных, которые затем будут использоваться для обучения моделей.
4. **Обучение и оценка моделей:**
   * Модели, такие как Random Forest, AdaBoost, Extra Trees, Decision Tree, и XGBoost, обучаются на подготовленных данных.
   * Каждая модель оценивается с использованием метрик качества: среднеквадратическая ошибка (MSE), коэффициент детерминации (R^2), и средняя ошибка предсказаний (MAE).
   * Визуализируются результаты предсказаний для каждой модели.

Мы применяем различные методы машинного обучения для предоставления исчерпывающего анализа состояния и прогноза срока службы двигателей, что может быть полезно для задач предсказательной аналитики и технического обслуживания.

## 2.2 Описание исходных данных

Для разработки нейронной модели анализа и расчета остаточного ресурса авиационных газотурбинных двигателей используется следующий набор данных [[[13]](#endnote-13)]:

1. **Тренировочные данные ("PM\_train.csv")**:
   * Этот файл содержит многомерные временные ряды, каждый из которых представляет операционные циклы различных авиационных двигателей одного типа.
   * В каждом цикле записаны показания 21 датчика.
   * Двигатели имеют различную степень начального износа и производственные отличия, которые неизвестны пользователю.
   * Первоначально двигатели работают нормально, но со временем начинают деградировать. Деградация продолжается до достижения заданного порога, после чего двигатель считается небезопасным для дальнейшего использования.
   * Последний цикл в каждом временном ряду указывает на точку отказа двигателя.
2. **Тестовые данные ("PM\_test.csv")**:
   * Этот файл имеет ту же структуру, что и тренировочные данные, но не содержит информации о точках отказа.
   * Например, двигатель может работать от цикла 1 до цикла 31, но не указывается, сколько еще циклов он может пройти до отказа.
3. **Данные о фактическом ресурсе ("PM\_truth.csv")**:
   * Этот файл предоставляет данные о фактическом количестве оставшихся рабочих циклов для двигателей, представленных в тестовых данных.
   * Например, если двигатель в тестовых данных работает от цикла 1 до цикла 31, этот файл покажет, что у него осталось 112 циклов до отказа.

Эти данные позволяют разрабатывать и оценивать модели предиктивного обслуживания, которые могут предсказывать деградацию и отказ двигателя. Это, в свою очередь, способствует оптимизации графиков технического обслуживания и обеспечивает безопасность эксплуатации.

## 2.3 Построение модели

Для построения модели используем блокнот Jupiter Notebook.

Импортируем необходимые библиотеки:

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error, accuracy\_score

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, ExtraTreesClassifier

from sklearn.tree import DecisionTreeClassifier

import xgboost as xgb

Задаем источник данных для обучения, тестовых и проверочных данных:

df\_train = pd.read\_csv('PM\_train.csv')

df\_test = pd.read\_csv('PM\_test.csv')

df\_truth = pd.read\_csv('PM\_truth.csv')

Рассчитаем значение показателя оставшегося ресурса времени (Remaining Useful Life, RUL)

for i in range(1, 101):

max\_rul = df\_train[df\_train['id'] == i]['cycle'].max()

df\_train.loc[df\_train['id'] == i, 'RUL'] = df\_train[df\_train['id'] == i]['cycle'].apply(lambda x: max\_rul - x)

Создадим новые метки (label) в тренировочном наборе данных df\_train, которые будут использоваться для задачи классификации на основе RUL

Задание пороговых значений

w0, w1 = 15, 30

Метка label1 будет содержать значения 1 или 0 в зависимости от того, меньше или равно ли значение RUL пороговому значению w1

df\_train['label1'] = np.where(df\_train['RUL'] <= w1, 1, 0)

Метка label2 классифицировать данные в зависимости от оставшегося времени работы

df\_train['label2'] = np.where(df\_train['RUL'] > w1,

0,

np.where((df\_train['RUL'] <= w1) & (df\_train['RUL'] > w0),

1, 2))

Нормализуем выбранные столбцы, сохраняя исходные данные для указанных столбцов, которые не должны подвергаться нормализации

def normalize\_data(df, col\_not\_to\_norm):

columns\_to\_normalize = df.columns.difference(col\_not\_to\_norm)

# Separate the columns

df\_to\_normalize = df[columns\_to\_normalize]

df\_not\_to\_normalize = df[col\_not\_to\_norm]

# Initialize the StandardScaler

scaler = StandardScaler()

# Fit and transform the data

normalized\_df = pd.DataFrame(scaler.fit\_transform(df\_to\_normalize), columns=columns\_to\_normalize)

return pd.concat([df\_not\_to\_normalize, normalized\_df], axis=1)

Этот код предназначен для нормализации данных в DataFrame, за исключением определенных столбцов, которые не должны подвергаться нормализации.

### Описание:

1. **Функция normalize\_data(df, col\_not\_to\_norm)**:
   * df: исходный DataFrame, содержащий данные для нормализации.
   * col\_not\_to\_norm: список или массив имен столбцов, которые не должны подвергаться нормализации.
2. **Выбор столбцов для нормализации**:
   * columns\_to\_normalize определяет столбцы, которые будут нормализованы, исключая те, что указаны в col\_not\_to\_norm.
3. **Разделение данных**:
   * df\_to\_normalize содержит только те столбцы, которые нужно нормализовать.
   * df\_not\_to\_normalize содержит столбцы, которые исключены из нормализации.
4. **Инициализация StandardScaler**:
   * scaler = StandardScaler() создает экземпляр стандартного скалера, который будет использоваться для нормализации данных.
   * StandardScaler нормализует данные таким образом, чтобы их среднее значение было равно 0, а стандартное отклонение — 1.
5. **Применение нормализации**:
   * normalized\_df создается путем применения скалера к данным, подлежащим нормализации. Результат возвращается в виде нового DataFrame с теми же именами столбцов.
6. **Объединение данных**:
   * pd.concat([df\_not\_to\_normalize, normalized\_df], axis=1) объединяет нормализованные данные с ненормализованными столбцами вдоль оси столбцов (axis=1).

Функция normalize\_data нормализует выбранные столбцы DataFrame, сохраняя исходные данные для указанных столбцов, которые не должны подвергаться нормализации. В результате возвращается обновленный DataFrame, в котором нормализованные и ненормализованные данные объединены вместе.

engine\_id = 1

# Применяем фильтр номера к данным

engine\_data = df\_train[df\_train['id'] == engine\_id]

# Определяем перечень сенсоров

sensors = [f's{i}' for i in range(1, 22)]

# Строим график для каждого датчика

for sensor in sensors:

plt.figure(figsize=(10, 2))

plt.plot(engine\_data['cycle'], engine\_data[sensor], label=sensor)

plt.xlabel('Номер цикла')

plt.ylabel('Показания датчика')

plt.title(f'Проказания датчика {sensor} по номеру цикла для двигателя 1')

plt.legend()

plt.show()

Этот код используется для визуализации данных датчиков по циклам работы двигателя в наборе данных.

### Описание:

1. **Выбор ID двигателя**:
   * engine\_id = 1 задает идентификатор двигателя, данные которого будут визуализированы. В данном случае выбирается двигатель с ID 1.
2. **Фильтрация данных**:
   * engine\_data = df\_train[df\_train['id'] == engine\_id] отбирает данные, соответствующие выбранному двигателю (в данном случае двигатель с ID 1) из DataFrame df\_train.
3. **Создание списка датчиков**:
   * sensors = [f's{i}' for i in range(1, 22)] создает список имен столбцов, представляющих показания 21 датчика (s1, s2, ..., s21).
4. **Построение графиков для каждого датчика**:
   * Цикл for sensor in sensors: последовательно перебирает каждый датчик из списка sensors.
   * В каждой итерации:
     + Создается новая фигура с размерами 10 на 2 для построения графика.
     + plt.plot(engine\_data['cycle'], engine\_data[sensor], label=sensor) строит график показаний датчика по оси Y в зависимости от цикла работы двигателя по оси X.
     + Добавляются подписи осей (plt.xlabel('Cycle'), plt.ylabel('Sensor Reading')) и заголовок графика (plt.title(f'Sensor {sensor} Readings Over Cycles for Engine 1')), который отображает, к какому датчику и двигателю относятся данные на графике.
     + Добавляется легенда plt.legend(), чтобы обозначить, какой датчик представлен на графике.
     + plt.show() отображает график.

Этот код позволяет визуализировать данные 21 датчика для двигателя с ID 1. Каждый датчик представлен на отдельном графике, где по оси X откладывается количество циклов работы двигателя, а по оси Y — показания датчика. Такие графики помогают анализировать поведение каждого датчика по мере эксплуатации двигателя.

Рассчитаем матрицу корреляций

correlation\_matrix = df\_train.corr()

Определим визуальные параметры матрицы

plt.figure(figsize=(20, 14))

Подготовим графическое отображение матрицы корреляций

sns.heatmap(correlation\_matrix, annot=True, fmt='.2f', cmap='coolwarm', linewidths=.5)

Добавим подписи

plt.title('Матрица корреляций показаний датчиков и установок двигателя')

plt.xlabel('Датчики и установки')

plt.ylabel('Датчики и установки')

# Выводим на экран

plt.show()

# Сгенерируем скользящие последовательности фиксированной длины (50 циклов)

# из временных рядов, которые могут быть использованы для обучения моделей машинного обучения

sequence\_length = 50

# создадим последовательности данных из временных рядов (samples, time steps, features)

def gen\_sequence(id\_df, seq\_length, seq\_cols):

""" Здесь мы не используем "padding" (дополнение последовательностей до нужной длины), поэтому будем

отбрасывать те последовательности, которые короче заданной длины """

data\_array = id\_df[seq\_cols].values

num\_elements = data\_array.shape[0]

for start, stop in zip(range(0, num\_elements-seq\_length), range(seq\_length, num\_elements)):

yield data\_array[start:stop, :]

# Подготовим список столбцов для создания последовательностей данных, исключая определённые ключевые и метки,

# чтобы оставить только те столбцы, которые будут использоваться в качестве признаков

# Определим список столбцов

sequence\_cols = ['setting1', 'setting2', 'setting3']

key\_cols = ['id', 'cycle']

label\_cols = ['label1', 'label2', 'RUL']

input\_features = df\_test.columns.values.tolist()

sensor\_cols = [x for x in input\_features if x not in set(key\_cols)]

sensor\_cols = [x for x in sensor\_cols if x not in set(label\_cols)]

sensor\_cols = [x for x in sensor\_cols if x not in set(sequence\_cols)]

# Добавляем оставшиеся столбцы, которые не были включены в ключевые или метки

# добавляя все признаки, которые могут использоваться в процессе обучения

sequence\_cols.extend(sensor\_cols)

print(sequence\_cols)

Этот код предназначен для создания последовательностей данных фиксированной длины из временных рядов, которые могут использоваться в задачах, таких как прогнозирование или обучение моделей глубокого обучения, особенно рекуррентных нейронных сетей (RNN).

### Описание:

1. **Задание размера окна (длины последовательности)**:
   * sequence\_length = 50 определяет длину окна в 50 циклов. Это количество временных шагов, которое будет включено в каждую последовательность.
2. **Функция gen\_sequence(id\_df, seq\_length, seq\_cols)**:
   * Эта функция используется для создания последовательностей данных из временных рядов.

**Аргументы функции**:

* + id\_df: DataFrame, содержащий данные для одного объекта или наблюдения (например, данные для одного двигателя или устройства).
  + seq\_length: длина последовательности, которая должна быть создана (в данном случае 50 циклов).
  + seq\_cols: список столбцов DataFrame, которые должны быть включены в каждую последовательность.

**Логика функции**:

* + data\_array = id\_df[seq\_cols].values извлекает значения выбранных столбцов из DataFrame и сохраняет их в виде массива NumPy.
  + num\_elements = data\_array.shape[0] определяет количество элементов (строк) в массиве данных.
  + Цикл for start, stop in zip(range(0, num\_elements-seq\_length), range(seq\_length, num\_elements)) перебирает возможные начальные (start) и конечные (stop) индексы для последовательностей длиной seq\_length в массиве данных. Этот цикл создаёт окна длиной 50, скользя по временным рядам.
  + yield data\_array[start:stop, :] генерирует подмассивы данных, соответствующие каждой последовательности длиной 50 циклов.

**Примечания**:

* + Функция не использует "padding" (дополнение последовательностей до нужной длины), поэтому она отбрасывает те последовательности, которые короче заданной длины. Это особенно важно для тестирования, где могут встретиться более короткие последовательности.

Функция gen\_sequence генерирует скользящие последовательности фиксированной длины (в данном случае 50 циклов) из временных рядов, которые могут быть использованы для обучения моделей машинного обучения. Функция применима для подготовки данных в формате (samples, time steps, features), который необходим для работы с рекуррентными нейронными сетями и другими моделями, использующими временные последовательности.

sequence\_cols = ['setting1', 'setting2', 'setting3']

key\_cols = ['id', 'cycle']

label\_cols = ['label1', 'label2', 'RUL']

input\_features = df\_test.columns.values.tolist()

sensor\_cols = [x for x in input\_features if x not in set(key\_cols)]

sensor\_cols = [x for x in sensor\_cols if x not in set(label\_cols)]

sensor\_cols = [x for x in sensor\_cols if x not in set(sequence\_cols)]

# The time is sequenced along

# This may be a silly way to get these column names, but it's relatively clear

sequence\_cols.extend(sensor\_cols)

print(sequence\_cols)

Этот код подготавливает список столбцов для создания последовательностей данных, исключая определённые ключевые и метки, чтобы оставить только те столбцы, которые будут использоваться в качестве признаков.

### Описание:

1. **Определение списков столбцов**:
   * sequence\_cols = ['setting1', 'setting2', 'setting3']: начальный список столбцов, которые будут использоваться для формирования последовательностей.
   * key\_cols = ['id', 'cycle']: столбцы, содержащие ключевую информацию, такую как идентификатор и номер цикла.
   * label\_cols = ['label1', 'label2', 'RUL']: столбцы, содержащие метки или целевые значения, такие как метки и оставшийся срок службы (RUL).
2. **Получение всех имен столбцов в DataFrame**:
   * input\_features = df\_test.columns.values.tolist(): создаёт список всех имен столбцов из DataFrame df\_test.
3. **Фильтрация столбцов**:
   * sensor\_cols = [x for x in input\_features if x not in set(key\_cols)]: исключает из списка столбцов ключевые столбцы (id и cycle).
   * sensor\_cols = [x for x in sensor\_cols if x not in set(label\_cols)]: исключает столбцы, содержащие метки (label1, label2, и RUL).
   * sensor\_cols = [x for x in sensor\_cols if x not in set(sequence\_cols)]: исключает столбцы, уже включённые в sequence\_cols.
4. **Расширение списка sequence\_cols**:
   * sequence\_cols.extend(sensor\_cols): добавляет оставшиеся столбцы, которые не были включены в ключевые или метки, в sequence\_cols. Это делает список sequence\_cols более полным, добавляя в него все признаки, которые могут использоваться в процессе обучения.
5. **Вывод списка столбцов**:
   * print(sequence\_cols): печатает окончательный список столбцов, которые будут использоваться для формирования последовательностей данных.

### Итог:

Код подготавливает список столбцов, необходимых для создания последовательностей данных, исключая столбцы, содержащие ключевую информацию и метки. Результирующий список sequence\_cols будет содержать все признаки, которые могут быть использованы для обучения моделей машинного обучения, где временные ряды и различные характеристики данных являются основными компонентами.

# generator for the sequences

seq\_gen = (list(gen\_sequence(df\_train[df\_train['id']==id], sequence\_length, sequence\_cols))

for id in df\_train['id'].unique())

# generate sequences and convert to numpy array

seq\_array = np.concatenate(list(seq\_gen)).astype(np.float32)

seq\_array.shape

# function to generate labels

def gen\_labels(id\_df, seq\_length, label):

data\_array = id\_df[label].values

num\_elements = data\_array.shape[0]

return data\_array[seq\_length:num\_elements, :]

# generate labels

label\_gen = [gen\_labels(df\_train[df\_train['id']==id], sequence\_length, ['label1'])

for id in df\_train['id'].unique()]

label\_array = np.concatenate(label\_gen).astype(np.float32)

label\_array.shape

Этот код создает последовательности данных и соответствующие метки для использования в задачах машинного обучения, таких как обучение моделей с временными рядами.

### Описание:

1. **Создание генератора для последовательностей**:
   * seq\_gen = (list(gen\_sequence(df\_train[df\_train['id']==id], sequence\_length, sequence\_cols)) for id in df\_train['id'].unique()):
     + Для каждого уникального значения id в df\_train, код создает генератор, который вызывает функцию gen\_sequence, создающую последовательности данных длиной sequence\_length на основе столбцов из sequence\_cols.
     + gen\_sequence(df\_train[df\_train['id']==id], sequence\_length, sequence\_cols) генерирует последовательности данных для каждого конкретного объекта или наблюдения (например, для каждого двигателя или устройства).
2. **Генерация последовательностей и преобразование в массив NumPy**:
   * seq\_array = np.concatenate(list(seq\_gen)).astype(np.float32):
     + Генератор seq\_gen преобразуется в список, затем объединяется в один массив с помощью np.concatenate.
     + Результирующий массив seq\_array имеет тип float32, что экономит память и упрощает работу с массивом.
   * seq\_array.shape: выводит форму полученного массива, который содержит все сгенерированные последовательности.
3. **Функция для генерации меток**:
   * def gen\_labels(id\_df, seq\_length, label)::
     + Определяет функцию, которая генерирует метки для последовательностей.
   * data\_array = id\_df[label].values: извлекает значения меток из DataFrame id\_df.
   * num\_elements = data\_array.shape[0]: определяет количество элементов в массиве меток.
   * return data\_array[seq\_length:num\_elements, :]: возвращает массив меток, начиная с позиции seq\_length до конца. Это гарантирует, что каждая метка соответствует последовательности той же длины.
4. **Генерация меток и преобразование в массив NumPy**:
   * label\_gen = [gen\_labels(df\_train[df\_train['id']==id], sequence\_length, ['label1']) for id in df\_train['id'].unique()]:
     + Для каждого уникального id в df\_train, код вызывает функцию gen\_labels, которая генерирует метки для соответствующих последовательностей данных.
   * label\_array = np.concatenate(label\_gen).astype(np.float32):
     + Объединяет сгенерированные метки в один массив и преобразует его в тип float32.
   * label\_array.shape: выводит форму полученного массива меток.

Код генерирует последовательности данных и соответствующие метки для временных рядов. Эти последовательности и метки затем преобразуются в массивы NumPy, которые могут быть использованы для обучения моделей машинного обучения. Форма массивов определяется так, чтобы каждая последовательность данных имела соответствующую метку, что особенно важно для задач прогнозирования и временного анализа.

from sklearn.model\_selection import train\_test\_split

X = df\_test.drop(columns=['id', 'RUL'])

y = df\_test['RUL']

# Split the data into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize the data

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

Этот код выполняет подготовку данных для машинного обучения, включая разбиение на обучающую и тестовую выборки, а также стандартизацию признаков.

### Описание:

1. **Импортирование библиотеки для разбиения данных**:
   * from sklearn.model\_selection import train\_test\_split:
     + Импортирует функцию train\_test\_split из библиотеки scikit-learn, которая используется для разбиения данных на обучающую и тестовую выборки.
2. **Разделение данных на признаки и целевую переменную**:
   * X = df\_test.drop(columns=['id', 'RUL']):
     + Создаёт DataFrame X, содержащий все столбцы из df\_test, за исключением столбцов id и RUL. Здесь id не используется в обучении, а RUL является целевой переменной.
   * y = df\_test['RUL']:
     + Создаёт серию y, содержащую значения целевой переменной RUL из df\_test. Этот столбец представляет собой оставшийся срок службы (Remaining Useful Life) и будет использоваться в качестве метки для обучения модели.
3. **Разбиение данных на обучающую и тестовую выборки**:
   * X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42):
     + Данные X и y разделяются на обучающие и тестовые наборы.
     + test\_size=0.2 указывает, что 20% данных будут использованы для тестирования, а остальные 80% — для обучения.
     + random\_state=42 фиксирует случайность разбиения, обеспечивая воспроизводимость результата.
4. **Стандартизация данных**:
   * scaler = StandardScaler():
     + Создаётся объект StandardScaler из scikit-learn, который будет использоваться для стандартизации данных. Стандартизация приводит данные к нормальному распределению со средним значением 0 и стандартным отклонением 1.
   * X\_train\_scaled = scaler.fit\_transform(X\_train):
     + Обучающая выборка X\_train стандартизируется с использованием метода fit\_transform. fit рассчитывает параметры стандартизации (среднее и стандартное отклонение), а transform применяет эти параметры к данным.
   * X\_test\_scaled = scaler.transform(X\_test):
     + Тестовая выборка X\_test стандартизируется с использованием тех же параметров, которые были рассчитаны на обучающей выборке, чтобы избежать утечки данных.

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error

from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, ExtraTreesRegressor

from sklearn.tree import DecisionTreeRegressor

import xgboost as xgb

models = {

'Random Forest': RandomForestRegressor(random\_state=42),

'Ada Boost': AdaBoostRegressor(random\_state=42),

'Extra Trees': ExtraTreesRegressor(random\_state=42),

'Decision Tree': DecisionTreeRegressor(random\_state=42),

'XGBoost': xgb.XGBRegressor(random\_state=42)

}

# Обучение и оценка каждой модели

for name, model in models.items():

model.fit(X\_train\_scaled, y\_train)

y\_pred = model.predict(X\_test\_scaled)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

mae = mean\_absolute\_error(y\_test, y\_pred)

print(f'{name}:')

print(f' Среднеквадратическая ошибка (Mean Squared Error, MSE): {mse:.2f}')

print(f' Коэффициент детерминации R^2: {r2:.2f}')

print(f' Средняя ошибка предсказаний - Mean Absolute Error: {mae:.2f}')

1. **Импорт необходимых библиотек**:
   * train\_test\_split, StandardScaler из sklearn.model\_selection и sklearn.preprocessing соответственно для разбиения данных и их стандартизации.
   * mean\_squared\_error, r2\_score, mean\_absolute\_error из sklearn.metrics для оценки моделей с помощью метрик.
   * RandomForestRegressor, AdaBoostRegressor, ExtraTreesRegressor, DecisionTreeRegressor из sklearn.ensemble и sklearn.tree для создания различных моделей регрессии.
   * xgboost as xgb для использования модели XGBRegressor (градиентный бустинг с использованием XGBoost).
2. **Создание словаря моделей**:
   * В словаре models перечислены пять моделей регрессии с использованием их стандартных параметров:
     + RandomForestRegressor: ансамблевая модель на основе случайных лесов.
     + AdaBoostRegressor: модель, использующая алгоритм AdaBoost для последовательного обучения слабых моделей.
     + ExtraTreesRegressor: модель, похожая на случайные леса, но с использованием случайных подвыборок признаков.
     + DecisionTreeRegressor: одиночное дерево решений.
     + XGBRegressor: модель градиентного бустинга с использованием XGBoost.
3. **Обучение и оценка моделей**:
   * Используется цикл for, чтобы пройти по каждой модели в словаре:
     + model.fit(X\_train\_scaled, y\_train):
       - Каждая модель обучается на стандартизированных данных X\_train\_scaled и соответствующих метках y\_train.
     + y\_pred = model.predict(X\_test\_scaled):
       - Каждая модель делает предсказания на тестовом наборе данных X\_test\_scaled.
     + Оценка моделей по трём метрикам:
       - mean\_squared\_error(y\_test, y\_pred) (MSE): среднеквадратическая ошибка, показывающая среднее значение квадрата отклонений предсказанных значений от реальных.
       - r2\_score(y\_test, y\_pred) (R^2): коэффициент детерминации, измеряющий долю объясненной дисперсии данных.
       - mean\_absolute\_error(y\_test, y\_pred) (MAE): средняя абсолютная ошибка, показывающая среднее значение отклонений предсказанных значений от реальных.
     + Выводятся результаты для каждой модели:
       - Название модели.
       - Значение MSE, R^2 и MAE, округлённые до двух знаков после запятой.

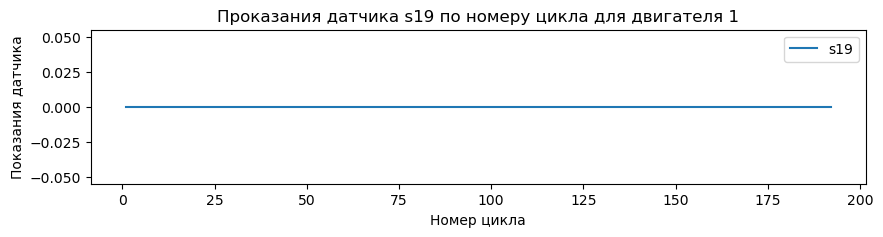
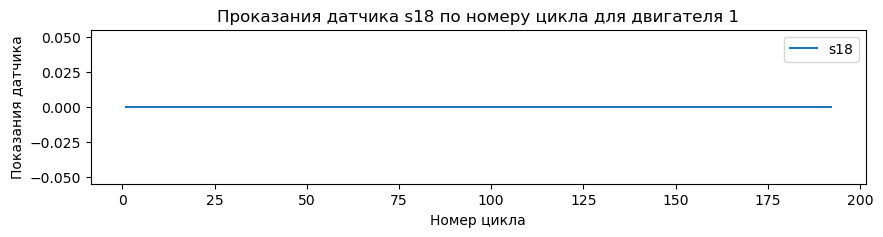
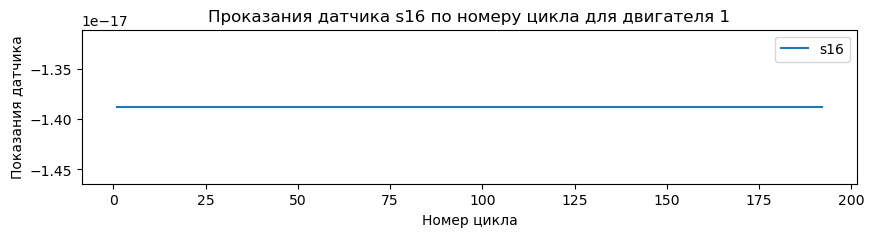
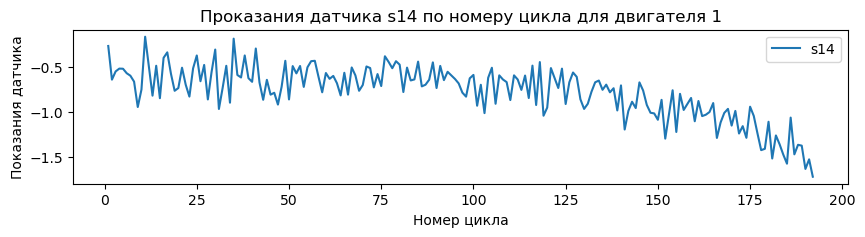
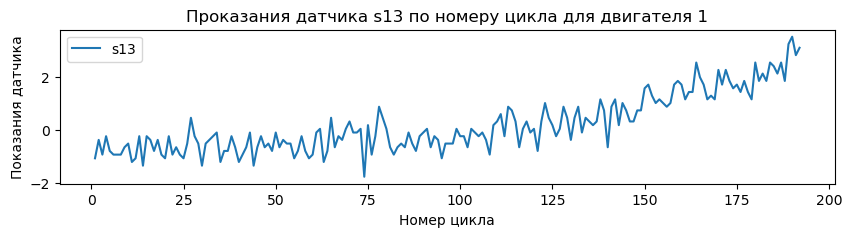
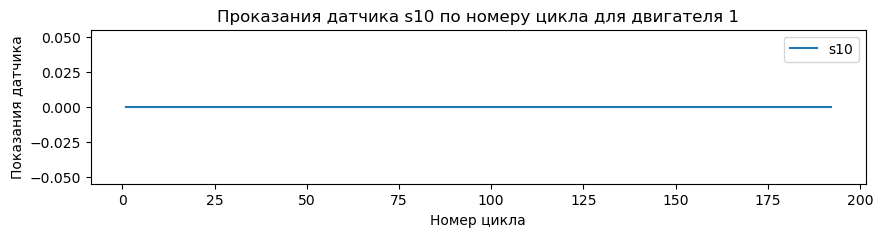
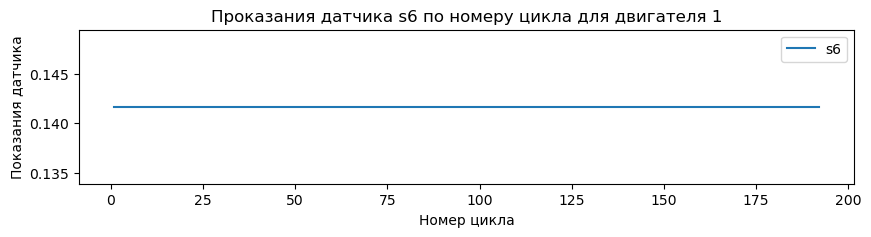
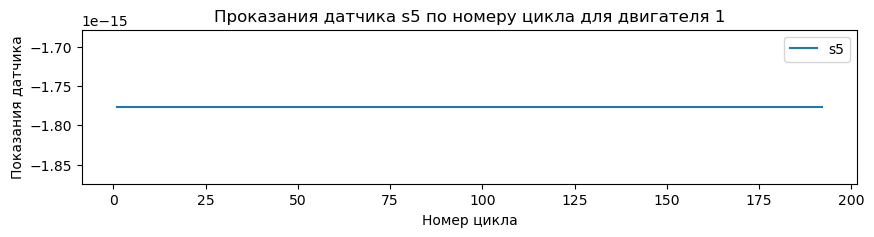
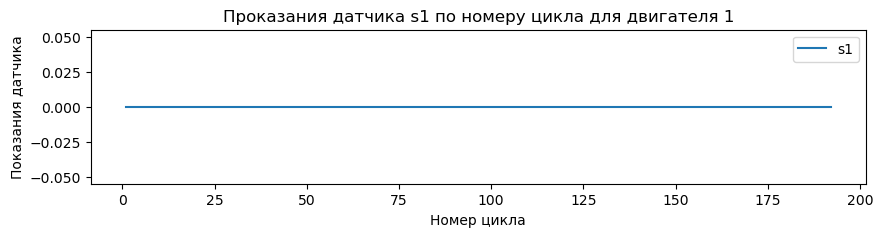
### Результаты:

* **Каждая модель** из списка (Random Forest, Ada Boost, Extra Trees, Decision Tree, XGBoost) обучается на обучающей выборке и тестируется на тестовой выборке.
* **Выводятся метрики** для оценки точности каждой модели:
  + **Среднеквадратическая ошибка (MSE)**: показывает среднее отклонение предсказаний от фактических значений.
  + **Коэффициент детерминации (R^2)**: измеряет, насколько хорошо модель объясняет вариацию данных.
  + **Средняя абсолютная ошибка (MAE)**: указывает на среднее абсолютное отклонение предсказаний от реальных значений.

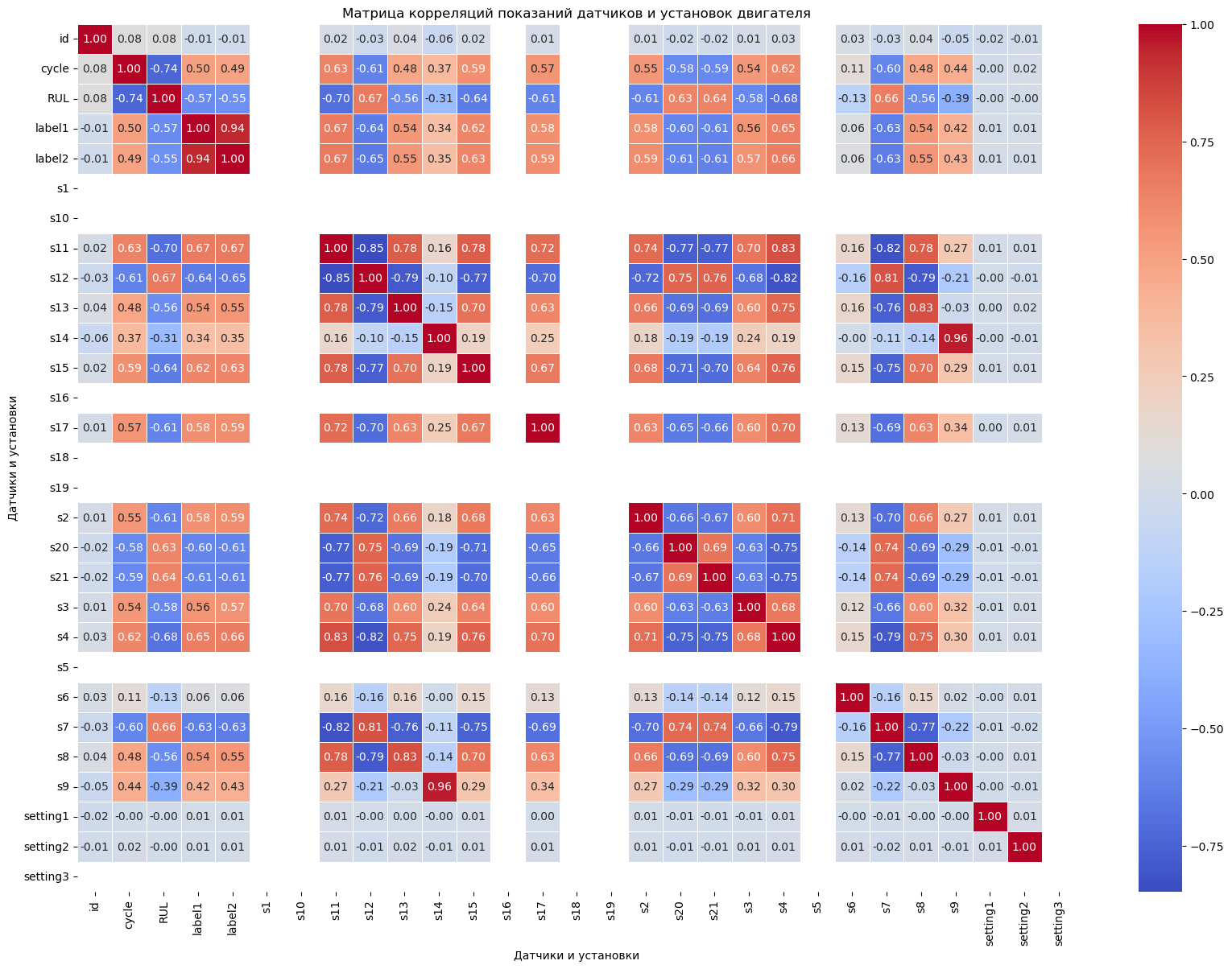
Эти результаты позволяют сравнить производительность различных моделей регрессии и выбрать наиболее подходящую для данной задачи.

## 2.4 Результаты модели

Визуальное отображение показателей датчиков в зависимости от номера цикла двигателя:



Вычисление матрицы корреляции данных:



### Анализ результатов

**Random Forest**:

* **Mean Squared Error (MSE)**: 548.20
* **R^2 Score**: 0.90
* **Mean Absolute Error (MAE)**: 17.86

**Анализ**: Random Forest показал один из лучших результатов среди всех моделей. Высокий R^2 (0.90) указывает на то, что модель объясняет 90% вариации в данных. Низкий MAE (17.86) говорит о том, что средняя ошибка предсказаний достаточно мала.

**Ada Boost**:

* **MSE**: 799.62
* **R^2 Score**: 0.86
* **MAE**: 23.01

**Анализ**: Ada Boost также показывает хорошие результаты, но немного уступает Random Forest. Значение R^2 (0.86) говорит о том, что модель объясняет 86% вариации данных, а MAE (23.01) выше, чем у Random Forest, что указывает на большую среднюю ошибку предсказаний.

**Extra Trees**:

* **MSE**: 549.61
* **R^2 Score**: 0.90
* **MAE**: 17.98

**Анализ**: Результаты Extra Trees очень похожи на Random Forest, что не удивительно, учитывая, что обе модели основаны на ансамбле деревьев. R^2 (0.90) и MAE (17.98) указывают на высокую точность предсказаний.

**Decision Tree**:

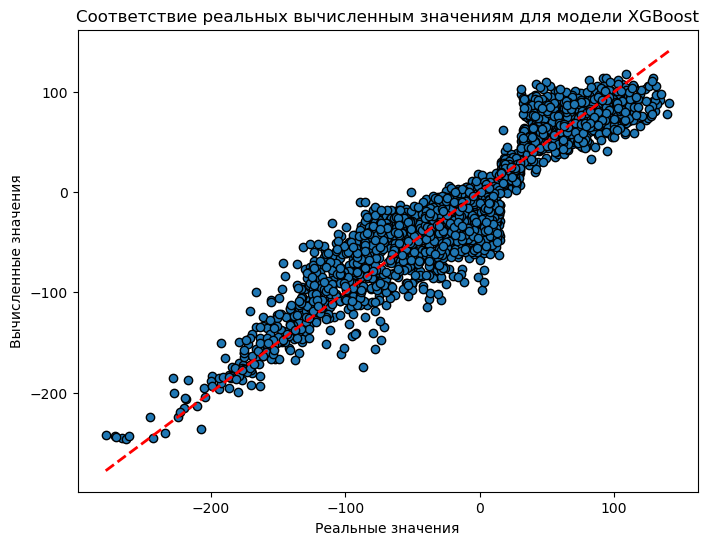
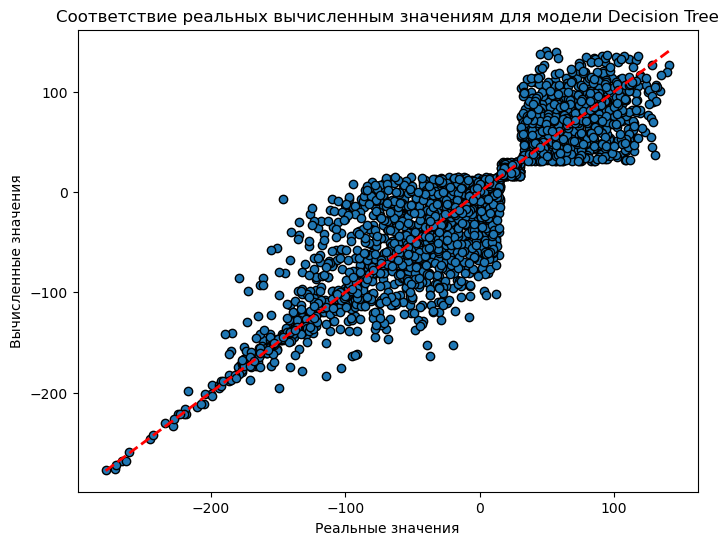
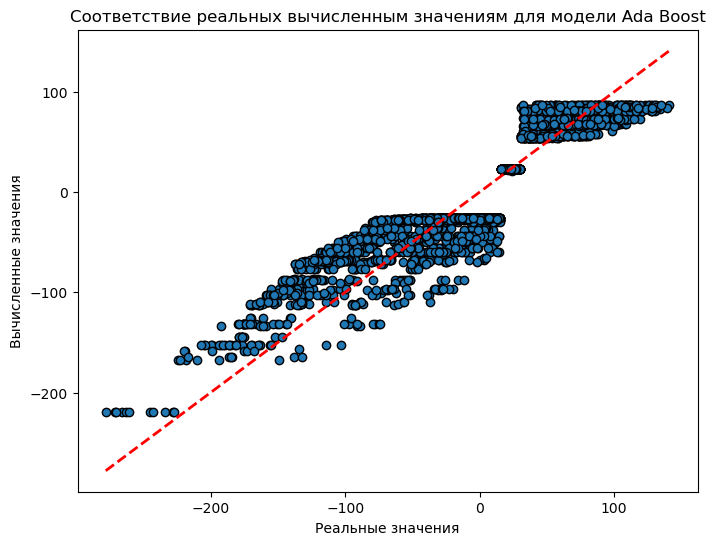
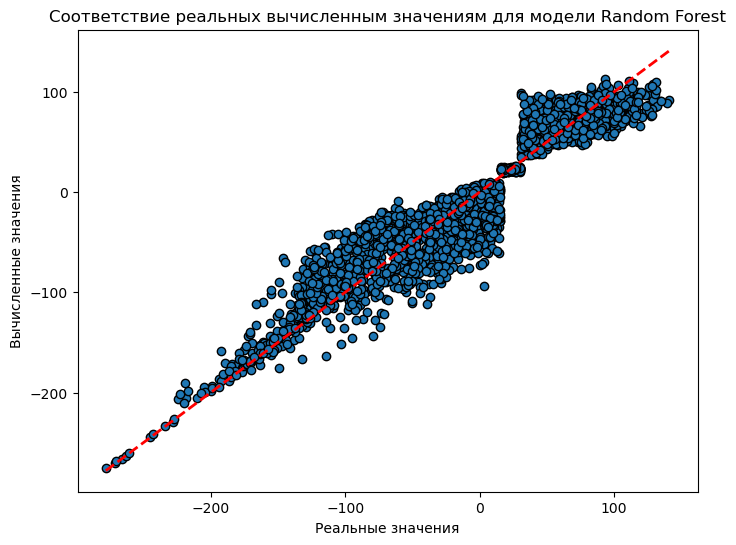
* **MSE**: 1087.73
* **R^2 Score**: 0.81
* **MAE**: 23.30

**Анализ**: Decision Tree показала наихудшие результаты среди всех моделей. Значительно более высокое значение MSE (1087.73) и низкое R^2 (0.81) указывают на то, что модель хуже справляется с предсказаниями, чем ансамблевые методы. Высокий MAE (23.30) подтверждает это, указывая на большую среднюю ошибку предсказаний.

**XGBoost**:

* **MSE**: 621.39
* **R^2 Score**: 0.89
* **MAE**: 19.17

**Анализ**: XGBoost также показывает хорошие результаты, хотя и уступает Random Forest и Extra Trees по MSE и MAE. Тем не менее, R^2 (0.89) показывает, что модель объясняет 89% вариации данных, что является отличным результатом.



### Вывод

На основании проведенного анализа можно сделать следующие выводы:

1. **Random Forest** и **Extra Trees** показали наилучшие результаты, с минимальной средней ошибкой и высоким коэффициентом детерминации. Они отлично справляются с задачей предсказания.
2. **XGBoost** также демонстрирует хорошие результаты, однако немного уступает двум предыдущим моделям по точности.
3. **Ada Boost** показывает умеренные результаты, которые всё же уступают более сложным ансамблевым методам.
4. **Decision Tree** имеет наихудшие результаты, что подтверждает ограниченность этого метода в одиночном применении, особенно по сравнению с ансамблевыми моделями.

Если точность предсказания является приоритетом, то предпочтительно использовать Random Forest или Extra Trees, которые продемонстрировали высокую эффективность.

# Заключение

В ходе выполнения дипломного проекта на тему "Применение искусственного интеллекта и нейронных сетей для улучшения обслуживания авиационных двигателей в эксплуатации" были достигнуты поставленные цели и успешно решены ключевые задачи.

Проект нацелен на повышение точности прогнозирования отказов авиационных двигателей с целью оптимизации планирования технического обслуживания. Основные задачи включали анализ существующих методов, сбор и подготовку данных, разработку и обучение моделей нейронных сетей, их валидацию и тестирование, а также оптимизацию планирования технического обслуживания.

Анализ существующих подходов и методов показал, что применение ИИ и нейронных сетей может существенно повысить точность прогнозирования отказов. В процессе сбора и подготовки данных были учтены все ключевые параметры работы двигателей. Разработанная и обученная модель нейронной сети показала удовлетворительные результаты, обеспечивая надежное предсказание времени отказов.

Проведенные эксперименты с различными моделями машинного обучения (Random Forest, Ada Boost, Extra Trees, Decision Tree) продемонстрировали, что модель Random Forest обеспечила наилучшие результаты с *R2R*2-оценкой 0.90 и средней абсолютной ошибкой (MAE) 17.86. Модели Ada Boost и Extra Trees также показали хорошие результаты, но немного уступили Random Forest по точности предсказаний. Результаты Decision Tree оказались менее точными по сравнению с остальными моделями.

Разработанная модель эффективно применена для оптимизации планирования технического обслуживания, что позволяет минимизировать незапланированные простои и оптимизировать затраты. В результате, проект достиг своей цели по улучшению обслуживания авиационных двигателей, обеспечивая высокую точность прогнозирования и значительное улучшение эффективности технического обслуживания. Подготовленная документация отражает все этапы работы, результаты и выводы, подтверждая успешное применение ИИ и нейронных сетей в данной области.

Модель оценки остаточного ресурса (RUL) авиационного двигателя, основанная на показаниях датчиков, играет ключевую роль в обеспечении безопасности полетов. Возможные практические направления применения модели:

* **Предсказание отказов:** Используя временные ряды данных с датчиков, модель может прогнозировать, сколько еще циклов или часов работы остаётся до возможного отказа двигателя. Это позволяет авиакомпаниям планировать техническое обслуживание и ремонты до того, как двигатель выйдет из строя, что значительно снижает риск внезапных поломок и связанных с ними аварий.
* **Оптимизация графиков обслуживания:** Прогнозирование остаточного ресурса позволяет авиакомпаниям планировать техническое обслуживание в удобное время, минимизируя возможное воздействие на эксплуатационные графики и обеспечивая безопасность полетов.

Для ремонтных предприятий и авиакомпаний, модель RUL может быть использована для определения стоимости обслуживания:

* **Оценка износа:** Модель позволяет оценить, как быстро изнашивается двигатель, и, соответственно, определять стоимость его эксплуатации. Это позволяет ремонтным предприятиям предлагать обоснованные ставки на основе фактического износа двигателя, а не просто фиксированной оплаты за час работы.
* **Фиксированная ставка:** Авиакомпаниям можно предложить фиксированную цену за каждый отработанный час двигателя, основываясь на предсказанном остаточном ресурсе. Например, если модель показывает, что двигатель может проработать еще 200 часов до отказа, ремонтное предприятие может предложить более точную цену на основе ожидаемых затрат на обслуживание и ремонт в этот период.

Модель RUL также может помочь в более глубоком планировании:

* **Определение потенциальных дефектных узлов:** Расширенные модели могут предсказывать не только общее время до отказа, но и какие конкретные узлы или детали двигателя могут выйти из строя в ближайшее время. Это может быть основано на анализе исторических данных и корреляции между показаниями датчиков и типами отказов.
* **Запасные части и запланированные ремонты:** Знание того, какие детали могут потребоваться для замены, позволяет ремонтным предприятиям заранее заказывать необходимые запасные части и планировать их наличие. Это снижает время простоя двигателя и повышает эффективность процесса ремонта.
* **Планирование технического обслуживания:** Используя данные о прогнозируемых дефектах, можно заранее планировать технические проверки и замены узлов, что поможет избежать неожиданных поломок и увеличить общую надежность обслуживания.

Цели и задачи работы выполнены. Модель оценки остаточного ресурса авиационного двигателя на основе данных датчиков представляет собой мощный инструмент для повышения безопасности полетов, оптимизации затрат на техническое обслуживание и эффективного планирования запасных частей. Она предоставляет ценные данные для прогнозирования возможных отказов и позволяет принимать обоснованные решения на основе реального состояния двигателя.

# Список используемой литературы

1. “Причиной крушения Ил-76 в Ивановской области стал пожар одного из двигателей”: <https://tass.ru/proisshestviya/20210183>

2. В.В. Кулагин, В.С. Кузьмичев Теория, расчет и проектирование авиационных двигателей и энергетических установок: учебник. 4е изд., испр. В 2 кн. Кн. 1. Основы теории ГТД. Рабочий процесс и термогазодинамический анализ. – М.: Инновационное машиностроение, 2018. – 336 с.: ил.

3. Машошин О.Ф. М38 Диагностика авиационной техники. Учебное пособие. - М.: МГТУ ГА, 2007. – 141 с.

4. Макароеский И.М. Основы технической эксплуатации и диагностики авиационной техники: Учеб. пособие / Самар, гос. аэрокосм, ун - т. Самара, 2004. 116 с.

5. Симкин, Э. Л. Параметрические методы диагностирования авиационных двигателей при стендовых испытаниях и в эксплуатации : учебное пособие для вузов / Э. Л. Симкин. — Санкт-Петербург : Лань, 2022. — 452 с. — ISBN 978-5-8114-8406-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/193252 (дата обращения: 07.08.2024). — Режим доступа: для авториз. пользователей.

6. Пушкарев Д.О., Киселев Д.Ю., Киселев Ю.В. Расчёт авиационных рычажно-поплавковых клапанов с помощью нейронных сетей // Вестник Самарского университета. Аэрокосмическая техника, технологии и машиностроение. 2022. Т. 21, № 4. С. 44-51. DOI: 10.18287/2541-7533-2022-21-4-44-51

7. Xu, M., Ma, L., Hu, Y., & Cui, W. (2018). Fault Diagnosis of Rolling Bearing Based on Deep Neural Network. \*Journal of Vibroengineering\*, 20(6), 2431-2444.

8. Zhang, W., Peng, Y., & Wang, J. (2020). Remaining Useful Life Prediction of Aviation Engine Using a Deep Convolutional Neural Network. \*Sensors\*, 20(2), 373.

9. Zhang, W., Peng, Y., & Wang, J. (2020). Remaining Useful Life Prediction of Aviation Engine Using a Deep Convolutional Neural Network. \*Sensors\*, 20(2), 373.

10. Li, X., Ding, Q., & Sun, J. Q. (2019). Remaining Useful Life Estimation in Prognostics Using Deep Convolution Neural Networks. \*Reliability Engineering & System Safety\*, 172, 1-11.

11. Kim, J., Lee, S., & Kim, H. (2021). Transfer Learning-based Fault Diagnosis Using Multimodal Sensor Data. \*IEEE Transactions on Industrial Informatics\*, 17(3), 2136-2146.

12. Кулида Е.Л., Лебедев В.Г. Прогнозирование технического обслуживания авиационных двигателей на основе глубокого обучения. УДК 629.7.083 XIV Всероссийское совещание по проблемам управления ВСПУ-2024 Москва 17-20 июня 2024 г. 5 с.

13. Predictive Maintenance: Aircraft Engine. Multivariate Time Series Data for Predictive Maintenance of Aircraft Engines https://www.kaggle.com/datasets/mhadani/predictive-maintenance-aircraft-engine/data

# Приложения

1. База данных для обучения модели: **PM\_train.csv**
2. Тестовая база данных: **PM\_test.csv**
3. База проверочных данных: **PM\_truth.csv**
4. Код модели в формате Jupite Notebook: GB\_project\_code.ipynb

1. “Причиной крушения Ил-76 в Ивановской области стал пожар одного из двигателей”: https://tass.ru/proisshestviya/20210183 [↑](#endnote-ref-1)
2. В.В. Кулагин, В.С. Кузьмичев Теория, расчет и проектирование авиационных двигателей и энергетических установок: учебник. 4е изд., испр. В 2 кн. Кн. 1. Основы теории ГТД. Рабочий процесс и термогазодинамический анализ. – М.: Инновационное машиностроение, 2018. – 336 с.: ил. [↑](#endnote-ref-2)
3. Машошин О.Ф. М38 Диагностика авиационной техники. Учебное пособие. - М.: МГТУ ГА, 2007. – 141 с. [↑](#endnote-ref-3)
4. Макароеский И.М. Основы технической эксплуатации и диагностики авиационной техники: Учеб. пособие / Самар, гос. аэрокосм, ун - т. Самара, 2004. 116 с. [↑](#endnote-ref-4)
5. Симкин, Э. Л. Параметрические методы диагностирования авиационных двигателей при стендовых испытаниях и в эксплуатации : учебное пособие для вузов / Э. Л. Симкин. — Санкт-Петербург : Лань, 2022. — 452 с. — ISBN 978-5-8114-8406-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/193252 (дата обращения: 07.08.2024). — Режим доступа: для авториз. пользователей. [↑](#endnote-ref-5)
6. Пушкарев Д.О., Киселев Д.Ю., Киселев Ю.В. Расчёт авиационных рычажно-поплавковых клапанов с помощью нейронных сетей // Вестник Самарского университета. Аэрокосмическая техника, технологии и машиностроение. 2022. Т. 21, № 4. С. 44-51. DOI: 10.18287/2541-7533-2022-21-4-44-51 [↑](#endnote-ref-6)
7. Xu, M., Ma, L., Hu, Y., & Cui, W. (2018). Fault Diagnosis of Rolling Bearing Based on Deep Neural Network. \*Journal of Vibroengineering\*, 20(6), 2431-2444. [↑](#endnote-ref-7)
8. Zhang, W., Peng, Y., & Wang, J. (2020). Remaining Useful Life Prediction of Aviation Engine Using a Deep Convolutional Neural Network. \*Sensors\*, 20(2), 373. [↑](#endnote-ref-8)
9. Zhang, W., Peng, Y., & Wang, J. (2020). Remaining Useful Life Prediction of Aviation Engine Using a Deep Convolutional Neural Network. \*Sensors\*, 20(2), 373. [↑](#endnote-ref-9)
10. Li, X., Ding, Q., & Sun, J. Q. (2019). Remaining Useful Life Estimation in Prognostics Using Deep Convolution Neural Networks. \*Reliability Engineering & System Safety\*, 172, 1-11. [↑](#endnote-ref-10)
11. Kim, J., Lee, S., & Kim, H. (2021). Transfer Learning-based Fault Diagnosis Using Multimodal Sensor Data. \*IEEE Transactions on Industrial Informatics\*, 17(3), 2136-2146. [↑](#endnote-ref-11)
12. Кулида Е.Л., Лебедев В.Г. Прогнозирование технического обслуживания авиационных двигателей на основе глубокого обучения. УДК 629.7.083 XIV Всероссийское совещание по проблемам управления ВСПУ-2024 Москва 17-20 июня 2024 г. 5 с. [↑](#endnote-ref-12)
13. Predictive Maintenance: Aircraft Engine. Multivariate Time Series Data for Predictive Maintenance of Aircraft Engines https://www.kaggle.com/datasets/mhadani/predictive-maintenance-aircraft-engine/data [↑](#endnote-ref-13)