Санкт–Петербургский государственный университет

Жадан Кирилл Сергеевич

Выпускная квалификационная работа

Прогнозирование сбоев IoT-устройств для гибкого планирования их технического обслуживания

Уровень образования: бакалавриат
Направление 02.03.02 «Фундаментальная информатика и информационные технологии»
Основная образовательная программа СВ.5003.2016 «Программирование и информационные технологии»
Профиль «Автоматизация научных исследований»

Научный руководитель:
кандидат физ.-мат. наук, доцент
кафедры компьютерного моделирования и
многопроцессорных систем:
Корхов Владимир Владиславович

Рецензент:
ведущий программист-математик компании
ООО «Т-Системс РУС»
Дмитриев Михаил Викторович

Санкт-Петербург
2020 г.

Содержание

Введение

На сегодняшний день многие крупные компании нуждаются в оптимизации производственных процессов. Приоритетными вопросами все чаще оказываются следующие задачи: повышение качества продукции, эксплуатационная безопасность, а также снижения издержек. С внедрением технологий больших данных (Big Data), системы интернета вещей (Internet of Things, IoT) и машинного обучения, прогнозная аналитика стала перспективной и быстроразвивающейся областью.

Внедрение технологии PdM (predictive maintenance, предиктивное обслуживание) в рамках технического обслуживания и инжиниринга в основном направлено на две группы задач. К первой относятся прогнозирование возможных поломок: определение аномальных режимов работы агрегатов, предсказание безопасного остаточного срока службы компонентов. Ко второй группе следует отнести предотвращение возникновения отказа оборудования путем выполнения технического обслуживания. Контроль за возможным отказом системы позволяет планировать техническое обслуживание до того, как произойдет отказ, что позволяет сократить необходимость в реализации внепланового ремонта, а также минимизировать риск поломок оборудования.

Незапланированный простой оборудования может причинить ущерб любому предприятию. Важно поддерживать рабочее состояние используемого оборудования, чтобы обеспечить максимально эффективное использование ресурсов и минимизировать время дорогостоящих незапланированных простоев, а также риски, связанные с работоспособностью и безопасностью. Целью стратегии прогнозного обслуживания является увеличение полезного срока службы оборудования и предотвращение сбоев. Часто используется обнаружение аномалий в работе устройств, потому что с его помощью можно определить, когда устройство ведет себя не так, как ожидалось.

При внедрении подобных технологий необходимо организовать сбор и хранения важной для исследований информации. По мере развития средств промышленного интернета вещей, в частности, благодаря оснащению обо-

рудования различными датчиками, сбор данных о его техническом состоянии выполняется не периодически, а непрерывно, без приостановки эксплуатации оборудования. Так, современные разработки позволяют записывать огромные массивы данных с показателей датчиков, установленных на оборудовании прямо во время работы, а по завершению получать информацию о работе всех компонентов. Учитывая большой объем получаемой информации, становится очевидной необходимость развития автоматических систем анализа и обработки данных. Поэтому, решение указанных задач требует как развития инфраструктуры по сбору и хранению данных, так и разработки новых алгоритмических подходов анализа данных.

Постановка задачи

Целью данной работы является разработка модели и программного модуля, позволяющего в режиме реального времени по данным, полученным с IoT устройств, предсказывать Remaining Useful Life (RUL) - количество оставшихся рабочих циклов до отказа системы, т. е. количество рабочих циклов после последнего выполненного цикла, в течение которого оборудование будет продолжать работать.

Для достижения цели были поставлены следующие задачи:

1. обзор существующих подходов и практик по прогнозированию сбоев оборудования;
2. предобработка и анализ исходных данных;
3. описание моделей прогнозирования, выявление преимуществ и недостатков;
4. описание архитектуры рекуррентных нейронных сетей LSTM;
5. разработка модели машинного обучения, основанной на использовании системы LSTM;
6. сравнение результатов прогнозирования различных моделей;

7. разработка программного модуля по прогнозированию сбоев оборудования;

Обзор литературы

Существующие решения, используемые в условиях реального производства, основные идеи и возможности прогностического обслуживания представлены в работах [1], [2].

Описание возможных моделей для анализа временных рядов, их достоинства и недостатки, тонкости применения моделей на практике и теоретическое описание процесса прогнозирования содержит работа [3].

В [4], [5] сформулирована задача поиска аномальных режимов работы оборудования, обозначена проблематика исследуемой области и предложены варианты решения. Так, в работе [4] предложена методика на основе модели Хольта-Винтерса, позволяющая выявлять аномалии в системах управления посредством краткосрочных прогнозов для состояний систем, представленных в виде временного ряда. А в исследовательской работе по прогнозированию отказов оборудования в условиях малого количества поломок[5], предложена модель прогнозирования отказов оборудования на основе алгоритма машинного обучения Random Forest.

В данной работе основной областью исследований является применение рекуррентных нейронных сетей в области предиктивного обслуживания. В книге [6] подробно представляется описание строения нейронных сетей, основные архитектуры и способы их обучения.

Для решения поставленной задачи приведено исследование нейронных сетей с долгой краткосрочной памятью. Данная архитектура была разработана с целью борьбы с проблемой угасающего градиента и описана в [7], [8].

Подробный процесс моделирования данных и их описание опубликованы в работе [9].

Методы и средства, описывающие подготовку данных к анализу подробно рассматриваются в книге [10].

Полный функционал, инструменты для практической реализации моделей и их описание, приводятся в документации соответствующих библиотек для языка python [11] , [12].

Для реализации нейронных сетей, оценки качества их работы была использована библиотека Keras [13], нацеленная на оперативную работу с сетями глубокого обучения.

Глава 1. Обзор существующих решений

1.1 Интуитивный и нормативный подходы

На сегодняшний день основным методом прогнозного обслуживания является интуитивный подход к построению прогноза. Интуитивные (экспертные) методы основаны на использовании накопленных знаний экспертов об объекте прогнозирования и обобщении их мнений о развитии предмета исследований в будущем. Экспертные методы в большей степени соответствуют прогнозированию скачкообразных процессов. Технический специалист должен проводить самостоятельную работу над оценкой текущей ситуации и тенденций прогнозируемого объекта, используя показания датчиков.

Еще одним методом прогнозирования работ является нормативный подход. Используя нормативную документацию по соответствующему виду оборудования, можно составить прогноз необходимых работ и дальнейшего состояния оборудования.

Таким образом, точность прогноза зависит от внимательности и компетентности каждого специалиста. Такая степень влияния человеческого фактора влечет за собой неточные оценки, а вследствие этого появляется большая погрешность при распознавании аномальных признаков работы оборудования и построении прогноза. Неточное прогнозирование влечет за собой финансовые потери, сбои в работе и аварийные ситуации. Предприятиям требуется нанимать очень квалифицированный штат сотрудников, что требует больших вложений, а постоянная нехватка кадров затрудняет поиск специалистов.

1.2 Коммерческие предложения

В России в настоящее время нет программной реализации, способной решить поставленную задачу в полном объеме. Технологии интернета вещей только приобретают массовый характер, разработка инструментов для прогнозирования осложняется нехваткой исторических данных, на основе которых можно приступить к исследованию методов для построения оптимальной модели. В основном, текущие решения направлены на поиск аномалий в режимах работы оборудования, что является неполной характеристикой технического состояния.

На рынке it-решений существуют коммерческие предложения, предоставляющие инструменты для прогнозирования и анализа данных, однако для их использования требуется лицензия и необходимы постоянных затраты на ее обновление. К тому же подобные программы с нужным функционалом могут быть трудно адаптируемыми под стандарты и возможности отечественных предприятий. К примеру, существует система IBM Watson (IoT Platform) - управляемое облачное решение IoT, применимое для получения, обработки и анализа данных с устройств интернета вещей. Еще одно программное обеспечение, предлагающее решения для прогнозного обслуживания, Azure – облачная платформа компании Microsoft. Для использования подобного программного обеспечения требуется специалист, который обладает фундаментальными знаниями в машинном обучении, математической статистике и прогнозной аналитике.

Глава 2. Анализ данных

2.1 Источник данных

NASA опубликовало в открытом доступе данные “Prognostics and Health Management PHM08”, которые используются для прогнозирования отказов реактивных двигателей. Для моделирования данных было использовано программное обеспечение C-MAPSS (Commercial Modular Aero-Propulsion System Simulation), позволяющее изменять параметры, влияющие на состояние системы, и регистрировать выходные измерения датчиков.

2.2 Описание данных

Наборы данных состоят из нескольких многомерных временных рядов. Каждый набор данных делится на обучающие и тестовые выборки. Отдельный временной ряд исходит от различных двигателей одного типа, и представляет изменения различных характеристик: давлений, температур, скоростей вращения оборудования и т. д. Системы запускаются с различной степенью первоначального износа и производственными изменениями. Каждый двигатель имеет 21 датчик, которые собирают различные измерения, связанные с состоянием двигателя во время работы. Двигатель работает нормально в начале каждой временной серии и в какой-то момент во время серии возникает неисправность. В учебном наборе ошибка растет по величине до тех пор, пока не произойдет сбой системы. В тестовом наборе временной ряд заканчивается за некоторое время до сбоя системы. Размер обучающей выборки - 20631, размер тестовой выборки - 13096.

2.3 Предобработка исходных данных

Перед применением методов машинного обучения необходимо выполнить первичный анализ данных. Это позволит понять некоторые аспекты структуры данных и выявить видимые закономерности в них. К тому же, неинформативные или слабо информативные признаки могут существенно понизить эффективность модели. Поэтому предварительная обработка

данных является важным шагом в процессе интеллектуального анализа данных.

Из гистограммы признаков (Рис. 1) следует, что Sensor 1, Sensor 5, Sensor 6, Sensor 10, Sensor 16, Sensor 18, Sensor 19 — константные признаки и их можно не учитывать в обучении модели.

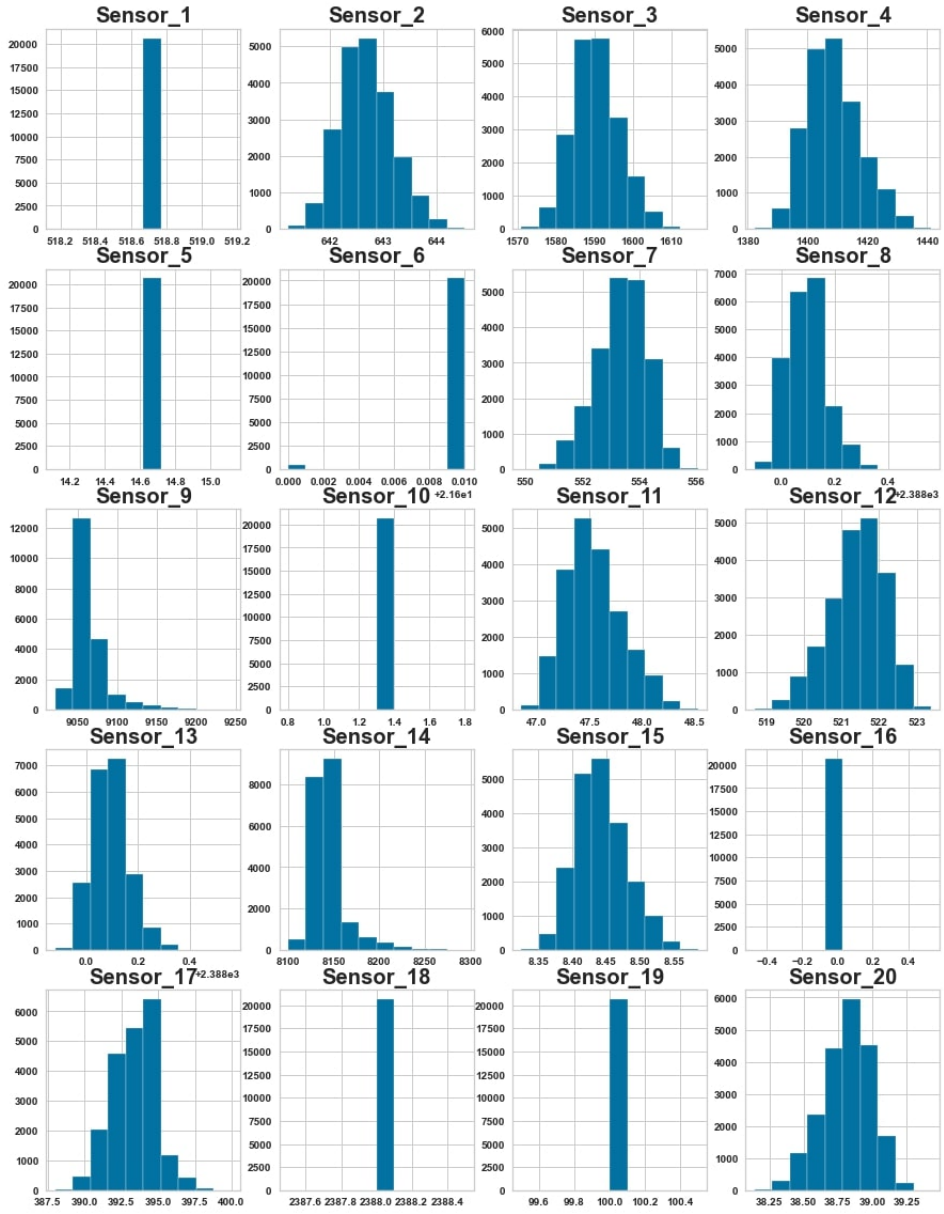

Рис. 1: Гистограмма признаков

С помощью корреляции Пирсона определены зависимости в данных. Корреляция Пирсона — это мера силы линейной взаимосвязи между двумя

случайными величинами, вычисляемая по следующей формуле:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}},$$

где $\bar{x}, \bar{y}$ — выборочные средние $x^n = (x_1, \cdots, x_n)$ и $y^n = (y_1, \cdots, y_n)$, $r_{xy} \in [-1, 1]$.

Статистическая взаимосвязь определена с помощью встроенной функции *corr*() из библиотеки pandas. На Рис. 2 представлена тепловая карта, описывающая взаимосвязи переменных. Высокая степень корреляции между Sensor 9 и Sensor 14 означает, что мы можем исключить один из них, так как они содержат схожую информацию.

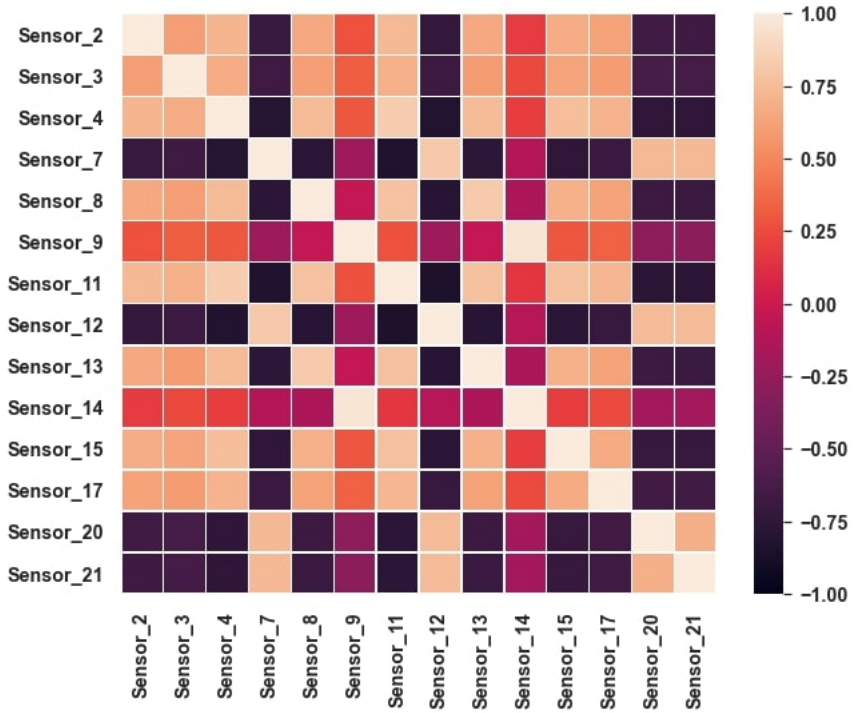

Рис. 2: Корреляционная матрица

На (Рис. 3) представлены показания датчиков во времени, каждому из двигателей соответствует определенный цвет. Можно заметить, что в начале эксплуатации оборудования значения датчиков изменяются незначительно, однако примерно за 50 циклов до поломки у показаний большинства датчиков выявляется тренд.

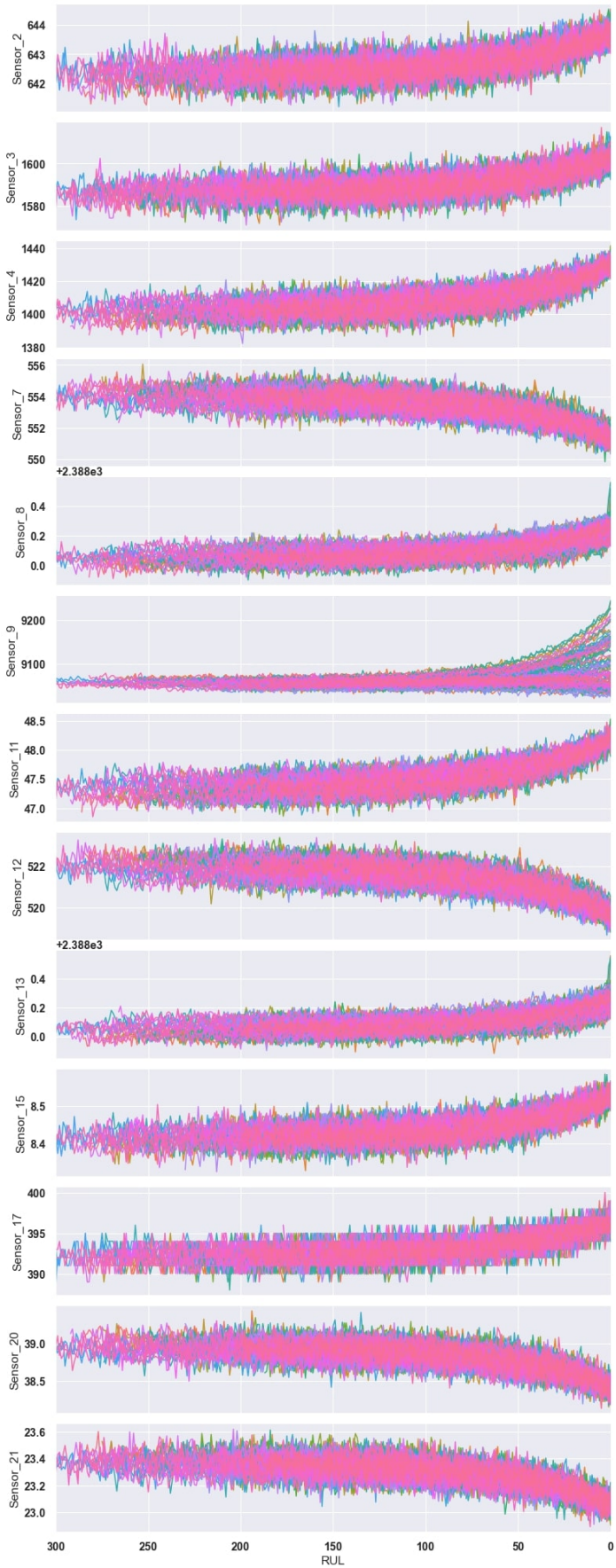

Рис. 3: Показания датчиков во времени

Для улучшения качества построения моделей произведена стандартизация показаний датчиков. Стандартизация набора данных включает в себя изменение масштаба распределения значений так, чтобы среднее наблюдаемых значений было 0, а стандартное отклонение -1. Для этого осуществляется Z-преобразование первичных данных по следующей формуле:

$$z_i = \frac{x_i - \bar{x}}{\sigma_x},$$

где z_i — стандартизованная величина для x_i, x_i — первичный результат, $\bar{x}$ — среднее арифметическое значение первичных результатов и σ_x — стандартное отклонение первичных результатов.

Z-преобразование выполнено с помощью функции $StandardScaler()$ из библиотеки sklearn.

Для масштабирования целевой переменной реализовано $MinMax$ преобразование — линейное преобразование данных в диапазон [0..1], где минимальное и максимальное масштабируемые значения соответствуют 0 и 1 соответственно:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}},$$

где x_{max}, x_{min} — максимальное и минимальное значение x^n.

Преобразования выполнены с помощью встроенной функции $MinMaxScaler()$ из библиотеки sklearn.

Глава 3. Методы прогнозирования в PdM

3.1 Методы и модели прогнозирования

На сегодняшний день предложено множество методов и моделей, которые успешно применяются в прогнозировании. Все они используют различный математический аппарат и различные подходы при реализации, однако эффективность этих методов зависит от конкретной решаемой задачи. Метод прогнозирования определяется как некоторый алгоритм, или последовательность действий, которые нужно совершить для получения модели прогнозирования. А модель прогнозирования это функциональное

представление, наилучшим образом описывающее исследуемый процесс и являющееся основой для получения его будущих значений. Модели прогнозирования можно условно разделить на два типа: модели временных рядов и модели предметной области[3].

В моделях предметной области используются математические модели, построенные на законах предметной области. Примерами могут служить прогнозирование уровня населения и уровня сахара в теле человека, для описания прогноза используется дифференциальное уравнение и система дифференциальных уравнений соответственно. В моделях временных рядов используются математические модели, в которых производится поиск зависимостей в массиве исторических данных, для прогнозирования будущих значений или значений целевой переменной. Такие модели универсальны для различных областей, ведь их общий вид не подвергается изменениям в зависимости от природы данных. К примеру, мы можем использовать нейронные сети для прогнозирования температуры воздуха, а после аналогичную модель применить для прогноза биржевых индексов.

При прогнозировании состояния оборудования и вычисления RUL требуется применять модели временных рядов, так как построенные модели должны иметь обобщающую способность и успешно справляться с решением поставленных задач в различных сферах деятельности.

В свою очередь, модели временных рядов можно разделить на статистические и структурные. В первом случае зависимость между целевым показателем и историческими данными определяется некоторым уравнением. Этот тип моделей включает в себя модели максимального правдоподобия, модели регрессии (линейной и нелинейной), авторегрессионные модели, модели с экспоненциальным сглаживанием.

В структурных же моделях зависимости между значениями задаются в виде некоторой структуры. Зачастую это подразумевает определение некоторых правил-переходов. К таким моделям можно отнести модели, основанные на классификационно-регрессионных деревьях, модели нейронных сетей, модели, использующие цепи Маркова.

В Таблице 1 приведено сравнение достоинств и недостатков методов прогнозирования.

Таблица 1: Сравнение достоинств и недостатков методов прогнозирования

Модели	Достоинства	Недостатки
Регрессионные модели	• Высокая скорость получения результата; • прозрачность моделирования; • вычислительная простата; • интерпретируемость модели.	• Чувствительность к входным данным; • сложность определения вида функциональной зависимости.
Авторегрессионные модели	• Высокая скорость получения результата; • прозрачность моделирования; • простота моделей; • широкий круг решения задач.	• Отсутствие возможности моделирования нелинейных процессов; • сложность вычисления параметров.
Модели регрессионных деревьев	• Маштабируемость; • интерпретируемость модели; • низкие требования к обучающей выборке; • широкий круг решения задач.	• Склонность к переобучению; • сложность выбора критерия остановки.
Модели экспоненциального сглаживания	• Простота моделей; • высокая скорость получения результата; • решение задач долгосрочного прогнозирования.	• Узконаправленность моделей; • отсутствие гибкости.

Нейросетевые модели	• Возможность моделирования нелинейных процессов; • масштабируемость; • адаптивность; • широкий круг решения задач.	• Отсутствие промежуточных вычислений; • сложность программной реализации.
Модели на базе цепей Маркова	• Простота моделей.	• Узконаправленность моделей.

В данной работе на основании адаптируемости и разнородности решаемых задач было решено рассмотреть нейросетевые модели. Так, модели на базе LSTM-сети могут быть применены, например, для формирования оперативных прогнозов по мере поступления новых данных, используя преимущество в виде долгосрочной памяти. Таким образом, появляется возможность построить адаптированную модель с большим потенциалом для дальнейшего масштабирования, что является преимуществом для прогнозирования сбоев устройств. Также, для сравнения результатов работы модели с данной архитектурой, способной находить долгосрочные зависимости, построены модели, которые строят предсказания независимо на каждом объекте из выборки:

- линейная регрессия;
- случайный лес.

3.2 Линейная регрессия

В задачах прогнозирования, базовой статистической моделью, способной работать с большим количеством факторов, является линейная регрессия. Существует множество задач, в которых требуется моделировать

и анализировать отношения между переменными. Для решения подобных задач используется регрессионный анализ, основная цель которого, определение зависимостей между целевой переменной и множеством внешних факторов (регрессоров). В основу модели линейной регрессии положено предположение, что между зависимой переменной и независимыми факторами определяется линейная зависимость. Регрессионная модель задается следующим уравнением:

$$y = f(x, b) = \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_k x_k = \sum_{j=1}^{k} \omega_j x_j = x^T \omega,$$

где $x^T = (x_1, x_2, \cdots, x_k)$ — вектор регрессоров, $\omega = (\omega_1, \omega_2, \cdots, \omega_k)^T$ — вектор столбец коэффициентов регрессии, k — количество факторов модели. В поставленной задаче под зависимой переменной понимается значение RUL, а вектор регрессоров - это показания оборудования, снятые при помощи IoT, k - количество исследуемых датчиков.

В основном коэффициенты регрессии находятся по методу максимального правдоподобия или по методу наименьших квадратов [14].

Данный метод очень чувствителен к входным данным и подвержен проблеме переобучения. В качестве решения предлагается регуляризация. К примеру, $L1$-регуляризация (1) приводит к отбору признаков, $L2$-регуляризация (2) позволяет избежать проблем мультиколлинеарности и переобучения. Подробный анализ и методы решения описаны в [15].

$$L_1 = \sum_{i=1}^{N} (y_i - \omega^T x_i) + \lambda \|\omega\| \tag{1}$$

$$L_2 = \sum_{i=1}^{N} (y_i - \omega^T x_i) + \lambda \|\omega\|^2 \tag{2}$$

3.3 Регрессионные деревья

Дерево принятия решений — алгоритм принятия решений, который используется в статистике, анализе данных и машинном обучении. Каж-

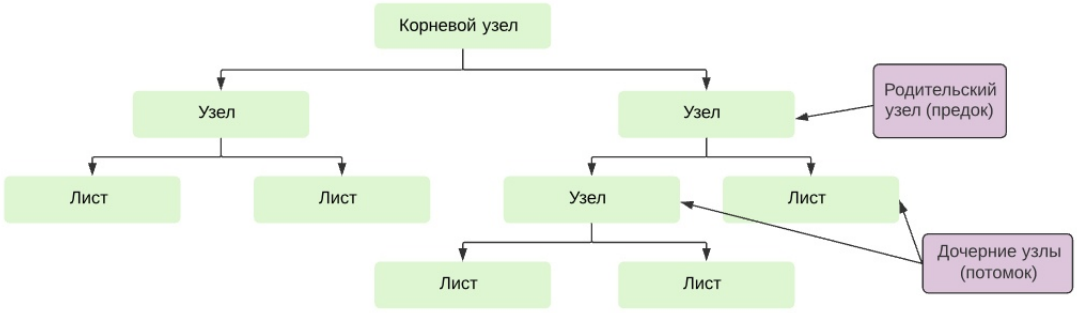

Рис. 4: Дерево принятия решений

дая внутренняя вершина содержит некоторую функцию от объекта, позволяющую определить, в какую из дочерних вершин нужно поместить рассматриваемый объект, а каждый лист представляет собой значение целевой переменной. (Рис. 4). Если решается задача регрессии прогноз в листе является вещественным числом, в нашей задаче — RUL. В основном на практике используются бинарные деревья, а условия во внутренних вершинах выбираются крайне простыми.

В машинном обучении решающее дерево строится жадно от корня к листьям. Сначала выбирается корень, который в общем случае разбивает выборку на k частей согласно выбранному правилу. Затем разбивается каждый из потомков этого корня и так далее. Дерево ветвится пока не выполнится критерий остановки. Существует много различных вариантов критерия остановки: ограничение максимальной глубины дерева, ограничение минимального числа объектов в листе, ограничение максимального количества листьев в дереве и т.д. [16]

3.3.1 Случайный лес

Одним из лучших способов объединения деревьев в композиции является случайный лес. Композиция — это объединение N алгоритмов $b_1(x) \cdots b_N(x)$ в один. Идея заключается в том, чтобы обучить алгоритмы

$b_1(x) \cdots b_N(x)$, а затем усреднить полученные от них ответы:

$$a(x) = \frac{1}{N} \sum_{n=1}^{N} b_n(x).$$

Алгоритм $a(x)$, который возвращает среднее, называется композицией N алгоритмов $b_1(x) \cdots b_N(x)$, а они сами называются базовыми алгоритмами.

Один из способов сделать базовые алгоритмы различными — обучать их на разных подвыборках. Бутстрап — один из популярных подходов к построению подвыборок. Он заключается в том, что из обучающей выборки длины l выбирают с возвращением l объектов. Также для уменьшения корреляции базовых алгоритмов используются:

- Беггинг: обучение базовых алгоритмов происходит на случайных подвыборках обучающей выборки [17];
- Метод случайных подпространств: очередной базовый алгоритм обучается на случайном подмножестве признаков [18].

Алгоритм построения случайного леса:

1. Построить с помощью бутстрапа N случайных подвыборок $\bar{X}_n, n = 1, \cdots, N$;
2. Каждая получившаяся подвыборка $\bar{X}_n$ используется как обучающая выборка для построения соответствующего решающего дерева $b_n(x)$. Причем на этапе выбора оптимального признака, по которому будет происходить разбиение, он ищется не среди всего множества признаков, а среди случайного подмножества размера q;
3. Построенные деревья объединяются в композицию:

$$a(x) = \frac{1}{N} \sum_{n=1}^{N} b_n(x).$$

3.4 Рекуррентные нейронные сети

Нейронные сети прямого распространения и классические методы машинного обучения строят предсказания независимо на каждом объекте из выборки. Если есть необходимость в моделировании протяженных во времени зависимостей в данных, следует использовать методы глубокого обучения, которые предназначены для моделирования таких зависимостей, например, рекуррентные нейронные сети. Данная модель позволяет работать с временными рядами и соответствует задаче прогнозирования.

Рекуррентные нейронные сети (Recurrent neural network, RNN) — это вид нейронных сетей, в которых связи между элементами имеют направленную последовательность. Благодаря такой архитектуре появляется возможность обрабатывать последовательные пространственные цепочки, т.е. некоторые серии событий во времени.

Рекуррентные сети имеют цепную структуру, состоящую из повторяющихся модулей (repeating module) нейронной сети. Модулем может быть единственный нейрон или последовательность нескольких (Рис. 5).

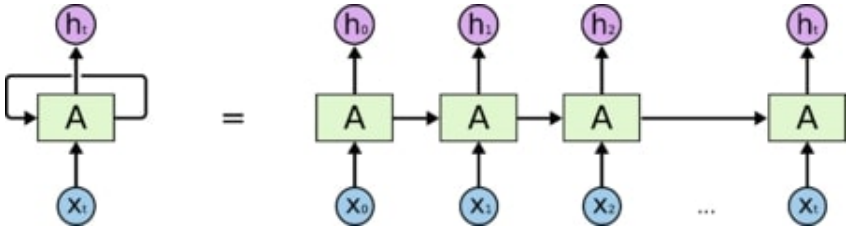

Рис. 5: RNN и ее развернутое представление

Рассмотрим простую Recurrent Neural Network (RNN). Repeating module будут иметь довольно простую структуру, например, всего один слой с функцией активации гиперболического тангенса ($tanh$) (Рис. 6)

Пусть имеется некоторая последовательность входных данных: $\{x_t\}_{t=1}^{T}$. В нашем случае $x_t = (x_1^t, \cdots, x_n^t)$ — векторное представление показаний датчиков. Выход t-го узла соединен с входом $t+1$-го узла.

Слои последовательно обрабатывают массивы данных и процесс обучения сети происходит следующим образом: на каждом шаге обучения t

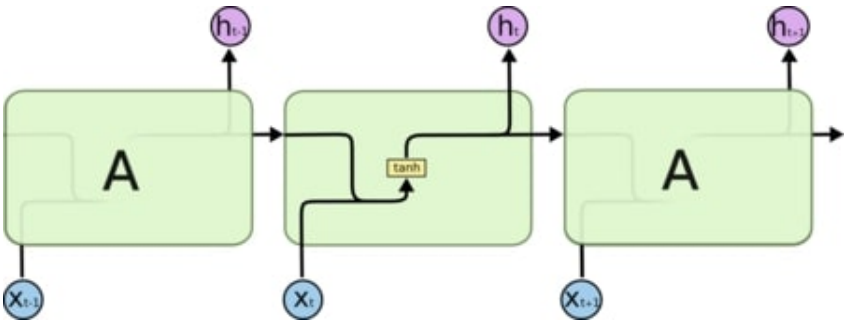

Рис. 6: Схема слоя рекуррентной сети

значения скрытого слоя нейронной сети вычисляется по формуле:

$$h_t = f_h(W_h x_t + U_h h_{t-1} + b_h)$$

$$y_t = f_y(W_y h_t + b_y)$$

где x_t — вектор входного слоя в момент времени t; h_t — вектор скрытого слоя; y_t — вектор выходного слоя; W, U и b — параметры нейронной сети, которые вычисляются при обучении; f_h, f_y — функция активации.

В процессе обучения нейронной сети обновление весов осуществляется с помощью алгоритма Backpropagation. Backpropagation (Метод обратного распространения ошибки) — метод вычисления градиента, который используется при обновлении весов многослойного перцептрона. На Рис. 7 представлен псевдокод алгоритма. Условием прекращения работы алгоритма являются следующие критерии: достигается выполнение определенного количества итераций, либо по методу ранней остановки. Метод ранней остановки заключается в том, что процесс обучения останавливается, если за заданное количество эпох потери не начинают значительно уменьшаться или результат становятся хуже.

Однако, при реализации данной архитектуры, память сети является довольно непродолжительной — при исполнении каждой итерации информация в памяти дополняется новой, а через некоторое время в полном объеме информация перезаписывается. В этом случае, в задаче прогнозирования RUL, при длительном использовании оборудования, показания датчиков полученные в начале не будут учитываться в результатах работы

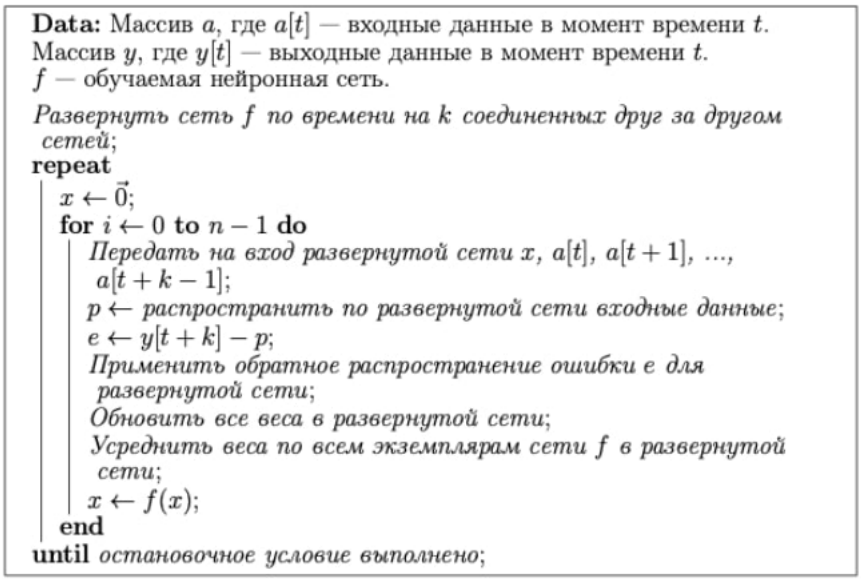

Рис. 7: Алгоритм метода обратного распространения ошибки

нейронной сети, хотя их значения могут значительно повлиять на реальный RUL. Это связано с проблемой затухания градиента: по мере увеличения количества слоев сеть в конечном итоге становится необучаемой. В работе [19] подробно рассматривается проблема исчезающего градиента, а ее влиянию на обучение и работу рекуррентных нейронных сетей посвящена работа [20]. Таким образом, RNN не всегда удается уловить долгосрочные зависимости в последовательности объектов. В качестве решения существует рекуррентная нейронная сеть с дополненной архитектурой, которая борется с этой проблемой.

3.4.1 LSTM

Long Short Term Memory networks (LSTM) — это особые нейронные сети, способные находить долгосрочные зависимости. LSTM тоже имеют цепную структуру, но повторяющий модуль отличается по строению. Вместо одного нейронного слоя, сеть состоит из четырех нейронов, соединенных специальным образом, схема такой сети представлена на Рис. 8. Желтыми блоками обозначены слои нейронной сети, а розовыми кругами поточные линейные операции, такие как сложение векторов. Сливающиеся линии обозначают конкатенацию, в то время как ветвящиеся линии обо-

значают, что их содержимое копируется, и копии отправляются в разные места. Данная нейронная сеть имеет сигмоидальную функцию активации, которая определяется по формуле: $\sigma(x) = \frac{1}{1+e^{-x}}$.

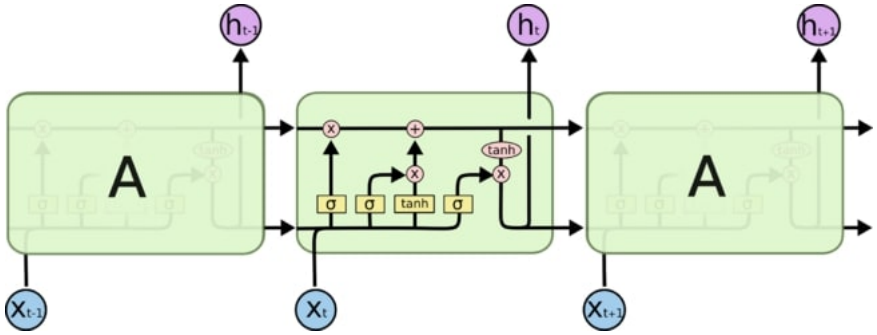

Рис. 8: Схема слоя LSTM

Отличительной особенностью LSTM сетей является то, что модули не используют функцию активации внутри своих рекуррентных компонентов. Она проходит напрямую через всю цепочку, участвуя лишь в нескольких линейных преобразованиях. (Рис. 9) Таким образом, хранимое значение не размывается во времени, и градиент или штраф не исчезает при использовании метода обратного распространения ошибки во времени при обучении сети.

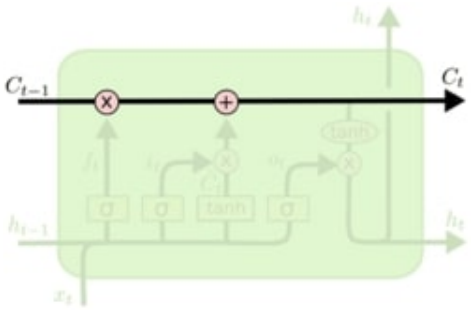

Рис. 9: Компонент LSTM - состояние ячейки (cell state)

LSTM может осуществлять удаление информации из рекуррентных компонент, данный процесс регулируется структурами, называемыми фильтрами (gates). Фильтры — это способ избирательно пропускать информацию. Они составлены из сигмоидного слоя нейронной сети и операции поточечного умножения (pointwise multiplication).

Соответственно, на первом шаге необходимо определить, какую информацию можно забыть в векторе состояний. Необходимо вычислить множители к компонентам вектора памяти. Сигмоидный слой принимает на вход вектор, составленный из значения предыдущего выхода h_{t-1} и текущего входа x_t, а на выход возвращает вектор $f_t \in [0,1]^m$ по формуле 3 (Рис. 10).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3}$$

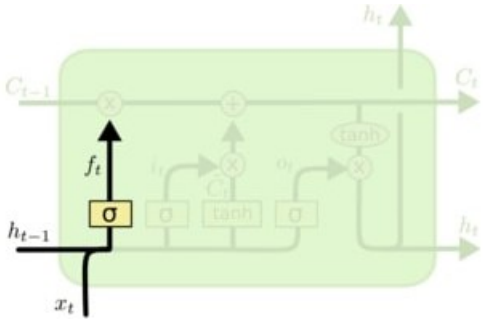

Рис. 10: Слой фильтра забывания

На следующем шаге необходимо определить, какая обновленная информация будет храниться в векторе состояний. Этот этап условно можно разделить на две составляющие. Сначала сигмоидальный слой под названием "слой входного фильтра"(input layer gate) определяет, какие компоненты вектора состояний и в какой степени следует обновить (4). Затем слой гиперболического тангенса задает вектор новых значений, которыми следует заменить компоненты вектора, установленные на предыдущем шаге (5). На вход слоям подается вектор (h_{t-1}, x_t) (Рис. 11).

$$i_t = \sigma(W_i x_i + U_i h_{t-1} + b_i) \tag{4}$$

$$\widehat{C}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{5}$$

Чтобы получить новый вектор состояний C_t, мы покоординатно умножаем старое состояние C_{t-1} на вектор f_t, определяющий какие компоненты состояния в какой мере мы забываем. Затем прибавляем покоординатное

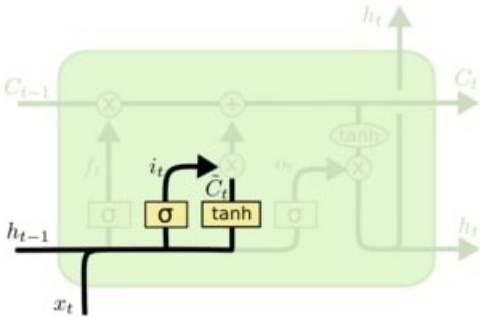

Рис. 11: Слои входного фильтра и гиперболического тангенса

произведение $i_t * C_t$, соответствующие новым значениям компоненты вектора состояний (6) (Рис. 12).

$$C_t = f_t * C_{t-1} + i_t * \widehat{C}_t \tag{6}$$

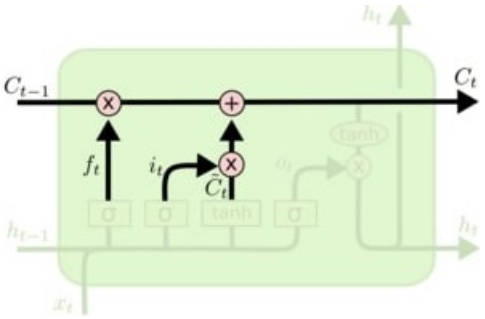

Рис. 12: Обновление вектора состояния

Последним этапом требуется обновить выходной вектор, вычислить значение выходного нейрона. Для определения выходного вектора, к вектору состояний применяются несколько фильтров. Сначала запускается сигмоидальный слой, который решает, какая информация из вектора состояний в какой степени перейдет в выходной вектор (7). Затем значения вектора состояний проходят через слой гиперболического тангенса (8). Полученные на выходе значения из диапазона $[-1, 1]$ покоординатно перемножаются с выходными значениями сигмоидального слоя, таким образом на выход отправляется только та часть информации, которая является необ-

ходимой. (Рис. 13)

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_0 C_t + b_o) \tag{7}$$

$$h_t = o_t * tanh(C_t) \tag{8}$$

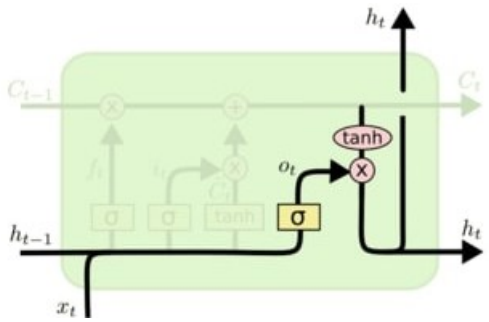

Рис. 13: Обновление выходного вектора

Значения h_t и C_t, полученные на предыдущих шагах, подаются на вход нейронной сети в момент времени $t + 1$. Далее, происходит обучение нейронной сети при помощи алгоритма обратного распространения ошибки, рассмотренного ранее.

3.4.2 Проблемы обучения глубоких нейронных сетей

Глубокие нейронные сети подвержены проблеме переобучения. При исполнении метода обратного распространения ошибки происходит минимизация функции потерь, т.е. параметры модели подстраиваются под особенности обучающего множества. Однако при этом не всегда происходит выявление общих закономерностей ряда, а лишь адаптация под особенности некоторой выборки - обучающего подмножества, что влечет уменьшение точности прогноза. Соответственно, модель теряет способность к обобщению. За последние годы было предложено множество решений данной проблемы, одним из самых простых, популярных и наиболее эффективных методов остается Dropout, так как он предотвращает взаимоадаптацию нейронов на этапе обучения. Главная идея Dropout заключается в следующем: вместо обучения одной нейронной сети обучается ансамбль

нескольких нейронных сетей, а затем полученные результаты усредняются. Сети для обучения определяются с помощью исключения из сети нескольких нейронов с заданной вероятностью p. После подобного преобразования сети, исключенный нейрон при любых входных данных или параметрах возвращает 0. Исключенные нейроны не вносят свой вклад в процесс обучения ни на одном из этапов алгоритма обратного распространения ошибки (*backpropagation*), поэтому исключение хотя бы одного из нейронов равносильно обучению новой нейронной сети. Подробный алгоритм метода описан в работе [21].

Глава 4. Реализация модели прогнозирования

4.1 Инструменты реализации

Для построения моделей использовался язык Python так, как он является кроссплатформенным, что обеспечивает совместимость почти с любой операционной системой.

Python - это интерпретируемый высокоуровневый язык программирования, ориентированный на повышение производительности разработчика и читаемости кода. Он обладает продуманной модульностью и масштабируемостью, а также большим функционалом и применяется для широкого круга задач.

Для визуализации данных использовалась библиотека Matplotlib. При помощи вызова пары функций она позволяет построить гистограммы, поля градиентов, диаграммы разброса и многое другое. Осуществляется поддержка основных форматов изображений: JPEG, PDF, PNG, SVG.

Для манипулирования данными и их анализа использовалась программная библиотека pandas. Этот пакет предоставляет специальные структуры данных и операции для обработки числовых таблиц и временных рядов. Из особых возможностей можно выделить наличие специального объекта DataFrame и инструментов для обмена данными между структурами в памяти и файлами различных форматов, а также удобную работу с временными рядами.

Для реализации алгоритмов машинного обучения использовалась библиотека Scikit-learn. Ее функционал включает задачи обучения с учителем (классификации, регрессии), а также задачи обучения без учителя (кластеризации, понижения размерности и детектирования аномалий). Помимо этого, Scikit-learn содержит функции для расчета значений метрик, препроцессинга данных и другие.

Для построения нейросетевой модели использовалась библиотека Keras. Она нацелена на оперативную работу с сетями глубокого обучения, содержит многочисленные реализации широко применяемых строительных блоков нейронных сетей, таких как целевые и передаточные функции, слои, оптимизаторы.

Для реализации программного комплекса использовался фреймворк Django. Он является фреймворком для веб-приложений на языке Python, который обеспечивает быструю разработку надежных и легко поддерживаемых сайтов. Django содержит в себе обширный функционал для веб-разработки например: аутентификация пользователей (вход, выход, регистрация), панель управления сайтом, формы, инструменты для загрузки файлов и т. д. Также он бесплатный и имеет подробную документацию.

Для реализации системы хранения данных было решено использовать объектно-реляционную систему управления базами данных PostgreSQL. Данная СУБД имеет широкие возможности и высокую производительность.

4.2 Выбор метрик

В задачах машинного обучения для оценки качества моделей и сравнения различных алгоритмов используются метрики. Для качественной оценки результатов работы каждого из методов воспользуемся MSE, MAE и R^2 так как они являются метриками качества регрессии и являются интерпретируемыми:

$$MSE(y, \widehat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \widehat{y}_i)^2, \tag{9}$$

$$MAE(y, \widehat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \widehat{y}_i| , \tag{10}$$

$$R^2(y, \widehat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}. \tag{11}$$

где $y, \widehat{y}$ — реальное и предсказанное значение соответственно, n — размер выборки.

Средняя квадратическая ошибка (MSE) (9) дает общее представление о величине ошибки, но не дает представление о направлении (например, больше или меньше при прогнозировании).

Средняя абсолютная ошибка (MAE) (10) представляет собой сумму абсолютных различий между прогнозами и фактическими значениями. Это дает представление о том, насколько неправильными были прогнозы.

R^2 (11) представляет собой долю дисперсии, которая была объяснена независимыми переменными в модели. Она интерпретируется как соответствие модели данным и, следовательно, меру того, насколько хорошо невидимые образцы могут быть предсказаны моделью, посредством доли объясненной дисперсии.

4.3 Построение моделей

Построение модели глубокого обучения требует принятия множества решений. Часть из них выбрана согласно рекомендациям в литературе [22], остальные значения подбирались методом экспериментальных прогонов. Гиперпараметр — это параметр, значение которого устанавливается перед началом процесса обучения. Далее описаны рассматриваемые гиперпараметры и их значения.

Фиксированные гиперпараметры:

- функция потерь — средняя квадратичная ошибка (стандартный выбор для задачи регрессии);
- оптимизационный алгоритм — Adam;
- метод регуляризации — Dropout с параметром 0.5.

Опробованные параметры:

- количество рекуррентных слоев нейронной сети: 2, 3, 4;
- количество входных нейронов в каждом LSTM слое: 30, 50, 100;
- количество эпох: 10, 20, 30;
- размер батча: 100, 200, 1000.

Итоговая оптимальная архитектура рекуррентной нейронной сети для задачи прогнозирования RUL состоит из следующих слоев:

1. входной слой;
2. слой LSTM (100 нейронов);
3. слой Dropout ($p = 0.5$);
4. слой LSTM (30 нейронов);
5. слой Dropout ($p = 0.5$);
6. выходной слой.

На следующем шаге, для построения модели необходимо преобразовать данные для подачи их в модель: из данных размерности (20631, 14) в новое признаковое пространство, где новыми объектом будет скользящее окно размером $window\ size$, чтобы учитывать исторические показания датчиков. Размер окна определен методом экспериментальных прогонов на тренировочной выборке. В таблице 2 представлены результаты работы модели с различным размером окна. В тестовой выборке отсутствуют показания датчиков более чем на 30 циклов, поэтому верхняя граница гиперпараметра 30. Можно заметить, что результаты показывают значительную тенденцию к снижению MSE/MAE на тренировочных данных почти для всех экспериментальных прогонов. Лучший результат представлен на Рис. 14.

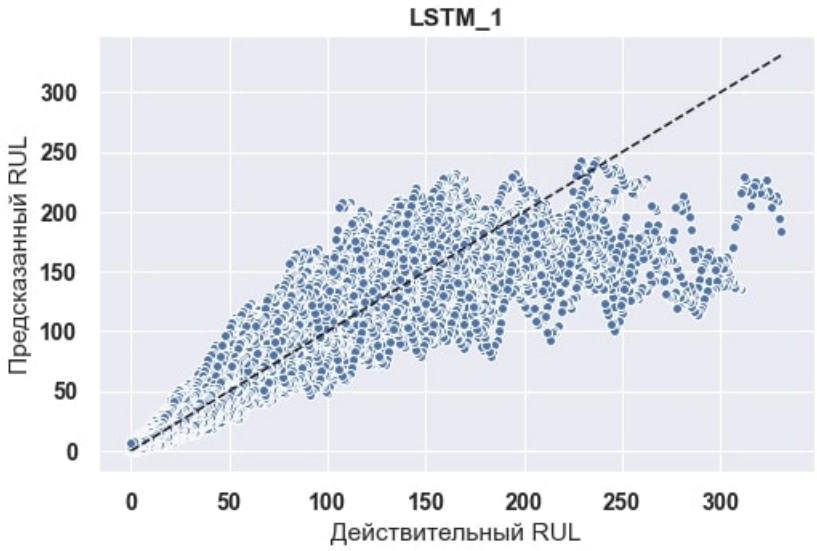

Рис. 14: Результат работы LSTM с размером окна 30

Таблица 2: Подбор гиперпараметра *window size*

window size	5	10	15	25	30
MSE	1615.674	1517.545	1266.430	948.480	890.299
MAE	28.310	27.088	25.009	21.070	21.376
R^2	0.647	0.656	0.702	0.759	0.766

Заметим, что при анализе показаний датчиков (Рис. 3) в разделе 2.3, значения датчиков изменялись постепенно, соответственно мы можем упростить модель рассматривая не подряд идущие значения, а с определенным интервалом.

Выполнены следующие преобразования входных данных: добавлены новые признаки — значения датчиков на 10, 20, 30 циклов назад от текущего цикла. Аналогично предыдущему шагу, с помощью экспериментальных прогонов на тренировочной выборке, производился подбор размера окна. Лучший результат представлен на Рис. 15.

Таблица 3: Подбор гиперпараметра *window size* в новом признаковом пространстве

window size	1	2	3	4	5
MSE	1156.291	1025.638	933.582	928.622	911.477
MAE	23.510	21.821	20.585	20.301	19.170
R^2	0.693	0.728	0.752	0.754	0.758

Из таблицы 3 можно заметить что при увеличении окна ошибка уменьшается незначительно, однако при этом значительно увеличивается признаковое пространство, для простоты модели и избегания переобучения было решено выбрать $window\ size = 3$.

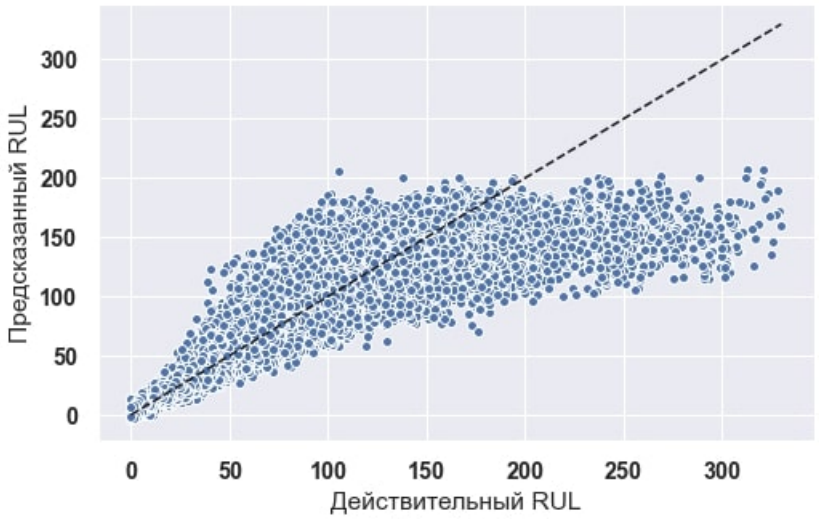

Рис. 15: Результат работы LSTM с размером окна 3 в новом признаковом пространстве

4.4 Сравнение работы моделей

Для оценки эффективности построения прогнозов с помощью моделей LSTM, случайного леса (RF) и линейной регрессии (LR), выполнены численные эксперименты. Результаты работы моделей на тестовой выборке представлены в таблице 4. Согласно значениям всех показателей, модель на основе сети LSTM 2 имеет лучшую точность.

Таблица 4: Результат работы методов на тестовой выборке в старом (1) и новом (2) признаковом пространстве

	LSTM 1	LSTM 2	RF 1	RF 2	LR 1	LR 2
MSE	556.08	497.41	1023.3	635.97	1024.6	636.32
MAE	16.425	16.589	23.469	18.126	25.545	19.626
R^2	0.677	0.711	0.407	0.648	0.406	0.624

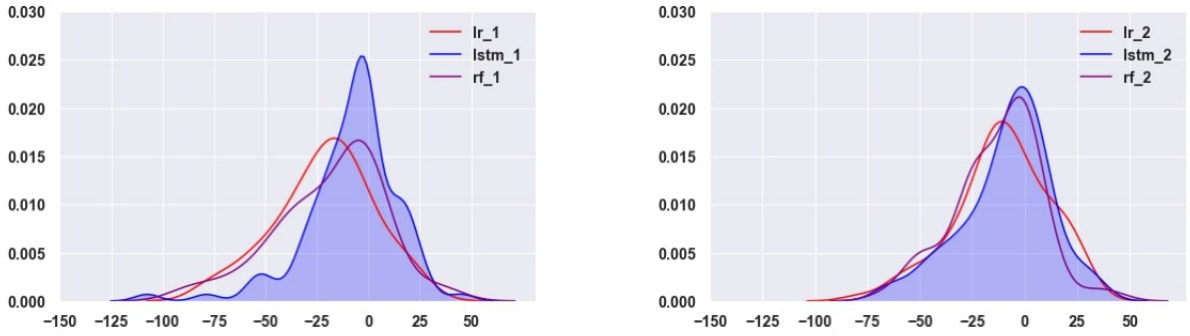

Рис. 16: Распределение остатков

По данным из таблицы 4 и графиков распределения остатков (Рис 16), можно сделать вывод о том, что добавление новых признаков положительно отразилось на качестве моделей, особенно на регрессионных, т. к. они не учитывали исторические значения датчиков.

По Рис. 17 можно наблюдать, что построенные модели являются менее эффективными при прогнозировании остаточного срока службы в начале использования оборудования, однако за некоторое время до сбоя системы, предсказание срока отказа двигателя вычисляется с более высокой

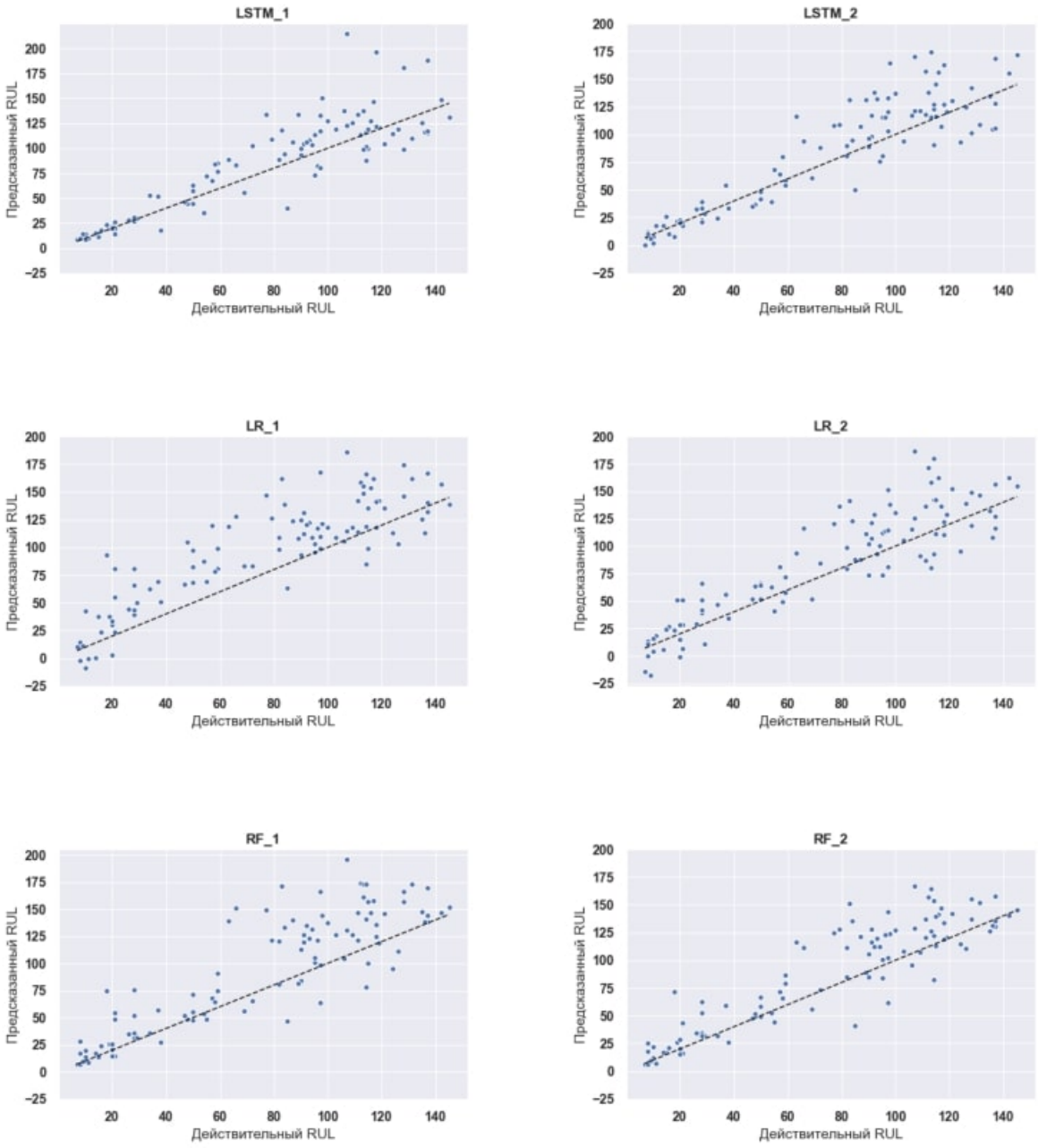

Рис. 17: Результат работы построенных моделей

точностью, что и необходимо с практической точки зрения. LSTM 2 показал себя не намного лучше LSTM 1, однако удалось значительно сократить признаковое пространство и обезопасить модель от переобучения, поэтому далее оптимальной моделью будем считать LSTM 2.

4.5 Программный комплекс

Рассмотрим реализацию программного комплекса на языке Python. Программный комплекс представляет собой веб-приложение. После снятия показаний датчиков, данные передаются на сервер, поступает запрос на анализ текущего состояния оборудования и прогнозирование остаточного срока работы. Сервер, в свою очередь, принимает запрос, формирует задачу и направляет ее вычислительному приложению. Вычислительное приложение по исходным данным подбирает оптимальные параметры метода LSTM и производит прогноз RUL. После решения задачи, вычислительное приложение отправляет результат прогноза на сервер, а далее происходит перенаправление результата пользователю. Также реализована учетная запись пользователя с возможностью редактирования и изменения исходных данных. Архитектура программного комплекса представлена на схеме (Рис. 18).

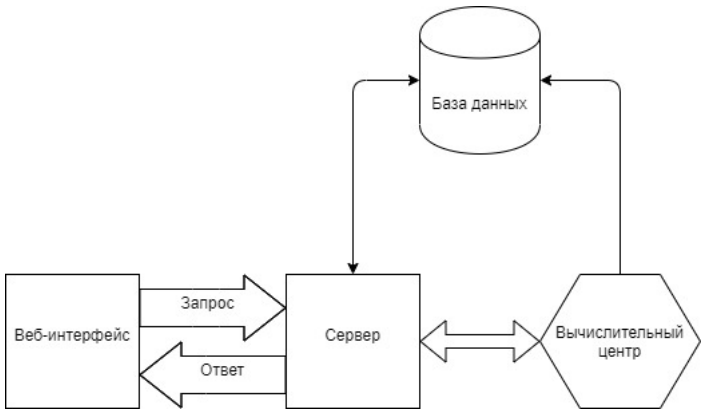

Рис. 18: Архитектура программного комплекса

4.5.1 База данных

При использовании программного комплекса, для подключения технического персонала к системе, потребуется хранить некоторые данные пользователей. Также, необходимо хранить данные с показателями датчиков для каждого оборудования для того, чтобы иметь возможность дообучать и переобучать модель, это может быть необходимо для улучшения качества работы построенной модели. На Рис. 19 представлена структура, полученной базы данных.

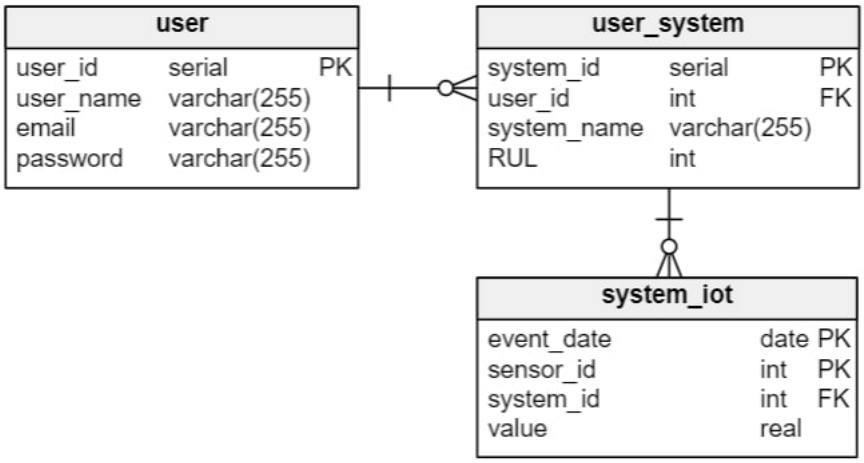

Рис. 19: Схема структуры базы данных

Таблица **user** используется для хранения персональных данных пользователей, они необходимы при процедуре аутентификации. Если есть необходимость в регистрации нового специалиста, в данной таблице создается запись, содержащая его ФИО, почту и захешированный пароль.

Таблица **user system** содержит информацию с техническими характеристиками оборудования, а также информацию о специалисте, который отвечает за диагностику системы. В работе использовались обезличенные данные, поэтому в таблице содержится только идентификационный номер оборудования, однако она может дополняться различной информацией о системе: тип модели, год выпуска, страна изготовителя и т.д.

Таблица **system iot** содержит информацию о показаниях датчиков.

Каждое показание ссылается на запись в user system для идентификации системы, с которой было снято показание, и пользователя системы. Также можно дополнить таблицу интерпретацией измерения датчика (температура, давление и т.п)

4.5.2 Функционал приложения

Использование программного комплекса техническими специалистами происходит с помощью различных запросов. Программный модуль имеет следующий функционал:

- **Регистрация нового пользователя.** Данный запрос создает новую запись в таблице user. Запрос содержит ФИО, почту и пароль.
- **Обучение модели с нуля.** Данный запрос запускает процесс обучения модели с нуля на новых данных. Запрос содержит исторический массив данных с показаниями датчиков.
- **Дообучение модели.** Данный запрос запускает процесс обновления весов модели используя новые данные. Запрос содержит исторический массив данных с показаниями датчиков.
- **Добавление новой системы.** Данный запрос создает новую запись в таблице user system. Запрос содержит идентификационный номер оборудования.
- **Удаление имеющейся системы.** Данный запрос удаляет данные о выбранной системе из таблицы user system. Запрос содержит идентификационный номер оборудования.
- **Загрузить показания датчиков и рассчитать RUL выбранной системы.** Данный запрос создает новые записи в таблице system iot. На выход подается предсказанный RUL оборудования. Запрос содержит текущие показания датчиков данной системы.

Выводы

В рамках реализации данной работы были решены поставленные задачи и получены следующие результаты:

1. Был произведен обзор существующих подходов и практик, применяемых в условиях реального производства, по внедрению предиктивного обслуживания оборудования;

2. Проведен анализ данных, учитывая специфику решаемой задачи. Выбраны необходимые методы для подготовки и анализа данных;

3. Рассмотрены математические модели, применимые к поставленной задаче, их преимущества и недостатки, а также приведено подробное описание исследуемого метода;

4. На основе нейронных сетей была предложена и реализована архитектура рекуррентной нейронной сети LSTM для задачи прогнозирования RUL. Подробнее об архитектуре нейронной сети написано в разделе 3.4, а о реализации — в разделе 4.3;

5. Произведено сравнение построенных моделей. В результате сравнительного анализа различных моделей на одном и том же наборе данных, вышеописанная модель показала лучший результат по метрикам MSE и R^2. Подробнее сравнение описано в разделе 4.4;

6. Реализован программный модуль, упрощающий взаимодействие пользователя с построенной моделью.

В качестве результата исследования получен программный комплекс, способный прогнозировать остаточный срок службы оборудования на производстве в условиях реального времени.

Заключение

На сегодняшний день, в погоне за эффективным использованием ресурсов, предприятиям необходимо развивать технологии гибкого планирования технического обслуживания. Прогнозирование сбоев оборудования и

остаточного срока службы систем является актуальной областью исследований, ведь в России предиктивная аналитика только набирает обороты и индустриальный сектор все активнее применяет PdM. Однако, до сих пор нет программного комплекса, который бы полностью решал данную задачу. Хотя в данном направлении и ведутся активные разработки, но перед исследователями стоит ряд проблем, которые затрудняют реализацию проектов. К примеру, медленное развитие технологий IoT, которые выступают необходимыми техническими компонентами в реализации поставленной задачи. Так, большая часть оборудования на заводах не оснащена датчиками для передачи информации, на предприятиях нет систем сбора данных и онлайн-мониторинга. К тому же на производствах журналы с информацией о дефектах и ремонтах зачастую ведутся некорректно. Неготовность персонала к IT-решениям и недоверие к новой концепции обслуживания сдерживают внедрение PdM-систем.

В данной работе на основе имеющихся данных было предложено решение для гибкого планирования технического обслуживания, оно включает в себя программный комплекс, способный прогнозировать остаточный срок службы оборудования на производстве в условиях реального времени. В качестве продолжения данной работы можно сформулировать и решить задачу классификации: по состоянию оборудования на текущей момент определять, может ли оно быть допущено в эксплуатацию, что позволит эффективнее принимать решение о безопасности его использования, исключая человеческий фактор полностью. Это требует дополнительных исследований по установлению безопасного доверительного интервала, для реализации необходимо привлечение специалистов, а также большего массива исторических данных.

Список литературы

[1] Diamond S., Marfatia A. Predictive maintenance for dummies. Hoboken, 2013. 44 p.

[2] Poor P., Basl J. Predictive Maintenance 4.0 as next evolution step in industrial maintenance development. // 2019 International Research

Conference on Smart Computing and Systems Engineering (SCSE), Colombo, Sri Lanka. 2019. P. 245–253.

[3] Чучуева И.А. Модель прогнозирования временных рядов по выборке максимального подобия: дис. ... канд. техн. наук: 05.13.18. Моск. гос. техн. ун-т им. Н.Э. Баумана Москва, 2012. 154 с.

[4] Андрюхин Е.В., Ридли М.К., Правиков Д.И. Прогнозирование сбоев и отказов в распределённых системах управления на основе моделей прогнозирования временных рядов // Вопросы кибербезопасности, 2019, №3. С. 24–32.

[5] Шаханов Н.И., Варфоломеев И.А., Ершов Е.В., Юдина О.В. Прогнозирование отказов оборудования в условиях малого количества поломок // Вестник Череповецкого государственного университета, 2016, № 6. С. 36–41.

[6] Хайкин С. Нейронные сети. Полный курс. : И.Д. Вильямс , 2016. 1104 с.

[7] Horichreiter S., Schmidhuber J. Long short-term memory // Neural Computation, 1997, № 9. P. 1735–1780.

[8] Имильбаев Р. Р. Методы и алгоритмы прогнозирования значений контролируемых параметров газораспределительной сети по результатам обработки телеметрической информации: дис. ... канд. техн. наук: Уф. гос. нефт. техн. ун-т Уфа, 2018. 152 с.

[9] Saxena A., Goebel K. Damage propagation modeling for aircraft engine run-to-failure simulation. // 2008 International Conference on Prognostics and Health Management, Denver, CO. 2008. P. 1–9.

[10] Cielen D., Meysman A., Ali M. Introducing Data Science Big Data, Machine Learning, and more, using Python tools, 2016. P. 22–48.

[11] Documentation Pandas python. `http://pandas.pydata.org/pandas-docs/version/0.15.2/tutorials.html`

[12] Documentation scikit-learn python. http://scikit-learn.org/stable/tutorial/index.html

[13] Documentation Keras models. https://keras.io/models/about-keras-models

[14] Стрижов В. В. Методы индуктивного порождения регрессионных моделей: ВЦ РАН. 2008. 55 с.

[15] Tibshirani R. J. Regression shrinkage and selection via the lasso // Journal of the Royal Statistical Society. Series B (Methodological), 1996. Vol. 58, No. 1. P. 267–288.

[16] Чистяков С. П. Случайные леса: обзор // Труды Карельского научного центра Российской академии наук, 2013 № 1. С. 117–136.

[17] Breiman L. Bagging predictors // Machine Learning, 1996. Т. 24, вып. 2. С. 123–140.

[18] Skurichina M., Duin R. P. W. Limited bagging, boosting and the random subspace method for linear classifiers// Pattern Analysis and Applications, 2002. Vol. 5, No. 2, P. 121–135.

[19] Hochreiter S. The vanishing gradient problem during learning recurrent neural nets and problem solutions // International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 1998. Vol. 6, No 2. P. 107–116.

[20] Hochreiter S., Bengio Y., Frasconi P., Schmidhuber J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In A Field Guide to Dynamic Recurrent Neural Networks. IEEE Press, 2001.

[21] Molchanov D., Ashukha A., Vetrov D. P. Variational dropout sparsifies deep neural networks // Proceedings of the 34th International Conference on Machine Learning, ICML, 2017, P. 2498–2507.

[22] Goodfellow I., Bengio Y., Courville A. Deep Learning: MIT Press, 2016. 775 p.