Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования «Московский физико-технический институт
(национальный исследовательский университет)»

Физтех-школа радиотехники и кибернетики

Кафедра инфокоммуникационных систем и сетей (ИСС)

Выпускная квалификационная работа
по направлению подготовки бакалавров
03.03.01 Прикладные математика и физика

Модель предиктивного анализа состояния железнодорожных путей на основе нейронных сетей

Студент
К. В. Просвирин

Научный руководитель
Е. Н. Платонов,
канд. физ.-мат. наук

Москва 2021

Ввиду значительного объема грузоперевозок железнодорожным транспортом в Российской Федерации поддержание железнодорожной инфраструктуры в исправном состоянии является критически важной хозяйственной задачей. Цель настоящей выпускной квалификационной работы — построение предиктивной модели на основе нейронных сетей, призванной прогнозировать опасный отказ верхних строений железнодорожного пути. Исследована применимость различных архитектурных решений: полносвязной нейронной сети, рекуррентных нейронных сетей на основе сети Элмана (RNN) и с управляемыми рекуррентными блоками (GRU), автокодировщика. Полученные результаты превосходят по качеству предсказаний существующие модели на основе градиентного бустинга. Достигнут практический порог по точности предсказаний модели с учетом зашумленности входных данных.

Содержание

Введение

Объем грузоперевозок в Российской Федерации железнодорожным транспортом исчисляется 6,38 млрд тонн или 755 млрд руб. в год (2020 г., [1]). Интенсивный грузооборот по железным дорогам, а также неблагоприятные климатические условия на большей части территории страны вызывают быстрое исчерпание ресурса верхних строений железнодорожного пути. *Поддержание инфраструктуры в исправном состоянии — это критически важная хозяйственная задача.*

Для сбора сведений о степени износа железнодорожных путей используют специальные поезда. Однако проведение ремонта всех участков в отличном от идеального состоянии не только экономически нецелесообразно, но и принципиально невозможно. Необходимо выделять потенциально наиболее опасные участки для своевременного проведения ремонтных мероприятий.

Нейронные сети в настоящее время широко распространены в решении различных классов задач, таких как классификация образов, кластеризация объектов, аппроксимация функций, предсказания свойств и событий.

Цель настоящей выпускной квалификационной работы — *построить предиктивную модель на основе нейронных сетей, призванную прогнозировать опасный отказ верхних строений железнодорожного пути.* По входным данным за определенный промежуток времени модель должна предсказывать вероятность отказа железнодорожного пути в следующем месяце.

Для достижения указанной цели необходимо решить следующие учебно-исследовательские задачи:

- рассмотреть существующие модели, которые в данный момент используются для решения данной проблемы;
- придумать архитектуру предиктивной модели на основе нейронных сетей;
- проверить, смогут ли модели на основе нейронных сетей решать данную проблему лучше, чем существующие аналоги;
- провести серии экспериментов, на основе которых можно будет судить об итоговом качестве моделей;
- проанализировать полученные результаты и сделать вывод о целесообразности использования данных моделей.

Внимание автора к данной теме обусловлено желанием применить известные методы машинного обучения в решении актуальной прикладной (хозяйственной) проблемы.

В первой главе настоящей работы описаны используемые архитектуры нейронных сетей, разъяснены основные принципы работы каждой из моделей, рассказано об их преимуществах и недостатках. Во второй главе приведены архитектуры сетей, предложенные автором данной работы, рассказано о ключевых этапах подготовки данных, проанализированы полученные результаты. В заключении эти результаты обобщены, обозначены возможные направления дальнейших исследований.

Глава 1

Архитектура искусственных нейронных сетей

1.1 Общая теория нейронных сетей

Искусственная нейронная сеть (ИНС) — математическая модель, которая работает по принципу, аналогичному работе сетей нервных клеток живого организма. ИНС основана на наборе связанных между собой узлов, называемых искусственными нейронами, которые моделируют биологические нейроны — клетки, которые принимают на вход информацию (сигналы) от других нейронов, обрабатывают ее, а затем передают другим нейронам посредством электрических сигналов.

В математической модели нейрона ядро, где накапливается заряд, заменяется на *сумматор*. Дендриты, по которым сигнал приходит от других нейронов в ядро нейрона, заменяются на *входы* в сумматор. Сила синапсов моделируется при помощи *весов*, которые задаются для каждого из входов. Аксон, по которому нейрон отправляет сигналы к другим нейронам,

соответствует *выходу* сумматора. Далее сигнал проходит через *функцию активации*, после чего уходит к другим нейронам. Простейшая функция активации — пороговая:

$$f(z) = \begin{cases} 1, \ z > 0, \\ 0, \ z \leq 0. \end{cases}$$

Иными словами, для n входов сумматора $\vec{x} = (x_1, x_2, \ldots, x_n)$ с заданными весами $\vec{w} = (w_1, w_2, \ldots, w_n)$ и смещением b, выходной сигнал есть

$$y = f(z) = f(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b) = f(\vec{x} \cdot \vec{w} + b),$$

где f — функция активации нейрона.

Таким образом, нейрон как объект производит некоторую *линейную* операцию, которая задаётся двумя параметрами: вектор весов и смещение. Эти параметры настраиваются в процессе обучения ИНС.

Разделяющая поверхность в соответствующем гиперпространстве задаётся уравнением

$$\vec{x} \cdot \vec{w} + b = 0,$$

определяющим гиперплоскость в пространстве весов $\vec{w} \in \mathcal{R}^n$, причем вектор $\vec{w}$ является нормалью к этой гиперплоскости. Получается, что один нейрон даёт *линейную* разделяющую поверхность. Если же соединить нейроны в некоторую более сложную конструкцию (то есть ИНС), возможно получить *нелинейные* разделяющие поверхности.

1.2 Полносвязные нейронные сети прямого распространения

Полносвязная нейронная сеть прямого распространения (англ. feed forward neural networks, FFNN) — сеть, которая представляет из себя набор слоев, где каждый слой состоит из входных, скрытых или выходных нейронов. В простейшем случае это выглядит следующим образом:

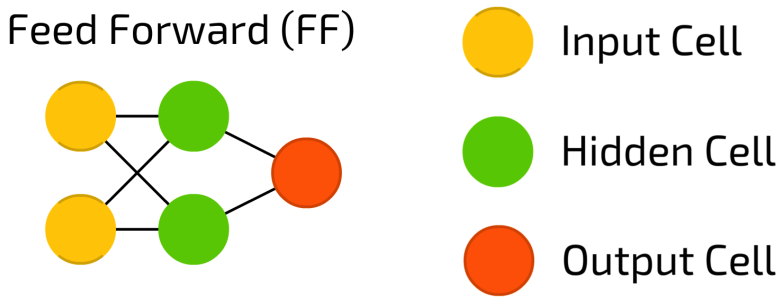

Рис. 1.1: Полносвязная НС

Нейроны одного слоя не связаны между собой, в то время как соседние слои *полносвязны* (англ. fully connected, FC), т. е. каждый нейрон связан со всеми нейронами предыдущего слоя, причем все связи направлены строго от входных нейронов к выходным.

Данный тип сетей обычно обучается по методу обратного распространения ошибки (англ. backpropagation, BP) [2], в котором сеть получает множества входных и выходных данных. Эта ошибка зависит от выбранной *функции потерь* (loss function), которая определяет величину ошибки между входом и выходом.

Этот процесс называется обучением с учителем. От обучения без учителя он отличается тем, что в последнем случае множество выходных данных сеть составляет самостоятельно. Если сеть имеет достаточное количество скрытых нейронов, она

теоретически способна смоделировать взаимодействие между входными и выходными данными.

Однако данный тип сетей имеет следующие существенные недостатки:

1. **Затухающий градиент.**
   При обратном распространении ошибки во время обучения ИНС градиенты ошибок отправляются обратно на вход сети [3], чтобы были подобраны веса, которые будут минимизировать функцию потерь. Когда в ИНС много слоев, эти градиенты близки к нулю. В этом случае веса вблизи входа практически не поменяют своих значений.
2. **Большое количество обучаемых параметров.**
   Например, если подать на вход ИНС с тремя скрытыми слоями черно-белую фотографию размером 28×28 пикселей, количество обучаемых параметров составляет порядка миллиона.

1.3 Автокодировщик

Автокодировщик (англ. autoencoder) — специальная архитектура искусственных нейронных сетей, которая позволяет применять обучение без учителя [**?**] при использовании метода обратного распространения ошибки.

Автокодировщик (см. простейшую схему на рис. 1.2) пытается восстанавливать объекты, принимаемые на вход сети, и состоит из двух частей:

1. Энкодер f, кодирующий исходную выборку X в свое внутреннее (англ. latent) представление: $h = f(X)$.

2. Декодер g, задача которого — восстановить исходную выборку: $\hat{X} = g(h) = g(f(X))$.

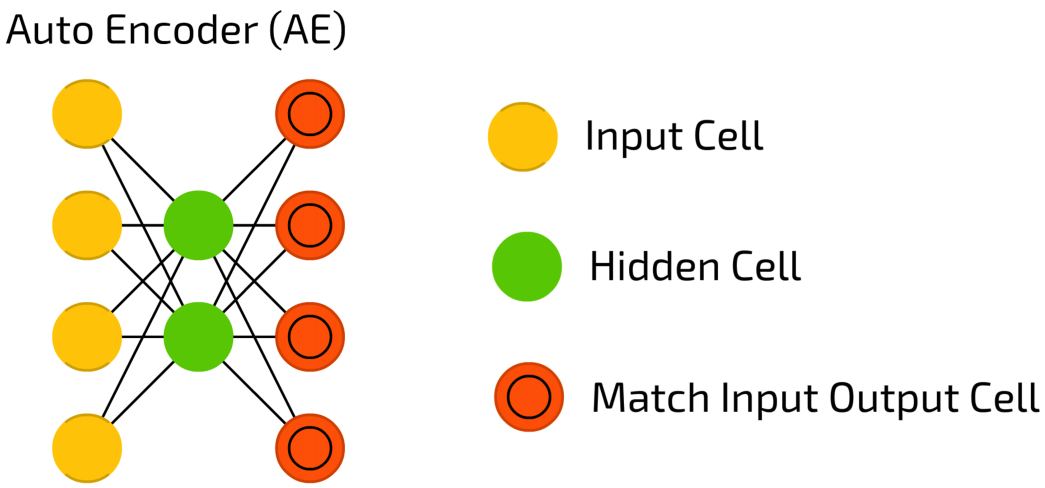

Рис. 1.2: Автокодировщик

Одно из основных предназначений автокодировщиков — снижение размерности исходного пространства. Сама процедура обучения нейросети заставляет автокодировщик запоминать основные признаки объектов, по которым будет проще восстановить исходные объекты выборки. Чаще всего во время обучения в качестве функции потерь выбирают средний квадрат ошибки:

$$\mathsf{MSE}\big[g(f(X)), X\big] = \mathsf{MSE}(\hat{X}, X).$$

Здесь возможно провести аналогию с методом главных компонент (англ. principal component analysis, PCA) [**?**]. Это метод снижения размерности, результатом работы которого является проекция выборки на подпространство, в котором дисперсия этой выборки максимальна.

По сути автокодировщик является обобщением метода главных компонент: если ограничиться рассмотрением линейных моделей, автокодировщик и метод главных компонент дают одинаковые векторные представления [**?**]. Разница возникает, если в качестве энкодера и декодера выбраны более сложные модели, например, многослойные *полносвязные нейронные сети*.

Как правило, размерность внутреннего представления меньше, чем размерность входного и выходного слоёв. Однако размерность латентного пространства не должна быть слишком маленькой, иначе модель может потерять обобщающие способности. В то же время она не должна быть слишком большой, чтобы не «заучивать» исходные данные (проблема переобучения [?]).

1.4 Рекуррентная нейронная сеть

Рекуррентная нейронная сеть (рекуррентная НС, англ. Recurrent neural network, RNN) — нейронная сеть, в которой разрешены циклы. Выход нейрона может подаваться снова на вход этого же нейрона, на вход всех нейронов в текущем слое либо к любому другому нейрону в любом слое.

Рекуррентные НС хорошо подходят для работы с последовательностями данных именно по той причине, что в них есть циклические соединения, через которые поступает информация с предыдущих шагов работы сети. Иными словами, рекуррентные НС могут анализировать входные данные не просто как набор изолированных друг от друга моментов времени, а как последовательность данных, где каждый последующий момент времени может некоторым образом зависеть или не зависеть от предыдущих состояний:

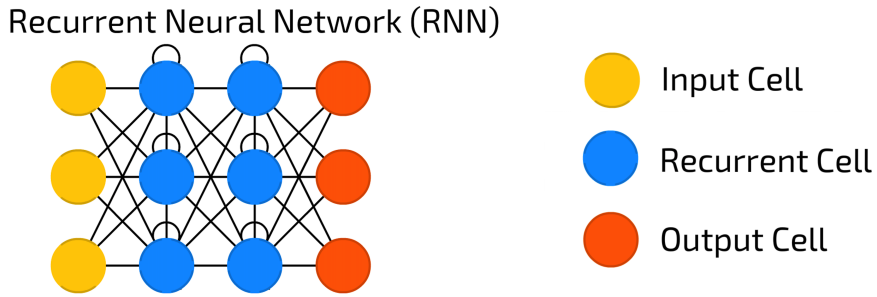

Рис. 1.3: Рекуррентная НС

Для обучения рекуррентной НС используется все тот же алгоритм обратного распространения ошибки. Рекуррентные НС представляют в виде НС с прямым распространением сигнала. Для этого используется *разворачивание во времени* [4], при котором создается несколько копий рекуррентной НС. Выглядит это следующим образом:

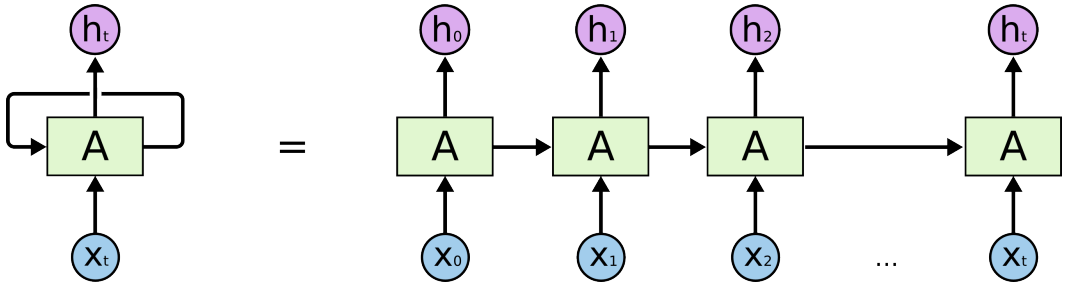

Рис. 1.4: РНС и ее развернутое представление

Поскольку одни и те же параметры используются на всех временных этапах в сети, градиент на каждом выходе зависит не только от расчетов текущего шага, но и от предыдущих временных шагов. Этот алгоритм называется *алгоритмом обратного распространения ошибки сквозь время* (англ. Backpropagation Through Time, BPTT) [5].

Рассмотрим некоторые реализации рекуррентных нейронных сетей, который используются в данной дипломной работе.

1.4.1 Нейронная сеть Элмана

Нейронная сеть Элмана — классический вариант рекурернтной нейронной сети. Она состоит из трех слоев: входного x, скрытого h, и выходного z (см. рис. 1.5). Для каждого элемента входной последовательности в момент времени t вычисляется новое состояние по следующему правилу:

$$h_t = \tanh(W_{xh}x_t + b_{ih} + W_{hh}h_{(t-1)} + b_{hh}),$$
$$z_t = W_{hz}h_t + b_z,$$

где h_t — скрытое состояние в момент времени t, x_t — входные данные в момент времени t, h_{t-1} — скрытое состояние на предыдущем шаге в момент времени $t-1$ или начальное состояние в момент времени 0, $\tanh[\circ]$ — функция активации (гиперболический тангенс), схему слоя см. на рис. 1.6

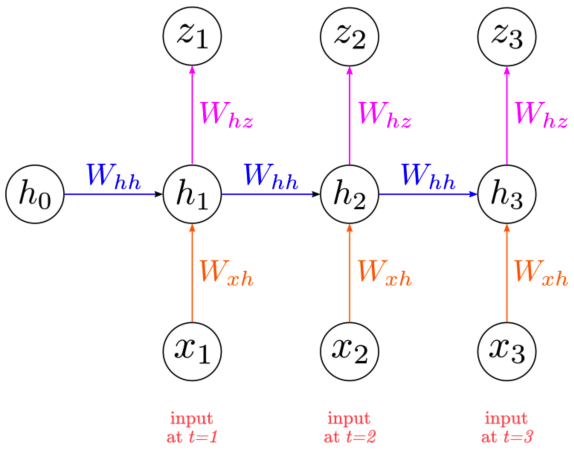

Рис. 1.5: Простейшая РНС

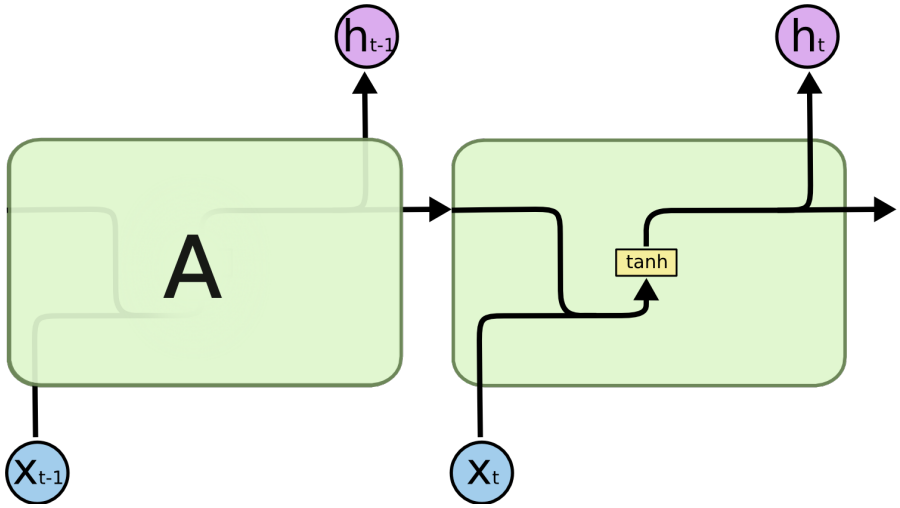

Рис. 1.6: Схема слоя рекуррентной сети

На каждом шаге на вход поступает информация, которая проходит прямой путь к выходному слою в соответствии с правилами обучения. Обратные связи сохраняют предыдущие значения скрытого слоя. Таким способом сеть сохраняет своё состояние, что может использоваться в предсказании последовательностей.

Однако если входная последовательность данных достаточно длинная (например, текст из более чем сотни слов), то такая простая сеть просто «забудет» слова, которые были в начале. Поэтому при необходимости проводить анализ длинных последовательностей необходимо научить нейронную сеть хорошо запоминать важные паттерны.

Другая проблема длинных последовательностей данных — большое количество слоев НС. Из-за этого снова встает вопрос о быстром затухании градиентов ошибок. Чтобы решить эти проблемы, были придуманы специальные блоки, заменяющие «обычные» нейроны.

1.4.2 Управляемые рекуррентные блоки

Управляемые рекуррентные блоки (англ. Gated Recurrent Units, GRU) — механизм вентилей для рекуррентных нейронных сетей. Для каждого элемента входной последовательности в момент времени t вычисляется новое состояние по следующему правилу:

$$
\begin{aligned}
r_t &= \sigma(W_{xr}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr});\\
z_t &= \sigma(W_{xz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz});\\
n_t &= \tanh\left[(W_{xn}x_t + b_{in}) + r_t \cdot (W_{hn}h_{(t-1)} + b_{hn})\right];\\
h_t &= (1 - z_t) \cdot n_t + z_t \cdot h_{(t-1)},
\end{aligned}
$$

где h_t — скрытое состояние в момент времени t, x_t — входные данные в момент времени t, h_{t-1} — скрытое состояние на предыдущем шаге в момент времени $t-1$ или начальное состояние в момент времени 0, r_t, z_t, n_t — вентиль сброса, вентиль обновления и новый вентель соответственно, $\tanh[\circ]$ — функция активации (гиперболический тангенс).

Основная идея таких блоков заключается в том, что обычный нейрон заменяется на некоторый блок, у которого есть память и есть вентили, которые контролируют сброс, перезапись или сохранение этой памяти. Данный блок схематично можно изобразить следующим образом:

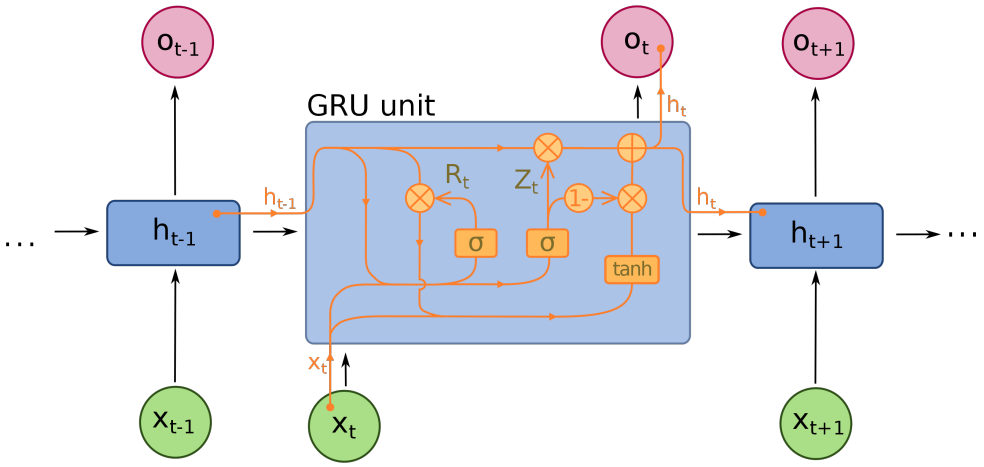

Рис. 1.7: Схема слоя с управляемым рекуррентным блоком

Нейронные сети, в которых используется данный подход, как правило, превосходят [6] классические рекуррентные сети.

Глава 2

Реализация архитектур

2.1 Подготовка данных

2.1.1 Описание

Исходные данные представляют собой таблицу в формате Microsoft Excel, где каждой строке соответствует определенный километр пути, а столбцу — определенный параметр состояния (например, дефект). Далее будем называть строки этой таблицы *объектами*, а столбцы — *признаками*.

Строго говоря, признак — результат измерения некоторой характеристики объекта, т. е. отображение

$$f : X \to D_f,$$

где X — множество объектов, D_f — множество допустимых значений признака. В зависимости от вида множества D_f признаки делятся на следующие типы:

- бинарный: $D_f = \{0, 1\}$,
- номинальный: D_f — конечное множество,
- порядковый: D_f — конечное упорядоченное множество,
- количественный: D_f — множество действительных чисел.

Перед тем как строить признаковое описание объектов, необходимо проанализировать, что из себя представляют исходные признаки и какие связи есть между ними:

Таблица 2.1: Типы признаков

Тип	**Количество**	**Пример**
Бинарный	23	Восемь и более просадок на отрезке длиной 100 м
Номинальный	91	Количество просадок 1 степени
Порядковый	6	Код дороги, год и месяц измерения
Количественный признак	20	Дистанция пути
Принимает единственное значение	4	Отступления с просадкой

Бóльшая часть признаков номинальна. Эти признаки показывают ту или иную степень отклонения железнодорожного полотна от нормы. В равной степени встречаются бинарные и количественные признаки. Меньше всего порядковых признаков и признаков, принимающих единственное значение.

Итоговые признаки, которые подаются на вход модели, необязательно должны состоять только из исходных сырых данных. В качестве итоговых признаков могут быть взяты:

- признаки, придуманные исходя из анализа основных свойств данных и нахождения в них общих закономерностей [7];
- признаки, полученные из исходных данных по некоторому правилу (косвенные признаки);
- смешанные признаки.

Выбор итоговых признаков для конкретных моделей описан в дальнейшем в соответствующих главах.

2.1.2 Анализ данных

Визуализируем исходные данные. Для начала узнаем, как меняется общее число отказов с течением времени (см. рис. 2.1). Можно заметить, что в летние месяцы в среднем количество отказов больше, чем в зимний период. Отсюда можно сделать следующие предположения:

- летом движение более интенсивное, т. е. нагрузка выше;
- летом рельсы под воздействием тепла больше склонны к деформации, следовательно, чаще выходят из строя;
- летом проводится больше измерений различных путей, и чем больше измерений делается, тем больше отказов находят.

Попробуем проверить последнее предположение. Для этого построим график с общим числом проверок сгруппировав их по дням (см. рис 2.2). Видно, что «горбы» на соответствующих графиках практически совпадают. Следовательно, что есть некоторая зависимость между числом проверок и количеством отказов, то есть чем больше проверок производится, тем больше отказов находят.

После большого количества проверок, а следовательно и ремонтов, количество выходов из строя железнодорожных путей резко падает. Поэтому было принято решение к исходным признакам добавить признак «количество проверок в прошлом месяце» на данном участке пути.

Из анализа данных об отказах железнодорожных путей видно, что не все отказы были правильно зарегистрированы. Отсутствует информация о том, где именно произошел отказ. Значит, не представляется возможным верно назначить данные об отказе определенному объекту железнодородного пути. Для таких объектов в стобце TARGET указано значение 0 (объект без опасного отказа) несмотря на то, что отказ был.

Таким образом, данные оказываются зашумленными. Это приводит к тому, что модель изначально обучается на некорректном наборе данных. Процент неверных отказов составляет порядка 17 %, что делает невозможным получение высокой точности предсказания потенциально опасных отказов.

При резком улучшении состояния километра данное событие классифицировалось как ремонтные работы и часть наблюдений удалялась, чтобы в дальнейшем не «запутывать» модель во время обучения. База данных по проводимым ремонтам железнодорожных путей находится в запущенном состоянии, содержит слишком много пропусков и ошибок, поэтому использовать ее непосредственно не представляется возможным.

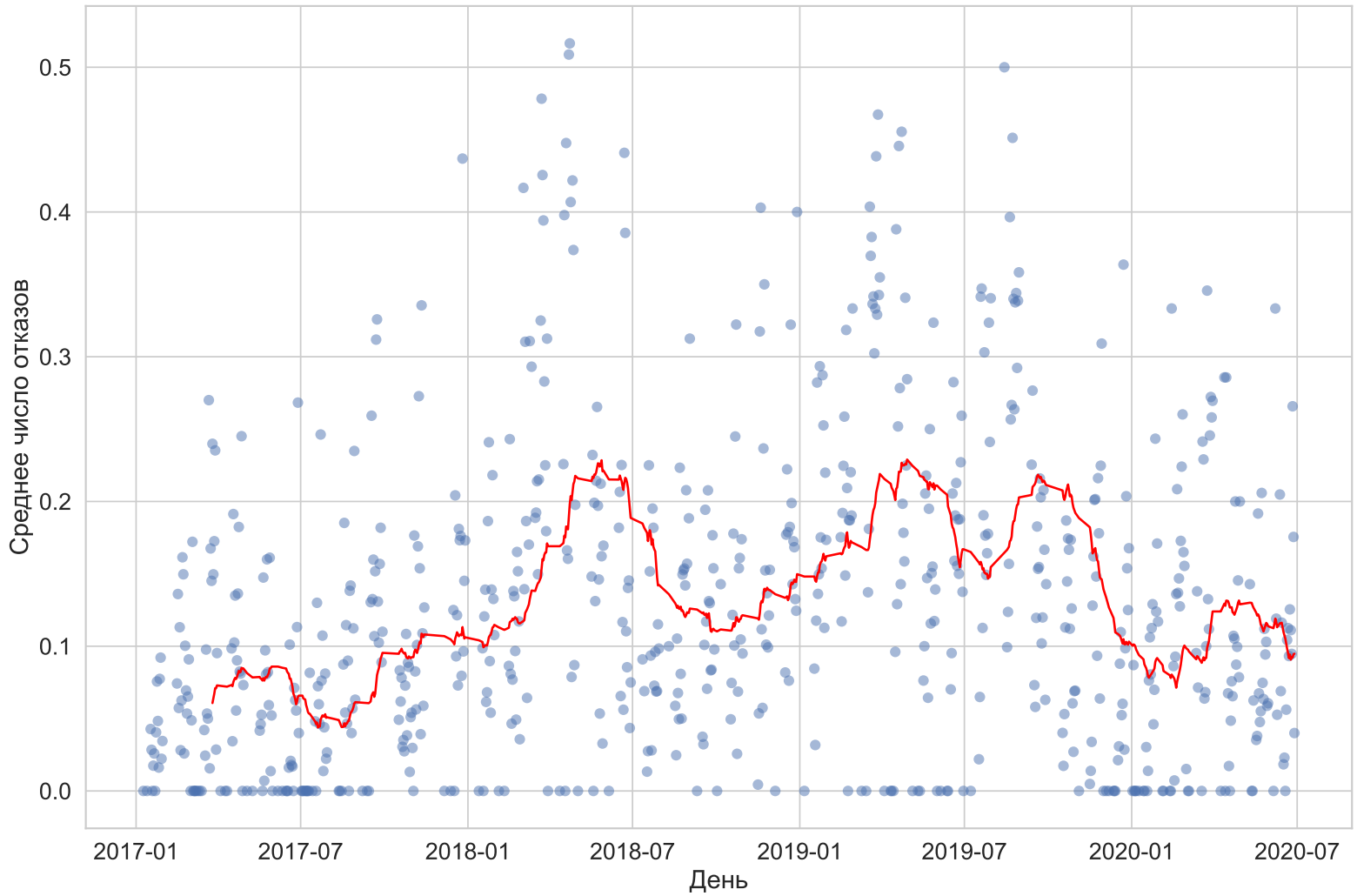

Рис. 2.1: Количество отказов от времени

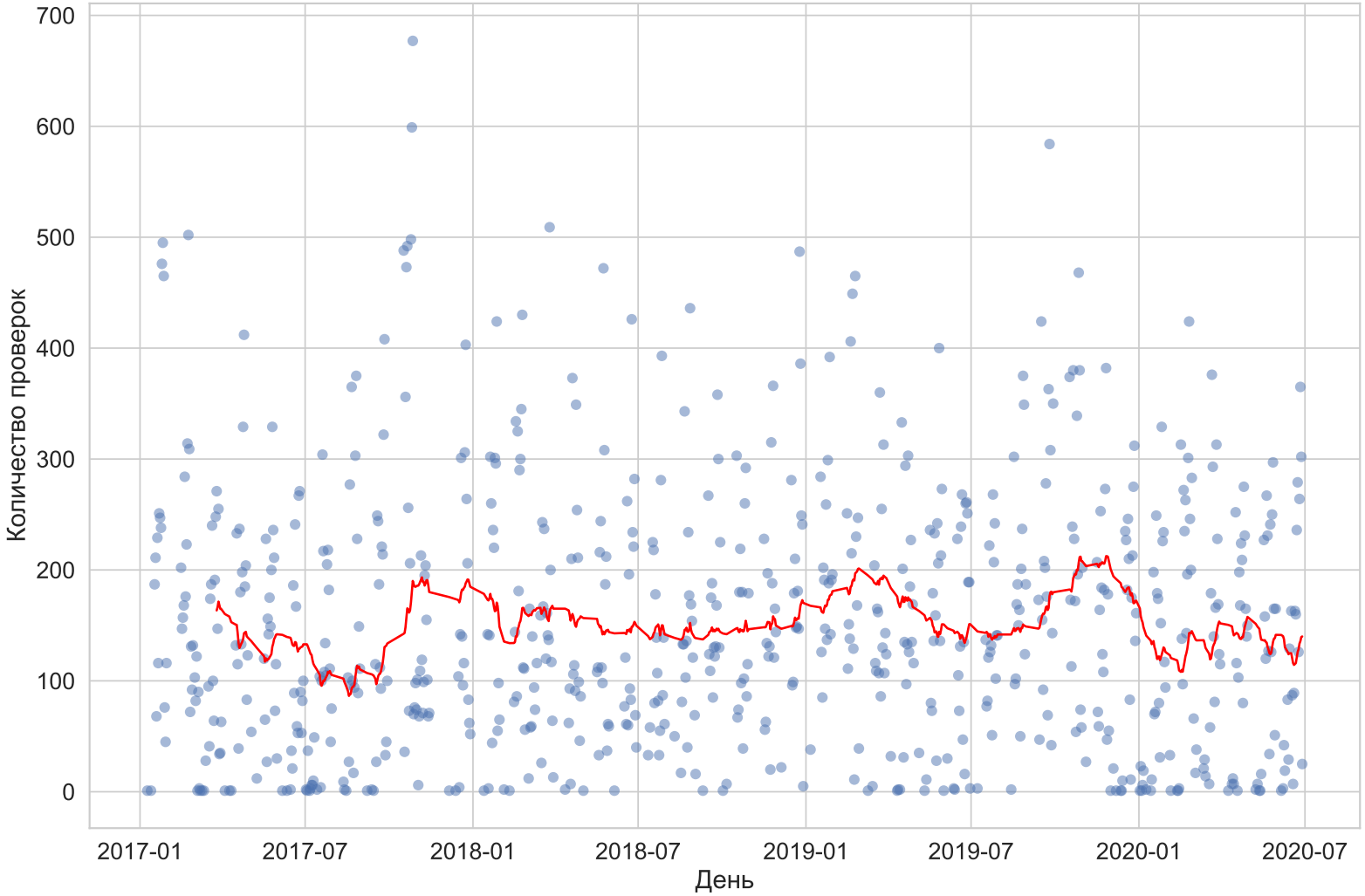

Рис. 2.2: Количество проверок от времени

2.1.3 Разбиение данных

Прежде всего для обеспечения корректного обучения моделей необходимо разбить данные на обучающую, валидационную и тестовую выборки.

На обучающей выборке модель будет подбирать нужные параметры сети для минимизации ошибки, на валидационной выборке будут настраиваться гиперпараметры модели а на тестовой выборке будет анализироваться итоговый результат. Ниже приведена таблица с данными по каждой из выборок:

Таблица 2.2: Разбиение данных

Суммарный объем выборки	107597
Объем обучающей выборки	74077
Объем валидационной выборки	17889
Объем тестовой выборки	15631
Обучающая выборка	январь 2017 – декабрь 2019
Валидационная выборка	январь 2017 – декабрь 2019
Тестовая выборка	январь 2020 – июнь 2020
Количество признаков	144

На выходе классификатора модели предиктивного анализа получается не строгое значение класса, а уверенность в «положительном» результате, которая может принимать значения от 0 до 1. Для каждого объекта пути классификатор рассчитывает вероятность того, что объект принадлежит к положительному классу. Прогноз возникновения опасного отказа строится для следующего месяца.

2.1.4 Выбор порога принятия решения

Для принятия окончательного решения необходимо задать порог для вероятности — threshold. Если вероятность меньше этого порога, то объект пути относится к отрицательному классу и ему присваивается метка 0 (объект без опасного отказа). Если вероятность больше заданного значения threshold, объекту присваивается метка 1 (объект с потенциальным опасным отказом).

Выбор порога threshold строится исходя из следующих положений:

1. Порог threshold равен 0,5, если количество 0 и 1 в поле кодировки целевого признака встречаются одинаково часто и ошибки в данных имеют симметричное распределение.

2. Если важнее правильно предсказать опасные отказы, то порог должен быть снижен. Если важнее не допустить появление объектов пути, которые ошибочно будут классифицированы как 1, то порог должен быть увеличен.

2.2 Построение моделей

2.2.1 Обучение

Настоящая работа предполагает работу с больши́ми объемами данных, которые требуют соответствующих затрат вычислительных ресурсов.

В силу того, что нет возможности загрузить сразу все данные в обработку, их делят на части меньшего размера, загружают их по очереди и обновляют веса нейронной сети в конце каждого шага, подстраивая их под данные.

Далее используются следующие понятия:

• **epoch** — эпоха: количество прохождений всего набора данных через нейронную сеть в обоих направлениях;

• **batch** — батч: часть набора данных (обычно значительно меньше, чем весь набор данных);

• **batch size** — общее число тренировочных объектов, представленных в одном батче;

• **iteration** — число батчей, необходимых для завершения одной эпохи.

2.2.2 Измерение результатов

Естественным, простым и распространённым функционалом качества является точность (англ. Accuracy) — доля объектов, на которых модель выдала правильные ответы. Недостаток такого функционала очевиден: он плох в случае дисбаланса классов, когда представителей одного из класса существенно больше, чем другого. В этом случае, с точки зрения точности, выгодно почти всегда выдавать метку самого популярного класса.

Поэтому для измерения качества модели было решено использовать следущую метрику, аналогичную точности, в случае дисбаланса классов:

$$\texttt{balanced accuracy} = \frac{\texttt{TPR} + \texttt{TNR}}{2},$$

где TPR — процент правильно классифицированных объектов положительно класса, TNR — процент правильно классифицированных объектов негативного класса.

Рассмотрим матрицу несоответствий (англ. confusion matrix) на рис. 2.3.

Объекты, которые модель относит к положительному классу (метка 1), называются положительными (Positive), те из них,

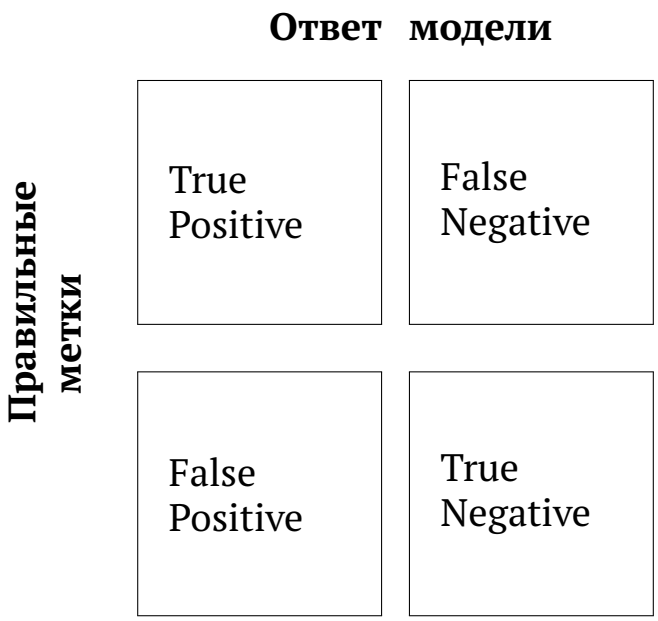

Рис. 2.3: Матрица несоответствий [8]

которые на самом деле принадлежат к этому классу — истинно положительными (True Positive), остальные — ложно положительными (False Positive). Аналогичная терминология есть для отрицательного (Negative) класса (метка 0). Дальше используем естественные сокращения:

- TP = True Positive,
- TN = True Negative,
- FP = False Positive,
- FN = False Negative.

В этих терминах исходный функционал качества принимает следующий вид:

$$\texttt{balanced accuracy} = \frac{\texttt{TPR} + \texttt{TNR}}{2} = \frac{1}{2}\left(\frac{\texttt{TP}}{\texttt{TP} + \texttt{FN}} + \frac{\texttt{TN}}{\texttt{TN} + \texttt{FP}}\right).$$

Если в бинарной задаче классификации представителей двух классов примерно поровну, то $\texttt{TP} + \texttt{FN} \approx \texttt{TN} + \texttt{FP}$, и сбалансированная точность примерно равна точности обычной (Accuracy).

2.2.3 Полносвязная нейронная сеть

Формат входных данных

Входные данные представляют собой тензор размера `(batch_size, num_features)`, где `num_features` — количество признаков в итоговом датасете.

Предложенная архитектура

Данная нейронная сеть состоит из четырех полносвязных слоев и функции активации ReLU [9] между ними. На выходе сети стоит функция активации Sigmoid, которая переводит множество выходных значений модели в отрезок $[0, 1]$, что по смыслу соответствует вероятности наличия положительного класса 1.

Используемые гиперпараметры

Количество эпох: `num_epoch = 50`
Размер батча: `batch_size = 1024`
Количество признаков: `num_features = 144`
Количество нейронов в скрытом слое: `num_hidden = 50`

Результаты

Оптимальный порог: `best threshold = 0.19`
При этом пороге матрица несоответствий имеет следующий вид:

Таблица 2.3: Матрица несоответствий для полносвязной НС

		Правильные метки		
		Истина	Ложь	Всего
Предсказания	Истина	10156	3253	13409
	Ложь	474	1367	1841
	Всего	10630	4620	15250

Функционал качества: `balanced accuracy = 0.7504`

2.2.4 Рекуррентная нейронная сеть

Для рекуррентной нейронной сети важна предистория наблюдений, поэтому по сравнению с предыдущим вариантом необходимо добавить размерность `history_size`.

Опишем процедуру формирования предыстории:

1. В исходных данных целевое значение, столбец TARGET, показывает отказ в *следующем* месяце, поэтому его необходимо сдвинуть на одну позицию вниз. Каждому километру пути соответствует строка признаков, в которой значение TARGET показывает, был ли отказ пути с заданными признаками.

2. Отсортировать все данные по признакам в следующем порядке: КМ (километр), ГОД, МЕСЯЦ, ДЕНЬ. Для каждого километра таких наблюдений в среднем около 70.

3. Удалить признаки «ГОД» и «ДЕНЬ», поскольку они не несут никакой дополнительной информации. Признак «МЕСЯЦ» оставлен в качестве некоторого климатического фактора: другие данные, которые можно было бы использовать в данном качестве, в исходных данных отсутствуют.

4. По наблюдениям для определенного километра пройтись окном размером `history_size` с шагом `step` (см. рис. 2.4).

В данном формате рекуррентная нейронная готова предсказывать следующее значение целевой переменной по заданному набору измерений.

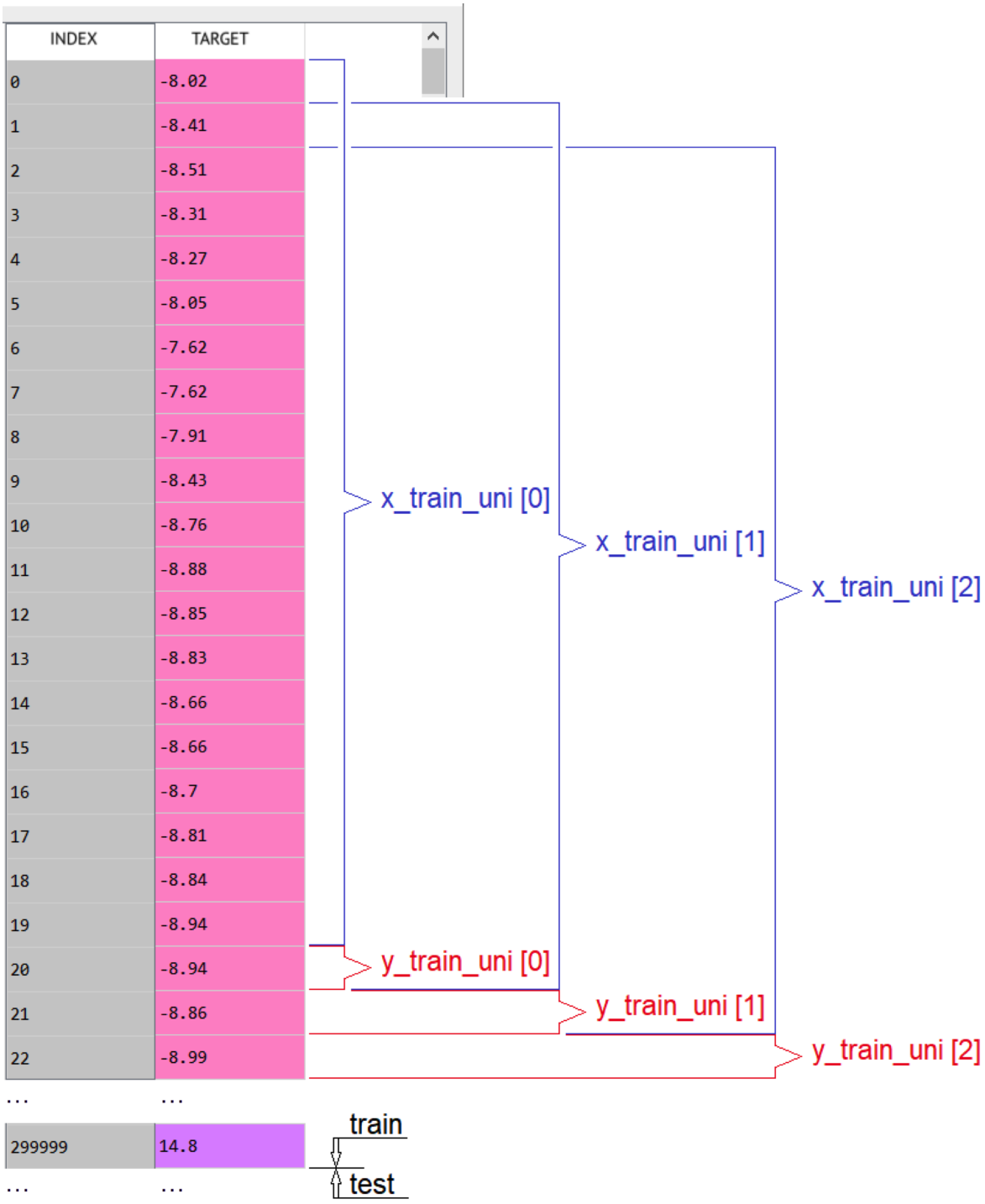

Рис. 2.4: Процесс прохода по данным окном

Если взять полученные данные и подать без предварительной обработки, качество на выходе будет сопоставимо с моделью случайного угадывания. Поэтому далее приведены этапы обработки данных для рекуррентной сети, которые помогли существенно повысить качество для обеих рекуррентных моделей RNN и GRU:

1. Перемешивание данных на обучающей выборке. Чтобы модель была более устойчива по входным данным, необходимо в процессе обучения подавать данные не в хронологическом порядке, а в случайном.

2. Нормировка данных. Все данные приводятся к нормальному виду путем вычитания среднего значения и деления на дисперсию. Это обычная практика, которая часто заметно улучшает итоговое качество моделей машинного обучения.

3. Уменьшение размерности пространства признаков с помощью метода главных компонент (PCA). Количество признаков в итоговом наборе данных задается параметром `num_features`.

4. Для увеличения количества объектов в обучающей, валидационной и тестовой выборках размер шага окна — гиперпараметр `step` — был установлен равным единице.

Чтобы модель лучше сходилась на последних эпохах обучения, использовался прием уменьшения шага обучения [10]. Через заданное количество эпох шаг обучения уменьшался в γ раз.

Формат входных данных

Входные данные представляют собой тензор размера `(batch_size, history_size, num_features)`,
где `history_size` — количество последних наблюдений для определенного километра пути, `num_features` — количество признаков в итоговом датасете.

Предложенная архитектура

на основе сети Элмана состоит из двух основных частей:

- рекуррентная сеть Элмана, которая выделяет наиболее важные признаки из данных (описана в подразд. 1.4.1);
- классификатор, который представляет собой один полносвязный слой, преобразующий выходные значения в вероятность того, что объект принадлежит к положительному классу.

на основе GRU блоков также состоит из двух основных частей:

- рекуррентная сеть из GRU-блоков, заменяющих привычные нейроны, которая, так же как и предыдущая модель, выделяет наиболее важные признаки (описана в подразд. 1.4.2);
- классификатор — аналогично архитектуре на основе сети Элмана.

Используемые гиперпараметры

Количество эпох: `num_epoch = 60`
Размер батча: `batch_size = 1024`
Количество признаков: `num_features = 10`
Размер окна: `history_size = 10`
Шаг окна: `step = 1`

Результаты для RNN

Оптимальный порог: `best threshold = 0.15`

При этом пороге матрица несоответствий имеет следующий вид:

Таблица 2.4: Матрица несоответствий для рекуррентной НС

		Правильные метки		
		Истина	Ложь	Всего
Предсказания	Истина	2516	591	3107
	Ложь	79	234	313
	Всего	2595	825	3420

Функционал качества: `balanced accuracy = 0.7900`

Результаты для GRU

Оптимальный порог: `best threshold = 0.12`

При этом пороге матрица несоответствий имеет следующий вид:

Таблица 2.5: Матрица несоответствий для рекуррентной НС

		Правильные метки		
		Истина	Ложь	Всего
Предсказания	Истина	2291	816	3107
	Ложь	52	261	313
	Всего	2343	1077	3420

Функционал качества: `balanced accuracy = 0.7926`

2.2.5 Автокодировщик

Формат входных данных
Входные данные представляют собой тензор размера `(batch_size, num_features)`, где `num_features` — количество признаков в итоговом датасете.

Предложенная архитектура
Данная нейронная сеть состоит из двух блоков: энкодера и декодера, как описано в разд. 1.3. Энкодер и декодер представляют собой три полносвязных слоя, разделенных функцией активацией ReLU, каждый. На вход энкодера поступает вектор с количеством признаков `num_features`, а на выходе имеем вектор размерности латентного пространства `dim_code`. Процесс прохода декодера зеркален относительно энкодера; размерность входа `dim_code`, выхода — `num_features`.

Используемые гиперпараметры
Количество эпох: `num_epoch = 40`
Размер батча: `batch_size = 1024`
Количество признаков: `num_features = 144`
Размер латентного пространства: `dim_code = 16`

Результаты
Оптимальный порог: `best threshold = 0.51`
При этом пороге матрица несоответствий имеет следующий вид:

Таблица 2.6: Матрица несоответствий для автокодировщика

		Правильные метки		
		Истина	Ложь	Всего
Предсказания	Истина	9819	3590	13409
	Ложь	521	1320	1841
	Всего	10340	4910	15250

Функционал качества: `balanced accuracy = 0.7285`

2.3 Анализ результатов

В настоящей работе рассмотрены несколько вариантов предиктивных моделей машинного обучения на основе нейронных сетей, призванных прогнозировать опасный отказ верхних строений железнодорожного пути:

- полносвязная нейронная сеть прямого распространения;
- рекуррентная нейронная сеть на основе сети Элмана;
- рекуреннтная нейронная сеть на основе GRU-блоков;
- автокодировщик.

Поскольку в исходных данных количество значений целевой переменной TARGET = 0 (объект без опасного отказа) гораздо меньше, чем значений TARGET = 1 (объект с потенциальным опасным отказом), показателем качества для всех четырех моделей был выбран функционал

$$\texttt{balanced accuracy} = (\texttt{TPR} + \texttt{TNR})/2,$$

который является аналогом точности (Accuracy) в случае дисбаланса классов.

Приведем сводную таблицу итоговых результатов по всем моделям. К общему сравнению также добавлена предиктивная модель на основе *градиентного бустинга* [11].

Самое высокое качество получилось у рекуррентной модели на основе GRU-блоков. Данный результат может быть обусловлен несколькими причинами:

- направленностью рекуррентных моделей на работу с последовательными типами данных;
- хорошей обобщающей способностью данной архитектуры.

Таблица 2.7: Итоговые результаты

Название	**Объем обучающей выборки**	**Качество**	**Время обучения**
Полносвязная НС	92347	0.7504	9 с
Рекуррентная НС на основе сети Элмана	78209	0.7900	62 с
Рекуррентная НС на основе GRU-блоков	78209	0.7926	66 с
Автокодировщик	92347	0.7285	79 с
Модели на основе градиентного бустинга	92347	0.7550	20 мин

Заключение

В настоящей выпускной квалификационной работе автором были предложены несколько принципиально различных архитектурных решений предиктивных моделей машинного обучения на основе нейронных сетей:

- полносвязная нейронная сеть;
- рекуррентные нейронные сети (RNN и GRU);
- автокодировщик.

Каждая из этих моделей была призвана прогнозировать опасный отказ верхних строений железнодорожного пути. Принимая на вход данные о состоянии объектов железнодорожного пути, модель возвращает вероятность отказа каждого из участков.

Полносвязная нейронная сеть — простейший вариант архитектуры — был рассмотрен автором в первую очередь. Данный тип сетей является довольно распространенным и используется для решения широкого круга задач, в том числе задачи классификации объектов.

Логичным продолжением исследования стало рассмотрение применимости рекуррентных нейронных сетей, которые исходно разрабатывались именно для работы с последовательностями (временными рядами). Эксперименты проведены с сетью на основе нейронной сети Элмана (RNN) и сетью на основе управляемых рекуррентных блоков (GRU).

Также был рассмотрен такой тип нейронных сетей как

автокодировщик. Это архитектура, позволяющая применять обучение без учителя при использовании метода обратного распространения ошибки. Идея применения их в приведенной задаче состоит в том, чтобы перевести все объекты в некоторое латентное пространство меньшей размерности и применить метод ближайших соседей для их классификации.

Лучше всего с поставленной задачей справилась рекуррентная нейронная сеть на основе управляемых рекуррентных блоков (GRU). Полученный результат превосходит остальные модели на основе нейронных сетей, причем как по итоговому качеству, так и по скорости работы. Кроме того, результаты работы рекуррентных нейронных сетей превосходят результаты, полученные на основе различных вариантов градиентного бустинга.

В дальнейшем можно постараться улучшить результаты работы, рассмотрев нейронные сети с архитектурой LSTM (Long Short-Term Memory), на основе внимания (Attention Neural Network) и Transformers. При этом следует учесть, что порог по точности с учетом примерно 17 % шума в объектах положительного класса практически достигнут. Незначительное увеличение точности, скорее всего, возможно получить за счет существенного увеличения времени обучения и дополнительных сложностей в применении моделей на практике.

Литература

[1] Sea News. Грузооборот российского транспорта.

[2] Goodfellow Ian, Bengio Yoshua, and Aaron Courville. *Back-Propagation and Other Differentiation Algorithms*. Deep Learning. MIT Press, 2016.

[3] Университет ИТМО. Проблемы нейронных сетей.

[4] Университет ИТМО. Рекуррентные нейронные сети.

[5] M. C. Mozer. *A Focused Backpropagation Algorithm for Temporal Pattern Recognition*. NJ: Lawrence Erlbaum Associates, 1995.

[6] Qidong Peng, Junqiang Lin, Dongsheng Wang, and Xuefei Liu. Simulating reservoir operation using a recurrent neural network algorithm. 2019.

[7] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng.

TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[8] Fawcett Tom. *An Introduction to ROC Analysis*. Pattern Recognition Letters, 2006.

[9] Simon S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[11] R. E. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. MIT Press, 2012.