

Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики
Кафедра теоретическая механика

Работа допущена к защите

Зав. кафедрой, д.ф.-м.н., чл.-корр. РАН


_____ А. М. Кривцов

"20" 01 2019

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

«ПРИМЕНЕНИЕ ТЕХНОЛОГИИ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПОВЫШЕНИЯ ТОЧНОСТИ ОЦЕНКИ ПЛАСТОВОГО ДАВЛЕНИЯ ДЛЯ ОПЕРАЦИЙ КРС»

по направлению 01.04.03 «Механика и математическое моделирование»
по образовательной программе
01.04.03_04 «Математическое моделирование процессов нефтегазодобычи»

Выполнил
студент гр. 23642/4



Т.Э. Баталяхин

Руководитель
Доцент, к.ф.-м.н.



В.А. Кузькин

Консультант
Нач. отдела



М.В. Наугольнов

Санкт-Петербург

2019

Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики
Кафедра теоретической механики

УТВЕРЖДАЮ

Зав. кафедрой, д.ф.-м.н., чл.-корр. РАН

_____ А. М. Кривцов

« 15 » 01 2019 г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы

студенту Баталяхину Тимуру Эдуардовичу, группы 23642/4

1. Тема работы: применение технологии машинного обучения для повышения точности оценки пластового давления для операций КРС.
2. Срок сдачи студентом законченной работы: 05.06.2019
3. Исходные данные по работе: исторические данные с месторождения компании Salym Petroleum Development (добыча и закачка нефти, измеренные значения давлений и прочее).
4. Содержание работы (перечень подлежащих разработке вопросов): Анализ данных, поступивших с месторождения. Разработка программы для прогнозирования пластового давления. Тестирование и сравнение с существующими подходами.
5. Консультант по работе: М.В. Наугольнов
6. Дата выдачи задания: 15.01.2019

Руководитель ВКР _____

(подпись)

В.А. Кузькин

Задание принял к исполнению _____

(дата)

15.01.2019

Студент _____

(подпись)

Т.Э. Баталяхин

РЕФЕРАТ

42 с., 10 рис., 3 табл., 9 источников

Ключевые слова: КРС (капитальный ремонт скважин), машинное обучение, нейронные сети, алгоритм.

Настоящая работа посвящена созданию и проверке алгоритма выдачи пластового давления на основании промысловых данных. Алгоритм настраивается на известные данные с месторождения и на основании этого выдает прогноз пластового давления для дальнейшей операции глушения скважины при капитальном ремонте. Применимость алгоритма проверена на исторических данных с месторождения. Данный алгоритм реализован в программном продукте Python.

THE ABSTRACT

Key words: well workover, machine learning, neural networks, algorithm.

This work is devoted to the creation and verification of the algorithm for issuing reservoir pressure on the basis of field data. The algorithm is adjusted to the known data from the field and on the basis of this produces a reservoir pressure forecast for further operation of well killing during major repairs. The applicability of the algorithm is tested on historical data from the field. This algorithm is implemented in the Python software product.

СОДЕРЖАНИЕ

Введение.....	5
Глава 1. Нейронные сети.....	8
1.1 Нейрон.....	8
1.2 Активационная функция.....	8
1.3 Многослойный персептрон.....	9
1.4 Обучение нейронной сети.....	9
1.5 Рекуррентные нейронные сети.....	12
1.6 Рекуррентная модель «вход-выход».....	12
1.7 Модель в пространстве состояний.....	14
1.8 Рекуррентный многослойный персептрон.....	15
1.9 Сеть второго порядка.....	16
1.10 Алгоритмы обучения.....	18
1.11 Обратное распространение во времени.....	19
1.12 Обратное распространение по эпохам во времени.....	20
Глава 2. Основная часть.....	22
2.1 Анализ и обработка исходных данных.....	22
2.1 Построение модели	24
2.1 Анализ результатов	25
Заключение.....	33
Список использованных источников.....	34
Приложение 1. Код алгоритма.....	35
Приложение 2. Динамика исследуемых параметров по скважинам.....	38

ВВЕДЕНИЕ

При эксплуатации нефтяных, нагнетательных или газовых скважин, особенно тех, которые работают достаточно длительное время, периодически возникает потребность в их ремонте. Ремонт обычно проводится на большой глубине и может быть подразделён на текущий и капитальный. Текущий ремонт – это такой тип ремонта скважины, при котором восстанавливают рабочее состояние техники и инструментов, меняют интенсивность, режим работы скважины и прочее. Капитальный же ремонт – такой, при котором изолируют отдельные нефтяные пропластки, полностью извлекают сломавшееся или устаревшее оборудование, производят ремонт ствола скважины, воздействуют на призабойную зону химическими или физическими способами, а в отдельных случаях даже полностью ликвидируют скважину в случае её нерентабельности, потери эффективности или других причин.

В основном при капитальном (а иногда и при текущем) ремонте требуется проводить глушение скважины. Глушение скважины – процесс, при котором создается достаточное противодавление, чтобы жидкость не поступала из пласта в скважину. Скважины обычно глушат либо пресной водой, либо соляным раствором (раствор глушения). Для определения параметров глушения важно знать параметры раствора глушения, а также параметры пласта (пластовое давление – давление на «глубине глушения») - для того, чтобы добываемая жидкость не поступала в скважину, но в то же время и не «заходила» глубоко в пласт (чтобы потом была возможность «вымыть» обратно эту жидкость при дальнейшей эксплуатации скважины). И если плотность раствора глушения определить достаточно легко, то значения пластового давления определить достаточно сложно, тем более в скважине, требующей ремонта.

Существуют множество методик определения пластового давления, они могут быть различными даже для разных компаний, которые занимаются этими вопросами. Но в основном все эти рекомендации выдаются специалистами с учетом уже проведенных замеров и с учетом предыдущего удачного или неудачного опыта глушения соседних скважин. Давления выдаются с большим диапазоном

(около 20-30 атм), что приводит к удлинению ремонта, либо к ухудшению режима работы скважины. Большое количество вводной информации по скважине также ведет к неточности определения пластового давления.

В современном мире высокая скорость развития технологий позволяет использовать их в тех сферах, в которых раньше они были неприменимы или недостаточно развиты. С каждым днем машинное обучение занимает всё больше и больше места в нашей жизни ввиду огромного количества вариантов его применений. Начиная от анализа пробок и заканчивая самоуправляемыми автомобилями, всё больше задач перекладывается на самообучаемые алгоритмы. Все чаще искусственный интеллект применяется и в нефтяной отрасли.

Однако в настоящее время еще не существует специализированного комплекса, в котором машинное обучение используется для прогнозирования параметров пласта по данным промысла. Данные методы в нефтяной отрасли применялись к другим задачам, например в работе [1] различные методы машинного обучения применяются для прогноза геологических свойств в межскважинном пространстве. В основном же машинное обучение сейчас применяется в тех отраслях, в которых уже существует в электронном виде большое количество известной и размеченной информации, как например в работе по предсказанию стоимости акций [2] или работе с текстовой информацией [3].

Цель работы - создание системы выдачи пластового давления на основании доступной информации с месторождения.

Задачи работы:

1. подготовить и предобработать данные для их последующей обработки алгоритмами машинного обучения;
2. написать программный код на языке Python;
3. выполнить валидацию данных на известных предсказаниях пластового давления, которые были выданы инженерами аналитически или получены при проведении операций КРС.

Проверяемая в ходе работы гипотеза: целесообразность и полезность использования технологии машинного обучения для повышения точности оценки

пластового давления. Вполне возможно, что специализированного комплекса до сих пор не существует как раз потому, что машинное обучение невозможно применить к данной задаче, либо оно не даёт существенного выигрыша в скорости или точности в сравнении с ранее использовавшимися методами.

Практическая значимость данной работы: повышение точности выдачи пластового давления позволит – снизить среднее время на проведение операций глушения, снизить среднее время вывода скважины на режим, снизить объем жидкости глушения. Также автоматизация процесса анализа позволит снизить время на выдачу пластового давления и повысить степень использования доступной информации по месторождению.

При написании данной работы использовались данные с месторождения компании Salym Petroleum Development N.V., которая находится под контролем ПАО «Газпром нефть». Объект исследования - участок Верхнесалымского месторождения, однопластовая залежь (пласт АС10-11). Практическая часть была выполнена после ознакомления с внутренними документами предприятия: «Инструкция по определению пластового давления при ремонтах и ГТМ на скважинах компании» [5]. В основном рекомендации, отраженные в данной инструкции носят экспериментальный характер и несут в себе обобщение всех тех результатов работ, которые были проведены на месторождениях компании.

Глава 1. НЕЙРОННЫЕ СЕТИ

1.1. Нейрон

Нейрон – это функция вида:

$$z = f(\mathbf{w}^T \mathbf{z})$$

где вектор $\mathbf{z} = (1, z_1, \dots, z_m)^T$, z_1, \dots, z_m – значения нейронов предыдущего слоя на выходе, $\mathbf{w} = (w_0, w_1, \dots, w_m)$ – веса нейронов (настраиваемые параметры модели), f – функция активации. Значение аргумента функции f (а именно $a = \mathbf{w}^T \mathbf{z}$) называется активацией данного нейрона. Нейрон является единицей обработки информации в нейронной сети.

1.2. Активационная функция

Функция активации преобразует входные сигналы нейрона в выходные и по сути является зависимостью входных сигналов от выходных. Обычно это монотонно возрастающая функция, область значений у которой находится в интервале $[-1; 1]$ (например гиперболический тангенс) или в интервале $[0; 1]$ (сигмоида). Для многих алгоритмов обучения требуется, чтобы функция активации была непрерывно дифференцируемой на всей области определения. Нейрон определяется своей активационной функцией (например существует «сигмоидальный нейрон»).

Далее представлены основные функции активации:

- функция Хевисайда (еще ее называют пороговой функцией)

$$f(x) = \begin{cases} 1, & x \geq -w_0 x_0 \\ 0 & \end{cases}$$

- сигмоидальная функция

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- гиперболический тангенс

$$\tanh(Ax) = \frac{e^{Ax} - e^{-Ax}}{e^{Ax} + e^{-Ax}}$$

1.3. Многослойный персептрон

Многослойный персептрон – такая нейронная сеть, в которой существует несколько слоев нейронов, каждый из которых связан со всеми нейронами предыдущего слоя и с каждым нейроном из последующего слоя.

Для каждого нейрона i -го слоя справедливо следующее утверждение: он принимает на вход активированные (взвешенные) выходные значения от каждого нейрона с предыдущего слоя, а в свою очередь его выходные значения с соответствующими весами передаются на вход каждому нейрону следующего слоя. На одном слое связи между нейронами не существует.

Входы (куда подаются начальные значения) нейронной сети формируют входной слой (input layer). Аналогично для выходов нейронной сети – из них образуется выходной слой (output layer). Те слои, которые находятся между ними, называются скрытыми слоями (hidden layers).

Если бы небольшое смещение (или изменение весов) вызывало бы аналогичное небольшое изменение на выходе многослойного персептрона, то нужное поведение сети получалось бы при помощи простого регулирования весов и смещений в процессе его обучения. Однако так не происходит, потому что даже небольшое смещение весов ведет к принципиально другому ответу на выходе. Поэтому в современных работах все чаще используются сигмоидальные нейроны (нейроны с сигмоидальной функцией активации).

1.4. Обучение нейронной сети

Обучение нейронной сети — это такой процесс, в котором свободные параметры нейронной сети настраиваются посредством моделирования среды, в которую эта сеть встроена. Тип обучения определяется способом подстройки параметров.

Это определение процесса обучения предполагает следующую последовательность событий:

1. в нейронную сеть поступают стимулы из внешней среды,
2. в результате этого изменяются свободные параметры нейронной сети,

3. после изменения внутренней структуры нейронная сеть отвечает на возбуждения уже иным образом.

Существует 3 парадигмы обучения нейронных сетей:

- обучение с учителем,
- обучение без учителя,
- обучение с подкреплением.

Обучение с учителем: участие учителя можно рассматривать как наличие знаний об окружающей среде, представленной в виде пар вход-выход. При этом сама среда неизвестна обучаемой нейронной сети. Теперь предположим, что учителю и обучаемой сети подается обучающий вектор из окружающей среды. На основе встроенных знаний учитель может сформировать и передать обучаемой нейронной сети желаемый отклик, соответствующий данному входному вектору. Этот желаемый результат представляет собой оптимальные действия, которые должна выполнить нейронная сеть. Параметры сети корректируются с учетом обучающего вектора и сигнала ошибки. Сигнал ошибки – это разность между желаемым сигналом и текущим откликом нейронной сети. Корректировка параметров выполняется пошагово с целью имитации нейронной сетью поведения учителя. Эта эмуляция в некотором статистическом смысле должна быть оптимальной. Таким образом, в процессе обучения знания учителя передаются в сеть в максимально полном объеме. После окончания обучения учителя можно отключить и позволить нейронной сети работать со средой самостоятельно.

Обучение без учителя (или обучение на основе самоорганизации) - при использовании этого подхода не существует маркированных примеров, по которым производится обучение сети. Обучение осуществляется без вмешательства внешнего учителя или корректора, контролирующего процесс обучения. Существует лишь мера качества представления, которому должна научиться нейронная сеть, и свободные параметры сети оптимизируются по отношению к этой мере. После обучения сети на статистические закономерности входного сигнала, она способна формировать внутреннее представление кодируемых признаков входных данных и таким образом автоматически создавать новые классы.

Обучение с подкреплением или нейродинамическое программирование - формирование отображения входных сигналов в выходные выполняется в процессе взаимодействия с внешней средой с целью минимизации скалярного индекса производительности. Такая система предполагает обучение с отложенным подкреплением. Это значит, что система получает из внешней среды последовательность сигналов возбуждения (т. е. векторов состояния), которые приводят к генерации эвристического сигнала подкрепления. Целью обучения является минимизация функции стоимости перехода, определенной как математическое ожидание кумулятивной стоимости действий, предпринятых в течение нескольких шагов, а не просто текущей стоимости. Может оказаться, что некоторые предпринятые ранее в данной последовательности действия были определяющими в формировании общего поведения всей системы. Функция обучаемой машины, составляющая второй компонент системы, определяет эти действия и формирует на их основе сигнал обратной связи, направляемый во внешнюю среду.

Практическая реализация обучения с отложенным подкреплением осложнена по двум причинам:

- не существует учителя, формирующего желаемый отклик на каждом шаге процесса обучения,

- наличие задержки при формировании первичного сигнала подкрепления требует решения временной задачи присваивания коэффициентов доверия. Это значит, что обучаемая машина должна быть способна присваивать коэффициенты доверия и недоверия действиям, выполненным на всех шагах, приводящих к конечному результату, в то время, как первичный сигнал подкрепления формируется только на основе конечного результата.

Обучение с подкреплением тесно связано с динамическим программированием — методологией, созданной Беллманом в 1957 году.

1.5. Рекуррентные нейронные сети

Рекуррентными называются нейронные сети, имеющие одну или несколько обратных связей. В основном рекуррентные сети используют глобальные обратные связи.

Обычно в качестве «строительного блока» используется многослойный персептрон, при этом применение обратной связи может принимать несколько форм. Во-первых, можно замкнуть выходной слой персептрона на его входной слой. Во-вторых, можно замкнуть выход скрытого слоя на вход. Так как многослойный персептрон может содержать несколько скрытых слоев, то последняя форма обратной связи может быть также сконфигурирована различными способами. Все это приводит к тому, что рекуррентные сети имеют богатый спектр архитектурных форм.

Далее будут описаны четыре основные архитектуры таких сетей, каждая из которых соответствует некоторой частной форме глобальной обратной связи. Все эти архитектуры имеют следующие общие характеристики:

- все они состоят из статического многослойного персептрона или его составных частей,
- все они используют способность многослойного персептрона выступать в качестве отображения вход-выход (нелинейного оператора).

1.6. Рекуррентная модель «вход-выход»

На рис.1.1 показана архитектура обобщенной рекуррентной сети, построенной на базе многослойного персептрона. Эта модель имеет единственный вход, который применяется к памяти на линиях задержки, состоящей из q элементов. Она имеет единственный выход, замкнутый на вход через память на линиях задержки, которая также состоит из q элементов. Содержимое этих двух блоков памяти используется для питания входного слоя персептрона. Вход модели обозначается как $u(n)$, а соответствующий выход - $y(n+1)$. Это значит, что

выход модели упреждает ее вход на одну единицу времени. Таким образом, вектор сигнала, подаваемый на вход персептрона состоит из окна данных, состоящего из следующих элементов:

Текущее и предыдущие значения входного сигнала $u(n)$, $u(n - 1)$, \dots , $u(n - q + 1)$, которые представляют сети, имеющие внешнее происхождение.

-значения выходного сигнала $y(n)$, $y(n - 1)$, \dots , $y(n - q + 1)$ в предшествующие моменты времени, от которых зависит выход модели $y(n + 1)$.

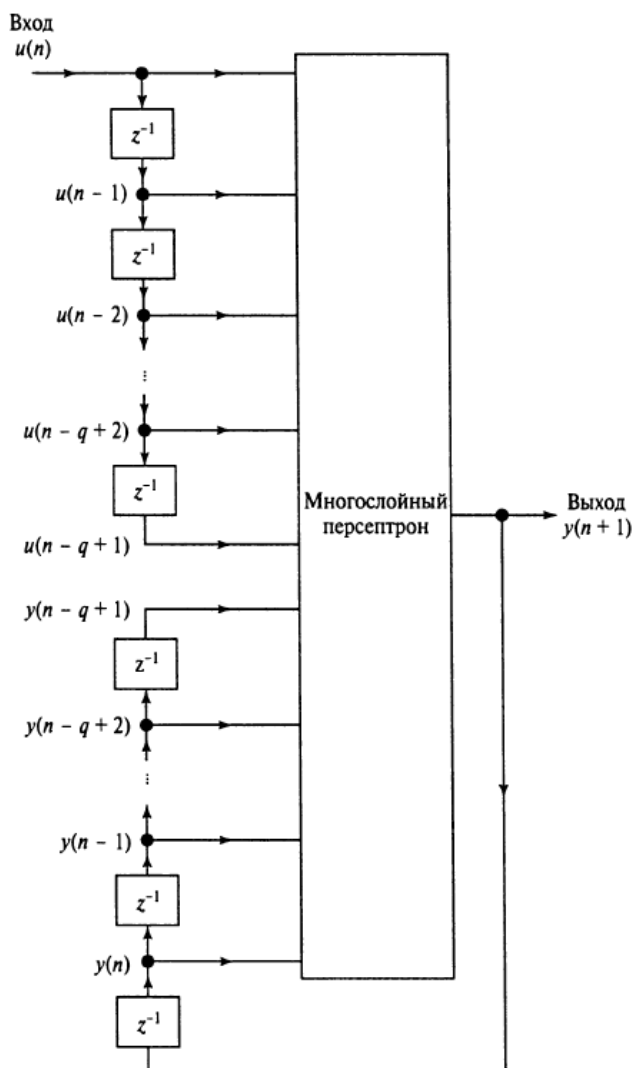


Рис. 1.1 Модель нелинейной регрессии с внешними входами (NARX).

Таким образом, рекуррентную сеть показанную на рис.1.1, можно рассматривать как модель нелинейной авторегрессии с внешними входами (nonlinear autoregressive with exogenous inputs model – NARX). Динамика модели NARX описывается следующим образом:

$$y(n + 1) = F(y(n), \dots, y(n - q + 1), u(n), \dots, u(n - q + 1)),$$

где F – некоторая нелинейная функция своих аргументов. Обратите внимание, на рис.1.1 предполагается, что обе памяти на дискретной линии задержки имеют размер q . В общем случае эти размеры могут отличаться.

1.7. Модель в пространстве состояний

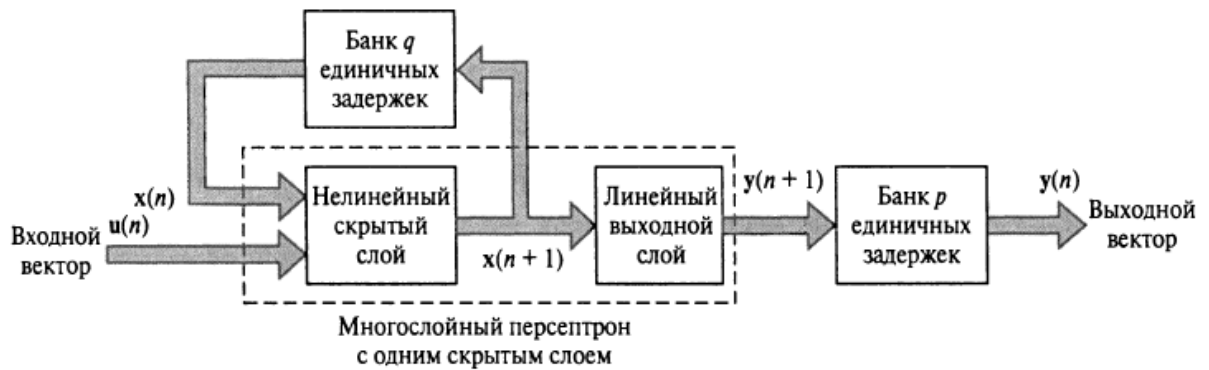


Рис. 1.2. Модель в пространстве состояний.

На рис.1.2 показана блочная диаграмма еще одной обобщенной рекуррентной сети, которая называется моделью в пространстве состояний (state-space model). Скрытые нейроны определяют состояние сети. Выход скрытого слоя замкнут на входной слой через банк единичных задержек. Входной слой сети состоит из объединения узлов обратной связи и узлов источника. Связь сети с внешней средой осуществляется через узлы источника. Количество единичных задержек, используемых для замыкания выхода скрытого слоя на входной слой, определяет порядок модели. Обозначим $\mathbf{u}(n)$ вектор входных сигналов размерности $m \times 1$ в момент времени n , а символом $\mathbf{x}(n)$ – вектор выходных сигналов скрытого слоя размерности $q \times 1$ в тот же момент времени. Тогда динамику этой модели можно описать следующей системой уравнений:

$$\mathbf{x}(n + 1) = \mathbf{f}(\mathbf{x}(n), \mathbf{u}(n)),$$

$$\mathbf{y}(n) = \mathbf{C}\mathbf{x}(n),$$

где \mathbf{f} – некоторая нелинейная функция, характеризующая скрытый слой; \mathbf{C} – матрица синаптических весов, характеризующих выходной слой. Скрытый слой

этой сети нелинеен, а выходной – линеен. Нейронная сеть на рис.1.2 включает ряд рекуррентных архитектур в качестве частных случаев.

1.8. Рекуррентный многослойный перцептрон

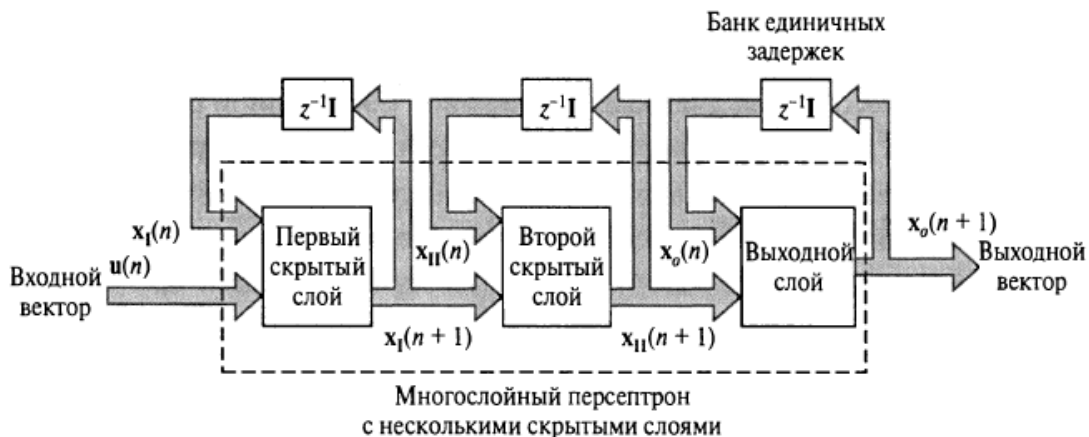


Рис. 1.3. Рекуррентный многослойный перцептрон.

Третья рассматриваемая рекуррентная архитектура известна как рекуррентный многослойный перцептрон (recurrent multilayer perceptron – RMLP). Этот перцептрон имеет один или несколько скрытых слоев (использование нескольких скрытых слоев определяется тем, что статичный многослойный перцептрон обычно является более эффективным и компактным, чем использующий только один скрытый слой). Каждый вычислительный слой RMLP замкнут на себя собственной обратной связью. На рис.1.3 – частный случай RMLP с двумя скрытыми слоями.

Пусть вектор $\mathbf{x}_1(n)$ обозначает выход первого скрытого слоя, вектор $\mathbf{x}_2(n)$ – выход второго скрытого слоя и т.д. Обозначим $\mathbf{x}_0(n)$ выход выходного слоя. Тогда динамика RMLP в ответ на подачу на вход вектора $\mathbf{u}(n)$ в общем случае будет описываться следующей системой уравнений:

$$\begin{aligned} x_1(n+1) &= \phi_1(x_1(n), u(n)) \\ x_2(n+1) &= \phi_2(x_2(n), x_1(n+1)) \\ &\dots, \\ x_0(n+1) &= \phi_0(x_0(n), x_k(n+1)) \end{aligned}$$

где $\phi_1()$, $\phi_2()$, $\phi_0()$ – функции активации, характеризующие первый, второй и другие скрытые слои, а также выходной слой RMLP, K – общее количество скрытых слоев в сети

1.9. Сеть второго порядка

При описании модели в пространстве состояний для обозначения количества скрытых нейронов, выходы которых замкнуты на выходной слой через банк единичных задержек, использовался термин «порядок».

В другом контексте этот термин иногда используется для обозначения способа определения индуцированного локального поля нейрона. Для примера рассмотрим многослойный персептрон, индуцированное локальное поле v_k нейрона k в котором определяется следующим выражением

$$v_k = \sum_j w_{a,kj} x_j + \sum_i w_{b,ki} u_i$$

где x_j – сигнал обратной связи, направленный от скрытого нейрона j , u_i – сигнал источника, применяемый к нейрону i входного слоя. Символами w обозначаются соответствующие синаптические веса сети. Нейрон, показанный на рис. 1.4 называют нейроном первого порядка (first order neuron). Когда же индуцированное локальное поле формируется с использованием перемножения:

$$v_k = \sum_i \sum_j w_{kij} x_i u_j,$$

то такой нейрон называют нейроном второго порядка. Нейрон второго порядка k использует один вес w_{kij} , который связывает его с входными узлами i и j .

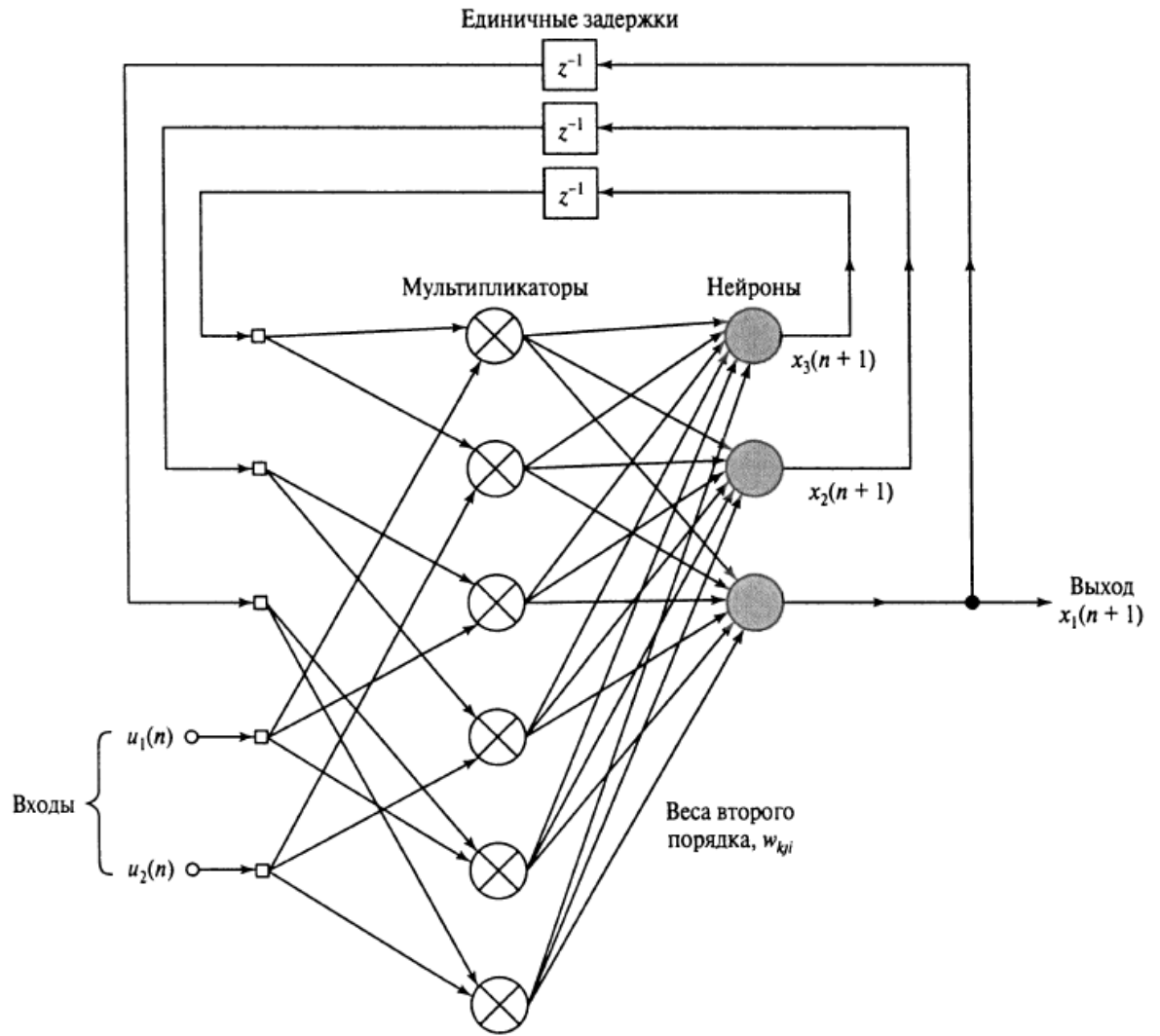


Рис. 1.4. Рекуррентная сеть второго порядка

Нейроны второго порядка лежат в основе рекуррентной сети второго порядка (пример на рис.1.4). Эта сеть принимает упорядоченную по времени последовательность входных сигналов и имеет динамику, описываемую следующей системой уравнений:

$$v_k(n) = b_k + \sum_i \sum_j w_{kij} x_i(n) u_j(n)$$

$$x_k(n+1) = \varphi(v_k(n)) = \frac{1}{1 + e^{-v_k(n)}}$$

где $v_k(n)$ – индуцированное локальное поле нейрона k ; b_k – соответствующее внешнее смещение (bias); $x_k(n)$ – состояние (выход) нейрона k ; $u_j(n)$ – входной сигнал, поступающий на узел источника j ; w_{kij} – вес нейрона второго порядка k .

Уникальным свойством рекуррентной сети второго порядка является то, что произведение $x_j(n)u_j(n)$ представляет собой пару {состояние, вход}; положительный вес w_{kij} описывает наличие перехода состояния (state transition) {состояние, вход} \rightarrow {следующее состояние}, а отрицательное значение веса – отсутствие такого перехода. Переход описывается выражением:

$$\delta(x_i, u_j) = x_k$$

1.10. Алгоритмы обучения

Как мы уже знаем, существует 2 режима обучения обычного (статического) многослойного пессетрона: пакетный и последовательный. В пакетном режиме перед коррекцией свободных параметров вычисляется чувствительность сети для всего множества обучения. С другой стороны, в последовательном режиме настройка параметров выполняется после подачи каждого примера обучения. Подобно этому, для обучения рекуррентной сети также существуют два режима обучения.

Обучение по эпохам (epochwise training). В заданной эпохе рекуррентная сеть начинает свою работу с некоторого исходного состояния и развивается, пока не достигнет некоторого другого состояния. В этой точке обучение приостанавливается, и сеть сбрасывается в исходное состояние, после чего начинается следующая эпоха обучения. Исходное состояние не обязательно должно быть одинаковым для всех эпох. Важно только, чтобы исходное состояние каждой эпохи отличалось от конечного состояния предыдущей.

Непрерывное обучение (continuous training). Второй метод обучения подходит для ситуаций, когда недоступны состояния сброса и (или) требуется обучение в реальном времени. Отличительной чертой непрерывного обучения является то, что сеть обучается во время самой реальной обработки сигнала, т.е. процесс обучения никогда не останавливается.

Теперь, после рассмотрения этих двух режимов обучения, рассмотрим следующие алгоритмы обучения:ри

- алгоритм обратного распространения во времени (back-propagation-through-time), работает в предположении того, что временные операции рекуррентной сети могут быть представлены посредством многослойного персептрона. Это позволяет применить стандартный алгоритм обратного распространения. Обратное распространение во времени можно реализовать как по эпохам, так и в непрерывном режиме. Можно также комбинировать эти два режима.

- алгоритм рекуррентного обучения в реальном времени, является производным от модели в пространстве состояний.

Эти два алгоритма имеют ряд общих характеристик. Во-первых, оба они основаны на методе градиентного спуска, при этом мгновенное значение функции стоимости (основанной на критерии среднеквадратичной ошибки) минимизируется по синаптическим весам сети. Во-вторых, оба эти метода довольно просты в реализации, но могут медленно сходиться. В-третьих, они связаны тем, что представление алгоритма обратного распространения во времени в виде графа движения сигнала может быть получено перестановкой из графа движения сигнала определенной формы алгоритма рекуррентного обучения в реальном времени.

1.11. Обратное распространение во времени

Алгоритм обратного распространения во времени (back propagation-through-time algorithm - ВРРТ), применяемый для обучения рекуррентных сетей, является расширением стандартного алгоритма обратного распространения. Он может быть получен путем развертывания временных операций сети в многослойной сети прямого распространения, топология которой расширяется на один слой для каждого шага времени.

Предположим, что \mathbf{N} – рекуррентная сеть, требуемая для обучения временной задаче, которая начинается с момента времени p_0 и продолжается до момента времени n . Пусть \mathbf{N}^* - сеть прямого распространения, которая получается после свертывания временных операций рекуррентной сети \mathbf{N} . Развернутая сеть \mathbf{N}^* связана с исходной сетью \mathbf{N} следующим образом:

1. Для каждого интервала $(n_0, n]$ сеть N^* содержит слой, состоящий из K нейронов, где K – количество нейронов исходной сети N ,
2. В каждом слое сети N^* содержатся копии всех нейронов сети N ,
3. Для каждого шага времени $l \in [n_0, n]$, синаптические связи от нейрона i слоя l к нейрону j слоя $l + 1$ сети N^* являются копиями синаптических связей между нейронами i и j сети N .

1.12. Обратное распространение по эпохам во времени

Пусть множество данных, используемое для обучения рекуррентной сети, разбито на независимые эпохи, каждая из которых представляет интересующий отрезок времени. Пусть n_0 – время начала эпохи, n_1 – время ее окончания. Рассматривая такую эпоху, можно определить следующую функцию стоимости:

$$E_{\text{общ.}}(n_0, n_1) = \frac{1}{2} \sum_{n=n_0}^{n_1} \sum_{j \in A} e_j^2(n),$$

где A – множество индексов j , относящееся к тем нейронам сети, для которых определены желаемые отклики; $e_j(n)$ – сигнал ошибки на выходе этих нейронов, измеренный по отношению к некоторому желаемому отклику. Требуется вычислить чувствительность этой сети, т.е. частные производные функции стоимости $E_{\text{общ.}}(n_0, n_1)$ по синаптическим весам сети. Для этого можно использовать алгоритм обратного распространения во времени по эпохам, который построен на пакетном режиме стандартного обучения методом обратного распространения. Алгоритм обучения по эпохам ВРТТ выполняется следующим образом.

- Сначала осуществляется прямая передача данных по сети на интервале времени (n_0, n_1) . Для записи входных данных состояние сети (т.е. ее синаптические веса) и желаемый отклик для этого интервала времени сохраняются.

- Для вычисления значений локальных градиентов выполняется единичная обратная передача через последнюю запись:

$$\delta_j(n) = -\frac{\partial E_{\text{общ.}}(n_0, n_1)}{\partial v_j(n)}$$

для всех $j \in A$ и $n_0 < n \leq n_1$. Это вычисление выполняется с помощью формулы:

$$\delta_j(n) = \begin{cases} \varphi'(v_j(n)) e_j(n), & n = n_1, \\ \varphi'(v_j(n)) \left[e_j(n) + \sum_{k \in A} w_{jk} \delta_k(n+1) \right], & n_0 < n < n_1 \end{cases}$$

где $\varphi'(\cdot)$ – производная функции активации по своему аргументу; $v_j(n)$ – индуцированное локальное поле нейрона j . Предполагается, что все нейроны сети имеют одну и ту же функцию активации φ . Вычисления по формуле повторяются с момента времени n_1 , шаг за шагом, пока не будет достигнут момент n_0 . Количество выполняемых шагов равно количеству шагов времени в данной эпохе.

-После выполнения вычислений по методу обратного распространения до момента времени $n_0 + 1$ к синаптическим весам w_{ji} нейрона j применяется следующая коррекция:

$$\Delta w_{ji} = -\eta \frac{\partial E_{\text{общ.}}(n_0, n_1)}{\partial w_{ji}} = \eta \sum_{n=n_0+1}^{n_1} \delta_j(n) x_i(n-1)$$

где η – параметр скорости обучения; $x_i(n-1)$ – входной сигнал, поданный на i -й синапс j -го нейрона в момент времени $n-1$.

Сравнивая описанную процедуру алгоритма ВРТТ по эпохам с пакетным режимом стандартного алгоритма обратного распространения, видим, что главное отличие заключается в том, что в первом случае желаемый отклик определяется для многих слоев сети, поскольку фактический выходной слой рекуррентной сети многократно воспроизводится при развертывании ее поведения во времени.

Глава 2.

ОСНОВНАЯ ЧАСТЬ

2.1. Анализ и обработка исходных данных

Сначала потребовалось выгрузить данные, полученные при добыче нефти на месторождении из специализированного программного обеспечения. Для этого был реализован вспомогательный макрос на языке программирования Visual Basic в приложении Excel. После этого эти данные (далее - выборка) были разделены на выборки по каждой скважине, а сами скважины в свою очередь по нескольким классам:

1. весь период времени проработавшие в режиме добычи (номера с 1 по 13),
2. весь период времени проработавшие в режиме закачки (нагнетательные),
3. проработавшие часть времени в режиме добычи, а далее переведенные в режим закачки.

Для последнего типа скважин выборка была также разделена на 2 по принципу: только добыча и только закачка. Таким образом получились 2 варианта скважин: те, которые только добывали (не обязательно весь период наблюдения) и те, которые только закачивали воду в пласт. Таким образом можно рассматривать только добывающие скважины и влияние на них нагнетательных (влияние соседних добывающих друг на друга на данном этапе не исследуется).

На рис.2.1 можно увидеть карту участка месторождения, на которой подписаны числами от 1 до 13 исследуемые добывающие скважины. По скважинам без номера (черный прямоугольник) оказалось слишком мало данных, тестовая и обучающая выборки были бы одинакового размера и поэтому было решено не включать их в обучение модели. Добыча нефти на диаграммах обозначена коричневым, добыча воды – зеленым, а нагнетательные скважины обозначены буквой «N». Размер круга зависит от объемов добычи/закачки.

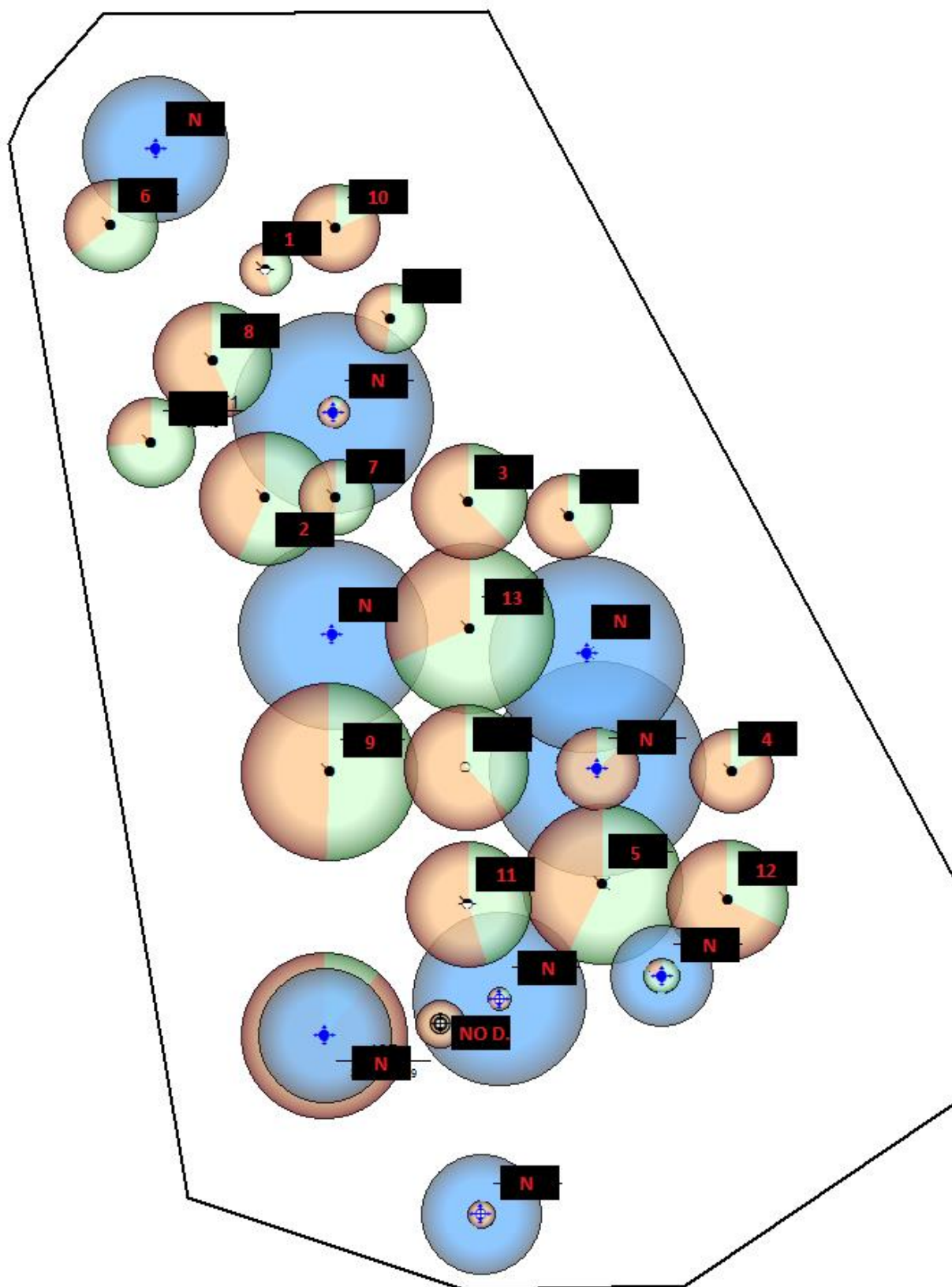


Рис. 2.1. Карта участка месторождения.

2.2. Построение модели

Для создания алгоритма машинного обучения была использована открытая нейросетевая библиотека Keras, написанная на языке Python. Она представляет собой надстройку над фреймворками DeepLearning4j, TensorFlow и Theano.

В процессе работы были использованы различные варианты архитектуры нейронной сети и различные варианты гиперпараметров, однако долго не удавалось достичь даже сходимости на обучающей выборке. По окончании настройки нейросети было принято решение остановиться на архитектуре, представленной на рис.2.2. Значения гиперпараметров сети: размер батча – 8 элементов (достаточно быстрое обучение, но без потери точности), количество эпох на каждой скважине – в среднем около 25000, время обучения алгоритма на каждой из скважин – порядка 15-15,5 минут. Применяемый оптимизатор – Adam, функция потерь вычислялась по методу среднеквадратичного отклонения, используемая метрика – доля правильных ответов алгоритма (accuracy). Также для более точного прогноза в модели были заданы веса значимости признаков – веса признаков на последний год работы скважины перед тестовым периодом были в 100 раз больше, чем в предыдущие года. Соотношение обучающей и тестовой выборки – 9:1 (10% тестируемых признаков – последние полгода работы скважины).

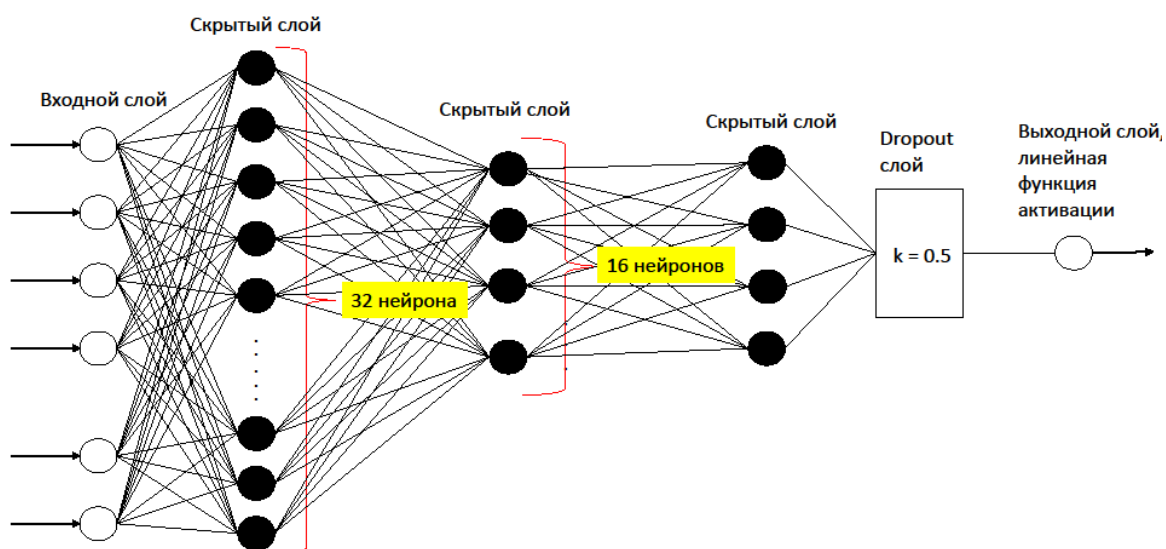


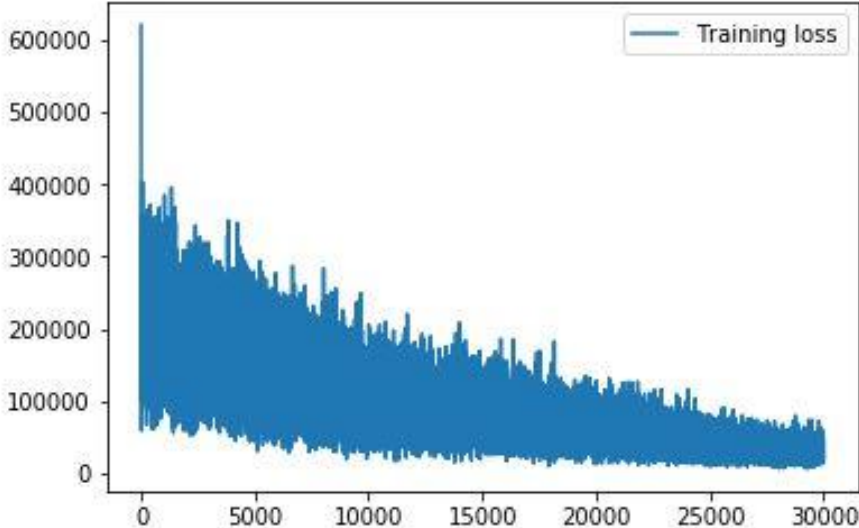
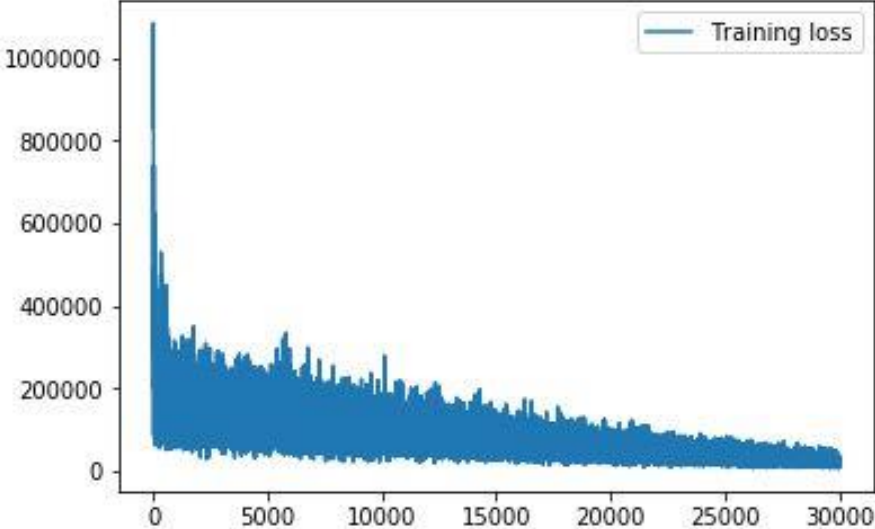
Рис. 2.2. Архитектура построенной нейронной сети.

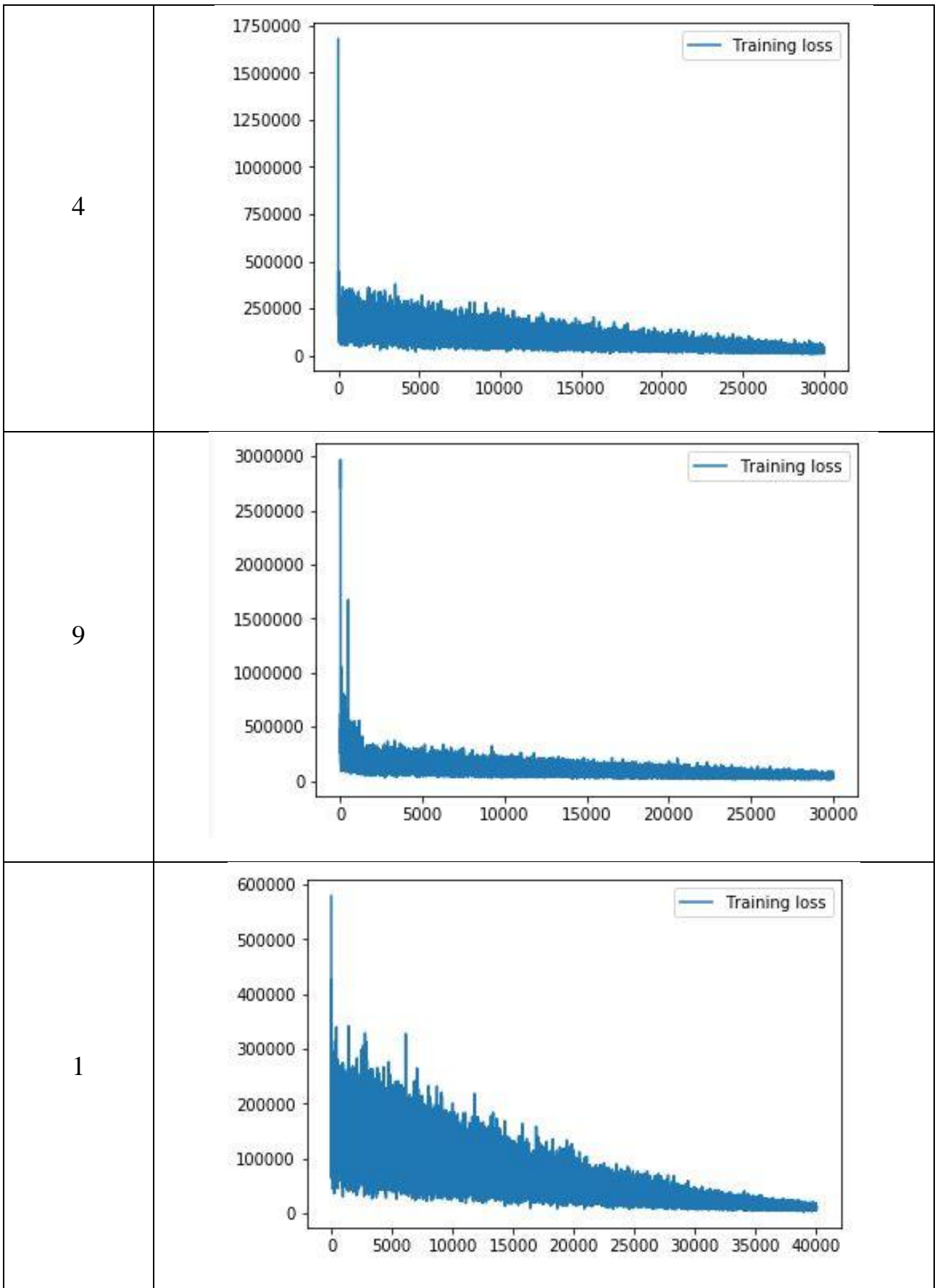
2.3. Анализ результатов

В таблице 2.1 выборочно для нескольких скважин приведены графики падения функции потерь (loss) в зависимости от очередной итерации (эпохи) обучения.

Таблица 2.1.

Функция потери (loss).

Номер скважины	Ошибка на обучающей выборке (loss)
7	
12	

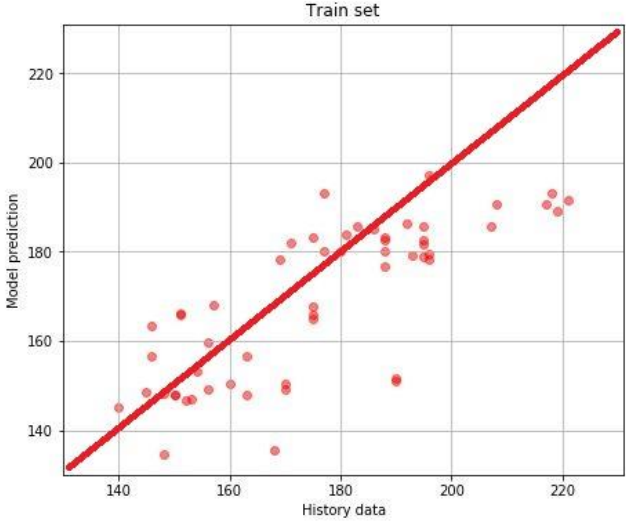
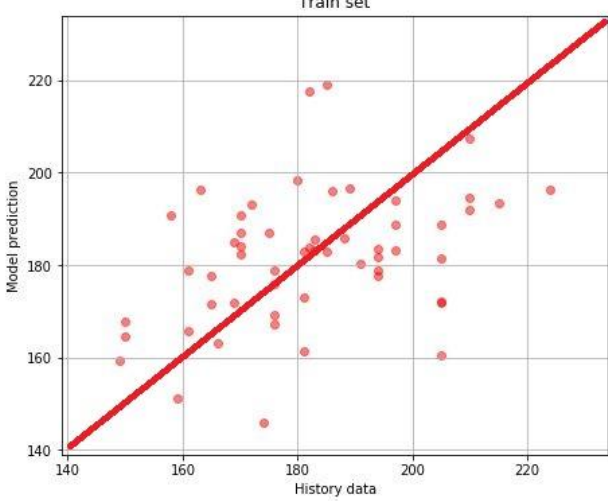


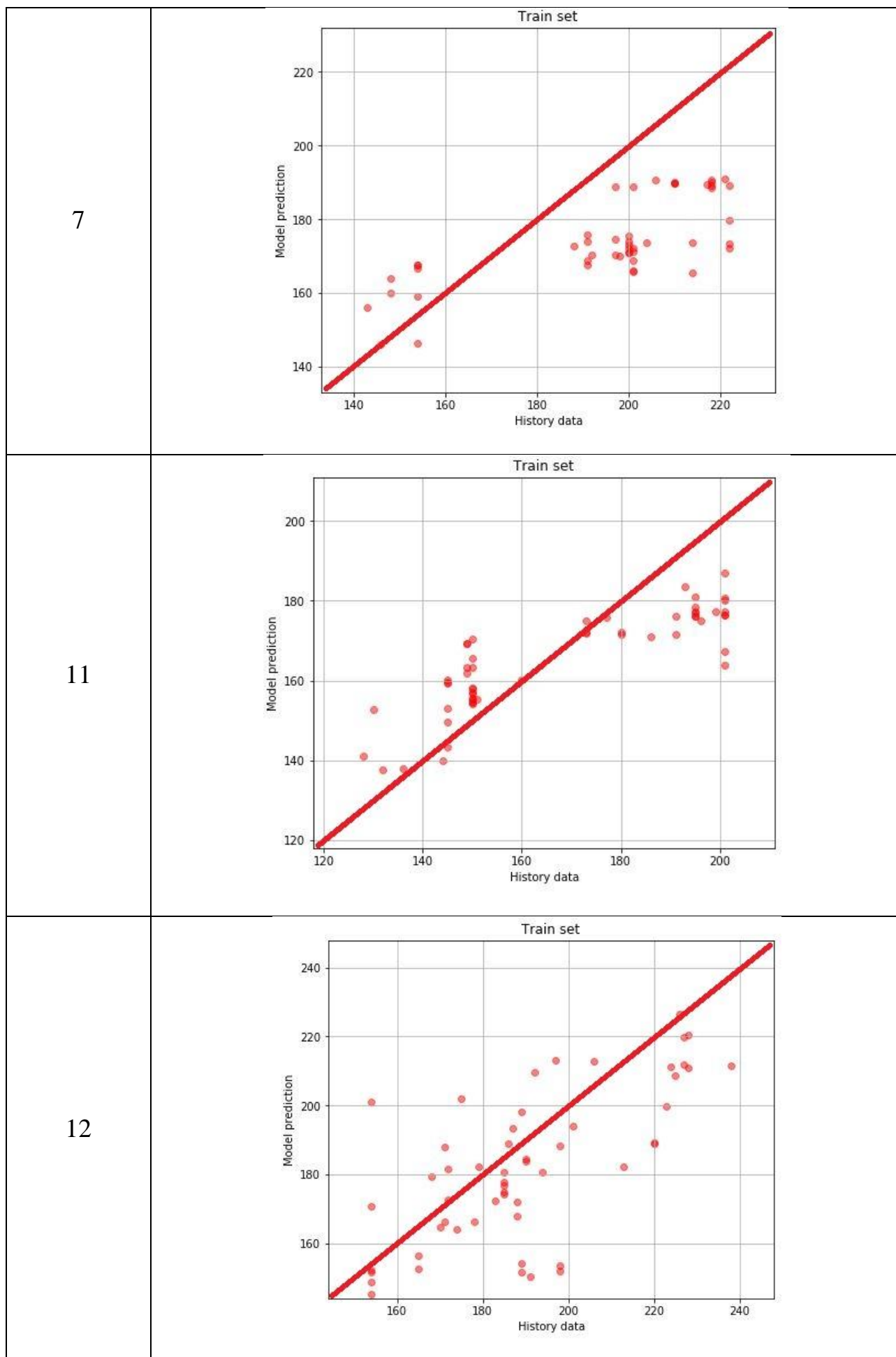
Функция «Loss» показывает, насколько классификатор «уверен» в своём прогнозе, соответственно, как можно убедиться из этих графиков, метод сходится, правда достаточно медленно, при этом существует осцилляция уверенности прогноза. Это можно объяснить тем, что мы на каждой эпохе обучения в модель на вход подаются перемешанные данные из разных частей выборки.

В таблице 2.2 выборочно для различных скважин представлены кросс-плоты, отражающие насколько хорошо обучилась модель на тестовой выборке. По оси абсцисс отложены исторические значения, а по оси ординат – полученные при помощи рекуррентной нейронной сети.

Таблица 2.2.

Исторические и модельные данные на обучающей выборке.

Номер скважины	Пластовое давление
2	
3	



Как можно увидеть из графиков, на многих скважинах существуют большие отличия предсказанных данных от исторических. Это может говорить нам о том, что модель «настроилась» не на те данные, либо недостаточно качественно. Однако дальнейшее обучение модели невозможно из-за так называемой проблемы переобучения: когда на обучающей выборке прогноз улучшается, а на тестовой ухудшается, т. е. модель настраивается на обучающую выборку и теряет способность предсказывать.

Далее будут приведены графики динамики пластового давления для нескольких скважин, на которых будет видна вся наша история и результаты работы алгоритма, а также все входные параметры (остальные графики помещены в приложение 2).

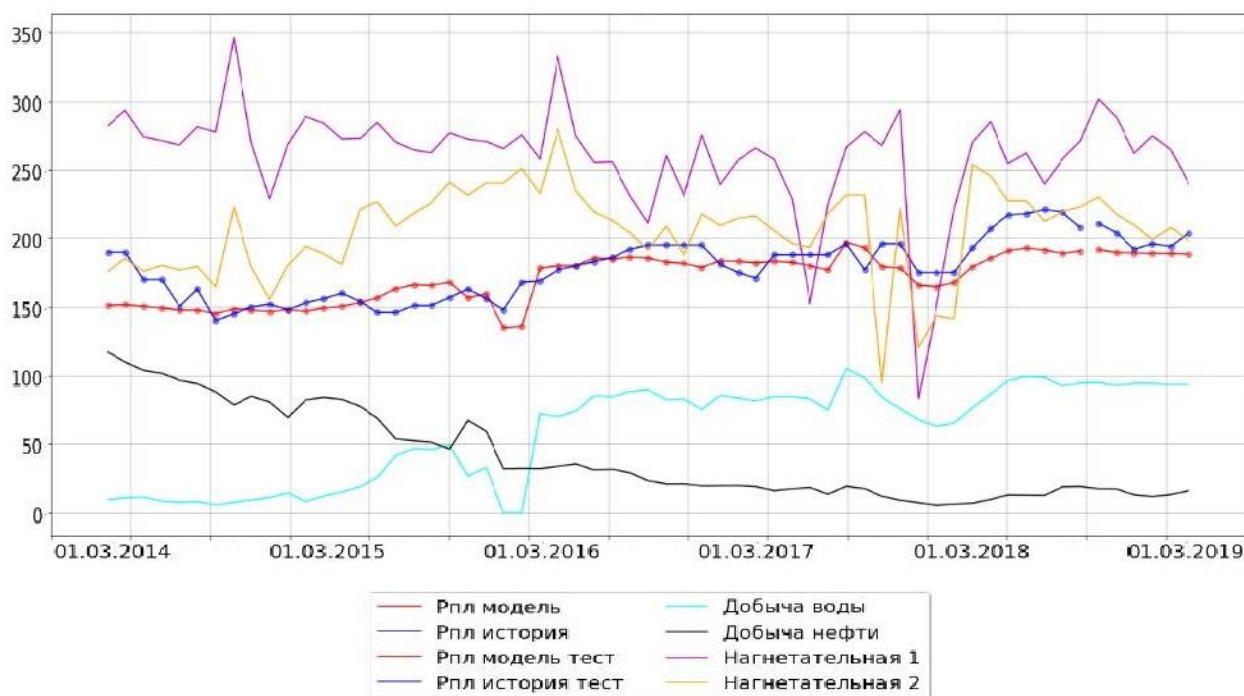


Рис. 2.3. Динамика исследуемых параметров для скважины № 2.

Видно, что модель «правильно» настроилась на обучающие данные и не зависит от «лишних» параметров, таких как закачка по соседним нагнетательным скважинам. Правильно воспроизведен общий тренд изменения пластового давления, например падение в районе 3 марта 2016 года. Также модель достаточно хорошо предсказала нужные нам параметры на тестовой выборке – последние 6 точек на синем и красном графиках (синие – исторические данные, красные – прогнозные данные модели).

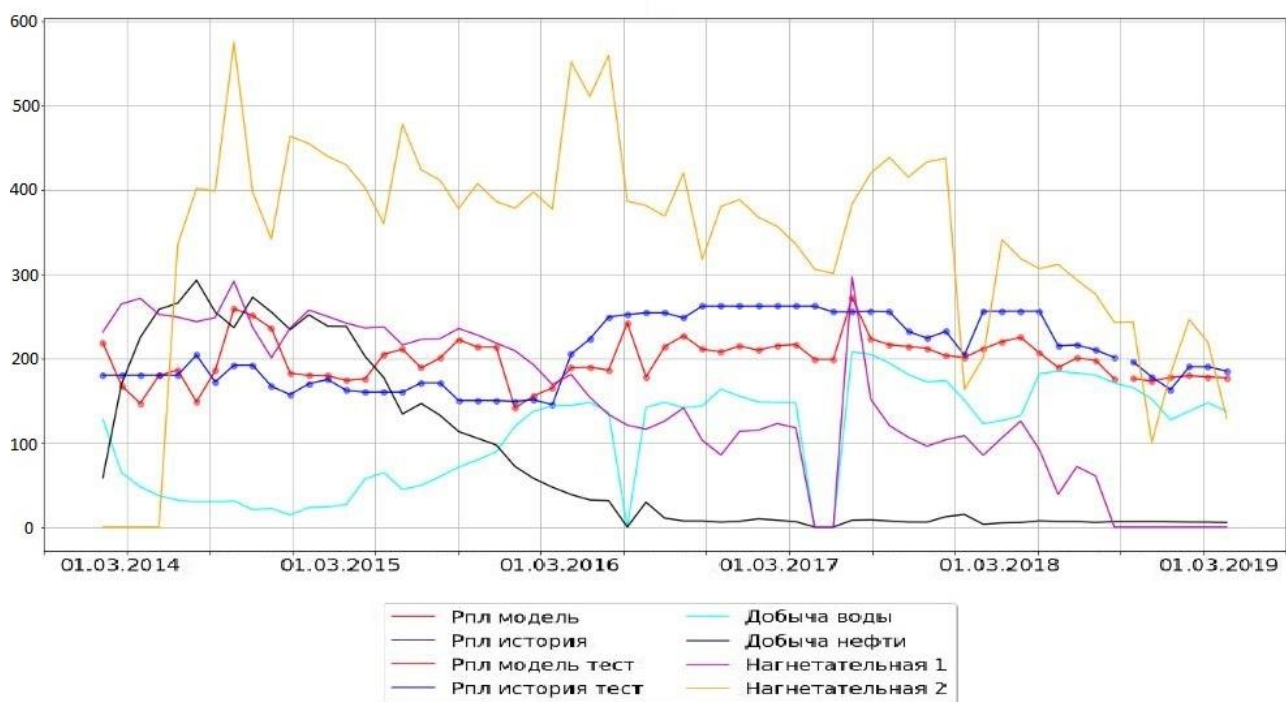


Рис. 2.4. Динамика исследуемых параметров для скважины № 5.

Аналогично, для скважины номер 5: видны участки подъема давления – участок конец 2014 - начало 2015 года. После марта 2016 года идет заметный подъем давления, что тоже согласуется с историческими данными.

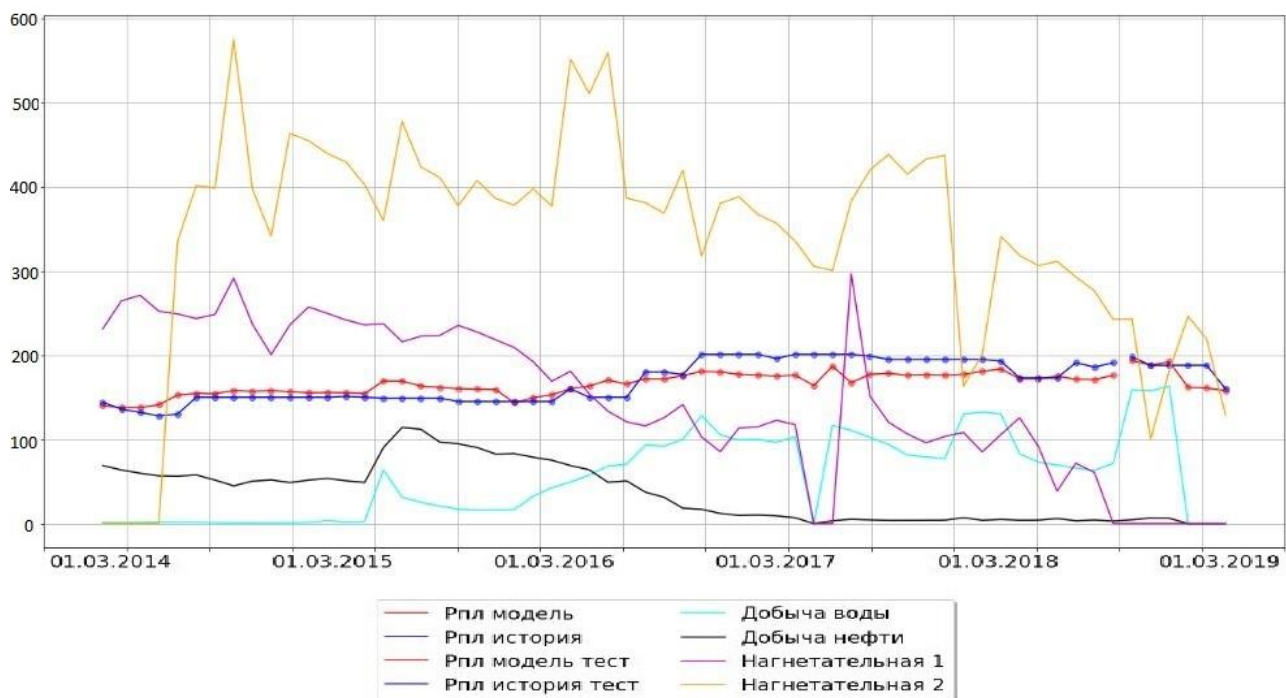


Рис. 2.5. Динамика исследуемых параметров для скважины № 11.

На рис. 2.5 представлены те же самые параметры для скважины номер 11. Здесь можно заметить некоторую корреляцию между количеством закачки с первой нагнетательной скважины и нашим пластовым давлением, при это на большом участке (с середины 2014 года по март 2016) историческое пластовое давление не менялось, что може говорить о двух вещах:

1. никакой связи между этой скважиной и закачкой с соседних нагнетательных скважин действительно нет, что не поддается объяснению,
2. возможная некорректная запись пластового давления простой интерполяцией в данном промежутке времени, по причине отсутствия прямых замеров (такое случается и происходит по вине инженеров по добыче на месторождении).

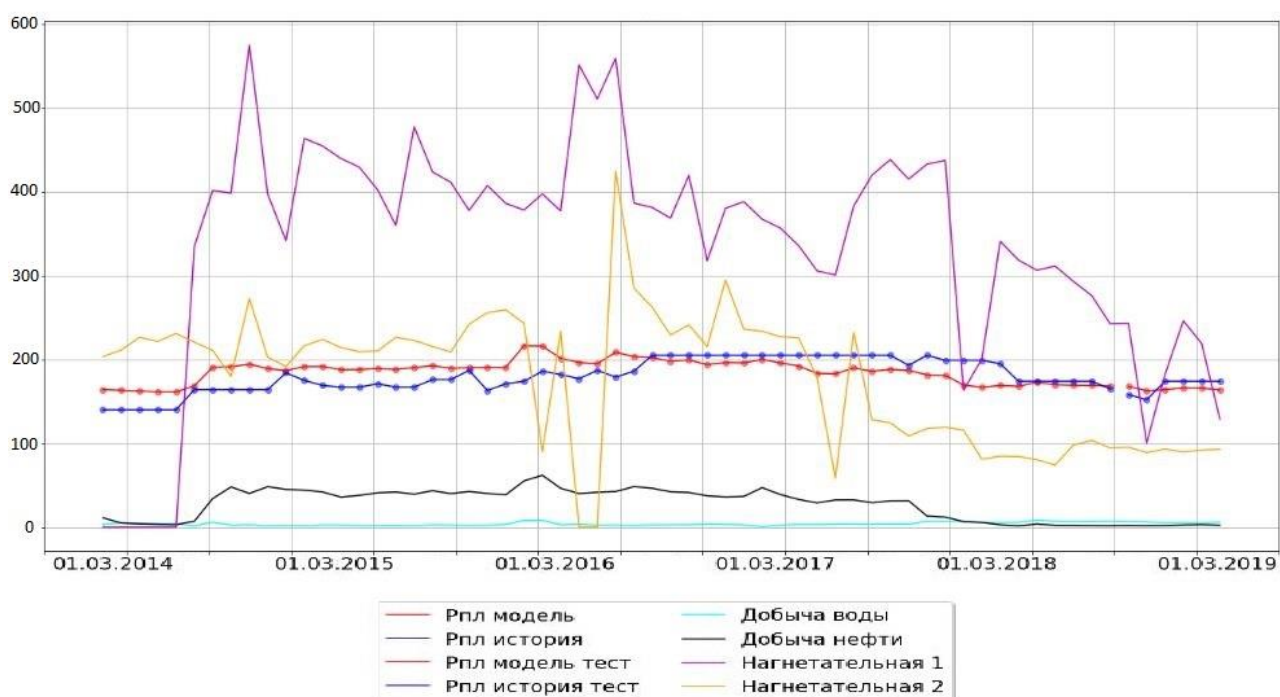


Рис. 2.6. Динамика исследуемых параметров для скважины № 4.

Можно провести аналогичные предыдущим рассуждения для скважины номер 4 (см. рис. 2.6) на интервале: середина 2016 года – конец 2017.

В общем, после проведенного анализа, можно заключить, что с определенной погрешностью, модель настроена верно.

Далее приведена таблица абсолютных отклонений для предсказаний модели на тестовой выборке (см. таблицу 2.3.).

Таблица 2.3.

Абсолютные значения отклонения на тестовой выборке за последний период.

	Октябрь 2018	Ноябрь 2018	Декабрь 2018	Январь 2019	Февраль 2019	Март 2019
1	17.34	20.43	7.14	-12.50	-27.39	11.57
2	19.26	14.33	2.62	6.86	5.04	15.58
3	16.46	15.45	16.75	12.71	12.61	20.82
4	-9.81	-10.72	10.00	7.85	8.21	10.38
5	19.90	4.91	-14.47	10.60	11.97	8.06
6	-23.89	-21.34	-8.22	-11.82	-21.36	10.51
7	11.47	-1.02	1.29	-13.94	4.83	14.45
8	-8.84	-12.12	3.07	-14.89	-17.44	12.88
9	15.55	12.66	9.24	9.77	13.18	14.20
10	7.88	10.03	10.54	6.53	2.45	16.46
11	4.43	-0.26	-4.18	16.10	27.03	2.13
12	0.36	2.08	28.31	10.94	-9.68	12.68
13	1.25	-15.71	-18.48	-6.42	-8.88	-12.21

Исходя из этой таблицы можно заключить, что в предсказании пластового давления даже на полгода вперед присутствуют значительные отклонения (до 30 атмосфер). Они выделены в таблице красным шрифтом. При этом относительные погрешности этих отклонений составляют не более 20-25%, что соответствует средней прогнозной способности современных методов машинного обучения.

ЗАКЛЮЧЕНИЕ

В ходе данной работы был создан алгоритм расчета оптимального пластового давления для операций глушения и дальнейшего ремонта скважины. Для этих целей были применены алгоритмы машинного обучения. Для реализации этих алгоритмов был выбран язык программирования Python. Как видно из графиков и сравнительной таблицы, исследованные методы машинного обучения не позволяют достичь желаемой точности при прогнозе пластового давления.

Поскольку сейчас инженером по поддержанию пластового давления значения пластового давления предсказываются с точностью в среднем до 10-15 атмосфер, то можно сделать вывод о том, что в данных условиях и при данных параметрах применение технологий машинного обучения нецелесообразно. Алгоритм выдает значения с недостаточной точностью, эта точность не превосходит точности, получаемой в результате анализа скважин «вручную».

Возможные причины: недостаточно данных – при обучении использовался датасет из 2000 элементов и всего 5 признаков. Возможно нужно увеличить количество входной информации. Также стоит рассмотреть вопрос о взаимовлиянии добывающих скважин друг на друга. Также стоит попробовать совместить аналитический и физический подход для решения данной задачи.

В будущем будут рассмотрены другие подходы для решения данной задачи (полуаналитический, 3D моделирование и прочее).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Анализ применимости алгоритмов машинного обучения для задач интерполяции и прогноза геологических свойств в межскважинном пространстве 24.12.2018. Источник: Журнал «ПРОнефть» Д.В. Егоров, Б.В. Белозеров.
2. Song Y., Zhou Y., Han R. Neural networks for stock price prediction //Journal of Difference Equation and Applications. – 2018. – May. – P. 1–14.
3. Дейк Л.П., Основы разработки нефтяных и газовых месторождений, Premium Engeneering, 2009, 570стр.
4. Автоматический анализ тональности рецензий с использованием библиотеки Tensorflow, Фанифатьева А.Д.
5. Геология нефти и газа: Учебник для вузов / Э. А. Бакиров, В. И. Ермолкин, В. И. Ларин и др. // Под ред. Э. А. Бакирова. – М.: Недра, 1990. – 240 с.
6. Инструкция по определению пластового давления при ремонтах и ГТМ на скважинах компании, внутренние документы компании SPD.
7. Т. Д. Голф-Рахт, Основы нефтепромысловой геологии и разработки трещиноватых коллекторов, пер. с англ. под ред. А. Г. Ковалева. — М.: Недра, 1986. — 608 с.
8. Sayarpour, M., Zuluaga, E., Kabir, C.S., and Lake, L.W. (2007). The Use of Capacitance-Resistive Models for Rapid Estimation of Waterflood Performance and Optimization. Paper SPE 110081 presented at the SPE Annual Technical Conference and Exhibition, Anaheim, California, 11-14 November.
9. Нейронные сети. Полный курс., С.хайкинг: пер. с англ. – М.: Вильямс, 2006. – 1104 с.

Код алгоритма.

```

import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
import matplotlib.ticker
import random

from sklearn.preprocessing import scale
from sklearn.utils import shuffle
from keras.models import Sequential
from keras.layers import Dense, Dropout, LeakyReLU, Activation
from keras.layers.recurrent import LSTM
from keras import regularizers
from keras.layers.normalization import BatchNormalization
from sklearn.preprocessing import MinMaxScaler
from keras import optimizers
from keras.layers.embeddings import Embedding
%matplotlib inline

skv = '23101'
df=pd.read_csv('DATASET_' + skv + '.csv',sep=';')
df = df.dropna()
yname="Pres"
kol = df.shape[1] - 2
df.head(10)

df0=df.drop(yname,axis=1)
df0.corrwith(df[yname])
df.corr()
sns.pairplot(df)

X = df.columns[1:]
y = df[yname]
print (df)
len1 = int(0.9*len(y))
trainX1 = df.drop(df[df.index > len1].index)
testX1 = df.drop(df[df.index <= len1].index)
trainX = trainX1[trainX1.columns[2:]]
testX = testX1[testX1.columns[2:]]
trainY = trainX1[yname]
testY = testX1[yname]
print (trainX1)
print (testX1)
print (trainX)
print (testX)
print (trainY)
print (testY)

model = Sequential()

```

```

model.add(Dense(32, input_dim=kol))
model.add(Dense(16))
model.add(Dense(4))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('linear'))
vesa = np.zeros(len(trainY))
vesa[0:len(trainY) - 12] = 1
vesa[len(trainY) - 12:len(trainY)] = 100
#sgd = optimizers.SGD(lr=0.05, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(optimizer='adam', loss='mse', metrics=['accuracy'])

h = model.fit(trainX, trainY, epochs = 30000, batch_size = 8, verbose = 1, sample_weight =
vesa, shuffle = True)

%pylab inline
scores = model.evaluate(trainX, trainY)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
print("\n%s: %.2f%%" % (model.metrics_names[0], scores[0]))
scores = model.evaluate(testX, testY)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
print("\n%s: %.2f%%" % (model.metrics_names[0], scores[0]))
loss = h.history['loss']
epochs = range(1, len(loss)+1)
plt.plot(epochs[0:], loss[0:], label='Training loss')
plt.legend()
plt.show()

print (model.predict(trainX))
print (trainY)
pylab.figure(figsize=(16, 6))
pylab.subplot(1,2,1)
pylab.grid(True)
pylab.scatter(trainY, model.predict(trainX), alpha=0.5, color = 'red')
pylab.title('Train set')
plt.xlabel('History data')
plt.ylabel('Model prediction')
min_l = min(trainY) - 10
max_l = max(trainY) + 10
plt.xlim(min_l, max_l)
plt.ylim(min_l, max_l)

print (model.predict(testX))
print (testY)
pylab.figure(figsize=(16, 6))
pylab.subplot(1,2,1)
pylab.grid(True)
pylab.scatter(testY, model.predict(testX), alpha=0.5, color = 'red')
pylab.title('Train set')
plt.xlabel('History data')
plt.ylabel('Model prediction')
min_l = min(testY) - 10

```

```

max_1 = max(testY) + 10
plt.xlim(min_1, max_1)
plt.ylim(min_1, max_1)

pylab.figure(figsize=(50, 10))
axes = pylab.subplot(1,2,1)
pylab.grid(True)
locator = matplotlib.ticker.LinearLocator (16)
axes.xaxis.set_major_locator (locator)
pylab.scatter(trainX1['Date'], model.predict(trainX), alpha=0.5, color = 'red', label = "")
pylab.scatter(trainX1['Date'], trainY, alpha=0.5, color = 'blue', label = "")
pylab.scatter(testX1['Date'], model.predict(testX), alpha=0.5, color = 'red', label = "")
pylab.scatter(testX1['Date'], testY, alpha=0.5, color = 'blue', label = "")
pylab.title(skv)
plt.plot(trainX1['Date'], model.predict(trainX), color = 'red', label = 'РПЛ модель')
plt.plot(trainX1['Date'], trainY, color = 'blue', label = 'РПЛ история')
plt.plot(testX1['Date'], model.predict(testX), color = 'red', label = 'РПЛ модель тест')
plt.plot(testX1['Date'], testY, color = 'blue', label = 'РПЛ история тест')
plt.plot(df['Date'], df['Water'], color = 'aqua', label = 'Добыча воды')
plt.plot(df['Date'], df['Oil'], color = 'black', label = 'Добыча нефти')
plt.plot(df['Date'], df[df.columns[5]], color = 'm', label = 'Закачка ' + df.columns[5])
plt.plot(df['Date'], df[df.columns[6]], color = 'orange', label = 'Закачка ' + df.columns[6])
pylab.legend ()
axes.legend(loc='upper center', bbox_to_anchor=(0.5, -0.1), shadow=True, ncol=2)

b = np.array(model.predict(testX))
b = b.ravel()
print(b)
a = np.array(testY)
print(a)
print(a - b)

```

Динамика исследуемых параметров по скважинам.

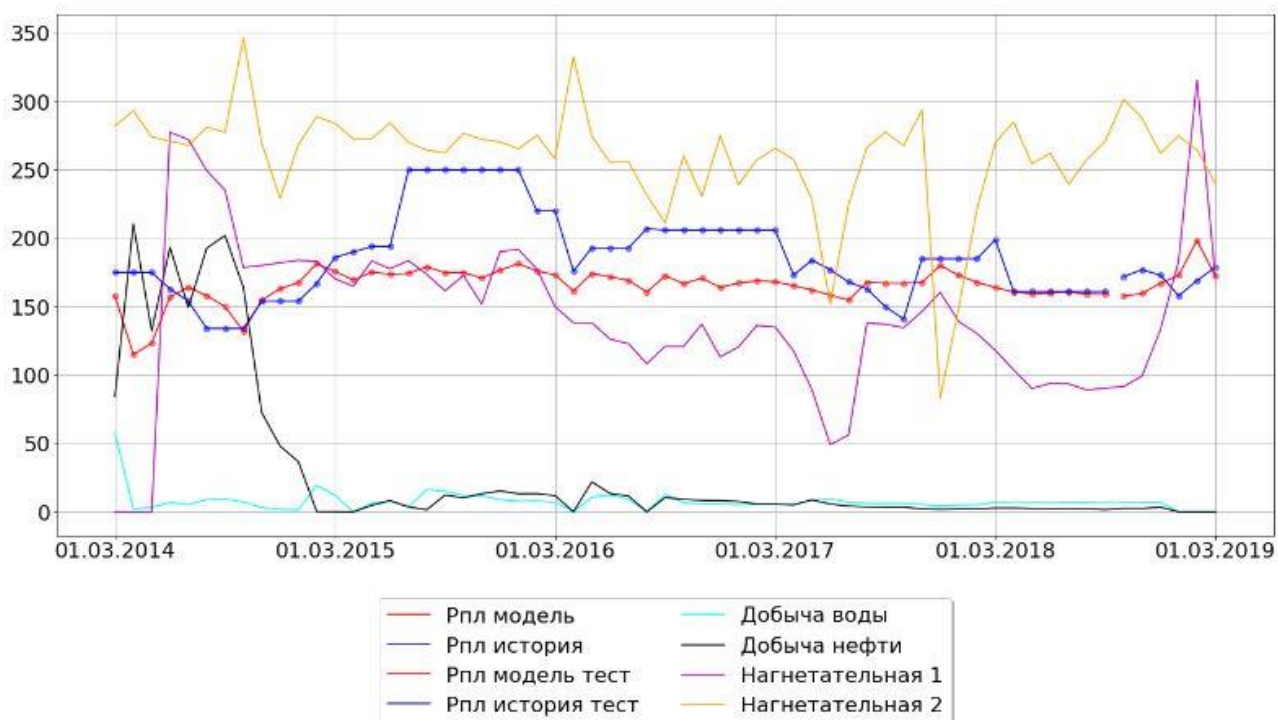


Рис. 1. Динамика исследуемых параметров для скважины № 1.

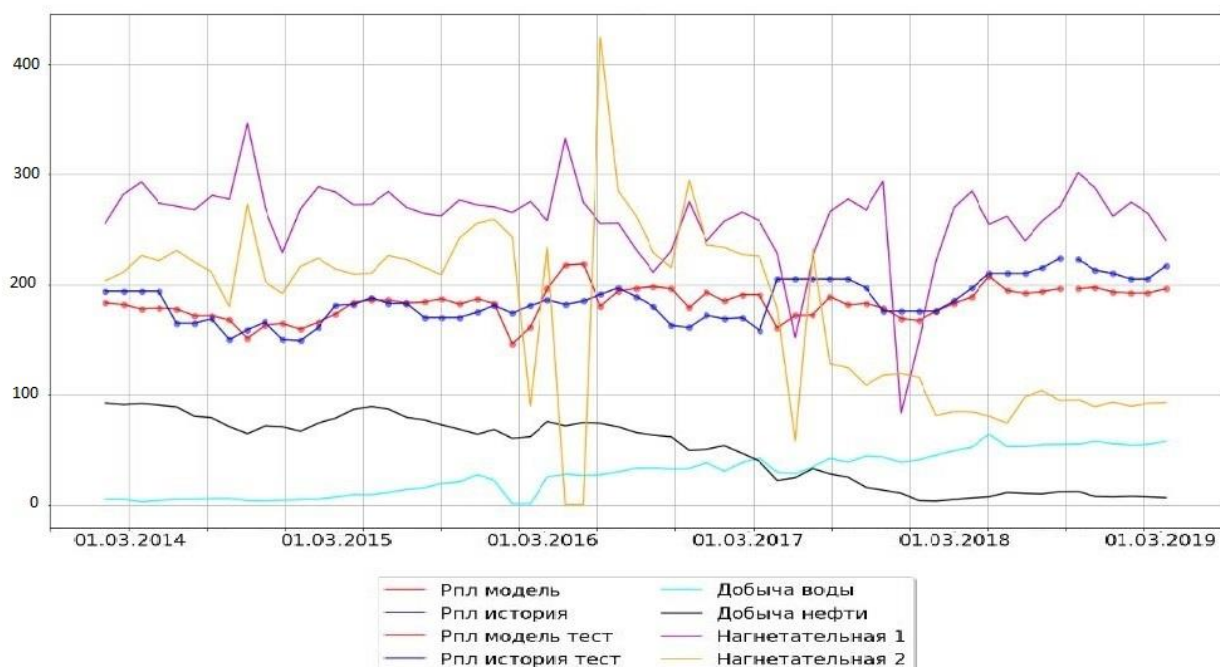


Рис. 2. Динамика исследуемых параметров для скважины № 3.

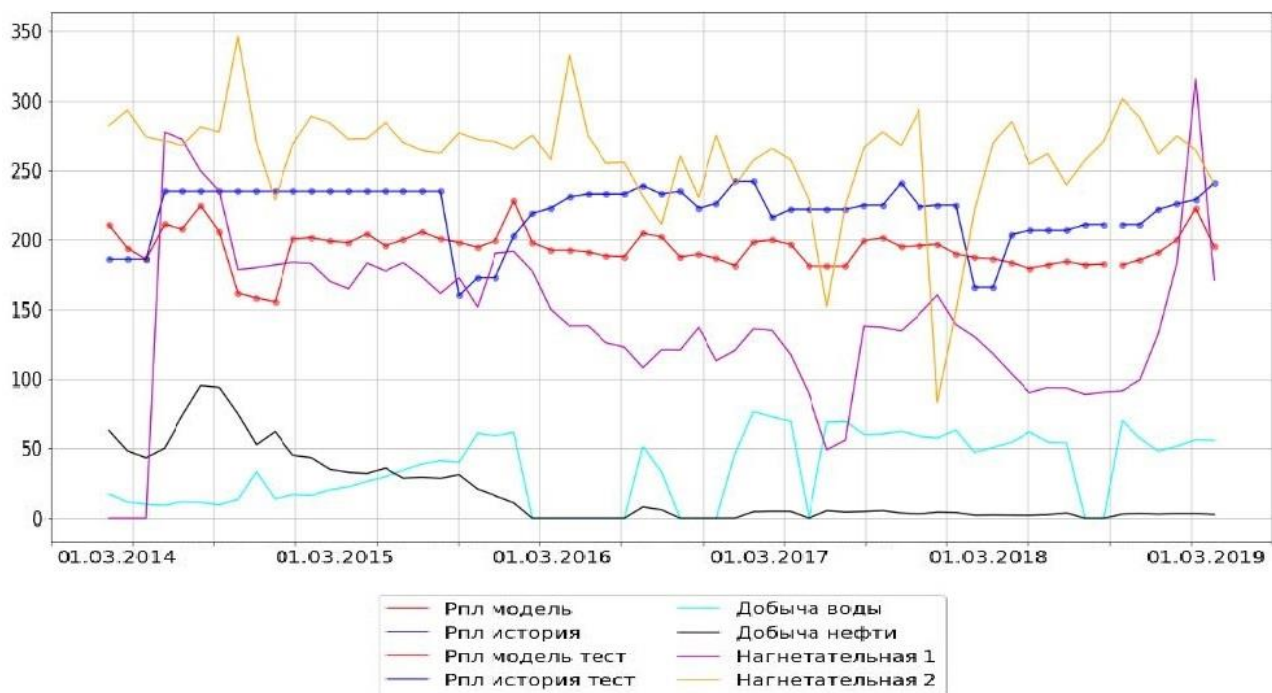


Рис. 3. Динамика исследуемых параметров для скважины № 6.

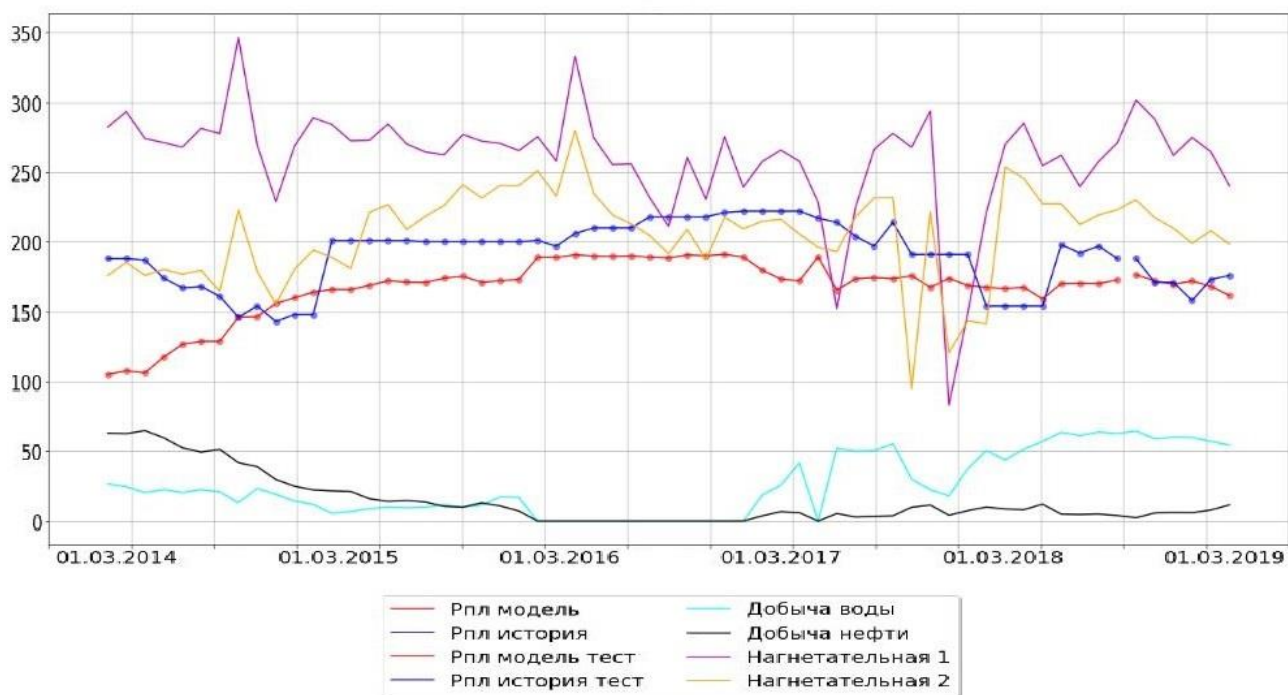


Рис. 4. Динамика исследуемых параметров для скважины № 7.

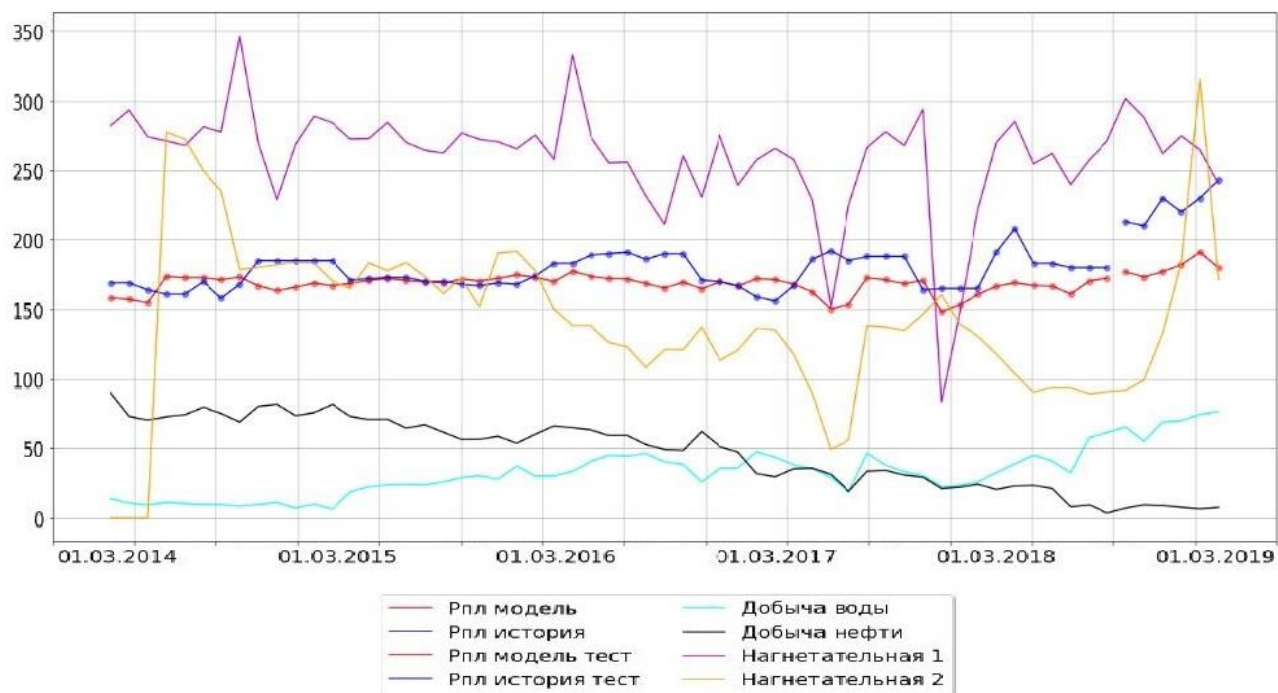


Рис. 5. Динамика исследуемых параметров для скважины № 8.

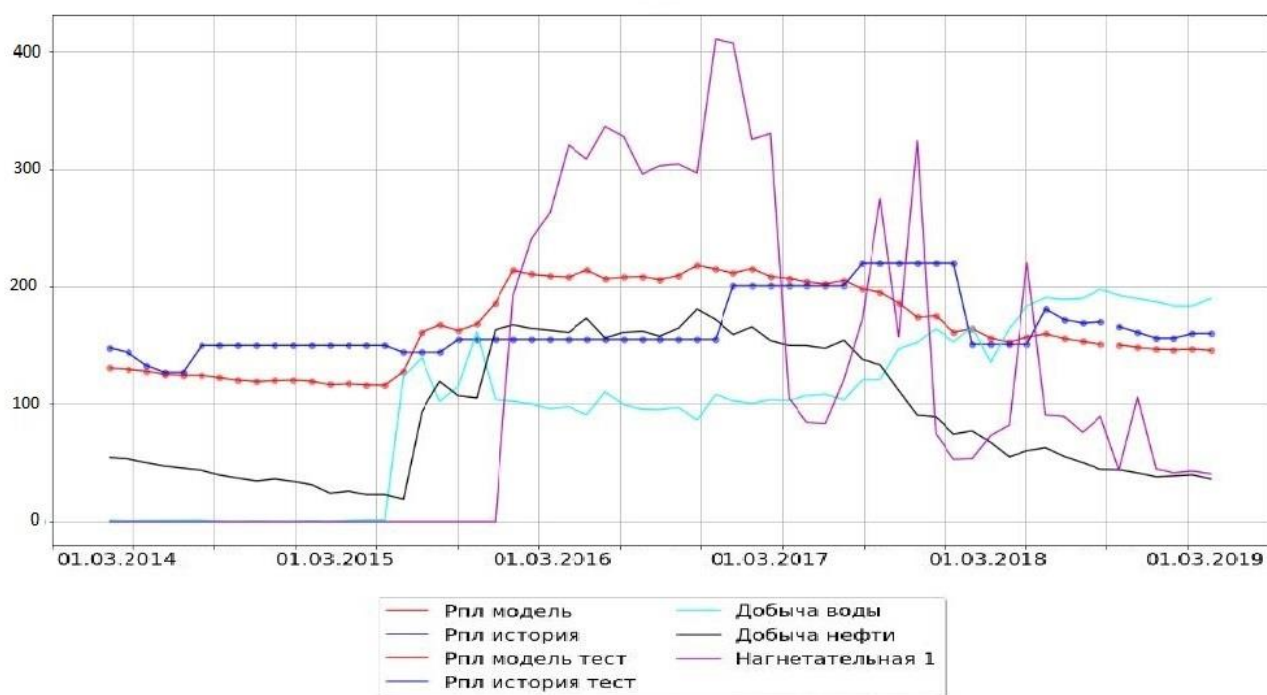


Рис. 6. Динамика исследуемых параметров для скважины № 9.



Рис. 7. Динамика исследуемых параметров для скважины № 10.

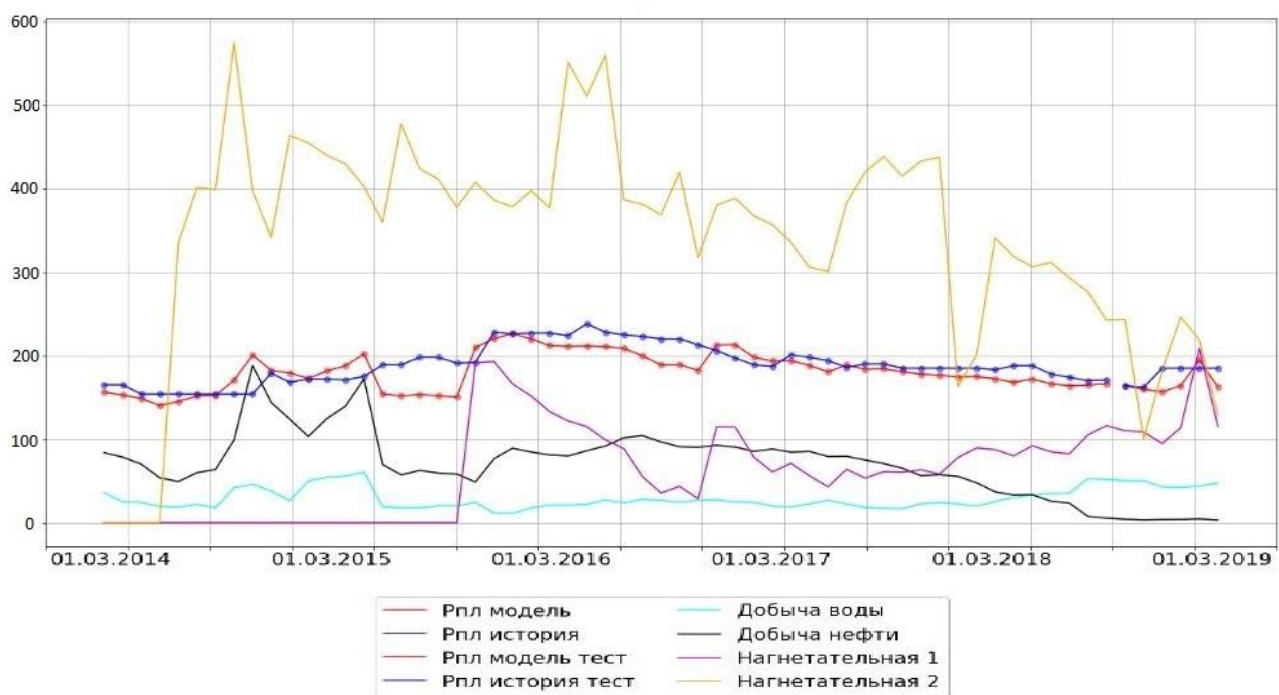


Рис. 8. Динамика исследуемых параметров для скважины № 12.

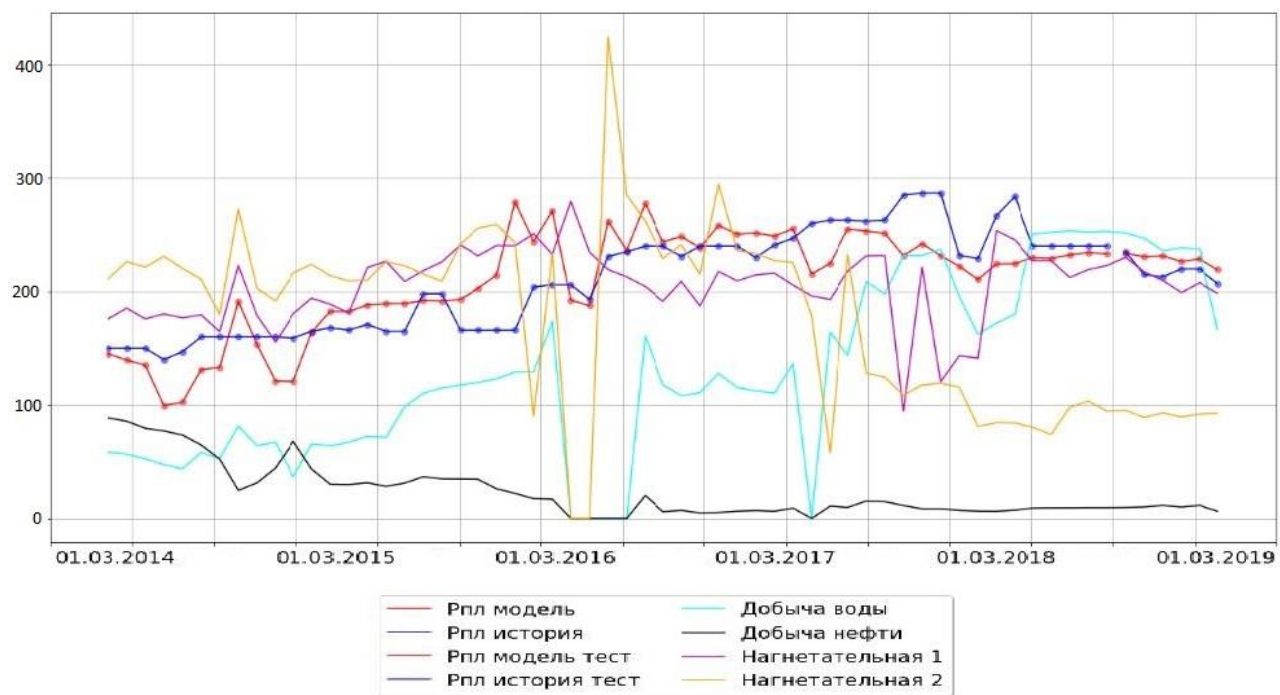


Рис. 9. Динамика исследуемых параметров для скважины № 13.