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

Работа допущена к защите
Зав.кафедрой, д.ф-м.н., чл.-корр.РАН
_____А.М. Кривцов
«___»_____20___г

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

“ПРИМЕНЕНИЕ АЛГОРИТМОВ ОБРАБОТКИ И АНАЛИЗА ГЕОЛОГИЧЕСКИХ ДАННЫХ В ЗАДАЧАХ ПРОГНОЗИРОВАНИЯ КАЧЕСТВА РАЗРАБОТКИ”

по направлению 01.04.03 “Механика и математическое моделирование”
01.04.03_04 “Математическое моделирование процессов нефтегазодобычи”

Выполнил
студент гр.23642/3 А.И.Севостьянов

Руководитель:
Доцент, к.ф-м О.С.Лобода

Консультант
начальник отдела Д.В.Солодов

Санкт-Петербург
2018

Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и механики

УТВЕРЖДАЮ

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

_____А.М. Кривцов

«_____»_____2018 г.

ЗАДАНИЕ

по выполнению выпускной квалификационной работы

студенту Севостьянову Андрею Игоревичу

фамилия, имя, отчество (при наличии), номер группы

1. Тема работы: ПРИМЕНЕНИЕ АЛГОРИТМОВ ОБРАБОТКИ И АНАЛИЗА ГЕОЛОГИЧЕСКИХ ДАННЫХ В ЗАДАЧАХ ПРОГНОЗИРОВАНИЯ КАЧЕСТВА РАЗРАБОТКИ

2. Срок сдачи студентом законченной работы: 12.06.18

3. Исходные данные по работе: Результаты исследований по скважинам Западной Сибири на целевой пласт

4. Содержание работы (перечень подлежащих разработке вопросов):

Описание существующих алгоритмов обработки и анализа данных, анализ исходных данных, обоснование выбора модели обучения, настройка модели

5. Перечень графического материала (с указанием обязательных чертежей):

29 рисунков, 1 таблица

6. Консультанты по работе: Начальник отдела сопровождения проекта «Большая Ачимовка” Солодов Данил Вячеславович

7. Дата выдачи задания _____

Руководитель ВКР _____ _____
(подпись) инициалы, фамилия

Задание принял к исполнению _____
(дата)

Студент _____ _____
(подпись) инициалы, фамилия

РЕФЕРАТ

На 42 с., 29 рисунков, 1 таблица, 1 приложение.

ГЕОЛОГИЯ, НЕФТЕДОБЫЧА, МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ, МАШИННОЕ ОБУЧЕНИЕ, НЕЙРОННЫЕ СЕТИ, АНСАМБЛИ АЛГОРИТМОВ

Данная работа посвящена обзору и разработке алгоритмов машинного обучения применительно к задачам нефтедобычи. В работе приводится краткий экскурс в существующие алгоритмы и модели обработки больших объемов данных. Описывается обоснование выбора подходящей в условиях поставленной задачи модели и производится ее настройка на пред обработанных входных данных.

THE ABSTRACT

42 pages, 29 pictures, 1 tables, 1 application.

GEOLOGY, OIL PRODUCTION, MATHEMATICAL MODELING, MACHINE LEARNING, NEURAL NETWORKS, ENSEMBLING

This work is devoted to review existing machine learning algorithms and their application in oil industry problems. The present work considers existing algorithms and big data models developed at the moment. The optimal algorithm was chosen, and the model was configured with respect to input preprocessed balanced dataset.

Оглавление

Введение

Повсеместная цифровизация приводит к накоплению огромных объемов данных в науке, производстве, бизнесе, транспорте, здравоохранении. Ранее задачи, связанные с обработкой этой информации либо не ставились вообще, либо решались совершенно другими методами.

С развитием информационных технологий и появлением почти неограниченных вычислительных ресурсов многие крупнейшие мировые стейкхолдеры нефтяной индустрии все чаще и чаще стали использовать при решении задач геологоразведки инструменты машинного обучения.

Цель данной работы – освящение основных инструментов машинного обучения, а также применение их в задаче построения карты перспективности разработки Ачимовской толщи Западной Сибири.

1 Объект исследования

Объект исследования – Ачимовская толща – глубоководные отложения, характеризующиеся низкими фильтрационно-емкостными свойствами, аномально-высоким пластовым давлением и отсутствием традиционных связей геологических параметров с результатами промышленной разработки (Рис 1). По этой причине, используя существующие подходы, составить наиболее реальную карту перспективности планируемой к разработке территории с достаточной для проведения ОПР точностью не представляется возможным.

Для решения этой задачи предлагается использовать модели машинного обучения, которые позволяют находить сложные, нелинейные зависимости между геологическими параметрами пласта и перспективностью разработки.

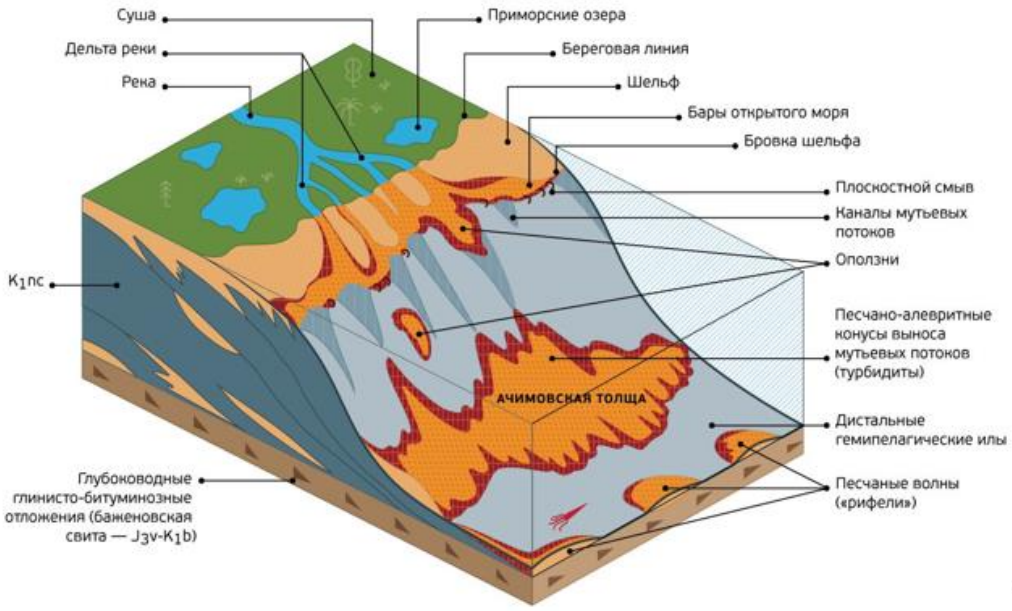


Рис 1. Схема формирования неокомских отложений

2 Теоретический экскурс

Машинное обучение (далее МО) находится на стыке математической статистики, методов оптимизации и классических математических дисциплин. Тем не менее это не только теоретическая, но и практическая, инженерная дисциплина. Теоретическая подоплека большинства методов подразумевает допущения в виде идеально выстроенных данных, что на практике почти никогда не встречается. Чтобы заставить их работать с приемлимой точностью, зачастую вводятся дополнительные эвристики, компенсирующие несоответствие сделанных в теории предположений условиям реальных задач.

2.1 Основные стандартные типы задач

Глобально, общая постановка задачи МО делится на 2 типа: обучение с учителем (supervised) и обучение без учителя (non supervised).

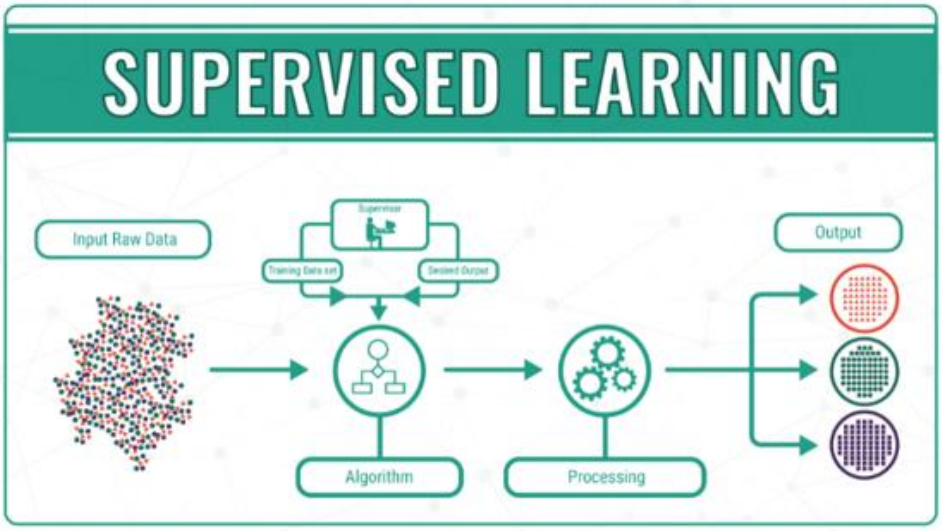


Рис 2.1 Схема обучения с учителем

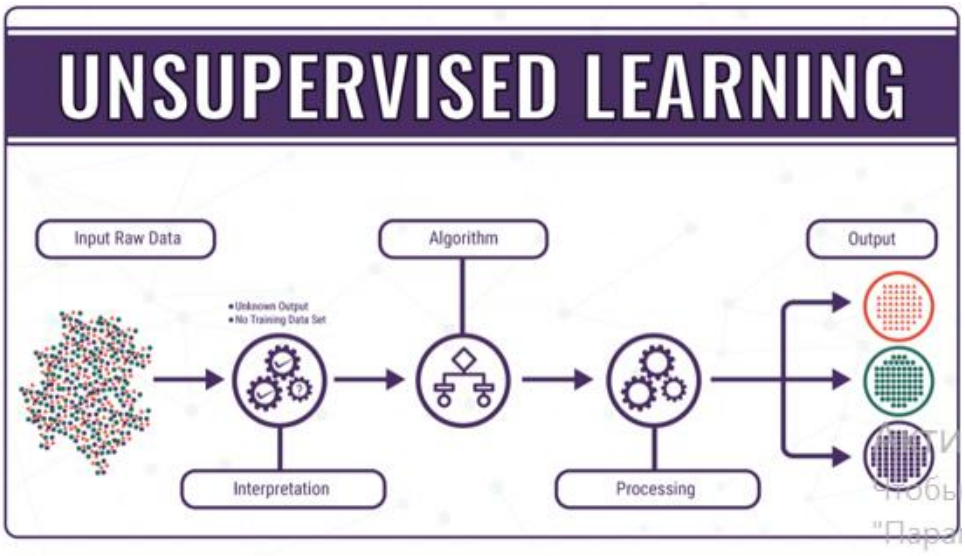


Рис 2.2 Схема обучения без учителя

В первом случае решаются задачи: классификации, регрессии, ранжирования, прогнозирования. Во втором: задачи кластеризации, заполнения пропущенных значений, сокращения размерности итд. Задачи обучение с учителем предполагают существование некой метки класса (классификация) либо значения целевой функции (регрессия). В качестве этого значения в задаче приоритезации территории Западной Сибири будем использовать запускной дебет нефти промышленной разработки месторождения, приведенный к общим условиям по диаметру штуцера, методу заканчивания и условиям работы.

2.2 Модели и методы их обучения

Часто для обучения модели пользуются принципом минимизации эмпирического риска. Риском гипотезы h называют ожидаемое значение функции стоимости L. В качестве меры эмпирического риска модели можно брать среднее значение функции стоимости L для всех n

$$Q_{emp}(h) = \frac{1}{n}\sum_{1}^{n} L(h((x)_i, y_i) \quad (1)$$

Тогда, согласно принципу минимизации эмпирического риска, мы должны выбрать такую гипотезу h∈H, которая минимизирует Q_{emp}

$$\hat{h} = \underset{h\in \mathrm{H}}{\operatorname{argmin}}\, Q_{emp}(h) \quad (2)$$

У данного принципа есть существенный недостаток, решения найденные таким путем будут склонны к *переобучению*. Мы говорим, что модель обладает *обобщающей способностью*, тогда, когда ошибка на новом (тестовом) наборе данных (взятом из того же распределения P(x,y) мала, или же предсказуема. Переобученная модель не обладает обобщающей способностью, т.е. на обучающем наборе данных ошибка мала, а на тестовом наборе данных ошибка существенно больше.

Инструментов МО и их модификаций большое множество. Некоторые из них применимы почти в любой задаче, но есть и такие, которые имеют довольно узкий спектр применения. За одними стоит довольно очевидный математический аппарат, а другие требуют глубокого понимания теории математического анализа и мат.статистики. Рассмотрим наиболее популярные из них по увеличению сложности

2.2.1 Линейная регрессия

Наиболее примитивный метод регрессионного анализа, характеризующийся всего одной степенью свободы, поэтому крайне редко использующийся на практике. Тем не

менее, метод имеет свои плюсы: простая реализация в любом программном продукте, низкая ресурс-востребованность.

Ограничив пространство гипотез h только линейными функциями:

$$\forall h \in \mathrm{H}, (\vec{x}) = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_m x_m = \quad (3)$$

$$= \sum_{i=0}^{m} w_i x_i = \vec{x}^T \vec{w}$$

Тогда эмпирический риск или функция стоимости, следуя формуле 1 окажется равной:

$$L(X, \vec{y}, \vec{w}) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - \vec{x}_i^T \vec{w_i})^2 = \frac{1}{2n} \|\vec{y} - X\vec{w}\|_2^2 = \frac{1}{2n} (\vec{y} - X\vec{w})^T (\vec{y} - X\vec{w}) \quad (4)$$

В качестве алгоритмы обучения подобных простых моделей используется метод наименьших квадратов (далее МНК). Приравняем производную от функции стоимости к нулю и найдем решение в явном виде:

$$\frac{\partial L}{\partial \vec{w}} = \frac{\partial}{\partial \vec{w}} \frac{1}{2n} (\vec{y}^T \vec{y} - 2\vec{y}^T X\vec{w} + \vec{w}^T X^T X\vec{w}) = \frac{1}{2n} (-2X^T \vec{y} + 2X^T X\vec{w}) \quad (5)$$

$$\frac{\partial L}{\partial \vec{w}} = 0 \Leftrightarrow \frac{1}{2n} (-2X^T \vec{y} + 2X^T X\vec{w}) = 0$$

$$\Leftrightarrow -X^T \vec{y} + X^T X\vec{w} \quad (5)$$

$$\Leftrightarrow X^T \vec{y} = X^T X\vec{w}$$

$$\Leftrightarrow \vec{w} = (X^T X)^{-1} X^T \vec{y}$$

Программная реализация описанной выше модели аппроксимирует точки функции синуса с шумом следующим образом:

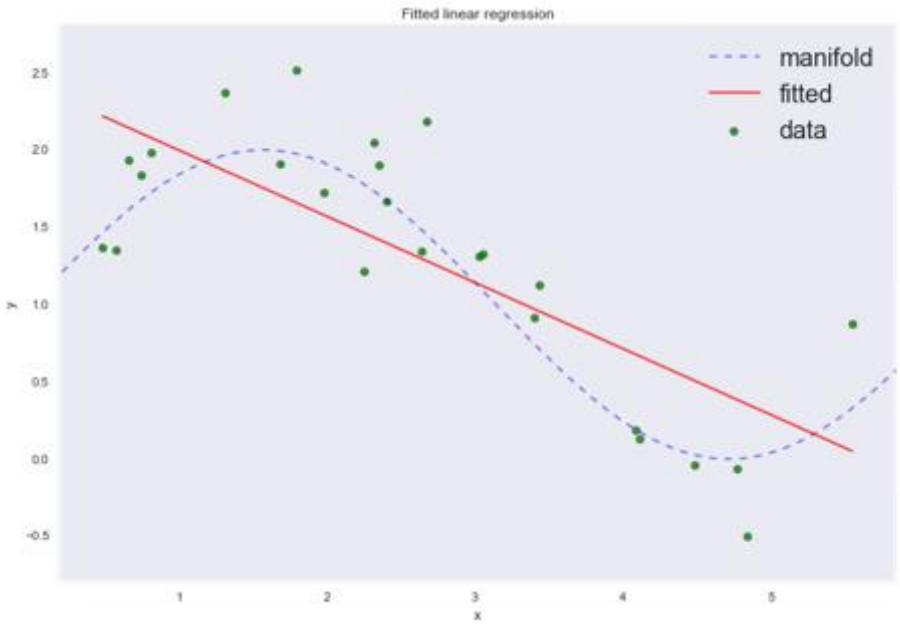


Рис 2.2.1 Аппроксимация точек линейной функцией

Среднеквадратичная ошибка приблизительно равна 0.3. Очевидно, что если бы вместо линии мы использовали кривую третьего порядка, то результат был бы куда лучше.

2.2.1.2 Полиномиальная регрессия

Если расширить пространство гипотез до полиномов степени n, получим:

$$\forall h \in \mathrm{H}, (\vec{x}) = w_0 + w_1 x_1 + w_2 x^2 + \cdots + w_n x^p = \sum_{i=0}^{p} w_i x^i \qquad (6)$$

Рассмотрим аппроксимацию вышеупомянутого кейса полиномами разных степеней:

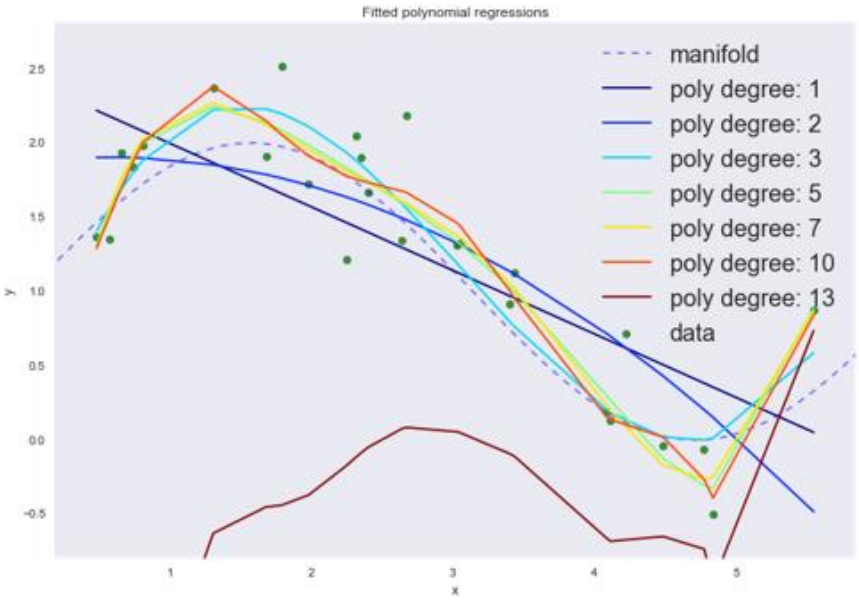


Рис 2.2.1.2 Аппроксимация полиномами разных степеней

Степень полинома	Error
1	0.26704
2	0.22495
3	0.08217
5	0.05862
7	0.05749
10	0.0532
13	5.76155

Табл 2.2.1.2 Зависимость ошибки от степени полинома

Как можно увидеть, при n>7, кривая стремится в точности повторять точки из тренеровочного набора, т.е переобучается, теряя при этом обобщающую способность. Это один из недостатков метода оптимизации МНК.

2.2.2 Решающие деревья

Решающие деревья реализуют логику иерархически организованной системы вопросов. Подобные модели издавна в том или ином виде используются в ботанике, зоологии, медицине, принятии решений и.т.д. Дерево состоит из корневой вершины (root edge), внутренних вершин, и листьев. Каждая внутренняя вершина инцидентна двум (бинарное дерево) или более выходящим рёбрам. Процесс обучения заканчивается при достижении концевой вершины (листа). Ключевой недостаток данного инструмента – высокая вероятность переобучения.

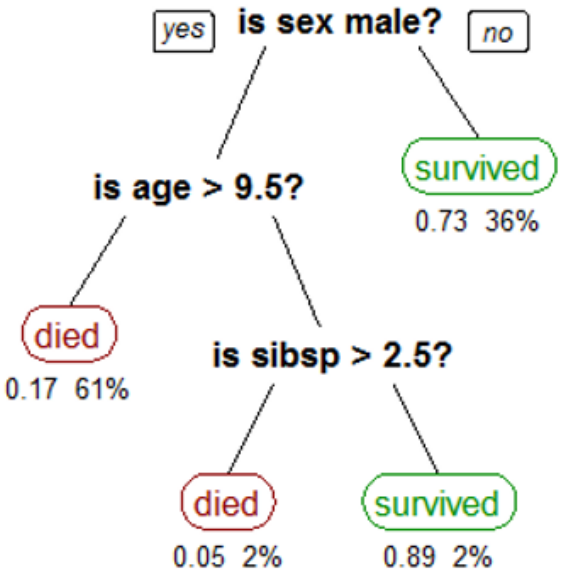


Рис 2.2.2 Пример простейшего решающего дерева

Возможен случай, когда на тренировочной выборке обучающий набор будет разделен таким образом, что в каждый класс попадет по одному примеру. Таким образом окажется что точность алгоритма достигает 100%. Для исключения переобучения вводят искусственные критерии остановки построения дерева до того, как будет достигнута полная однородность концевых вершин:

1. Ограничение на минимум объектов в выборках, соответствующих концевым вершинам
2. Точность на контрольной выборке
3. Критерии однородности (критерий Джини, энтропийный индекс неоднородности, индекс ошибочной классификации)

2.2.3 Случайный лес

Как упоминалось во введении, в ходе решения задач на реальных данных возникает потребность в модификациях алгоритмов. Так, решающие деревья обладают крайне негативной особенностью – склонностью к быстрому переобучению. Эту проблему была решена Лео Брейманом, который предложить использовать так называемые ансамбли или композиции моделей, в частности деревьев решений. Прогноз получается в результате агрегирования ответов множества деревьев. Тренировка деревьев происходит на независимых друг от друга выборках. Для бэггинга (независимого обучения алгоритмов классификации, где результат определяется голосованием) есть смысл использовать большое количество деревьев решений с достаточно большой глубиной.

Во время классификации финальным результатом будет тот класс, за который проголосовало большинство деревьев, при условии, что одно дерево обладает одним голосом. Для задач регрессии финальный результат усредняется по всем деревьям.

Стоит заметить, что при увеличении количества моделей в ансамбле время обучения приблизительно линейно их количеству. Поэтому, если раньше для нас метрикой успешности алгоритмы была только его точность, то теперь необходимо учитывать и его требования к ресурсам

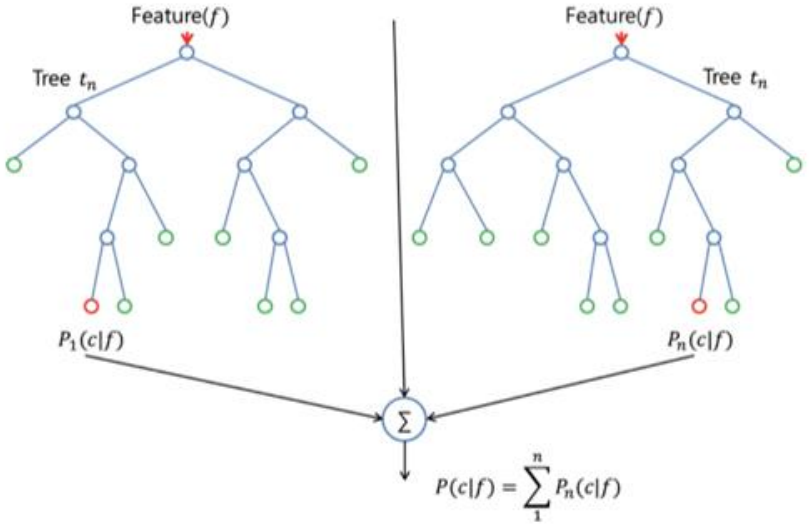


Рис 2.2.3 Пример простейшего алгоритма случайного леса.

2.2.4 Бустинг алгоритмов

Исторически считалось, что в машинном обучении обобщающая способность напрямую связана с интуитивной «сложностью» алгоритма. Действительно, всегда можно написать сложный алгоритм, абсолютно точно подстраивающийся под обучающую выборку, но имеющий нулевую обобщающую способность. Научное сообщество интересовал вопрос, о том, можно ли из большого количества слабых и простых моделей

получить одну сильную. Под слабыми моделями подразумеваются произвольный алгоритмы МО, точность которых немногим выше 50%.

Ответ на этот вопрос был впервые опубликован профессурой Стэнфордской кафедры статистики, уже принесшей миру Lasso, Elastic Net, Random Forest и.т.д.

2.2.4.1 Постановка задачи машинного обучения

Имеем набор пар признаков x и целевых переменных y, $\{(x\_i, y\_i)\}_{i=1,\dots,n}$,на котором мы будем восстанавливать зависимость вида y=f(x). Восстанавливать будем приближением $\hat{f}(x)$, с функцией потерь L (y, f), которую мы будем минимизировать:

$$y \approx \hat{f}(x)$$

$$\hat{f}(x) = \underset{f(x)}{\operatorname{argmin}} L(y, f(x)) \qquad (7)$$

Перепишем последнее равенство в терминах матожидания:

$$\hat{f}(x) = \underset{f(x)}{\operatorname{argmin}} E_{x,y}[L(y, f(x))] \qquad (8)$$

Ограничим пространство функций f(x) конкретным параметризованным семейством f (x, θ), θ ∈ ℝ. Тогда задача оптимизации параметров сводится к:

$$\hat{f}(x) = f(x, \hat{\theta})$$

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} E_{x,y}[L(y, f(x))] \qquad (9)$$

Выпишем эмпирическую функцию потерь:

$$\hat{\theta} = \sum_{i=1}^{M} \hat{\theta}_i$$

$$L_\theta(\hat{\theta}) = \sum_{i=1}^{N} L(y_i, f(x_i, \hat{\theta}) \qquad (10)$$

Используя итеративный алгоритм градиентного спуска минимизируем L. Будем добавлять итеративные оценки $\hat{\theta}_i$ вдоль градиента $L_\theta(\hat{\theta})$. Алгоритм поиска оптимального приближения, следующий:

- Расчет градиента функции потерь:

$$\nabla L_\theta(\hat{\theta}) = [\frac{\partial L(y, f(x, \theta))}{\partial \theta}]_{\theta = \hat{\theta}} \qquad (11)$$

- Задаем текущее приближение $\hat{\theta}_t$ на основе посчитанного градиента

$$\hat{\theta}_t \leftarrow -\nabla L_\theta(\hat{\theta})$$

- Обновляем приближение параметров $\hat{\theta}$:

$$\hat{\theta} \leftarrow \hat{\theta} + \hat{\theta}_t = \sum_{i=0}^{t} \hat{\theta}_i$$

- Сохраняем итоговое приближение $\hat{\theta}$:

$$\hat{\theta} = \sum_{i=0}^{M} \hat{\theta}_i$$

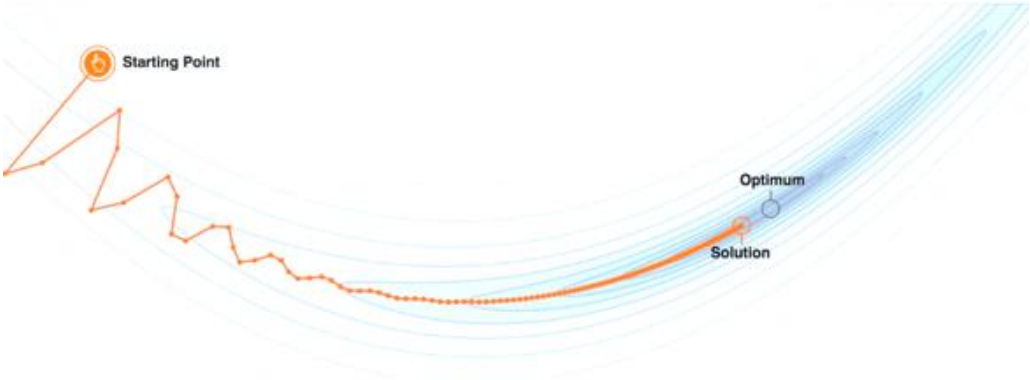


Рис 2.2.4.1 Визуализация поиска оптимального решения

Попробуем искать приближение $\hat{f}(x)$ не в виде аргумента минимизирующего функционал ошибки одной большой модели, а в виде самих функций:

$$\hat{f}(x) = \sum_{i=0}^{M} f_i(x)$$

Ограничим $\hat{f}(x)$ семейством функций $\hat{f}(x) = h(x, \theta)$. Теперь на каждом шаге для функций нам понадобится подбирать оптимальный коэффициент $\rho \in \mathbb{R}$. На шаге t задача сводится к:

$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x)$$

$$\left(\rho_t, \hat{\theta}_t\right) = argmin\ \mathbb{E}_{x,y}[L(y, \hat{f}(x) + \rho \cdot h(x, \theta))]$$

$$\hat{f}_t(x) = \rho_t \cdot h(x, \theta_t)$$

Мы принимали в этом теоретическом экскурсе что любую модель $h(x, \theta)$ можно обучить с помощью любой функции потерь $L(y_i, f(x_i, \hat{\theta})$, что на практике далеко не всегда оказывается возможным. Поэтому для того, чтобы свести задачу к чему-то решаемому принимается, что мы будем обучать модели так, чтобы наши предсказания были наиболее скоррелированными с градиентом функции потерь:

$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x)$$

$$r_{it} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}(x)}, \text{for i} = 1, \dots, \text{n},$$

$$\theta_t = \ = \arg\min_{\theta} \sum_{i=1}^{n} (r_{it} - h(x_i, \theta))^2,$$

$$\rho_t = \arg\min_{\rho} \sum_{i=1}^{n} L(y_i, \hat{f}(x_i) + \rho \cdot h(x_i, \theta_t))$$

2.2.4.2 Алгоритм GBM Friedman'a

На вход алгоритма подаются следующие данные:

- Набор данных {(x,y)}
- Чисто итераций M
- Выбор функции потерь L(y,f)
- Выбор базового алгоритма h(x,θ) и процедура их обучения
- Гиперпараметры h(x θ)

2.2.4.3 Пример работы алгоритма

Чтобы описать, какие действия происходят в процессе обучения, давайте сгенерируем зашумленную функцию косинуса:

$$y = cos(x) + \epsilon, \epsilon \sim \mathcal{N}(0, \frac{1}{5}), x \in [-5,5]$$

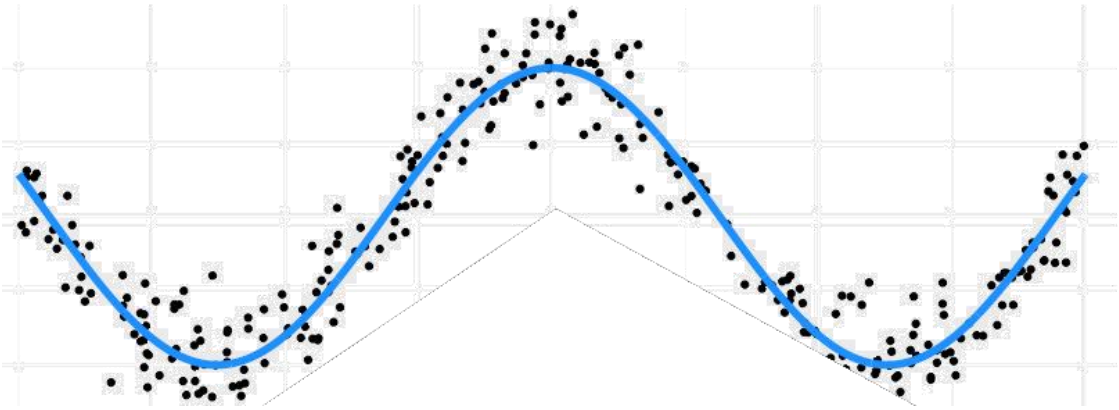


Рис 2.2.4.3 Функция cos (синяя кривая) и шум относительно кривой (черные точки)

Зададим 300 тестовых точек со случайным шумом. Проведем 3 итерации обучения используя деревья решений глубины 2 и среднеквадратичную функцию потерь.

$$L(y, f) = (y - f)^2$$

Для простоты будем инициализировать GBM мы будем средним значением:

$$\gamma = \frac{1}{n} \cdot \sum_{i=1}^{n} y_i$$

а все ρ_t равными 1.

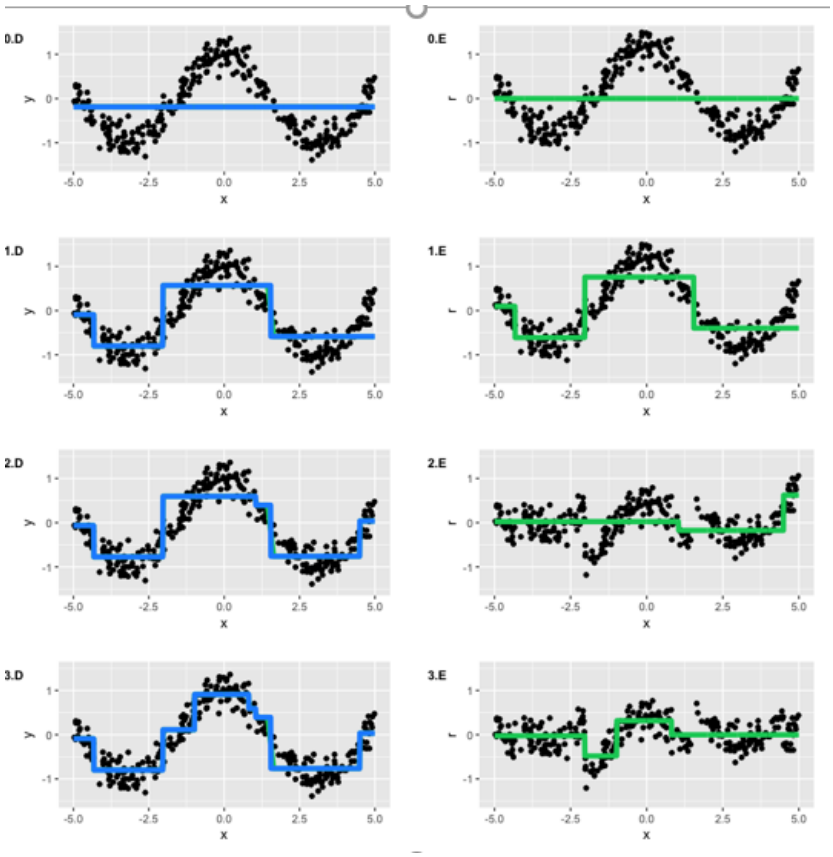


Рис. 2.2.4.3.1 - Синий график – актуальное приближение $\hat{f}(x)$, зеленый график – псевдо-остатки $\hat{f}_t\,(x)$

Как можно увидеть к четвертой итерации функция приближения достаточно хорошо аппроксимирует наши точки, при этом от шага к шагу функция псевдо-потерь уменьшается.

2.2.5 Нейронные сети

Данная модель представляет из себя последовательность нейронов, связанных синапсами. Функция нейронной сети – преобразование входных факторов в выходной. Подбор весов этих факторов и порога возбуждения называется процессом обучения.

На рисунке ниже представлена полная модель искусственного нейрона.

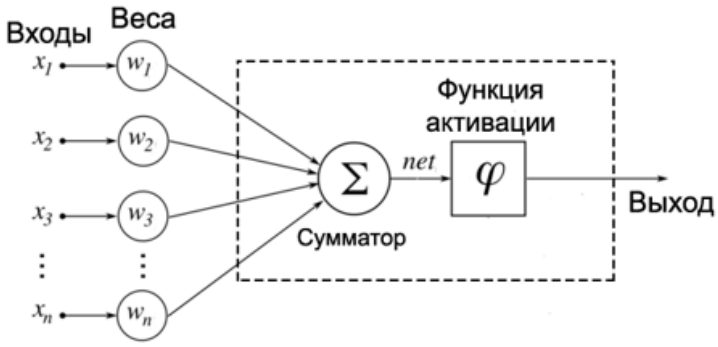


Рис 2.2.5.1 Схема работы нейрона

Поступающие на вход каждого нейрона сигналы умножаются на соответствующие веса. Каждое произведение поступает в качестве слагаемого на вход сумматора. На выходе получается взвешенная сумма входных сигналов.

Чтобы усиливать слабые сигналы и не позволять сети перенасыщаться сильными используются функции активации. В зависимости от целей задачи могут быть использованы самые разные функции, в том числе и сформированные собственноручно. Главное требование – существование градиента функции, который будет использован в алгоритмах оптимизации весов при обучении.

На рисунке ниже представлены основные функции активации для нейронных сетей.

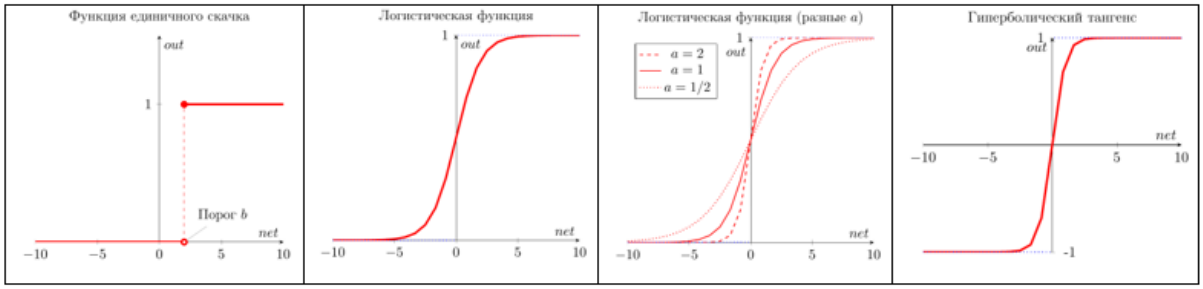


Рис 2.2.5.2 Функции активации

2.2.5.1 Виды нейронных сетей

По количеству слоев выделяют однослойные и многослойные модели. В первом случае взвешенная сумма сигналов входного слоя подается сразу на выходной слой.

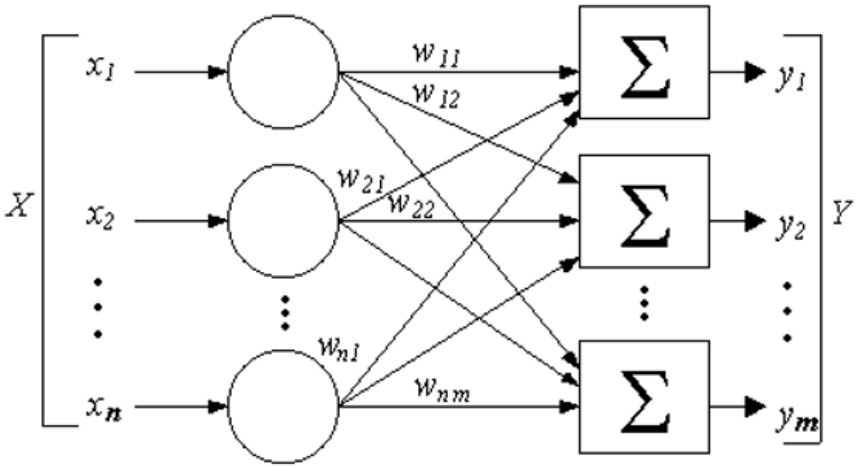


Рис 2.2.5.1 Схема однослойного перцептрона

Многослойные нейронные сети помимо входного и выходного слоя характеризуются еще и скрытым(и) слоями. На каждом скрытом слое задается количество нейронов. Каждый нейрон получает взвешенный сигнал каждого нейрона с предшествующего слоя и передает результат на каждый нейрон последующего слоя. Таким образом сложность модели возрастает, что позволяет находить сложные закономерности. Полносвязные сверточные нейронные сети (fully connected convolutional neural networks) на первом слое в задачах распознавания образом с изображения учатся распознавать простейшие геометрические примитивы (прямые, углы, окружности). На последующих слоях эта информация помогает распознавать элементы лица (глаза, брови, уши, нос) и в итоге получается модель, которая позволяет распознавать возраст человека на фотографии.

2.2.5.2 Обучение нейронных сетей

Разделяют обучение модели с учителем (supervised) и без учителя (unsupervised). В первом случае модель обучается на прецедентах, следовательно, необходимо иметь размеченную обучающую выборку и дифференцируемую функцию потерь. Таким образом задача сводится к оптимизации весов и решается численными методами

2.2.5.2.1 Обратное распространение ошибки

Обновление весов происходит, двигаясь от выходов к входам сети. Алгоритм метода, следующий:

- Инициализация весов случайными значениями из близкого к 0 интервалу.
- До тех пор, пока условие останова не выполнено, do:
  1. Каждый входной нейрон ($X_i, i = 1,2, \dots, n$) воздействуем импульсом всем нейронам следующего (скрытого) слоя

2. Каждый скрытый нейрон $(Z_j, j = 1,2, \dots, p)$ обрабатывает поступающий сигнал в соответствии с заданными в пункте 1 весами, а также добавляет bias (смещение) : $Z\_in_j = Vo_j + \sum_{i=0}^{n} x_i * v_{ij}$, далее сигнал обрабатывается активационной функцией $z_j = f(z_{in_j})$. Полученный сигнал посылается на вход нейронов следующего уровня
3. Выходные нейроны $(Y_k, k = 1,2, \dots, m)$ действую также как обычные нейроны скрытых слоев, взвешивают сигнал и добавляют смещение: $y\_in_k = w_{0k} + \sum_j Z_j * w_{jk}$ в конце концов к полученному сигналу применяется активационная функция $y_k = f(y_{in_k})$
4. Каждый выходной нейрон вычисляет невязку полученного и целевого сигнала и перемножает разность с производной функции активации $\delta_k = \left(t_k - y_k\right) * f'\left(y_{in_k}\right)$

   а также вычисляет насколько необходимо изменить веса

   $$\Delta w_{jk} = a * \delta_k * z_j$$

   и смещение

   $$\Delta w_{ok} = a * \delta_k$$

   и посылает δ_k нейронам предыдущего слоя
5. Каждый скрытый нейрон суммирует приходящие к нему из выходного слоя ошибки и вычисляет величину ошибки как:

   $$\delta_j = \delta in_j * f'\left(z_{in_j}\right)$$

   а также вычисляет величину, на которую будем менять вес связи

   $$\Delta v_{ij} = a * \delta_j * x_i$$

   и вычисляет величину корректировки смещения:

   $$v_{0j} = a * \delta_j$$
6. Веса в слоях меняются добавлением к случайно заданным величины Δ
7. Проверка условий останова.

В условиях останова обычно задается достижение функции потерь определенной отбивки или выполнения определенного количества итераций. Функции останова помимо всего прочего выполняют также роль контроллера переобучения. При выходе функции ошибки на тесте, но последовательном снижении ошибки на обучении процесс обучения принято останавливать, так как достигнут явный идентификатор переобучения

2.2.5.3 Начальные значения весов

От того, по какой логике задаются начальные значения весов зависит то, сумеет ли достичь оптимизатор глобального минимума или нет. Поэтому необходимо убедиться, что случайные значения весов не обнуляют сигнал на функции активации или ее производной. Рекомендуется присваивать им значение в интервале [-0.5 до 0.5].

3 Построение карт перспективности

На этапе ранжирования опций, задача выявление перспективных для проведения ОПР зон является одной из ключевых задач, от корректного решения которой может зависит успех всего проекта в целом. На первом этапе необходимо выявить какие показатели являются целью, показателем перспективности участка (target), а какие косвенно определяют перспективность (features)

3.1 Входные данные

Вся территория Западной Сибири в разной степени исследована сейсмическими методами (далее СРР), отбором керна, гидродинамическими исследованиями (далее ГДИС), геофизическими исследованиями (далее ГИС), испытаниями итд.

Экспертно, а также основываясь на формуле Дюпюи:

$$\eta_0 = \frac{kh}{\mu B} * \frac{2\pi}{\ln\left(\frac{R_k}{r_c}\right)},$$

можно сделать вывод, что дебет жидкости явно зависит от таких параметров как проницаемость, эффективная мощность пласта, вязкость флюида, а также неявно от пористости, нефтенасыщенности, давления (пластового и устьевого) итд, то есть от т.н hard-data, геологических параметров. Связь всех этих параметров с показателем успешности скважины (высоким запускным дебетом и низкой обводненностью промышленной разработки или результаты испытаний для разведочных скважин) интуитивно прослеживается, тем не менее для пластов ачимовской толщи такие канонические зависимости как ($h_{эфф}$ – $Q_н$) не всегда имеют место быть.

В качестве показателей разработки будем учитывать запускной дебет скважины. По причине того, что ачимовская толща ранее крайне редко являлась рентабельным объектом, количество скважин с отработкой на данный момент ограничено ~150 для Аганского и Ватинского месторождения. Для пополнения базы для обучения будет также учитывать результаты испытаний, приведенных к одним условиям со скважинами промышленной разработки (длина ствола, тип заканчивания, диаметр штуцера, параметры НКТ)

Среди геологических карт в обучение принято включить следующие:

1. Карта изученности
2. Карта пластовых температур
3. Карта TVDSS Бажена
4. Карта TVDSS Ачимовской толщи
5. Карта общих толщин
6. Карта эффективных толщин
7. Карта пористости
8. Карта проницаемости
9. Карта плотности флюида
10. Карта пластовых давлений
11. Карта АВПД
12. Карта общего количества сгенерированных УВ

3.1.1 Методика картопостроения

Средствами ПО Schlumberger Petrel на основании результатов СРР были построены карты глубины Баженовской и Ачимовской толщи, а также карта эффективных толщин

Карта изученности строилась на основе информации о пробуренных скважинах, а также о проведенных 2D и 3D СРР. В зависимости от степени изученности точке присваивается значение от 0 до 1

На основе данных РИГИС были построены карты пористости

На основе данных разработки были получены карты пластовых давлений и карта аномальности пластового давления

На основе седиментологических данных были получены карты общего количества сгенерированных углеводородов.

Карты строились методом криггинга ПО Petrel. Также добавлялись подложки в виде сопутствующих карт на основе эмпирических зависимостей

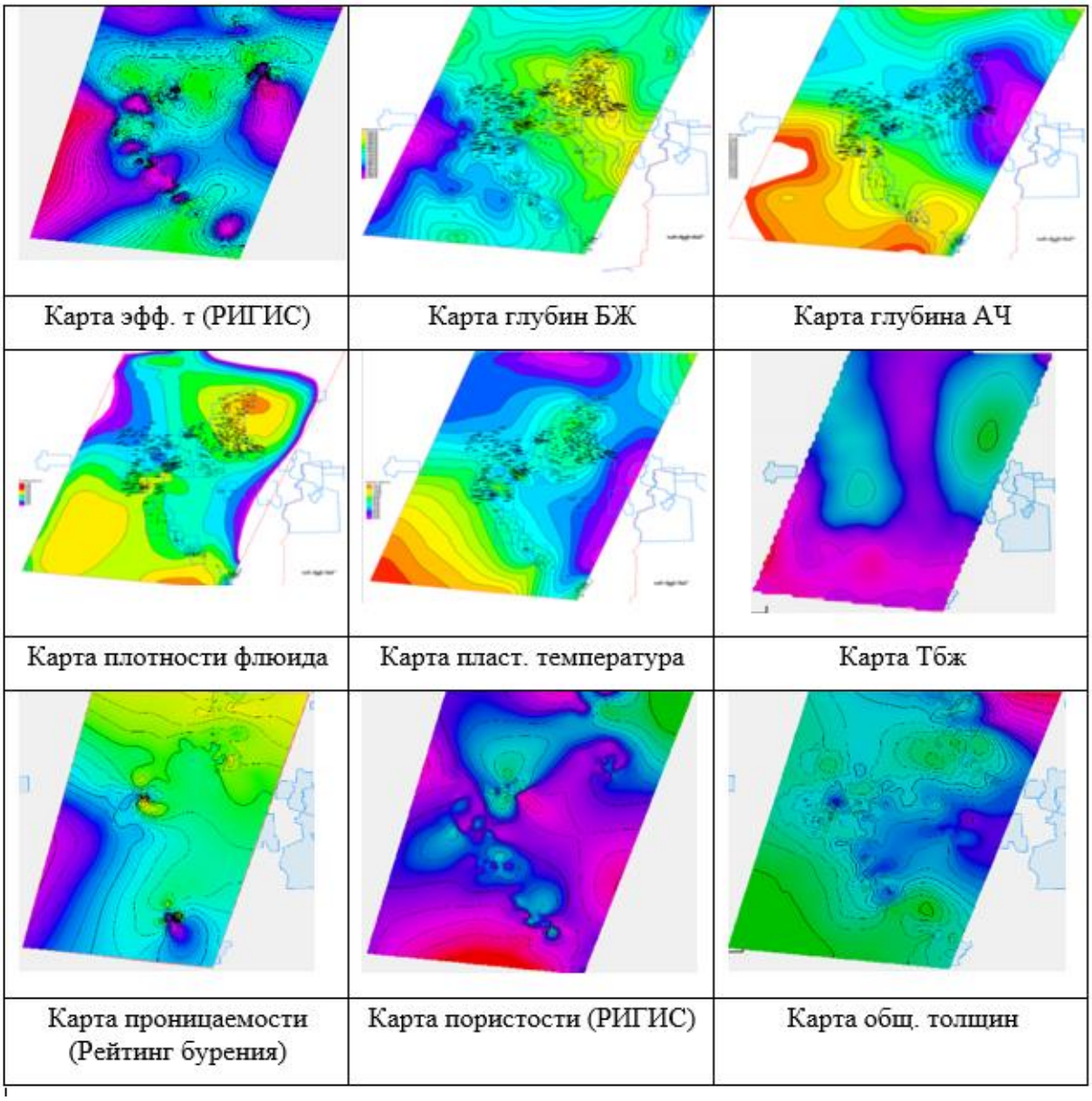


Рис.3.1.1 Итоговые карты для обучения

3.1.2 Формирование обучающей выборки

Для формирования исходной обучающей выборки будем “снимать” значения с геологических карт описанных выше в точках эксплуатационных и испытанных скважин (чтобы также получать значение запускного дебета либо результата испытания). Для пополнения обучающей выборки было сделано допущение, что скважина со значением дебета может располагаться в радиусе 200м от скважины с геологическими исследованиями.

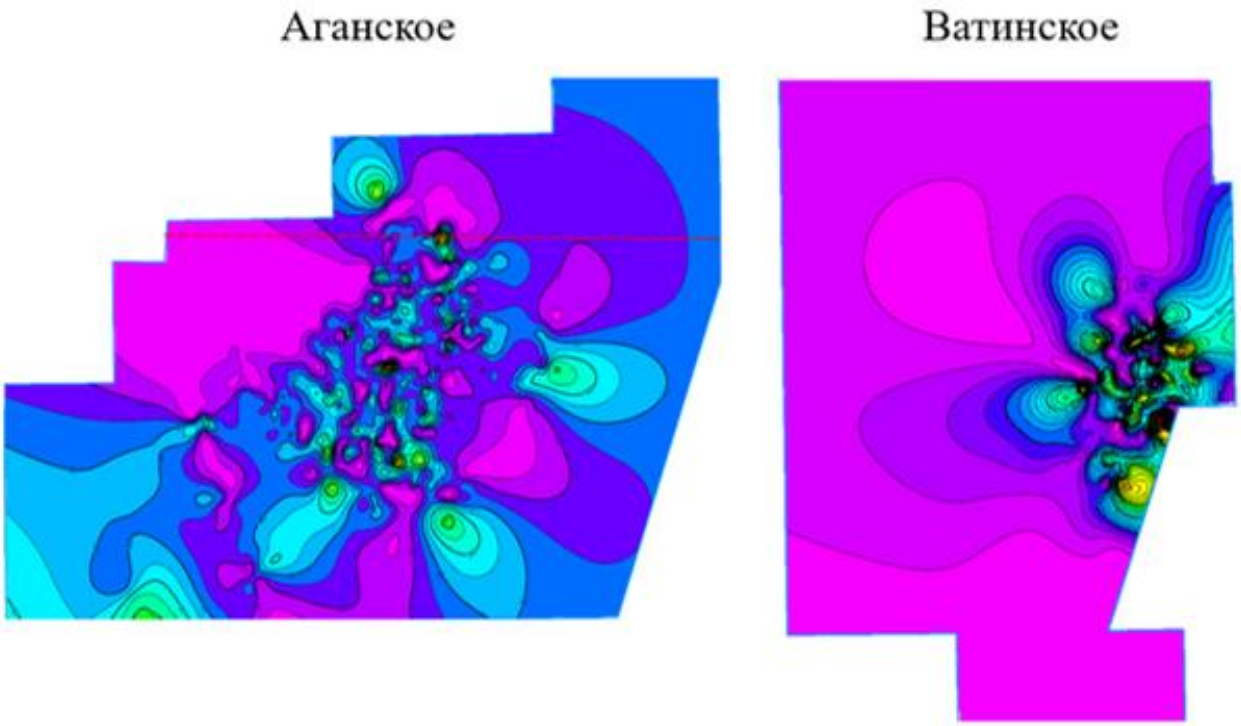


Рис 3.1.2 Экстраполяция точек с дебетом скважин для пополнения обучающей выборки

В результате была получена выборка размером 70 точек для обучения.

3.1.3 Препроцессинг входных данных

Имея дело с реальными данными с полей зачастую приходится производить их предварительную обработку. Наиболее важная часть в подготовке датасета для обучения – поиск аномалий, пропущенных значений, взаимных корреляций (некоторые модели МО чувствительны к взаимно коррелирующим входным параметрам) исходных данных.

3.1.3.1 Анализ распределения параметров

Прежде всего необходимо взглянуть на распределение параметров. Как можно увидеть (рис 3.1.3.1), распределение каждого параметра является нормальным либо близким к нормальному. Тем не менее можно также заметить аномальные выбросы на концах кривой распределения.

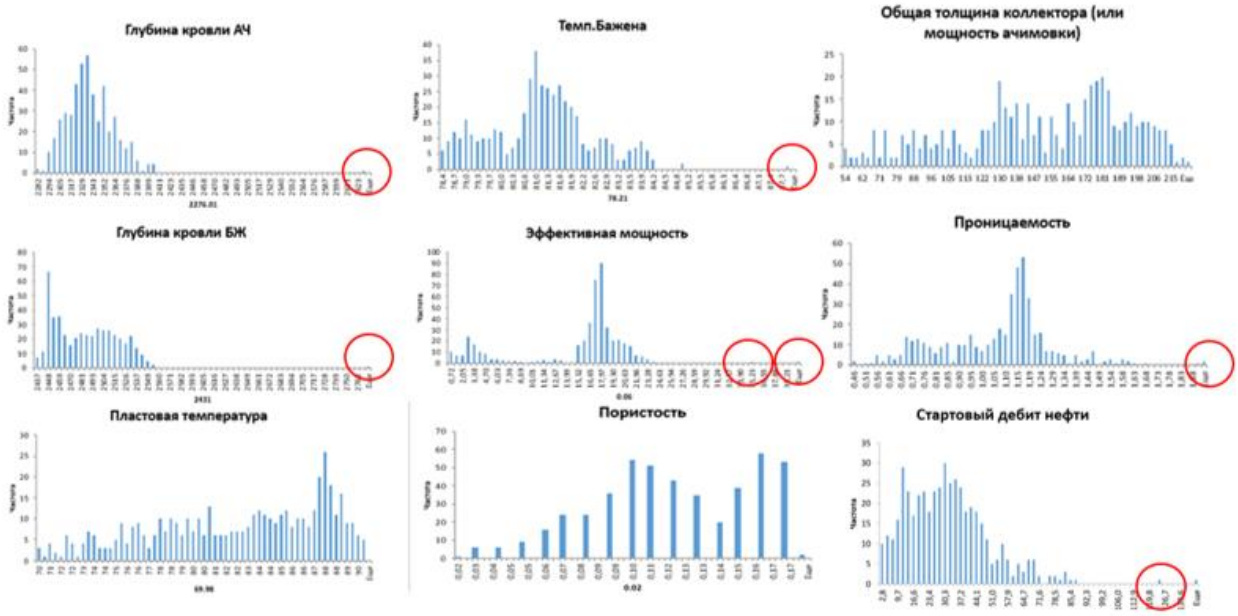


Рис 3.1.3.1 – Распределение исходных параметров (красным выделены аномалии)

3.1.3.2 Методы выявления выбросов

Существует два направления, которые занимаются поиском аномалий: **детектирование выбросов** (Outlier Detection) и **«новизны»** (Novelty Detection). Разница заключается в том, что в первом случае мы последовательно определяем аномально высокие или аномально низкие значения, а во втором мы определяем значения, которые не похожи на все предыдущие в последовательности.

Аномалии в исходных данных могут быть по следующим причинам:

- Человеческий фактор (ошибки при округлении, переносе данных в базу и.т.д)
- Технологический фактор (погрешности приборов, неисправность приборов)

3.1.3.2.1 Статистические тесты

Такие тесты применяются для отдельных признаков и способны хорошо отлавливать экстремальные значения (Z-value, Kurtosis measure)

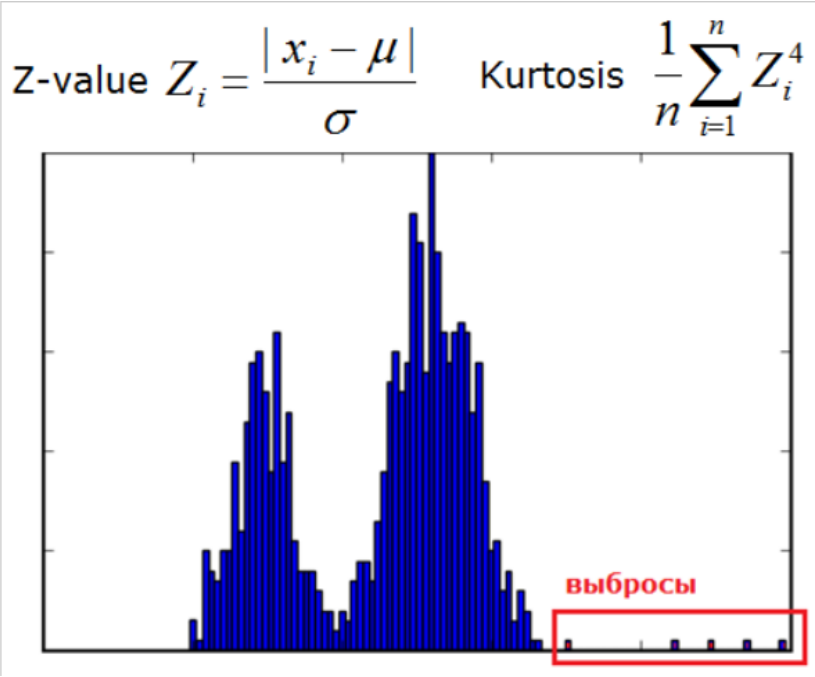


Рис 3.1.3.2.1 Пример выбросов на гистограмме распределения

3.1.3.2.2 Модальные тесты

Суть метода в том, что мы строим модель, которая описывает наши данные и делаем выводы о наличии аномалий по невязкам с реальными значениями. Такие методы работают для аномалий, но хуже для выбросов, так как модель хорошо может под них “затачиваться”. Так, например, используя сингулярное разложение для нахождения матрицы признаков наиболее похожей на исходную мы можем идентифицировать аномалии путем выявления элементов, которые сильно отличаются между матрицами.

Рис. 3.1.3.2.2 Применение SVD

3.1.3.2.3 Итерационные методы

В n-мерном признаковом пространстве можно построить т.н выпуклую оболочку и принимать за выбросы элементы выходящую в первую оболочку.

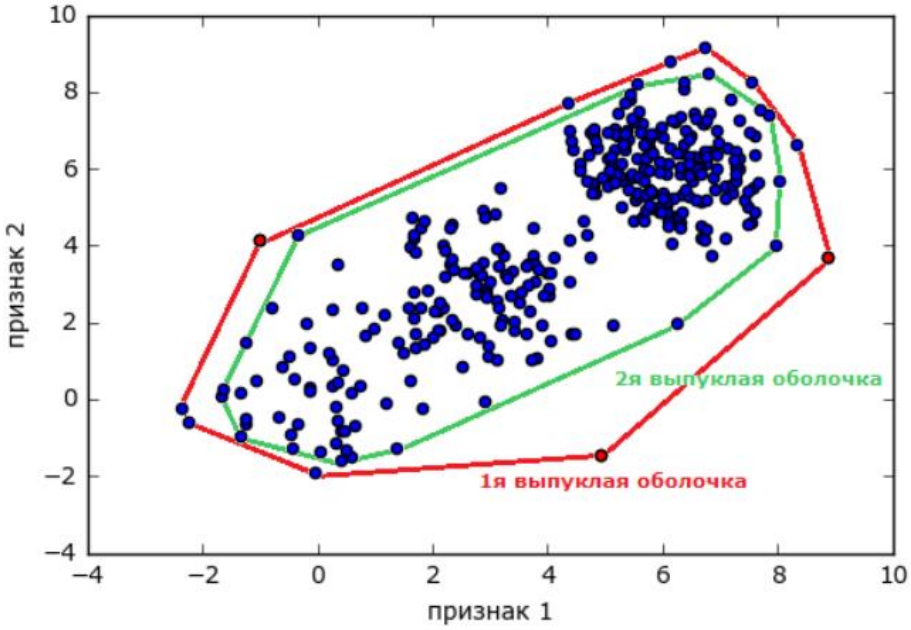


Рис. 3.1.3.2.3 Выпуклые оболочки

3.1.3.2.4 Метрические методы

Наиболее популярные методы среди дата-исследователей. За основу метода берется идея о том, что у типичной точки много соседей, а у нетипичной – мало. Метрикой близости может служить например “расстояние до К-го соседа”.

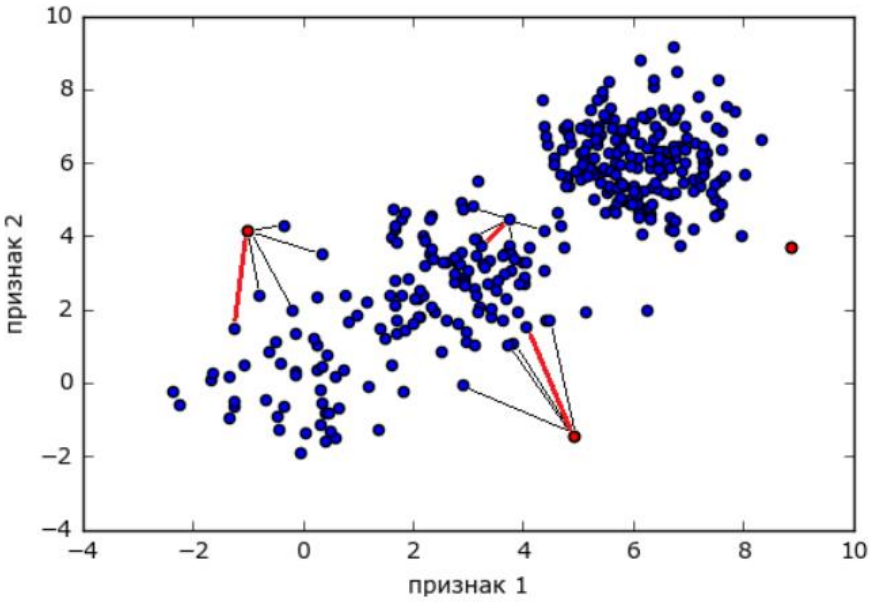


Рис. 3.1.3.2.4 Метод ближайших соседей

3.1.3.2.5 Обработка выбросов в исходных данных

При допустимо нормальном распределении параметра для выявления аномалий и выбросов распространен критерий Диксона. В зависимости от объема используются скорректированные формулы для расчета данного коэффициента. Полученное значение сравнивается с табличным значением при соответствующем значении доверительной вероятности. Сомнительное значение считают ошибкой и выбрасывают

n	Число односторонних выбросов в вариационном ряду	
	Один	Два и больше
3..7	r_{10}	r_{20}
8..10	r_{11}	r_{20}
11..13	r_{21}	r_{21}
14..30	r_{22}	r_{22}

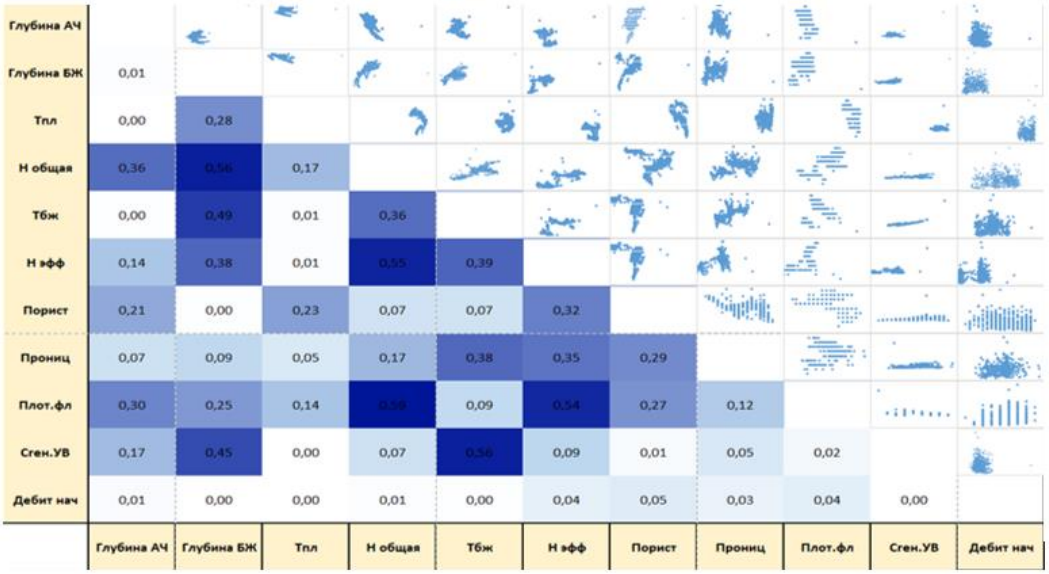
Коэффициент Диксона для выброса	
Наименьшего	Наибольшего
$r_{10} = (x_2 - x_1)/(x_n - x_1)$	$r_{10} = (x_n - x_{n-1})/(x_n - x_1)$
$r_{11} = (x_2 - x_1)/(x_{n-1} - x_1)$	$r_{11} = (x_n - x_{n-1})/(x_n - x_2)$
$r_{21} = (x_3 - x_1)/(x_{n-1} - x_1)$	$r_{21} = (x_n - x_{n-2})/(x_n - x_2)$
$r_{22} = (x_3 - x_1)/(x_{n-2} - x_1)$	$r_{22} = (x_n - x_{n-2})/(x_n - x_3)$
$r_{20} = (x_3 - x_1)/(x_n - x_1)$	$r_{20} = (x_n - x_{n-2})/(x_n - x_1)$

Рис. 3.1.3.2.5 Алгоритм расчета критерия Диксона для разных объемов выборок (Xn - наибольший элемент в ряду, X1 – наименьший элемент в ряду)

Проводить расчет коэффициента Диксона для каждого элемента нашей выборки будем при помощи библиотеки sklearn, которая автоматически исключает наборы с аномальными значениями. При граничащих расчетных критериях Диксона будем восстанавливать действительные значениям методом ближайших соседей библиотеки sklearn.

3.1.3.2.6 Оценка взаимных корреляций

Построим корреляционный кросс-плот для анализа взаимозависимости между исходными переменными.



	Глубина АЧ	Глубина БЖ	Тпл	Н общая	Тбж	Н эфф	Порист	Прониц	Плот.фл	Сген.УВ	Дебит нач
Глубина АЧ											
Глубина БЖ	0,01										
Тпл	0,00	0,28									
Н общая	0,36	0,56	0,17								
Тбж	0,00	0,49	0,01	0,36							
Н эфф	0,14	0,38	0,01	0,55	0,39						
Порист	0,21	0,00	0,23	0,07	0,07	0,32					
Прониц	0,07	0,09	0,05	0,17	0,38	0,35	0,29				
Плот.фл	0,30	0,25	0,14	0,58	0,09	0,54	0,27	0,12			
Сген.УВ	0,17	0,45	0,00	0,07	0,56	0,09	0,01	0,05	0,02		
Дебит нач	0,01	0,00	0,00	0,01	0,00	0,04	0,05	0,03	0,04	0,00	

Рис. 3.1.3.2.6 Взаимная корреляция

Как видно, явно, дебет нефти не зависит ни от одного параметра, поэтому такой набор данных будет слабо описываться линейными моделями.

При этом есть параметры, коэффициент корреляции R^2 которых >0.5. что часто плохо сказывается на точности моделей

3.1.3.3 Снижение размерности. Метод главных компонент

Метод главных компонент – PCA – базовых подход, который применяется в задачах снижения размерности пространства признаков, а также при анализе взаимных корреляций данных. Метод предложен Карлом Пирсоном в 1901 году, а в дальнейшем разработан экономистом и статистиков Г.Хоттелингом.

3.1.3.3.1 Сущность метода

Сущность метода заключается в снижении размерности пространства признаков – столбцов матрицы X, путем перехода к новым матрицам T и P, размерность которых, A, меньше, чем число переменных (столбцов) J у исходной матрицы X.

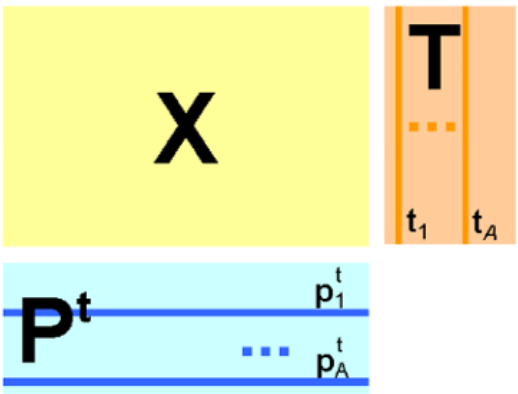


Рис 3.1.3.3.1 Декомпозиция матрицы X

Вводятся новые переменные t_a(a=1,…,A), которые являются линейной комбинацией исходных переменных x_j (j=1,…,J) :

$$t_a = p_{a1}x_1 + \cdots + p_{aj}x_j$$

Таким образом исходная матрица признаков X может быть разложена в произведение двух матриц T и P –

$$X = TP^t + E = \sum_{a=1}^{A} t_a p_a^t + E$$

Где матрица T это матрица **счетов**, размерностью (IxA), а P матрица **нагрузок,** ее размерность (JxA). Матрица E – матрица остатков, размерностью (IxJ).

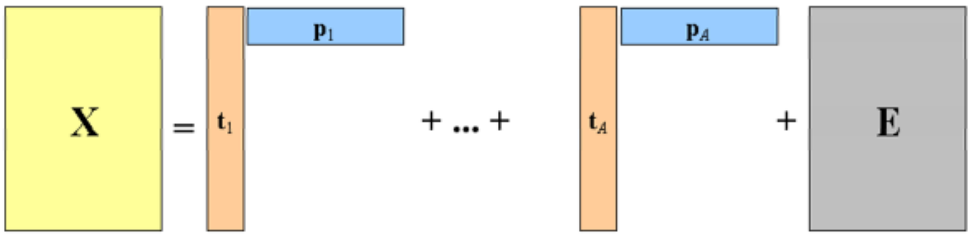
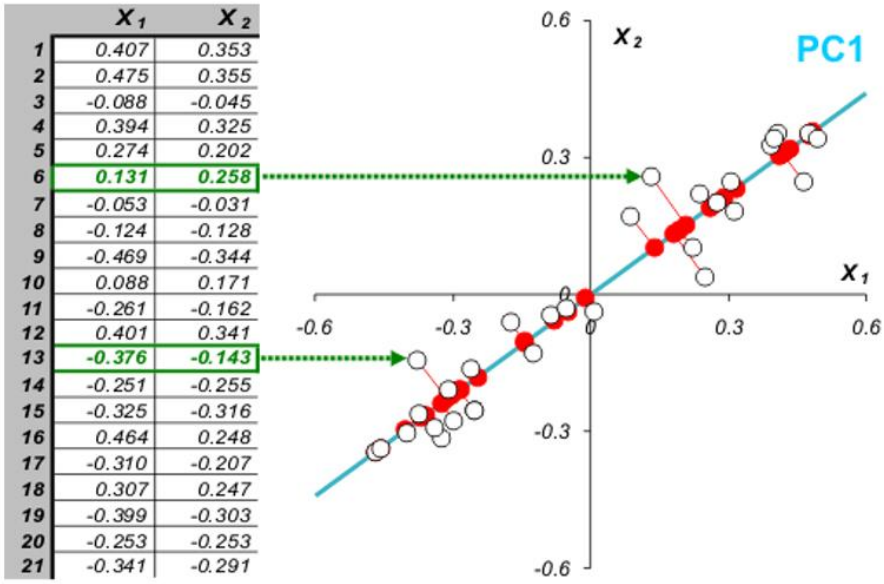


Рис 3.1.3.3.2 – Разложение матрицы X на главные компоненты



	X_1	X_2
1	0.407	0.353
2	0.475	0.355
3	-0.088	-0.045
4	0.394	0.325
5	0.274	0.202
6	0.131	0.258
7	-0.053	-0.031
8	-0.124	-0.128
9	-0.469	-0.344
10	0.088	0.171
11	-0.261	-0.162
12	0.401	0.341
13	-0.376	-0.143
14	-0.251	-0.255
15	-0.325	-0.316
16	0.464	0.248
17	-0.310	-0.207
18	0.307	0.247
19	-0.399	-0.303
20	-0.253	-0.253
21	-0.341	-0.291

Рис 3.1.3.3 Первая главная компонента

Все операции метода базируются на нахождении собственных значений и собственных векторов ковариационной матрицы признаков. Это матрица, элементы которой – ковариации признаков i и j:

$$Cov(X_i, X_j) = E\left[\left(X_i - E(X_i)\right) * \left(X_j - E(X_j)\right)\right] = Cov(X_i, X_j) - E(X_i) * E(X_j)$$

Так как $E(X_i) = E(X_j) = 0$, то уравнение упрощается до:

$$Cov(X_i, X_j) = E(X_i * X_j)$$

Когда $X_i = X_j$:

$$Cov(X_i, X_i) = Var = (X_i)$$

Таким образом, ковариационная матрица будет состоять из диагональных элементов равных дисперсии соответствующего признака, а остальные элементы будут равняться ковариации соответствующих пар признаков. Заметим, что ковариационная матрица будет симметрична

Для того, чтобы найти первую главную компоненту, необходимо вычислить такой случайный вектор, проекция нашей выборки признаков на который $v^T X$ давала бы максимальную дисперсию этой выборки. Дисперсия центрированных величин в векторной форме в общем случае выражается следующим образом:

$$Var(X) = \sum = E(X * X^T)$$

Таким образом дисперсия проекции:

$$Var(X^*) = \sum^* = E\,(X^* * X^{*T}) = E\big((\vec{v}^T X) * (\vec{v}^T X)^T\big) =$$

$$E(\vec{v}^T X * X^T \vec{v}) = \vec{v}^T E(X * X^T)\vec{v}^T = \vec{v}^T \sum \vec{v}$$

Следовательно, дисперсия максимальна при максимальном $\vec{v}^T \sum \vec{v}$. Используя уравнение Релея для ковариационных матриц:

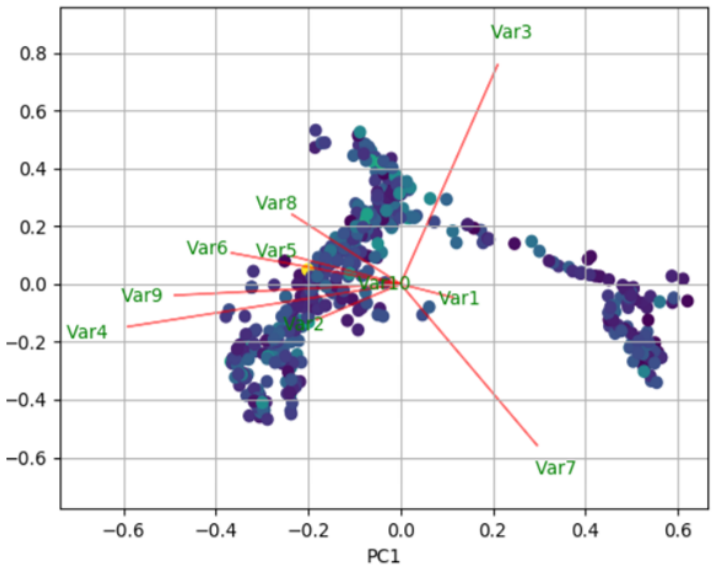
$$R(M, \vec{x}) = \frac{\vec{x}^T M \vec{x}}{\vec{x}^T \vec{x}} = \lambda \frac{\vec{x}^T \vec{x}}{\vec{x}^T \vec{x}} = \lambda$$

$$M\vec{x} = \lambda\vec{x}, \text{ где } \lambda - \text{собственное значение}$$

Таким образом можно заключить, что направление максимальной дисперсии совпадает с направлением собственного вектора, имеющего максимальное собственное значение, которое количественно равно этой дисперсии. Такой вывод также справедлив для n измерений.

Наибольший вектор имеет направление, схожее с линией регрессии и, спроецировав на него нашу выборку, мы потеряем информацию, сравнимую с суммой остаточных членов регрессии

Для выявления наиболее влияющих на целевую функцию параметров, необходимо построить матрицу нагрузок (loadings). Все действия легко реализуются методом PCA библиотеки sklearn языка программирования python:



Explained Variance Ratio:

PC1: 4.77454673e-01

PC2: 3.14780212e-01

PC3: 1.05798830e-01

PC4: 3.99414569e-02

PC5: 3.09441077e-02

PC6: 1.50255923e-02

PC7: 1.06316591e-02

PC8: 4.72351421e-03

PC9: 6.99829703e-04

PC10:1.24577513e-07

Рис 3.1.3.2 Главные компоненты

Таким образом, первые три главные компоненты описывают ~95% исходных данных, следовательно, размерность пространства признаков можно снизить с 10 до 3.

Матрица нагрузок (loadings) может быть интерпретирована как показатель связи между компонентами. Возьмем первые 2 главных компоненты и посмотрим, какую нагрузку дает каждая исходная переменная на главные компоненты:

X	5.683872890675861,
Y	4.772864115132131,
Карта изученности	6.875140142713248,
Карта пластовых температур	3.715965098726352,
Глубина АТ	9.362507956946587,
Карта общих толщин	5.675918398463982,
Кара эффективных толщин	5.613774044329784,
Карта пористости	3.6251530733202504,
Карта проницаемости рейтинг бур	6.850547685997503,
Карта плотности флюида	8.414483917631618,
Карта пластовых давлений	8.860704060134779,
Карта АВПД	3.525724106042169,
Глубина БЖ	4.585997038989785,
Heff	9.778368443719405,
Общее колво сген УВ	4.002085932

Табл 3.1.3.3 Интерпретация матрицы факторных нагрузок

Как видно, наиболее сильное влияние на целевую функцию производят параметры из формулы Дарси: эффективная толщина, плотность флюида, депрессия (пластовое давление), проницаемость. Слабое влияние оказывают параметры: общее кол-во сген.УВ, АВПД (что подтверждается отсутствием аномальности пластового давления на юге Западной Сибири), пористости. Данная информацию может быть использована при настройке модели

3.1.3.4 Нормировка данных

По причине того, что в обучении используются параметры разных порядков, некоторые модели могут работать некорректно. Для приведения параметров к нормальному виду необходимо воспользоваться встроенными методами библиотеки sklearn: Standart Scaler.

3.2 Подбор моделей

Из всего многообразия моделей, используемых в машинном обучении в задачах регрессии, были отобраны 4 наиболее подходящие: линейные модели, случайный лес, градиентный бустинг алгоритмов, нейронные сети. Был проведено тестирование данных моделей на обучающей выборки без дополнительной настройки. Результаты показаны на рисунке ниже:

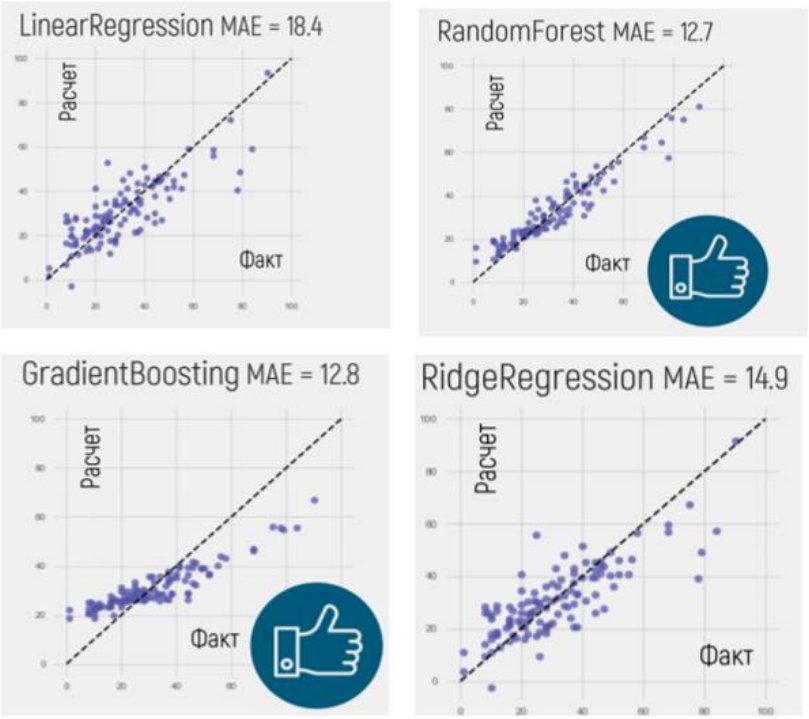


Рис 3.2 Результаты работы моделей

Как оказалось самую высокую точность показали 2 модели: нейронные сети прямого распространения (многослойный перцептрон из библиотеки Keras) и градиентный бустинг случайных лесов (из библиотеки catboost)

Считается, что на однородных данных (видео, изображения) лучшую точность показываются нейронные сети. Неоднородные же данные лучше описываются алгоритмами градиентного бустинга.

Для выбора модели был произведен прогон нескольких отобранных ранее на аналогичных задачах алгоритмов без настройки параметров на исходных данных. Наиболее точный оказался алгоритм библиотеки CatBoost, разработанный компанией Яндекс.

Yandex CatBoost

Рис.3.2.1 Библиотека Catboost

Данная библиотека показывает лучшие оценки по сравнению с существующими на общепринятых данных как с настройкой гиперпараметров так и без. Преимущества модели:

- Самая быстрая библиотека в классе
- Поддержка и числовых и категориальных данных
- Поддержка компиляции на GPU

- Интерактивные инструменты визуализации данных и процесса обучения

3.2.1 Настройка модели

На первом этапе работы, в качестве отправной точки, зафиксируем результаты работы модели на стандартных параметрах. Модель будет оптимизировать по метрике Mean Absolute Error

$$MAE = \frac{1}{n} * \sum_{j=1}^{n} |\hat{y}_j - y_j|$$

Во избежание эффекта переобучения будем наблюдать за метрикой MAE на тестовой части выборки. При выходе функции MAE (n) на плато будем останавливать обучение.

Рис.3.2.1 Схема обучения

Процесс обучения проиллюстрирован на рисунке ниже:

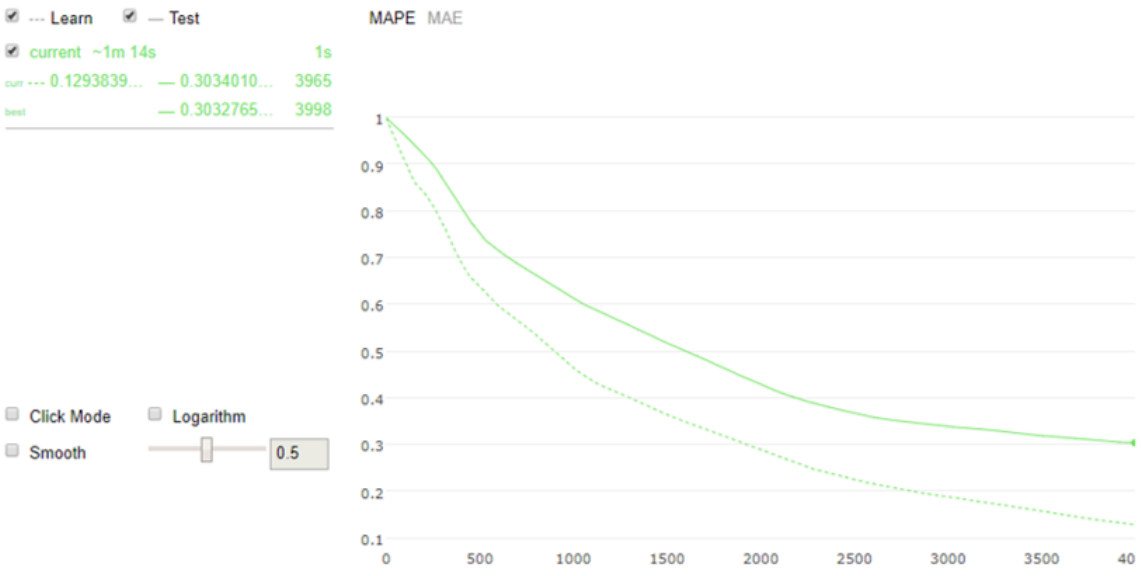


Рис 3.2.1.1 Кривая обучения на исходных данных

Для обучающей выборки из 70 сэмплов, производим разделение на 30% для теста и 70% для обучения

Как видно, к 4000 итерации кривая MAPE выходит на плато, а кривая обучение продолжает падать. Это один из признаков переобучения.

Таким образом, на стандартных гиперпараметрах относительная ошибка на тренировочную выборку составила 12.9%, а на тестовой выборке (скрытые от обучения данные) 30.3%.

3.2.2 Тонкая настройка гиперпараметров модели

Одно из преимуществ данной библиотеки – высокая точность без настройки гиперпараметров. Тем не менее можно получить небольшое снижение ошибки путем цикличного перебора определенного диапазона параметров на исходных данных

Искать наилучшие параметры будем с помощью библиотеки Sklearn и метода GridSearchCV. Особенность метода в том, что он делит тренировочную выборку на части и тестирует комбинации параметров по отдельности на каждой части, а затем усредняет полученный результат. Таким образом можно решить проблему маленькой выборки и произвести тонкую настройку модели, при этом не переобучив ее.

Согласно теоретической основе алгоритма, модель наиболее чувствительна к изменению следующих параметров:

depth (глубина единичного дерева)

learning_rate (шаг обучения)

iterations (максимальное число деревьев).

Разобьем выборку для обучения на 3 части и будем обучать модель на каждой комбинации параметров сразу на всех частях, а затем усреднять результат прогноза:

- `'depth': [6, 8, 10, 12]`
- `'learning_rate': [0.01, 0.05, 0.1, 0.3, 0.5, 0.9]`

В результате проведения 24 экспериментов выявлены лучше параметры: глубина дерева 8, а шаг обучения 0.8. Эти результаты дают улучшение точности на 1.7 %. Таким образом точность модели по абсолютной ошибке стала 8 т/сутки, а по относительной 26.2%

```
Params Tuning:: 100%|##########| 24/24 [1:07:19<00:00, 168.31s/it]
Best MAE at all: 8.031820650321807
Best Params, provided best mae: {'depth': 8, 'learning_rate': 0.9}
```

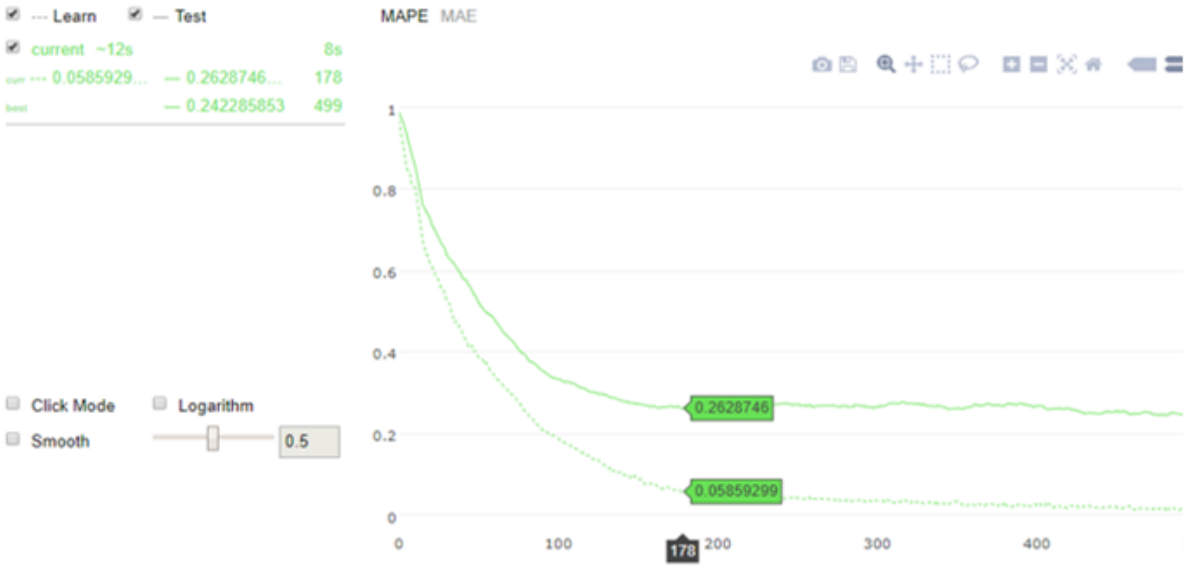


Рис 3.2.2 Кривая обучения после тонкой настройки параметров модели

4. Выводы

В результате работы была создана предварительно обработанная база данных геологических параметров и показателей разработки для обучения, а также подобран класс алгоритмов, наиболее подходящий к условиям данной задачи. Была произведена тонкая настройка гиперпараметров наилучшим образом характеризующих связь между геологическими данными и перспективой разработки. Полученная модель настроена на обучающую выборку, при этом не теряется обобщающая способность и на скрытых от обучения данных.

Помимо предиктивной модели была получена матрица важности признаков, описывающая степень влияния исходных параметров на целевую функцию – запускной дебет нефти.

Несмотря на высокую точность алгоритма необходимо осознавать, как именно он работает. Очевидно, что в районах, слабо исследованных разведочным бурением исходные данные будут браться из результатов экстраполяции. Поэтому достоверность данных в таких районах должна учитываться с долей допущений при интерпретации полученный прогнозов.

В качестве дальнейших шагов развития рассматривается актуализация модели на полученных от подрядной организации картах геологических параметров. При улучшении точности прогноза будет построена карта перспективности всего региона, которая в дальнейшем будет использована при обосновании проводимых опытно-промышленных мероприятий, а также для кластеризации региона по геологическим параметрам.

Список литературы

- Friedman J. Greedy Function Approximation: A Gradient Boosting Machine. _ IMS
- 1999 Reitz Lecture.
- Воронцов К.В. Машинное обучение (курс лекций)
- Friedman J. Stochastic Gradient Boosting. _1999
- https://tech.yandex.com/catboost/doc/dg/concepts/about-docpage/
- А.А.Баргасян, М.С.Куприянов, В.В.Степаненко, И.И.Холод “Методы и модели анализа данных: OLAP и DataMining”
- А.Б.Барский “Нейронные сети: распознавание, управление, принятие решений”
- Дж.-О.Ким, Ч.У.Мьюллер, У.Р.Клекка, М.С.Олдендерфер, Р.К.Блэшфилд “Факторный, дискриминантный и кластерный анализ”, перевод А.М.Хотинского, С.Б.Королева 1989

Приложение 1

Листинг исходных программ

1. Поиск гиперпараметров Catboost

```
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import catboost
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.grid_search import ParameterGrid
from tqdm import tqdm
from sklearn.metrics import mean_absolute_error
import sklearn
import codecs
#!/usr/bin/env python
# -*- coding: utf8 -*-

def cross_val(X, Y, X_test, param, n_splits):
    skf = sklearn.model_selection.KFold(n_splits=3, shuffle=True,
random_state=42)
    MAE = []
    predict = None
    i=0
    print(len(Y))
    for tr_ind, val_ind in skf.split(X, Y):
        Y_train = Y.iloc[tr_ind]
        print(len(Y_train))
        X_train = X.iloc[tr_ind]
        X_valid = X.iloc[val_ind]
        Y_valid = Y.iloc[val_ind]
        print(len(Y_valid))

        clf = catboost.CatBoostRegressor(iterations=3500,
                                learning_rate=param['learning_rate'],
                                loss_function='MAE',
                                depth=param['depth'],
                                eval_metric='MAE',
                                use_best_model=True,
                                #logging_level='Silent'
                                )

        clf.fit(X_train,Y_train,eval_set=(X_valid, Y_valid))

        y_pred = clf.predict(X_valid)
        mean_abs_err = mean_absolute_error(Y_valid, y_pred)
        MAE.append(mean_abs_err)
        i= i+1
        print("                          ")
        print("____________________________________________________________
")
        print("Прогон номер",i)
        print('Параметр depth = ', param['depth'])
        print('Параметр learning_rate = ', param['learning_rate'])
        print('Абсолютная ошибка MAE = ', MAE[i-1])

    return sum(MAE)/n_splits
```

```
def catboost_GridSearchCV(X, Y, X_test, params):
    ps = {'MAE': 1000,'param': []}
    predict = None
    for prms in tqdm(list(ParameterGrid(params)), ascii=True, desc='Params
Tuning:'):

        MAE = cross_val(X, Y, X_test, prms, n_splits=3)

        if MAE < ps['MAE']:
            ps['MAE'] = MAE
            ps['param'] = prms
    print('Best MAE at all: ' + str(ps['MAE']))
    print('Best Params, provided best mae: ' + str(ps['param']))

    return ps['param']

def main():
    #df = pd.read_csv('Train_476.csv', sep=";", header=0, index_col=0)
    #Y = df['Target']
    #X = df.ix[:,['Глубина АЧ', 'Глубина БЖ', 'Тпл', 'Н общая', 'Тбж', 'Н
эфф', 'Пористт', 'Прониц', 'Плот.фл', 'Сген.УВ']]

    df = pd.read_csv('train_54_cleared.csv', sep=";", header=0, index_col=0)
    Y = df['18']
    X = df.ix[:, ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11',
'12', '13', '14', '15', '16', '17']]


    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.05,
random_state=42)

    params = {'depth': [6, 8, 10, 12], 'learning_rate': [0.01, 0.05, 0.1,
0.3, 0.5, 0.9]}
    param = catboost_GridSearchCV(X_train, Y_train, X_test, params)
    print('Элементов выборки для финального прогна', len(Y_train))
    print(Y_train)
    clf = catboost.CatBoostRegressor(learning_rate=param['learning_rate'],
                            loss_function='MAE',
                            depth=param['depth'],
                            eval_metric='MAPE',
                            use_best_model=True
                            )
    X_train, X_valid, Y_train, Y_valid = train_test_split(X_test,Y_test,
                                                          shuffle=True,
                                                          random_state=42,
                                                          train_size=0.01)


    clf.fit( X_train, Y_train, eval_set=(X_valid, Y_valid))

 # Расчитаем предсказание на ВАЛИДАЦИИ части
    predict_valid = clf.predict(X_valid)
    MAPE = 100 * (np.mean(abs((Y_valid - predict_valid) / Y_valid)))
    MAE = (np.mean(abs((Y_valid - predict_valid))))
    print(predict_valid)
    print(Y_valid)
    print('Valid set with MAPE = %d' % (MAPE), 'and MAE =%d' % (MAE))


if __name__ == '__main__':
    main()
```

2. Тестирование нейронной сети

```
import seaborn as sns
import numpy as np
import tensorflow as tf
sns.despine()
from livelossplot import PlotLossesKeras
from keras.callbacks import TensorBoard
from time import time
from matplotlib import pyplot as plt
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.advanced_activations import LeakyReLU
from keras.layers import BatchNormalization, LeakyReLU
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, CSVLogger,
EarlyStopping, Callback
from keras.optimizers import RMSprop, Adam, SGD, Nadam
from keras import regularizers
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from keras import optimizers
def shuffle_in_unison(a, b):
    # courtsey http://stackoverflow.com/users/190280/josh-bleecher-snyder
    assert len(a) == len(b)
    shuffled_a = np.empty(a.shape, dtype=a.dtype)
    shuffled_b = np.empty(b.shape, dtype=b.dtype)
    permutation = np.random.permutation(len(a))
    for old_index, new_index in enumerate(permutation):
        shuffled_a[new_index] = a[old_index]
        shuffled_b[new_index] = b[old_index]
    return shuffled_a, shuffled_b


def create_Xt_Yt(X, y, percentage=0.8):
    p = int(len(X) * percentage)
    X_train = X[0:p]
    Y_train = y[0:p]
    X_train, Y_train = shuffle_in_unison(X_train, Y_train)
    X_test = X[p:]
    Y_test = y[p:]
    return X_train, X_test, Y_train, Y_test
```

```
dataset = np.loadtxt('Train_476.csv', delimiter=";",skiprows=1)
X = dataset[:,:10]
Y = dataset[:,10]
Z = dataset[:,:11]
```

```
pca= PCA(n_components=10)
pca.fit(Z)
```

```
loadings =  pca.components_.T * np.sqrt(pca.explained_variance_)
with open("loadings.txt", "w") as file:
    for item in loadings:
        file.write("%s\n" % item)


#histogram
#import pandas as pd
#x=pd.DataFrame(data=X)
#x.hist(bins=100)
#plt.show()
from scipy import stats


#for j in range (0,14):
 #print(stats.kstest(X[:,j],'norm'))



scaler = MinMaxScaler(feature_range=(0, 1))
scaler.fit(X)
X = scaler.transform(X)
Y = Y.reshape(-1, 1)
scaler1 = MinMaxScaler(feature_range=(0, 1))
scaler1.fit(Y)
Y = scaler1.transform(Y)
scaler.fit(Z)
Z = scaler.transform(Z)


dropouts = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]

X_train, X_test, Y_train, Y_test = create_Xt_Yt(X, Y)

for dropout  in dropouts:
 model = Sequential()
 model.add(Dense(10, input_dim=10, activation='sigmoid'))
 model.add(Dropout(dropout))
 model.add(BatchNormalization())
 model.add(Dense(50, activation='sigmoid'))
 model.add(BatchNormalization())
 model.add(Dropout(dropout))
 model.add(Dense(1, activation='linear'))

 reduce_lr = ReduceLROnPlateau(monitor='loss', factor=0.9, patience=10,
min_lr=0.00001, verbose=1)

 checkpointer = ModelCheckpoint(filepath="test.hdf5", verbose=1,
save_best_only=True)

 SGD=optimizers.SGD()
 model.compile(optimizer='sgd', loss='mae')

#tensorboard =
TensorBoard(log_dir="PycharmProjects/achimovka/test".format(time()))


 history = model.fit(X_train, Y_train,
                    epochs=40,
                    batch_size=5,
                    verbose=2,
                    validation_data=(X_test, Y_test),
                    callbacks=[reduce_lr,checkpointer],
                    shuffle=True)
```

```
 score1 = model.evaluate(X_test, Y_test, verbose=0)
 score2 = model.evaluate(X_test, Y_test, verbose=0)
 #y[dropout] = model.predict(X_test)
 print('Train_mae is', score1, 'and test_mae is', score2)
 pred = model.predict(X_test)
 Y_test = scaler1.inverse_transform(Y_test)
 pred = scaler1.inverse_transform(pred)
### ЛОСС ПО ТРЕН И ТЕСТУ ###

 plt.plot(history.history['loss'])
 plt.plot(history.history['val_loss'])
 plt.title('model loss')
 plt.ylabel('loss')
 plt.xlabel('epoch')
 plt.legend(['train', 'test', ''], loc='best')
 plt.show()

 ### Кривая теста ###
 MAPE = 100 * (np.mean(abs((Y_test - pred) / Y_test)))
 MAE = (np.mean(abs(Y_test - pred)))
 plt.plot(Y_test, color='black', label='Original data')
 plt.plot(pred, color='blue', label='Predicted data')
 plt.legend(loc='best')
 plt.title('Test set with MAPE = %d', MAPE)


pred=model.predict(X_test)
pred1=model.predict(X_train)
Y_test = scaler1.inverse_transform(Y_test)
pred = scaler1.inverse_transform(pred)
Y_train = scaler1.inverse_transform(Y_train)
pred1 = scaler1.inverse_transform(pred1)
print("Оригинал:", Y_test)
print("Прогноз:", pred)


MAPE=100*(np.mean(abs((Y_test-pred)/Y_test)))
MAE=(np.mean(abs(Y_test-pred)))

print("Относительная ошибка на тесте:",MAPE)


original1 = Y_train
predicted1 = pred1



plt.plot(Y_test, color='black', label='Original data')
plt.plot(pred, color='blue', label='Predicted data')
plt.legend(loc='best')
plt.title('Test set with MAPE = %d' % (MAPE), 'and MAE =%d' %(MAE))
plt.show()

plt.plot(original1, color='black', label='Original data')
plt.plot(predicted1, color='blue', label='Predicted data')
plt.legend(loc='best')
plt.show()
```