ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В странах с большой железнодорожной сетью и большим потоком перемещения поездов, таких как РФ, США, Китай, Индия, существует проблема схода составов с рельс, которые могут быть обусловленны различными факторами, их можно классифицировать на:

- внешние: кривизна пути, профиль пути, состояние транспортного пути, проблемы со стрелочным переводом, погодные условия (при экстремальных температурах рельсы могут сильно расширяться или сжиматься);
- внутренние: количество вагонов в составе, загруженность, скорость, невнимательность машиниста, состояние состава.

Некоторые пути могут проходить через национальные парки, национальные заповедники и другие типы особо охраняемых объектов. По этой причине аварии, произошедшие на таких участках, могут привести к экологической катастрофе, особенно велика опасность, если поезд был грузовым и перевозил легко воспламеняемые грузы (нефть, газ, метан, уголь, древесина) или высокотоксичные грузы. Следует отметить, что помимо экологической проблемы могут возникнуть и другие проблемы, например, такие как:

- логистическая - если состав сошел с рельс, следующим поездам приходится идти в обход, в некоторых случаях обхода может не быть;
- экономическая - связанна с издержками транспортной компании по решению экологической проблемы, потери части вагонов, локомотива, утрата части груза, временные издержки;
- инфраструктурная - повреждение строения железнодорожного пути, стыков, моста, обрушение тоннеля и др.

В данной работе рассматривается проблема схода состава с рельс, поскольку проблема является одной из самых опасных. В зависимости от масштаба происшествия сходы классифицируют на аварии и крушения. Согласно [1] за период с 2013 г. по 2016 г. в Российской Федерации имеется 262 протокола сходов с рельс вагонов как в грузовых поездах, так и в пассажирских,

без учета протоколов транспортных происшествий, классифицированных как крушения. Соответственно, при вычислении среднего числа дней без аварий выходит 4 дня, поэтому проблема представляет интерес для железнодорожных компаний.

В данной работе будет проведен анализ причин схода железнодорожного подвижного состава, а также будут построены предсказательные модели числа сошедших вагонов. Для достижения поставленных задач будут использованы методы теории вероятностей и математической статистики.

ГЛАВА 1

ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Постановка задачи машинного обучения

Одним из первых, кто дал определение предмету машинного обучения стал американский ученый Артур Самуэль. В 1959 году в своей работе [6], посвященной созданию искусственного интеллекта по игре в шашки с помощью алгоритма minimax, Артур Самуэль дал определение тому, что есть машинное обучение – процесс обучения, в результате которого компьютеры способны показывать поведение, которое в них явно не было заложено.

Более современное и точное определение дал Том Митчелл в 1998 году. Корректно поставленная задача обучения определяется следующим образом: говорят, что компьютерная программа обучается на основе опыта E (experience) по отношению к некоторому классу задач T (task) и меры качества P (performance), если качество у задачи из T, измеренное на основе P, улучшается с приобретением опыта E.

Большинство алгоритмов машинного обучения условно можно разбить на несколько классов: обучение с учителем, обучение без учителя, обучение с подкреплением, рекомендательные системы, а также частичное обучение, существуют и другие менее используемые классы алгоритмов.

В алгоритмах обучения с учителем подразумевается обучение на размеченных данных, то есть когда дана матрица, описывающая объекты с помощью признаков (матрица ”объекты-признаки”) и вектор ответов для каждого объекта. Таким образом методы обучения с учителем можно представлять как функциональную зависимость: на каждый набор признаков $x \in X$ есть ответы из Y такие, что $y : \ X \to Y$, где y – искомая зависимость.

Рассмотрим такой подход более подробно [3]. Пусть X – множество объектов, Y – множество ответов, $y : \ X \to Y$ – неизвестная зависимость, ставящая в соответствие объекты и ответы.

Пусть также нам известны:

- $\{x_1, \ldots, x_l\} \subset X$ – известное подмножество объектов;
- $y_i, \ \ \forall\, i = \overline{1,n}$ – известное множество результатов.

Ставится задача найти $a: \ X \to Y$ – искомый алгоритм, дающий минимальное расхождение с целевой функцией y.

Замечание: как правило множество объектов описывается с помощью признаков. Пусть есть n объектов, тогда под признаками объекта будем иметь в виду следующее отображения: $f_j: \ X \to D_j, \ \ \forall\, j = \overline{1,n}$. Признаки могут быть: количественными $D_j = \mathbb{R}$, бинарными $D_j = \{A, B\}$, номинальными $|D_j| = k < \infty$, упорядочено номинальными. Один объект может задаваться набором признаков разных типов.

Тогда любой объект $x \in X$ может быть описан вектором признаков $(f_1(x), \ldots, f_n(x))$. Следовательно, все объекты можно описать с помощью матрицы "объекты-признаки":

$$F = \begin{pmatrix} f_1(x_1) & f_2(x_1) & \ldots & f_n(x_1) \\ f_1(x_2) & f_2(x_2) & \ldots & f_n(x_2) \\ \ldots & \ldots & \ldots & \ldots \\ f_1(x_l) & f_2(x_l) & \ldots & f_n(x_l) \end{pmatrix}$$

Каждой строчке соответствуют правильные (ожидаемые) ответы, полученные в результате эксперимента, наблюдения, опроса и т.д. По типу представления множества ответов можно разбить на 3 класса:

- классификация: $Y = \{A, B\}, \ Y = \{A_1, \ldots, A_k\}, \ Y = \{A, B\}^k$;
- регрессия: $Y = \mathbb{R}$ (одномерный случай), $Y = \mathbb{R}^k$ (k-мерный случай);
- ранжирование: в отличие от классификации на множестве Y задано отношение частичного порядка.

Предсказательная модель $a(x)$ строится как параметрическое семейство функций над некоторой фиксированной функцией $g(x)$. Более формально модель $A = \{a(x) = g(x, \theta) | \theta \in \Theta\}$, где $g: \ X \times \Theta \to Y$ – фиксированная функция, Θ – множество допустимых параметров θ.

В обучении без учителя данные об ответах не известны, поэтому можно говорить лишь о том, как данные расположены друг относительно друга. Данные методы машинного обучения происходят без участия экспериментатора и применяется для обнаружения внутренних взаимосвязей. Обычно

так решаются задачи кластеризации, понижения размерности, визуализации данных.

Также используются методы обучения с подкреплением, когда набор данных дается из некоторого потока, а также рекомендательные системы.

Существуют задачи частичного обучения, которые занимают промежуточное положение между задачами регрессии и задачами кластеризации. Задача возникает когда только на части обучающей выборки даны ответы, а другая часть неразмечена, обычно такие задачи возникают при больших объемах данных, когда разметить всю выборку либо невозможно, либо очень дорого. При этом задачи частичного обучения не сводятся ни к классификации, ни к кластеризации.

1.2 Методология решения задач машинного обучения

Вне зависимости от метода машинного обучения задача состоит из 2-х этапов: обучение и применение. На первой стадии происходит построение оптимального алгоритма a – функция, дерево, набор инструкций и др. На второй стадии алгоритм выдает ответы для новых объектов.

Оптимальным алгоритмом будем называть такой алгоритм, который на большинстве объектов обучающей выборки дает правильные ответы или достаточно близкие ответы к ожидаемым. Для того чтобы это сделать нужно определять точность или расстояние между объектами, другими словами нужно задать метрику в пространстве объектов. Для этого вводится понятие функции потерь $\mathfrak{L}$ – величина ошибки алгоритма $a \in A$ на объекте $x \in X$:

- $\mathfrak{L}(a, x) = \begin{cases} 1, & \text{если } a(x) \neq y(x), \\ 0, & \text{иначе.} \end{cases}$ – индикатор ошибки для случая классификации;
- $\mathfrak{L}(a, x) = |a(x) - y(x)|^p$ – для случая регрессии.

На практике для случая регрессии обычно берут $p = 2$ т.к. при $p = 1$ возникает проблема с дифференцированием функции потерь. Однако, если в выборке присутствуют большие по модулю выбросы, то можно брать

$p = 1$, поскольку тогда модель будет меньше реагировать на большие отклонения.

Чтобы в целом оценить алгоритм берут сумму функций потерь по всем объектам из обучающей выборки, деленную на ее мощность, получившуюся величину называют эмпирическим риском (функционал качества алгоритма a на объектах X^l): $Q(a, X^l) = \frac{1}{l} \sum_{i=1}^{l} \mathfrak{L}(a, x_i)$.

Таким образом, задача обучения сводится к задаче оптимизации (минимизация функционала эмпирического риска на обучающей выборке): $\mu(X^l) = \arg\min_{a \in A} Q(a, X^l)$.

Замечание: иногда сумму делят не на l, а на $2 \cdot l$, с той целью, чтобы при дифференцировании функции эмпирического риска сократились некоторые коэффициенты (для $p = 2$).

Для решения задачи минимизации применяют различные численные методы. Например, метод наименьших квадратов (МНК).

1.2.1 CRISP-DM

Таким образом, любая задача машинного обучения: классификация, регрессия, кластеризация сводится к оптимизационной задаче, возможно с ограничениями. По этой причине были предприняты попытки по созданию единого алгоритма решения задач. Чтобы облегчить процесс решения задач, был разработан и предложен CRISP-DM (CRoss Industry Standard Process for Data Mining) – межотраслевой стандарт решения задач интеллектуального анализа данных. CRISP-DM – модель жизненного цикла исследования данных. Первая версия данного стандарта была принята в 1999 году. Стандарт призван формализовать схему решения

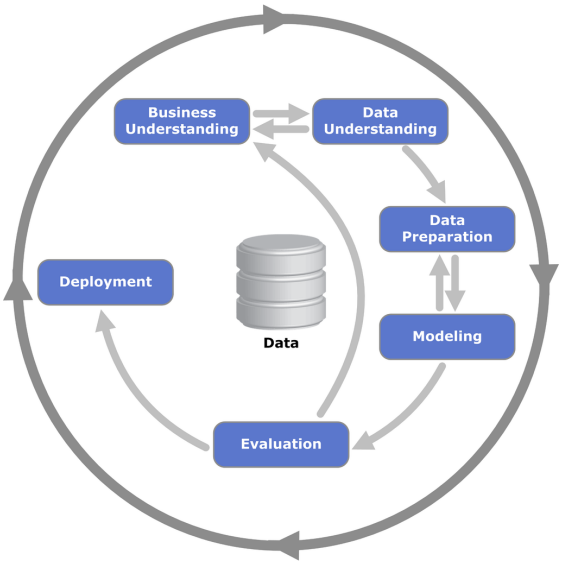

Рисунок 1.1 — Жизненный цикл исследования данных [7]

задач анализа данных. Предлагается алгоритм решения произвольной задачи анализа данных в 5 шагов, причем они могут замыкаться в цикле.

1. Вначале исследователь должен понять предметную область или сферу бизнеса;
2. Далее необходимо понять как собирались данные;
3. На следующем шаге нужно определить есть ли в данных шумы, пропуски, выбросы, все ли признаки несут полезную информацию, можно ли вычислить полезные признаки по уже имеющимся, данный этап можно назвать подготовкой данных;
4. После происходит моделирование или построение предсказательной модели;
5. Полученная модель оценивается с помощью выбранных метрик;
6. Если качество полученной модели удовлетворяет исследователя, модель внедряется в производственные процессы и эксплуатируется.

Помимо CRISP-DM существуют менее известные стандарты: My own, SEMMA и другие. На сайте [8] публикуются результаты опросов по популярности методологий анализа данных.

1.3 Методы оптимизации

Во многих задачах науки, экономики и бизнеса возникают проблемы нахождения экстремальных значений целевой функции. При этом на множество допустимых решений могут быть наложены ограничения. Такие задачи называются задачами оптимизации. Если есть ограничения, то говорят о задаче условной оптимизации, иначе безусловной оптимизации. Рассмотрим некоторые методы решения задач оптимизации.

1.3.1 Градиентный спуск

Метод градиентного спуска является наиболее часто используемым методом ввиду быстроты работы для большинства задач и многообразия модификаций метода (метод наискорейшего спуска, метод сопряженных гра-

диентов, метод Нестерова, метод стохастического спуска и многие другие) Для отыскания экстремальной точки в градиентном спуске используют итеративную формулу:

$$x := x - \alpha \nabla f(x)$$

Где $\nabla f(x)$ – градиент функции $f(x)$, в векторной форме градиент можно записать как $\nabla f(x) = \frac{\partial f(x)}{\partial x_1}\vec{i_1} + \frac{\partial f(x)}{\partial x_2}\vec{i_2} + \cdots + \frac{\partial f(x)}{\partial x_n}\vec{i_n}$, причем $\vec{i_1}, \vec{i_2}, \ldots, \vec{i_n}$ - единичные векторы. Направление градиента указывает на то направление из точки x, которое имеет наибольшую скорость роста функции из данной точки. Соответственно, антиградиент $-\nabla f(x)$ показывает направление наискорейшего убывания;

α – шаг градиента (в машинном обучении называют скоростью обучения). Данный параметр выбирается вручную, при этом, если α мал, то методу потребуется много итераций для сходимости, если же α большой, то метод градиентного спуска может начать расходится, то есть значение x будет отдалятся от точки экстремума. Существует условие Липшица [12], которое дает достаточное условие сходимости градиентного спуска: если $\exists L : \ \forall x, y \hookrightarrow ||\nabla f(x) - \nabla f(y)|| \leq L||x - y||$, то для $\forall \alpha < \frac{2}{L}$ гарантируется убывание функции $f(x)$. На практике обычно выбирают $\alpha = 10^{-2}$, либо еще меньше в зависимости от того, какая точность необходима.

Важной особенностью градиентных методов является выбор начального приближения x_0, так, если производится поиск минимума и функция имеет несколько минимумов, как на рисунке 1.2, то выбор x_0 будет определять найденный минимум. Например, если $x_0 = 2$, метод найдет минимум $f(x^*) = -0.18$, если $x_0 = 1$, то $f(x^*) = -0.809$, если же $x_0 = 1.5$, то метод не сможет найти ни один минимум функции, поскольку в точке $x = 1.5$ производная функции $f(x)$ равна 0, поэтому $\nabla f(x) = 0$ и формула градиентного спуска вырождается в $x := x$. Для того чтобы метод искал глобальный ми-

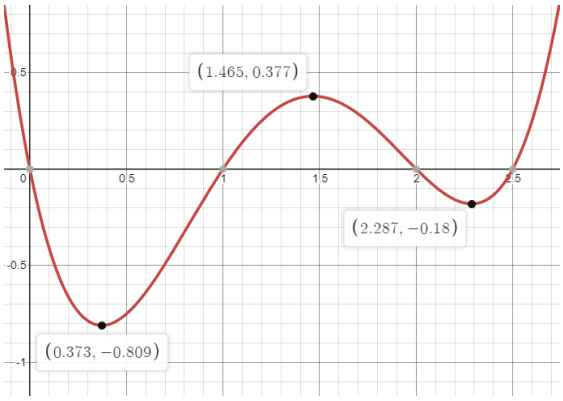

Рисунок 1.2 — Проблема много экстремальности функции

нимум можно запустить данный метод несколько раз из разных начальных точек. Выбрать начальные приближения можно как самому, так и выбрать их случайным образом из равномерного распределения на отрезке $[a, b]$. Для проверки того, что был найден глобальный минимум можно запустить метод из точек $x^* - x_L,\ x^* + x_L$, где x^* – найденный минимум, x_L – точка с большими по модулю координатами. Если метод не сойдется к тому же решению, то необходимо сравнить 2 минимума и выбрать наименьший.

При поиске экстремума в овражных функциях градиентный спуск сходится медленно, причем если число аргументов функции велико, то довольно часто можно встретить овражные области. Для решения данной проблемы используют модификации градиентного спуска, так называемые, овражные методы.

Для ускорения сходимости градиентного спуска применяют различные техники. Можно определять α на каждом шаге по следующей формуле: $\alpha_k = \arg\min\limits_{\alpha \in [0,\infty)} f(x^k - \alpha f'(x_k))$ метод с таким выбором градиентного шага носит называние метода наискорейшего спуска, также можно уменьшать параметр α на каждом шаге. Идея заключается в том, что после каждого шага расстояние до экстремума уменьшается, следовательно, нужно уменьшить градиентный шаг.

1.3.2 Инерционный градиентный спуск

Метод известен с середины 20 века и изначально носил название метод тяжелого шарика. Идея данного метода – добавить в формулу градиентного спуска свойство инерционности, то есть чтобы на каждом следующем шаге аргумент x_{k+1} зависел не только от значения антиградиента, но и также от значения предыдущего шага. Для этого предлагается добавить в формулу градиентного спуска следующее слагаемое $\beta(x_k - x_{k-1})$, где β – коэффициент инерции, также как и α является гиперпараметром, то есть задается вручную. Как правило коэффициент инерции берется немного меньше единицы.

Все семейство инерционных градиентных методов можно описать сле-

дующей формулой:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) + \beta_k (x_k - x_{k-1})$$

Добавление нового слагаемого в метод градиентного спуска не меняет асимптотической сложности алгоритма, при этом, если выбрать оптимальные значения в паре (α_k, β_k), можно добится ускорения метода на порядок.

1.3.3 Стохастический градиентный спуск

Идея метода заключается в использовании вместо градиента функции $\nabla f(x)$ другую функцию (случайный процесс) $g(x, \theta)$ такую, что математическое ожидание $E[g(x, \theta)] = \nabla f(x)$, где θ – случайная величина. Метод стохастического градиента можно описать следующей формулой:

$$x_{k+1} = x_k - \alpha_k g(x_k, \theta_k)$$

Если x – вектор с большим количеством компонент, то вычисление градиента может происходить продолжительное время. Поэтому если использовать метод стохастического градиента и в качестве θ взять случайный индекс у x, то есть x_i, где i – случайная величина, то вычисление градиента сведется к вычислению частной производной по аргументу со случайным индексом. За счет этого происходит существенное ускорение сходимости, это называется процедурой Роббинса-Монро и Кифера–Вулфовица, представленные в 1951 и 1952 годах соответственно [13]. В настоящее время данный метод активно применяется в алгоритмах машинного обучения и в нейронных сетях, где количество признаков у объектов может быть велико. Следует сказать, что данный метод реализован в крупных библиотеках машинного обучения: TensorFlow, PyTorch. Метод используется для обучения моделей с большим и сверх большим объемом данных из-за того, что даже на небольшой подвыборке объектов модель может хорошо обучиться.

1.3.4 Нормальное уравнение (Normal Equation)

Нормальное уравнение позволяет найти экстремум функционала аналитически, без необходимости применять итеративный подход. Пусть функционал задан следующим образом: $\mathfrak{L}(\theta_1, \theta_2, \ldots, \theta_m) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$ поставим задачу минимизации этого функционала по всем параметрам θ, при условии, что известны m пар $(x^{(i)}, y^{(i)})$. Составим из $x^{(i)}$ матрицу: $X = (x^{(1)}, x^{(2)}, \cdots, x^{(m)})$, тогда $\theta = (X^T X)^{-1} X^T y$.

В отличии от градиентных методов в данном методе не нужно проводить итерационную процедуру, а также в методе отсутствуют гиперпараметры. Однако для вычисления ответа нужно вычислить обратную матрицу порядка $n \times n$, алгоритм вычисления обратной матрицы работает за $O(n^3)$ по времени, поэтому для объектов с большим числом признаков, например $n = 10^6$ потребуется $C \cdot 10^{18}$ операций, где C – некоторая константа из алгоритма обратной матрицы. Современные компьютеры могут обрабатывать порядка 10^9 операций в секунду, учитывая нужное количество операций алгоритм не успеет за разумное время вычислить минимум функции. Поэтому область применимости метода – объекты с небольшим числом параметров $(< 10^4)$. На объектах с большим числом признаков хорошо работает стохастический градиентный спуск.

1.4 Регрессионный анализ

Задача регрессионного анализа состоит в определении наиболее оптимальной функции $f(x)$, которая бы наилучшим образом восстанавливала закономерность во входных парах данных (x_i, y_i). Тогда можно записать следующее соотношение: $y = f(x) + \varepsilon$, где $f(x)$ – функция, предсказывающая поведение y, ε – случайная величина с нулевым математическим ожиданием. В большинстве задач полагается, что ε имеет нормальное распределение.

Задача исследователя состоит в определении вида функции $f(x)$. После происходит параметризация данной функции, выбирается функция штра-

фа за отклонение от целевого значения, например, сумма квадратов отклонений. Нахождение оптимальных параметров θ означает минимизацию функции потерь. Таким образом задача регрессии сводится к задаче оптимизации. Рассмотрим конкретные виды аппроксимирующих моделей.

1.4.1 Линейные модели

Иногда, искомую зависимость можно хорошо аппроксимировать линейными моделями, например это могут быть: прогноз стоимости дома или квартиры, классификация типа опухоли, предсказание оттока клиентов, прогноз оклада по описанию вакансии, некоторые физические законы (Гука, Ома, Паскаля).

Линейная регрессия

Модель линейном регрессии называется так из-за того, что функция прогноза выглядит как линейная комбинация компонент объекта и компонент вектора параметров. Пусть $f(x, \theta) = \sum\limits_{i=1}^{N} x_i\theta_i = (\theta, x)$. Чтобы записать последнее равенство в вектор x добавляют фиктивную компоненту, равную единице для того, чтобы размерности вектора параметров и вектора объекта совпадали. В качестве функции штрафа можно взять, например $SSE = \sum\limits_{i=1}^{N}(y_i - f(x_i, \theta))^2 = \sum\limits_{i=1}^{N}(y_i - (\theta, x_i))^2$. Тогда, определив, с помощью методов теории оптимизации, вектор θ, будет найдена искомая функция.

В общем случае можно считать что модель $f(x, \theta)$ задает гиперплоскость в n-мерном пространстве и чем больше расстояние объекта до гиперплоскости тем больше штраф.

Если среди признаков есть линейно зависимые, то модель теряет свою точность на тестовой выборке, хотя на обучающей выборке можно достичь максимальной точности. Данная проблема называется мультиколлинеарностью линейной модели. При этом возможны 2 случая: функциональная зависимость, когда набор признаков однозначно и точно определяет другой признак и частичную мультиколлинеарность, когда линейная комбинация

части признаков сильно коррелирует с другим признаком. При функциональной зависимости вектор параметров определяется неоднозначно, следовательно появляется степень свободы при выборе параметров и можно подобрать их так, чтобы на тестовой выборке они давали завышенные метрики качества. Частичная мультиколлинеарность приводит к неустойчивости оценок.

Существует несколько способов борьбы с мультиколлинеарностью:

- Метод главных компонент – позволяет уменьшить размерность пространства признаков и следовательно избавиться от их коррелированности;
- гребневая регрессия – добавим в функционал эмпирического риска штрафное слагаемое за большие по модулю значения вектора параметров: $Q = \frac{1}{N}\mathfrak{L}(\theta, x_i, y_i, f) + \frac{\tau}{2}||\theta||^2$, где τ – коэффициент регуляризации. При минимизации Q будет минимизироваться как сумма функций потерь, так и квадрат нормы вектора параметров. Подбор параметра τ можно сделать вручную, посмотрев на качество метрик на текстовой выборке или определить по критерию скользящего контроля;
- метод LASSO – (Least Absolute Shrinkage and Selection Operator) в отличие от гребневой регрессии, в методе LASSO берется сумма модулей компонент вектора параметров: $Q = \frac{1}{N}\mathfrak{L}(\theta, x_i, y_i, f) + \tau \sum\limits_{i=1}^{N} |\theta_i|$, также некоторые значения θ_j могут стать в точности нулем, следовательно соответствующий признак больше не будет учитываться. Таким образом в данном методе происходит отбор признаков.

Гребневая регрессия и метод LASSO позволяют ограничить вектор параметров и тем самым избежать проблемы мультиколлинеарности, при этом в методе LASSO можно провести селекцию признаков то есть убрать часть из них.

Говорят, что регуляризация приводит к сокращению размерности пространства, хотя само пространство остается той же размерности, сокращается эффективная размерность, поскольку мы накладываем на вектор параметров ограничение.

Полиномиальная регрессия

Линейная регрессия требует чтобы между целевой переменной и переменными признаков была линейная зависимость. На практике возникают случаи, когда зависимость между данными невозможно описать линейным образом, для этого рассмотрим модель регрессии с полиномом. Определим модель следующего вида: $f(x, \theta) = \sum_{i=0}^{k} \theta_i x^i = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_k x^k$. Тогда f задает полиномиальную регрессию. При этом x может быть как скалярной величиной, так и векторной. Такая модель все еще остается линейной так как степени при весах остаются первого порядка. Следует отметить, что полиномиальная регрессия может привести к эффекту переобучения, если наибольшая степень k будет высокой, поскольку выразительная способность (число степеней свободы) модели будет велико, то модель сможет подстроиться под имеющие данные и не будет обладать обобщающей способностью.

Пуассоновская регрессия

Пуассоновская регрессия применяется, когда целевая переменная имеет пуассоновское распределение или в основе которой лежат события, счетчиком которых она является. При этом частота возникновения событий не обязательно стационарна, а может меняться со временем. Пуассоновское распределение имеют, например количество звонков в колл-центр за период времени или число вакцинированных людей за период времени. Для описания частотности событий введем обозначение: λ, которое возможно зависит от времени.

Пуассоновская регрессия основывается на Пуассоновском распределении, которое имеет следующее распределение вероятности: $p(k) = \frac{e^{-\lambda t}(\lambda t)^k}{k!}$ – вероятность получения k событий за время t с интенсивностью потока λ. Математическое ожидание распределения Пуассона численно равно интенсивности потока λ, следовательно для интервала времени $[0, t]$ можно ожидать λt событий.

Если целевая переменная определяется распределением Пуассона и нет

оснований говорить о нестационарности потока событий, в качестве оценки можно использовать значение интенсивности потока λ.

Если же λ может изменяться от одного события к другому, то предполагается, что λ зависит от объясняющих признаков по некоторому закону. Задача регрессии в таком случае состоит в приближении значений целевой переменной y к $\lambda(x)$. Для этого функция гипотезы $\lambda(x)$ параметризуется. Как правило [1] $\lambda(x, \theta) = e^{x \cdot \theta}$, где θ – вектор обучаемых параметров, x – вектор признаков, описывающих объект.

Таким образом вероятность обнаружить y_i событий для объекта x_i равна условной вероятности $p(y_i|x_i;\theta) = \dfrac{e^{-\lambda_i(x,\theta)}\lambda_i^{y_i}(x,\theta)}{y_i!}$.

Для нахождения θ воспользуемся методом максимального правдоподобия (ММП). Функция правдоподобия имеет следующий вид: $L(\theta|X,y) = \prod\limits_{i=1}^{N} p(y_i|x_i;\theta) = \prod\limits_{i=1}^{N} \dfrac{e^{-\lambda_i(x,\theta)}\lambda_i^{y_i}(x,\theta)}{y_i!}$.

Максимизируем полученное выражение по θ (переменная λ_i зависит от θ), для удобства будем максимизировать не $L(\theta|X,y)$, а $l(\theta|X,y) = \ln L(\theta|X,y)$ на значение максимума это не повлияет, поскольку логарифмическая функция монотонно возрастающая. Данное преобразование удобно тем, что при вычислении производной удобнее считать производную суммы, чем производную произведения. По свойству логарифма произведения: $l = \sum\limits_{i=1}^{N} \ln\left(\dfrac{e^{-\lambda_i(x,\theta)}\lambda_i^{y_i}(x,\theta)}{y_i!}\right) = \sum\limits_{i=1}^{N} (-\lambda_i(x,\theta) + y_i \ln(\lambda_i(x,\theta)) - \ln(y_i!))$. Функция l называется логарифмической функцией правдоподобия. Найдя максимум этой функции по θ одним из численных методов, получим такие значения θ, которые при подстановке в исходную функцию правдоподобия максимизируют ее.

В данном методе метрикой качества является значение функции максимального правдоподобия. Та модель лучше, у которой большее значение функции, учитывая, что эффекта переобучения нет, либо его влиянием можно пренебречь.

Когда модель обучилась, то есть подобран вектор параметров для получения прогноза количества событий для объекта $x_{predict}$ вычисляется значение $\lambda(x_{predict}, \theta)$, далее искомая величина $y_{predict}$ вычисляется по следующей формуле: $y_{predict} = \arg\max\limits_{k \in \mathbb{Z}^+} p(k|x_{predict}, \theta)$.

Помимо Пуассоновской регрессии существуют и другие модели, основанные на подсчетах: модель Пуассона с нулевым завышением, отрицательная биномиальная регрессия, обобщенная Пуассоновская регрессия [5].

Геометрическая регрессия

Геометрическая регрессия применяется, когда целевая функция имеет геометрическое распределение. Функция вероятности первого 'успешного' испытания в серии Бернулли с n повторениями и вероятностью 'успеха' p имеет вид: $P(X = n) = (1 - p)^n p$, где p – некоторая функция, выбираемая исследователем, причем на данных из набора данная функция должна принимать значения от нуля до единицы, из-за вероятностных ограничений. Для нахождения θ воспользуемся ММП. Функция правдоподобия имеет следующий вид: $L(\theta|X, y) = \prod_{i=1}^{N}(1 - p(x_i, \theta))^{y_i} p(x_i, \theta)$.

Используя численные методы вычислим максимум полученной функции, однако для удобства будем искать максимум не $L(\theta|X, y)$, а $l(\theta|X, y) = \ln L(\theta|X, y)$. Тогда функция логарифмического правдоподобия будет иметь следующий вид: $l(\theta|X, y) = \sum_{i=1}^{N}(y_i \ln(1 - p(x_i, \theta)) + \ln(p(x_i, \theta)))$.

Как и в модели пуассоновской регрессии метрикой качества является значение функции максимального правдоподобия. Та модель лучше, у которой большее значение функции, учитывая, что эффекта переобучения нет, либо его влиянием можно пренебречь.

Когда модель обучилась, то есть подобран вектор параметров для получения прогноза количества событий для объекта $x_{predict}$ вычисляется значение $p(x_{predict}, \theta)$, далее искомую величину $y_{predict}$ можно вычислить по следующей формуле: $y_{predict} = \dfrac{1}{p(x_{predict}, \theta)}$.

Проблема переобучения и метрики качества

Проблема переобучения или эффект переподгонки – одна из самых частых проблем машинного обучения. Данная проблема возникает, когда модель очень сильно подстроилась под объекты на обучающей выборке и

из-за этого потеряла обобщающую способность. Эффект можно обнаружить, если сопоставить функции эмпирического риска, вычисленные для обучающей и тестовой выборок, тогда на значение функционала на обучающей выборке будет близко к нулю, а значение на тестовой выборке будет существенно больше.

На рисунке 1.3 изображены две кривые, соответствующие двум модели, решающие задачу классификации. Видно, что черная разделительная кривая классифицирует данные более общо, чем зеленая кривая. Можно говорить, что модель с зеленой разделительной кривой переобучена. Эффект связан с тем, что в выборке существуют случайные отклонения, которые могут быть обусловленны ошибкой измерения или погрешностью, при этом, если модель обладает высокой выразительной способностью, она настраивается на этих данных, что приводит к ухудшению качества на тестовой выборке.

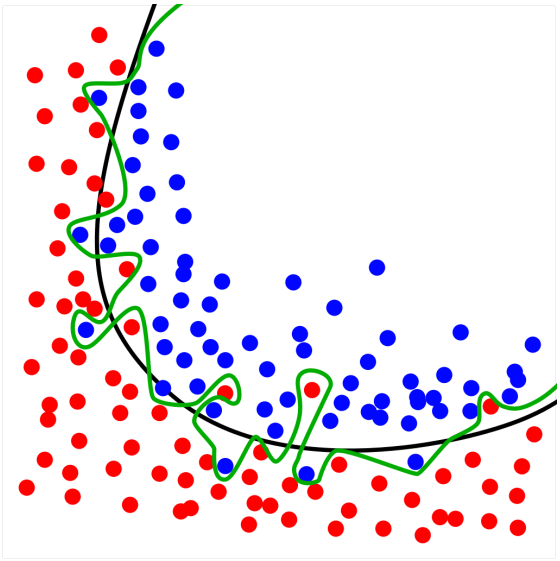

Рисунок 1.3 — Пример переобучения модели [15]

Для того чтобы измерять эффект переобучения считают функционалы, измеряющие качество построенной модели на данных на которых модель не обучалась.

Рассмотрим 3 таких метода:

- Отложенная выборка (hold-out) – деление всей выборки на 2 части (обучающая и валидационная). Недостатком данного функционала качества является то, что можно неудачно сделать разбиение так, что оценка будет смещенной, субъективной. Важным параметром данного метода является отношение размеров разбиений, если обучающая выборка мала, оценка качества будет пессимистической, если же валидационная выборка мала, оценка будет неточной. В большинстве случаев обучающую выборку берут в размере 70-80% от общего размера;

- Скользящий контроль (leave-one-out) – по очереди будем выбирать один объект из выборки, на оставшейся части будем обучаться, а на выбранном объекте тестировать. После результаты тестирований усредняются. Таким образом невозможно получить смещенную выборку как в hold-out, однако обучение нужно проводить столько же раз, сколько объектов в обучающей выборке;
- Кросс-валидация (cross-validation) – выберем число групп N, примерно одинакового размера, равномерно распределим объекты выборки по группам. После будем проводить обучение на $N - 1$ группе, а тестирование на оставшейся. Результаты усредняются. При выборе малого числа групп оценки будут пессимистичными, но при этом точными. При большом числе групп оценки будут несмещенными, но с большой дисперсией. Обычно выбирают число групп от 5 до 10.

Данные методы основываются на предположении об одинаковости распределении в группах данных и их независимости. Если данные зависимы или их можно разбить на группы так, чтобы в разных группах распределение были разными, возникает риск переобучения.

Также потеря качества может возникнуть при неправильном проведении отбора признаков или понижения размерности. Корректно в начале разбить выборку на обучающую и валидационную, а отбор признаков или понижение размерностей необходимо проводить не над всей выборкой, а только над обучающей. После проводится проверка качества на валидационной выборке. Было бы ошибочно проводить отбор признаков или понижение размерности на всей выборке до разбиения.

1.4.2 Нелинейные модели

Нелинейные модели регрессии – такие методы регрессионного анализа, в котором в моделирующей функции параметры модели входят нелинейным образом.

Решающие деревья

Данный метод исходно разрабатывался для решения задач классификации, однако в настоящее время существует алгоритм CART, расширяющий класс решаемых задач методом решающих деревьев на случай регрессии, поэтому данный метод помещен в раздел "Регрессионный анализ".

Решающие деревья – попытка формализовать человеческое мышление при принятии решений. Решающее дерево можно приближенно проиллюстрировать на примере работы врача: врач задает пациенту уточняющие вопросы, исходя из его ответов происходит спуск по дереву на уровень ниже. Вершина, в которую нужно перейти определяется ответом на вопрос. Если у вершины нет дочерних вершин, она листовая в данном примере листовые вершины – диагноз больного или совет. Таким образом за конечное количество вопросов можно дойти от корневой вершины до листовой.

Все вершины дерева можно разделить на листовые и внутренние. Любая внутренняя вершина v содержит предикат $\beta_v : X \to A$, для случая бинарного решающего дерева $A = \{0, 1\}$. Любая листовая вершина v содержит метку класса c_v.

Для построения решающего дерева используется алгоритм Induction of Decision Tree (ID3). На вход алгоритм может принимать часть выборки U.

```
def LearnID3(U):
  if все объекты из U лежат в одном классе:
    return (новый лист v, c_v = c)
  найти предикат с максимальной информативностью beta = argmax I(beta,U)
  разбиваем выборку на U0 и U1 по предикату beta
  if len(U0) == 0 or len(U1) == 0: # не смогли найти информативный предикат
    return (новый лист v, c_v = мажоритарный_класс(U))
  создать новую вершину v
  построить левое  дерево: L_v = LearnID3(U0)
  построить правое дерево: R_v = LearnID3(U1)
  return v
```

Рассмотрим случай, когда при разбиении входной выборки U по найденному предикату beta один из классов U0 или U1 оказывается пустым. В этом случае критерий информативности I не смог разделить выборку на 2 класса, хотя в выборке были представители как одного, так и другого классов. В этом случае образуем вершину по данному предикату и отнесем ее

к тому классу, которых больше в выборке. При этом в этой вершине будут ошибки.

Для поиска предиката с максимальной информативностью используются различные критерии ветвления:

- критерий Джини: $I(\beta, X) = |\{(x_i, x_j) : \quad y_i = y_j \ \ \& \ \ \beta(x_i) = \beta(x_j)\}|$ Из определения следует, что критерий Джини позволяет определить число вершин из одного класса, которые были классифицированы предикатом одинаково. При нормировании критерия на число вершин критерий будет показывать частоту объединения вершин из одинаковых классов.
- D-критерий Донского: $I(\beta, X) = |\{(x_i, x_j) : \quad y_i \neq y_j \ \ \& \ \ \beta(x_i) \neq \beta(x_j)\}|$ позволяет определить число вершин из разных классов, которые были классифицированы предикатом в разные ветви дерева. При нормировании критерия на число вершин критерий будет показывать частоту разделения вершин из разных классов.

Для оптимальной обработки пропусков в данных на стадии обучения для каждой вершины считают частоты прохождения объекта в левую и правую ветви. На этапе классификации, если для объекта невозможно вычислить предикат, он отправляется в оба поддерева с определенными весами, после ответ усредняется по ним и выбирается наиболее вероятный класс.

Для избежания эффекта переобучения и улучшения качества алгоритма можно ограничить максимальную грубину дерева, процедура называется pruning и реализована в алгоритме C4.5, предложенным Джоном Квинланом.

Рассмотрим обобщение алгоритма ID3 для случая регрессии, для этого рассмотрим метод CART (Classification And Regression Trees).

Будем считать, что $Y = \mathbb{R}, c_v = \mathbb{R}$. Пусть U_v – множество объектов, дошедших до вершины v. Тогда терминальные вершины c_v определим как среднее значение по всем достигшим вершины v объектам: $c_v = \frac{1}{|U_v|} \sum\limits_{x_i \in U_v} y_i$.

В качетсве критерия ветвления возьмем среднеквадратичную ошибку:

$I(\beta, U_v) = \sum\limits_{x_i \in U_v} (\hat{y}_i(\beta) - y_i)^2$, где $\hat{y}_i(\beta) = \beta(x_i)\hat{y}_i(U_{v1}) + (1 - \beta(x_i))\hat{y}_i(U_{v0})$ – прогноз β и разбиения $U = U_{v0} \sqcup U_{v1}$.

При обнаружении эффекта переобучения можно воспользоваться методом MCMP (Minimal Cost-Complexity Pruning). Идея метода заключается в регуляризации количества терминальных вершин, для этого составляется критерий из двух частей: среднеквадратичная ошибка и $\alpha|V_{terminal}|$, где $|V_{terminal}|$ – количество терминальных вершин, α – коэффициент регуляризации. При увеличении α дерево решений упрощается.

Градиентный бустинг

Если качество базовых алгоритмов машинного обучения для решения задачи не удовлетворяет исследователя, можно воспользоваться композицией данных алгоритмов, качество которой будет выше отдельно взятого алгоритма данной композиции. Градиентный бустинг – один из таких методов, объединяющий в себе несколько алгоритмов. Алгоритмы в градиентном бустинге обучаются независимо друг от друга. Каждому алгоритму $a_i(x)$ сопоставлен коэффициент w_i. Пусть всего алгоритмов T, тогда линейной композицией будем называть взвешенную сумму: $a(x) = C\left(\sum\limits_{i=1}^{T} w_i a_i(x)\right)$, где $C : \mathbb{R} \to Y$ – решающее правило.

Решающее правило необходимо для обобщения градиентного бустинга на задачи классификации. Так, если решается задача регрессии, то решающее правило не учитывается $C(x) = x$, для задачи классификации на 2 класса $C(x) = sign(x)$.

Выберем произвольную функцию потерь $\mathcal{L}(a(x), y)$ и пусть $T - 1$ алгоритм уже обучен и соответствующие им коэффициенты w_i определены, тогда запишем функционал эмпирического риска как функцию от нового алгоритма a и веса w: $Q(w, a) = \sum\limits_{i=1}^{N} \mathcal{L}\left(\sum\limits_{t=1}^{T-1} w_t a_t(x_i) + w a(x_i), y_i\right) \to \min\limits_{w,a}$.

Для решения такой задачи оптимизации вначале находится базовый алгоритм $a(x)$, после решается задача одномерной оптимизации поиска w. После решения задачи функционал эмпирического риска обновляется с учетом новой пары $(a(x), w)$. Это и есть алгоритм градиентного бустинга.

ГЛАВА 2

ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ ДАННЫХ

Анализ данных будет проведен при помощи языка программирования Python3. Данный язык был выбран по нескольким причинам:

- большое количество модулей для анализа данных;
- подробная документация как языка, так и его модулей;
- удобство и простота работы с данными в форматах csv, xlsx;
- множество встроенных функций и выразительность языка.

2.1 Обзор признаков

В данном наборе данных представлена информация о случаях схода составов с рельс по причине излома боковой рамы вагона.

Набор данных содержит следующие признаки:

- дата – дата происшествия;
- кол-во вагонов – кол-во вагонов в составе (без локомотивов);
- максимальное число вагонов в сходе – кол-во вагонов с конца до сшедшего поезда;
- общее кол-во вагонов – кол-во вагонов в составе (с локомотивами);
- кол-во сошедших вагонов – целевая переменная; кол-во вагонов, сшедших с рельс;
- скорость – средняя скорость состава;
- вес – масса состава, включая груз, выраженная в тоннах;
- загрузка – отношение текущего веса к максимальному весу;
- стрелочный перевод – наличие стрелочного перевода в месте схода;
- кривизна – обратная величина к радиусу кривизны;
- профиль пути – знак величины определяет направление (положительный, если подъем и отрицательный, если спуск);
- Режим движения – дискретная величина, обозначающая тягу, выбег, торможение;

Вывод первых пяти записей из набора, таблица разделена на две части:

№	Дата	Кол-во вагонов	Макс. число вагонов в сходе	Общее кол-во вагонов	Кол-во сшедших вагонов
1	2013-01-08	56.0	19.0	58.0	1
2	2013-01-09	60.0	25.0	62.0	1
3	2013-01-10	60.0	4.0	64.0	1
4	2013-01-12	66.0	63.0	68.0	21
5	2013-01-19	67.0	34.0	69.0	1

№	Скорость	Вес	Загрузка	Стрелочный перевод	Кривизна	Профиль пути	Режим движения
1	57.0	3402.0	0.547101	0	0.000000	0.0007	NaN
2	72.0	4082.0	0.652657	0	0.000000	0.0009	NaN
3	15.0	4420.0	0.734300	0	0.001639	NaN	3.0
4	67.0	5699.0	0.918094	0	0.002326	0.0060	NaN
5	69.0	5854.0	0.932944	0	0.000000	0.0006	2.0

Таблица 2.1 — первые 5 записей в наборе данных

2.2 Описательные статистики

Мощность выборки равна 56, при этом в записях содержится существенное количество пропусков.

Вывод описательных статистик для набора данных (примечание: признак ’Дата’ исключен):

	Кол-во вагонов	Макс. число вагонов в сходе	Общее кол-во вагонов	Кол-во сшедших вагонов	Скорость
count	54.000000	51.000000	54.000000	56.000000	53.000000
mean	63.870370	37.137255	66.407407	3.875000	49.150943
std	9.790342	21.543463	10.053665	6.081455	18.450971
min	24.000000	2.000000	26.000000	1.000000	9.000000
25%	60.000000	17.500000	62.500000	1.000000	35.000000
50%	66.000000	43.000000	68.000000	1.000000	51.000000
75%	68.000000	56.500000	71.750000	2.250000	64.000000
max	96.000000	72.000000	100.000000	26.000000	78.000000

	Вес	Загрузка	Стрелочный перевод	Кривизна	Профиль пути	Режим движения
count	54.000000	54.000000	56.000000	46.000000	44.000000	33.000000
mean	5126.629630	0.817678	0.107143	0.000806	-0.000384	1.666667
std	1438.743887	0.243936	0.312094	0.001171	0.005689	0.777282
min	998.000000	0.179710	0.000000	0.000000	-0.011500	1.000000
25%	4155.500000	0.690451	0.000000	0.000000	-0.004750	1.000000
50%	5722.000000	0.925519	0.000000	0.000000	0.000000	1.000000
75%	6010.250000	0.995586	0.000000	0.001479	0.001875	2.000000
max	8806.000000	1.076087	1.000000	0.005000	0.010900	3.000000

Таблица 2.2 — описательные статистики

Следует отметить, что такие признаки как: ’Режим движения’, ’Кривизна’, ’Профиль пути’ имеют наибольшее количество пропусков в данных: 41%, 17%, 21% пропусков соответственно.

2.3 Корреляция признаков

Построим матрицу корреляции признаков:

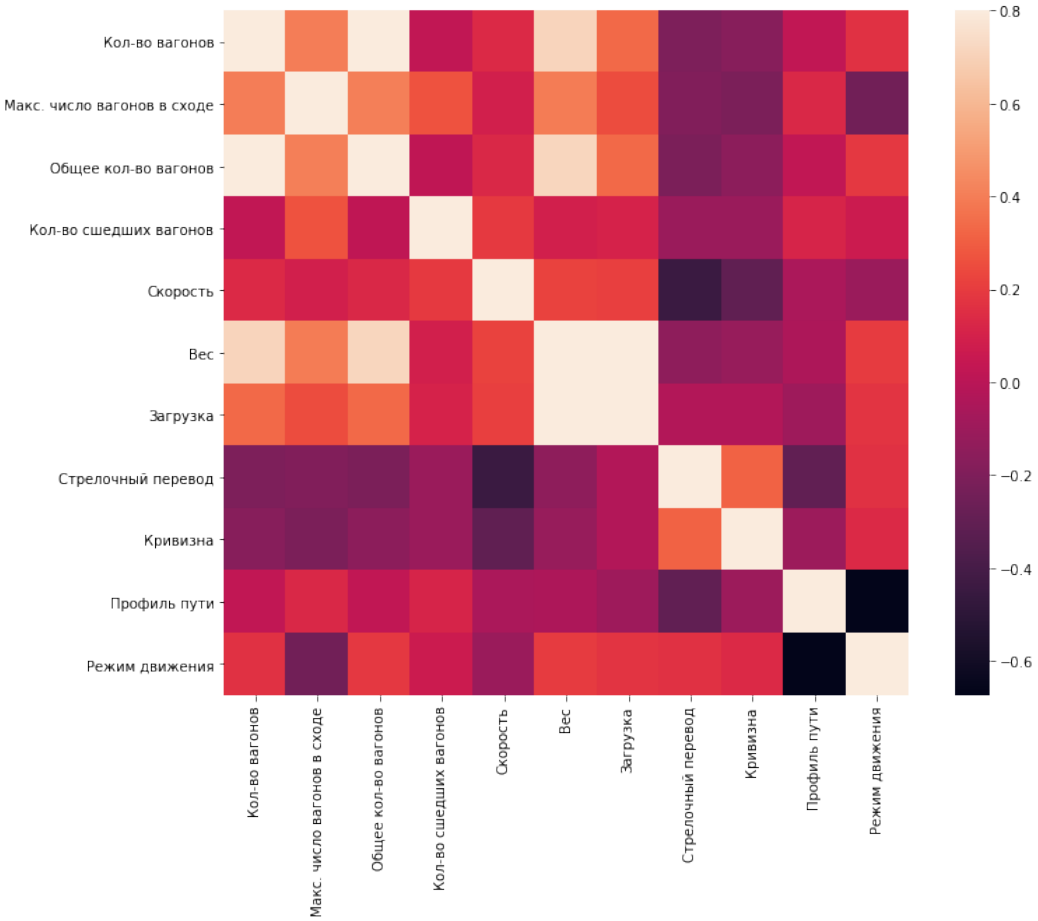

Рисунок 2.1 — Корреляция признаков

Из рисунка 2.1 видно, что признаки ’Количество вогонов’ и ’Общее число вагонов’ имеют сильную корреляцию, что можно объяснить тем, что общее число вагонов равно количеству вагонов плюс количество локомотивов, при этом количество локомотивов в составах как правило равняется двум, следовательно получается функциональная зависимость. Также можно заметить, что признаки ’Вес’ и ’Загрузка’ сильно коррелируют. Менее

сильная корреляция наблюдается у признаков ’Вес’ и ’Общее число вагонов’. Заметим, что у ’Профиль пути’ и ’Режим движения’ наблюдается сильная обратная корреляция. Многие зависимости можно нетрудно объяснить: чем больше вагонов в составе, тем больше вес, чем больший вес, тем, как правило, большая загруженность. Таким образом, можно прийти к выводу, что в данные в наборе избыточны, поскольку несколько признаков несут одинаковое количество информации. Поэтому эти зависимости приводят к проблеме мультиколлинеарности, что приведет к эффекту переобучения в линейных моделях. Для решения данной проблемы нужно исключить избыточные признаки.

2.4 Пропуски в данных

Из таблицы 2.2 по строке ’count’ видно, что в последних трёх признаках присутствуют пропуски в данных. Для решения проблемы пропусков в данных существует ряд методов:

- удалить все записи из используемого в модели подмножества признаков, в которых есть хотя бы одно пустое поле. При использовании этого метода для данного набора данных существует риск того, что оставшегося множества записей не хватит для получения приемлемого качества построенной модели;
- заменить пропуски на средние значение по признаку;
- заменить пропуски на медианные значение по признаку. В отличие от среднего значения замена на медианное позволяет избежать сильного влияния выбросов на итоговое значение;
- заменить пропуски на крайне большие значения, если в качестве модели так или иначе используется структура данных дерево. Таким образом, все записи, имеющие пропуски будут выделены в отдельную ветку дерева;
- заменить пропуски нулевыми значениями, таким образом для логистической регрессии пропуски в данных не будут влиять на предсказанные значения.

При решении данной задачи будет использован метод удаления записей с

пропусками. Замена значений на медианное, среднее для данного набора данных может оказаться не оптимальным выбором, поскольку мощность выборки не велика.

2.5 Экстремальные значения

Для поиска выбросов построим графики, изображающие отношения между парами признаков, и гистограммы распределений.

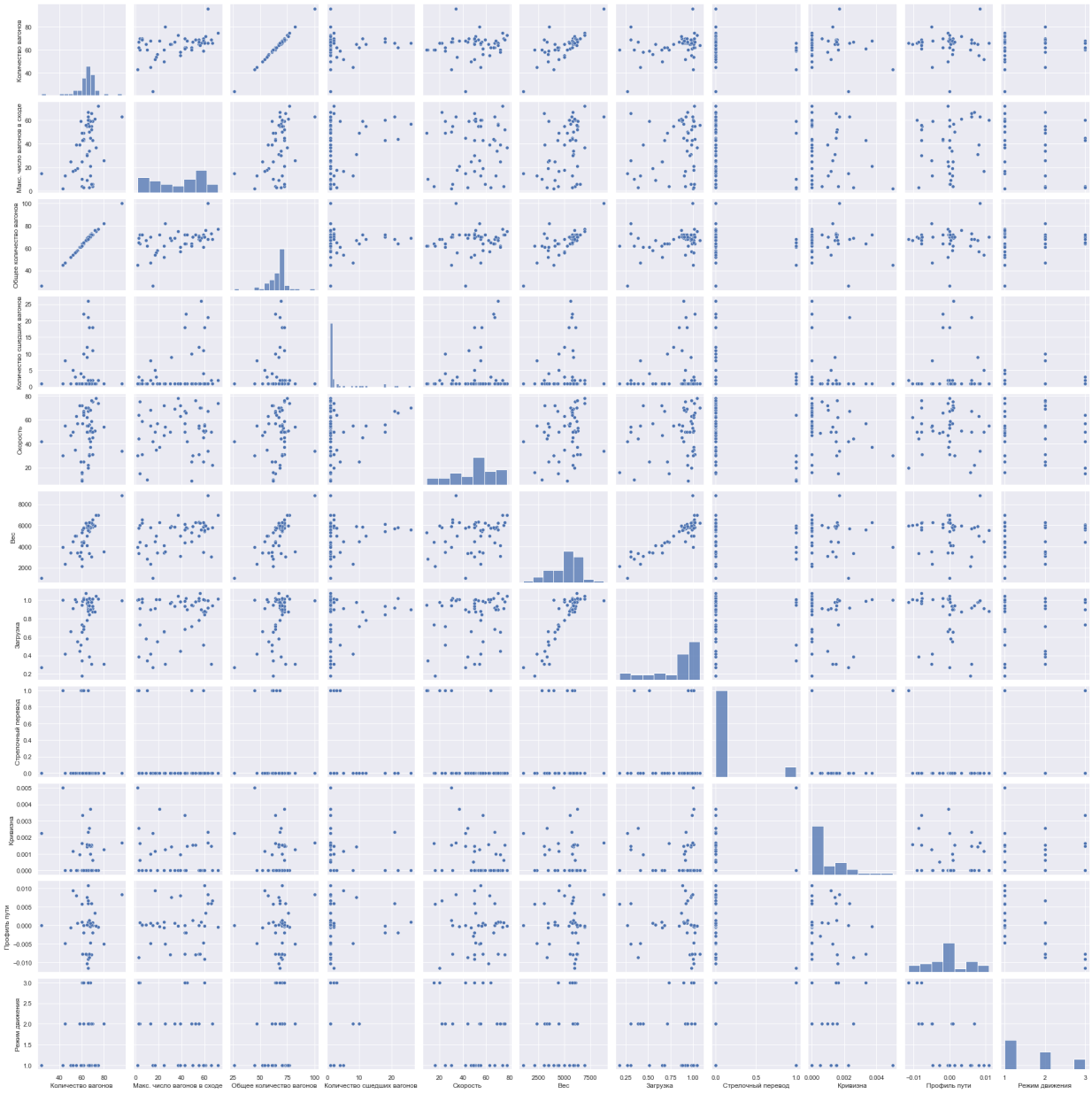

Рисунок 2.2 — Пары признаков

Изучив таблицу с описательными статистиками 2.2, а также при детальном рассмотрении графиков пар признаков и распределений на рисунке 2.2 сильных выбросов в данных обнаружить не удалось.

2.6 Оценка плотности распределения

Построим ядерную оценку плотности распределения признака количества сошедших вагонов. В качестве ядра было взято стандартное гауссово ядро с шириной окна парзена $h = 0.5$.

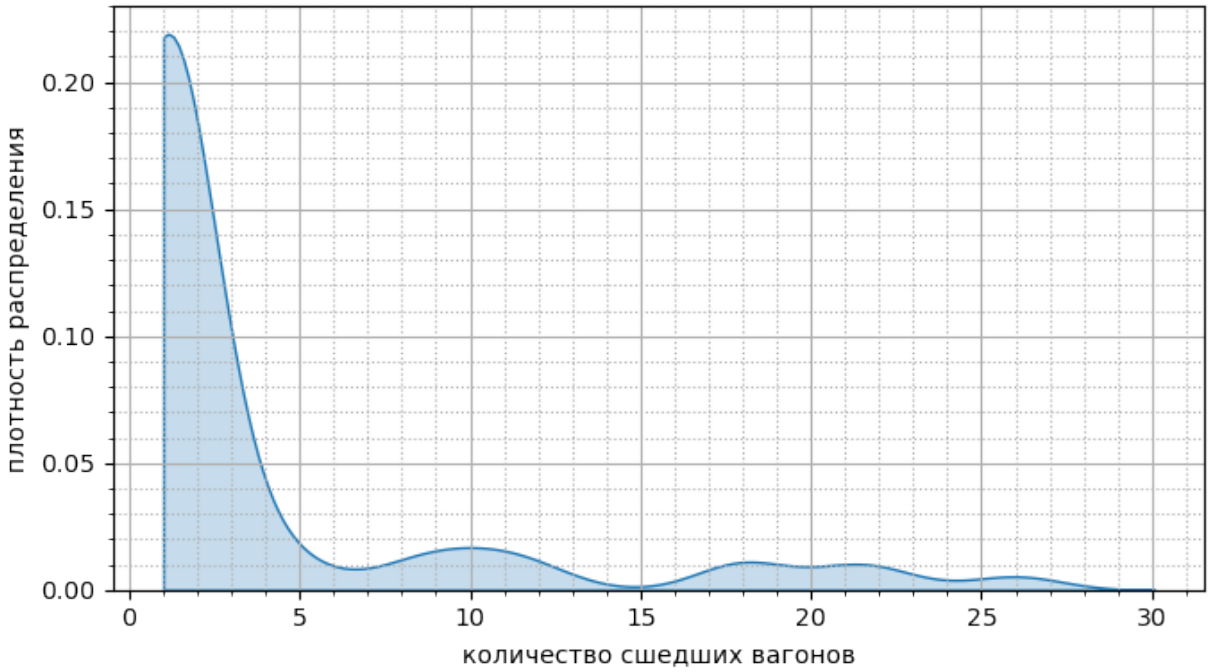

Рисунок 2.3 — Ядерная оценка плотности распределения

Из графика видно, что оценка плотности распределения признака количества сшедших поездов напоминает распределение Пуассона или геометрическое распределение.

ГЛАВА 3

МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ

3.1 Признаковые пространства

Сформируем признаковые пространства для моделей, основанных как на Пуассоновской регрессии, так и на геометрической регрессии:

1. $features_1$: (кривизна);
2. $features_2$: (кривизна, профиль пути);
3. $features_3$: (кривизна, профиль пути · макс. число вагонов в сходе);
4. $features_4$: (кривизна, $1 - \frac{\text{макс. число вагонов в сходе}}{\text{общее кол-во вагонов}}$);
5. $features_5$: (кривизна, профиль пути, скорость · загрузка);
6. $features_6$: (кривизна, профиль пути, скорость · загрузка, $1 - \frac{\text{макс. число вагонов в сходе}}{\text{общее кол-во вагонов}}$);
7. $features_7$: (кривизна, скорость · загрузка, $1 - \frac{\text{макс. число вагонов в сходе}}{\text{общее кол-во вагонов}}$);
8. $features_8$: (скорость · загрузка, $1 - \frac{\text{макс. число вагонов в сходе}}{\text{общее кол-во вагонов}}$).

Также в каждый набор признаков добавим новый признак $Intercept$, реализация которого всегда равна единице. Данный признак необходим для появления свободного члена θ_0 в результате скалярного произведения: $(\theta, x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$, где $n + 1$ – размерность соответствующего признакового пространства.

Были добавлены новые признаки, такие как: профиль пути · макс. число вагонов в сходе, $1 - \frac{\text{макс. число вагонов в сходе}}{\text{общее кол-во вагонов}}$, скорость · загрузка. Вычислим коэффициенты корреляции новых признаков с целевой целевым признаком количество сшедших вагонов, а также вычислим коэффициенты корреляции между самими признаками (для удобства чтения таблицы сделаем соответствующие переименование признаков: target, f_1, f_2, f_2, f_3):

	target	f_1	f_2	f_3
target	1.0	0.101375	-0.286535	0.198847
f_1	0.101375	1.0	-0.086693	-0.228508
f_2	-0.286535	-0.086693	1.0	-0.124420
f_3	0.198847	-0.228508	-0.124420	1.0

Таблица 3.1 — корреляция целевого признака с введеными

Из таблицы видно, что признаки f_1 и f_3 имеют значительную величину корреляции между собой. При этом признак f_2 слабо коррелирует как с f_1, так и с f_3. Из этого следует, что в один набор признаков нежелательно включать f_1 и f_3 вместе.

Также можно заметить, что признак f_2 сильно коррелирует с целевым признаком.

3.2 Программная реализация метода максимального правдоподобия

Для построения предсказательной модели будут использованы Пуассоновская и геометрическая регрессии. Оба метода в своей основе используют метод максимального правдоподобия. Именно поэтому был реализован общий класс для метода максимального правдоподобия. Сигнатура конструктора данного класса выглядит следующим образом:

```
def MLM(neg_log_likelihood_fun,
        const_log_likelihood_fun,
        optimization_method,
        borders, predict_fun,
        count_of_param_fun,
        features, target, df)
```

где:

- log_likelihood_fun – функция логарифмического правдоподобия;
- const_log_likelihood_fun – необязательное числовое поле, используется когда из логарифмической функции правдоподобия можно выделить константу;
- optimization_method – метод глобальной оптимизации;
- borders – границы поиска оптимальной точки;
- predict_fun – функция предсказания целевой переменной по заданному объекту;
- count_of_param_fun – количество параметризованных функций;

- features – вектор наборов названий признаков из признаковых пространств;
- target – название целевого признака;
- df – набор данных;

Таким образом, для построения регрессионной модели с заданным распределением целевой переменной достаточно определить функцию, вычисляющую логарифмическую функцию правдоподобия. Также можно переопределить функцию предсказания результата predict, либо воспользоваться готовой реализацией. Оставшиеся параметры в конструкторе класса не требуют от пользователя реализации функций.

```
class MLM():

    neg_log_likelihood_fun = 0
    const_log_likelihood_fun = 0

    predict_fun = None

    optimization_method = shgo
    borders = 10 * [(-1000, 1000)]

    df = None
    features = None
    target = None

    count_of_param_fun = 0


    def __init__(self, df, features, target,
            neg_log_likelihood_fun, const_log_likelihood_fun,
            optimization_method, borders, predict_fun,
            count_of_param_fun):
        self.neg_log_likelihood_fun = neg_log_likelihood_fun
        self.const_log_likelihood_fun = const_log_likelihood_fun
        self.optimization_method = optimization_method
        self.borders = borders
        self.predict_fun = predict_fun
        self.count_of_param_fun = count_of_param_fun
        self.features = features
        self.target = target
        self.df = df


    def const_log_likehood_zero(y):
```

```
34          return 0
35
36
37      def calc_aic_c(self, logL, feature) -> float:
38          logL = float(logL[2:-2])
39          k = len(feature)
40          n = len(self.y)
41          return round(2 * (k - logL) + 2 * k * (k + 1) / (n - k - 1), 2)
42
43
44      def to_fixed2(self, a, is_tex_output=False):
45          result_str = ''
46          for i in a:
47              if is_tex_output:
48                  result_str += ('$%.2f$, ' % i)
49              else:
50                  result_str += ('%.2f, ' % i)
51          result_str = '[' + result_str[:-2] + ']'
52          return result_str
53
54
55      def fit_all(self, get_tex_code=False):
56          num_of_feature = 0
57          for feature in self.features:
58              num_of_feature += 1
59
60              data = self.df[feature + self.target].dropna()
61              self.y = data[self.target]
62              self.y = np.array(self.y)
63              self.X = data.drop(self.target, axis=1)
64              self.X = np.matrix(self.X)
65
66              thetas_for_tex = ''
67              thetas_without_round_for_tex = ''
68              AIC_c_for_tex = ''
69
70              print('Признаковое пространство', num_of_feature, ':', feature)
71              print('Мощность выборки n = ', len(self.y))
72
73              for param_index in range(self.count_of_param_fun):
74                  count_of_attempt = 0
75                  result =
                    ↪  self.optimization_method(self.neg_log_likelihood_fun,
76                  self.borders[:len(feature)],
77                  args=(self.X, self.y, param_index))
78
```

```
79                  while math.isnan(result['fun']) and count_of_attempt < 15:
80                  result =
                     ↪  self.optimization_method(self.neg_log_likelihood_fun,
81                  self.borders[:len(feature)],
82                  args=(self.X, self.y, param_index))
83                  count_of_attempt += 1
84
85                  if get_tex_code:
86                  thetas_for_tex += self.to_fixed2(result['x'],
                     ↪  is_tex_output=True) + ' & '
87                  thetas_without_round_for_tex += str(result['x']) + ' & '
88                  AIC_c_for_tex += '$' +
                     ↪  str(self.calc_aic_c(self.to_fixed2([-result['fun'] +
                     ↪  self.const_log_likelihood_fun(self.y)],
                     ↪  is_tex_output=True), feature)) + '$' + ' & '
89
90                  print('success: ', result['success'])
91                  print('ln L: ', self.to_fixed2([-result['fun'] +
                     ↪  self.const_log_likelihood_fun(self.y)]))
92                  print('AIC_c', self.calc_aic_c(self.to_fixed2([-result['fun'] +
                     ↪  self.const_log_likelihood_fun(self.y)]), feature))
93                  print('thetas: ', self.to_fixed2(result['x']))
94                  print()
95
96              if get_tex_code:
97                  print('latex code:')
98                  print('thetas_for_tex:', thetas_for_tex[:-2], '\n')
99                  print('thetas_for_tex:', thetas_without_round_for_tex[:-2],
                     ↪  '\n')
100                 print('AIC_c_for_tex: ', AIC_c_for_tex[:-2], '\n')
101                 print('#######################################################')
102
103
104     # TO DO
105     def predict(self, x_pred):
106         return 0
```

3.3 Пуассоновская регрессия

Предположим, что количество сшедших вагонов имеет Пуассоновское распределение. Тогда функция вероятности $p(k) = \frac{\lambda^k}{k!}e^{-\lambda}$. Пусть плотность потока событий $\lambda = \lambda(\theta, x)$, где θ – вектор параметров, x – вектор, описы-

вающий объект. Выберем набор функций λ, параметризованных по θ:

1. $\lambda_1(\theta, x) = e^{\theta, x}$;
2. $\lambda_2(\theta, x) = e^{-(\theta, x)^2}$;
3. $\lambda_3(\theta, x) = \sqrt{|5^2 - ((\theta, x) - 5)^2|} + 1$;
4. $\lambda_4(\theta, x) = ((\theta, x) - 1)^2$;
5. $\lambda_5(\theta, x) = \frac{1}{1+(\theta, x)^2}$;
6. $\lambda_6(\theta, x) = (\theta, x)(\frac{\pi}{2} + \arctan(\theta, x)) + 1$;
7. $\lambda_7(\theta, x) = \log(1 + (\theta, x)^2) + 1$.

В качестве оптимизационного метода был выбран топологический метод глобальной оптимизации SHGO (Simplicial Homology Global Optimisation), реализованный в модуле scipy.optimize. В качестве граничного множества для искомых параметров был взят гиперкуб со 2000 и центром в начале координат (т.е. $-1000 \leq \theta_i \leq 1000 \;\; i = \overline{0, n}$, где n – размерность соответствующего признакового пространства).

В главе 1 была получена функция логарифмического правдоподобия: $l = \sum_{i=1}^{N} (-\lambda_i(x, \theta) + y_i \ln(\lambda_i(x, \theta)) - \ln(y_i!))$. Программно реализовав данную функцию и запустив метод fit_all у объекта класса MLM, были получены следующие результаты:

$n = 46$	модели с признаковым пространством $features_1$						
λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7
AIC_c	516.82	617.6	617.6	516.56	617.6	516.84	517.6
$\hat{\theta}$	[1.18, −271.63]	[−0.00, 0.00]	[0.00, 0.00]	[−0.80, 208.15]	[−0.00, 0.00]	[0.95, −236.29]	[2.66, −549.73]
$n = 41$	модели с признаковым пространством $features_2$						
λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7
AIC_c	491.77	606.39	485.09	486.11	606.39	483.01	477.09
$\hat{\theta}$	[1.17, −160.85, 39.30]	[−0.00, 0.00, 0.00]	[0.56, −114.67, 35.95]	[−0.83, 212.70, −56.93]	[−0.00, 0.00, 0.00]	[1.05, −281.75, 83.44]	[3.28, −486.77, 276.36]
$n = 37$	модели с признаковым пространством $features_3$						
λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7
AIC_c	466.01	598.47	457.63	458.97	598.47	453.11	448.95
$\hat{\theta}$	[1.29, −201.69, 0.82]	[−0.00, 0.00, 0.00]	[0.78, −67.92, 1.64]	[−0.94, 280.75, −1.44]	[−0.00, 0.00, 0.00]	[1.21, −426.41, 2.49]	[3.91, −1000.00, 7.26]
$n = 42$	модели с признаковым пространством $features_4$						
λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7
AIC_c	450.45	611.95	474.13	444.81	611.95	444.77	460.89
$\hat{\theta}$	[2.11, −180.43, −2.36]	[−0.00, 0.00, 0.00]	[1.98, 3.85, −2.09]	[−1.70, 207.42, 1.96]	[−0.00, 0.00, 0.00]	[2.19, −302.64, −2.53]	[5.94, −74.64, −6.39]
$n = 42$	модели с признаковым пространством $features_5$						
λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7
AIC_c	458.47	586.35	586.35	447.81	586.35	446.29	454.43
$\hat{\theta}$	[−0.09, −237.26, 0.03]	[−0.00, 0.00, −0.00]	[0.00, 0.00, 0.00]	[0.44, 216.06, −0.03]	[−0.00, 0.00, −0.00]	[−0.64, −262.77, 0.03]	[−0.39, −431.93, 0.07]

$n = 37$	модели с признаковым пространством $features_6$						
λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7
AIC_c	435.49	575.39	504.13	427.71	575.39	425.77	436.55
$\hat{\theta}$	[−0.15, −148.04, 53.13, 0.03]	[−0.00, 0.00, 0.00, −0.00]	[9.24, 3.94, 3.55, −0.11]	[0.29, 244.79, −52.47, −0.02]	[−0.00, 0.00, 0.00, −0.00]	[−0.47, −328.73, 75.88, 0.03]	[−0.31, −633.78, 7.19, 0.08]
$n = 35$	модели с признаковым пространством $features_7$						
λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7
AIC_c	400.13	574.21	574.21	401.53	574.21	401.33	409.59
$\hat{\theta}$	[0.92, −105.35, 34.34, 0.02, −2.24]	[−0.00, 0.00, 0.00, −0.00, 0.00]	[0.00, 0.00, 0.00, 0.00, 0.00]	[−0.98, 242.26, −22.62, −0.01, 1.73]	[−0.00, 0.00, 0.00, −0.00, 0.00]	[1.28, −373.54, 41.10, 0.02, −2.34]	[2.69, −57.95, 235.44, 0.08, −8.15]
$n = 37$	модели с признаковым пространством $features_8$						
λ_i	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7
AIC_c	410.65	575.39	575.39	419.57	575.39	419.63	419.13
$\hat{\theta}$	[0.55, 39.28, 0.03, −2.14]	[−0.00, 0.00, −0.00, 0.00]	[0.00, 0.00, 0.00, 0.00]	[−0.49, −15.96, −0.02, 1.44]	[−0.00, 0.00, −0.00, 0.00]	[0.31, 26.23, 0.03, −1.97]	[1.91, 281.49, 0.08, −6.90]

Таблица 3.2 — модели Пуассоновской регрессии с признаковыми пространствами $\{features_i\}_{i=1}^{8}$ и $\{\lambda_i\}_{i=1}^{7}$

3.4 Геометрическая регрессия

Предположим, что количество сшедших вагонов имеет геометрическое распределение. Тогда функция вероятности $p(n) = (1-p)^n p$. Пусть вероятность успеха в серии испытаний Бернулли $\lambda = \lambda(\theta, x)$, где θ – вектор параметров, x – вектор, описывающий объект. Выберем набор функций p, параметризованных по θ:

1. $p_1(\theta, x) = e^{(\theta,x)}$;
2. $p_2(\theta, x) = e^{-(\theta,x)^2}$;
3. $p_3(\theta, x) = \frac{1}{1+e^{-(\theta,x)}}$;
4. $p_4(\theta, x) = \frac{1}{1+(\theta,x)^2}$;
5. $p_5(\theta, x) = (\theta, x)(\frac{\pi}{2} + \arctan(\theta, x)) + 1$.

Метод SHGO, использованный при оптимизации в Пуассоновской регрессии, оказался не успешным для геометрической регрессии и данного набора данных. Ни для одной модели геометрической регрессии не удалось найти оптимальную точку. Вероятно, это связано со сложным видом оптимизационной функции в признаковых пространствах. По этой причине были использованы другие методы глобальной оптимизации (Dual Annealing, Differential Evolution, Basin-hopping), также реализованные в модуле scipy.optimize.

Наиболее лучшие результаты были получены с помощью метода Dual Annealing (алгоритм имитации обжига). В качестве граничного множества для искомых параметров был взят гиперкуб со 2000 и центром в начале координат (т.е. $-1000 \leq \theta_i \leq 1000 \quad i = \overline{0, n}$, где n – размерность соответствующего признакового пространства).

В главе 1 была получена функция логарифмического правдоподобия: $l = \sum_{i=1}^{N}(y_i \ln(1 - p(x_i, \theta)) + \ln(p(x_i, \theta)))$. Программно реализовав данную функцию и запустив метод fit_all у объекта класса MLM, были получены следующие результаты:

$n = 46$	модели с признаковым пространством $features_1$				
p_i	p_1	p_2	p_3	p_4	p_5
AIC_c	200.42	200.66	200.7	200.36	200.3
$\hat{\theta}$	[−1.46, 215.57]	[1.20, −90.37]	[−1.18, 278.32]	[1.82, −233.20]	[−0.95, 150.96]
$n = 41$	модели с признаковым пространством $features_2$				
p_i	p_1	p_2	p_3	p_4	p_5
AIC_c	182.17	186.19	185.53	178.59	180.33
$\hat{\theta}$	[−1.57, 304.90, −70.79]	[1.14, −853.89, −106.71]	[−1.27, 388.34, −103.97]	[−1.97, 392.24, −123.78]	[−1.00, 150.21, −64.07]
$n = 37$	модели с признаковым пространством $features_3$				
p_i	p_1	p_2	p_3	p_4	p_5
AIC_c	170.43	173.19	172.89	165.23	168.99
$\hat{\theta}$	[−1.52, 238.17, −2.18]	[−1.26, 171.59, −0.90]	[−1.40, 560.11, −3.07]	[2.05, −523.11, 3.38]	[−1.03, 169.06, −1.39]
$n = 42$	модели с признаковым пространством $features_4$				
p_i	p_1	p_2	p_3	p_4	p_5
AIC_c	176.33	176.33	176.67	174.73	176.85
$\hat{\theta}$	[−2.10, 111.27, 1.61]	[−1.57, 111.19, 0.86]	[−2.35, 342.71, 2.72]	[2.75, −255.76, −1.99]	[−1.31, 81.32, 0.94]
$n = 42$	модели с признаковым пространством $features_5$				
p_i	p_1	p_2	p_3	p_4	p_5
AIC_c	171.09	173.51	173.91	168.97	179.65
$\hat{\theta}$	[0.03, 133.64, −0.03]	[−0.58, 111.49, −0.01]	[0.80, 332.24, −0.04]	[−0.39, 279.20, −0.03]	[−0.77, 201.93, −0.01]
$n = 37$	модели с признаковым пространством $features_6$				
p_i	p_1	p_2	p_3	p_4	p_5
AIC_c	670.07	170.15	164.29	160.83	165.11
$\hat{\theta}$	[−22.08, 927.97, −55.98, 0.27]	[0.97, −838.88, −93.59, 0.00]	[0.39, 439.55, −89.77, −0.03]	[0.91, −413.88, 76.69, 0.02]	[−0.54, 105.09, −50.14, −0.01]

$n = 35$	модели с признаковым пространством $features_7$				
p_i	p_1	p_2	p_3	p_4	p_5
AIC_c	3365.29	157.83	158.03	153.79	160.31
$\hat{\theta}$	[−122.42, 1000.00, −1000.00, 1.36, 21.71]	[1.29, −147.10, 15.81, 0.00, −0.72]	[−1.45, 456.03, −52.52, −0.01, 2.32]	[−0.68, 89.66, −113.05, −0.05, 3.31]	[−1.42, 49.85, −3.56, −0.00, 1.34]
$n = 37$	модели с признаковым пространством $features_8$				
p_i	p_1	p_2	p_3	p_4	p_5
AIC_c	1467.35	177.75	164.23	155.27	352.91
$\hat{\theta}$	[−51.83, −1000.00, 0.52, 18.16]	[−0.18, −49.54, −0.03, 1.64]	[−0.59, −15.12, −0.02, 1.88]	[0.34, 99.11, 0.05, −3.08]	[−19.40, −550.73, 0.17, 8.78]

Таблица 3.3 — модели геометрической регрессии с признаковыми пространствами $\{features_i\}_{i=1}^{8}$ и $\{p_i\}_{i=1}^{5}$

ГЛАВА 4

СРАВНИТЕЛЬНЫЙ АНАЛИЗ

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Замышляев А.М., Игнатов А.Н., Кибзун А.И., Новожилов Е.О. Функциональная зависимость между количеством вагонов в сходе из-за неисправностей вагонов или пути и факторами движения // Надежность. 2018. Т. 18, № 1. С. . . . . DOI: 10.21683/1729-2646-2018-18-1. . .
2. Andrew Ng., Machine Learning from Stanford University. https://www.coursera.org/learn/machine-learning
3. Воронцов К.В., Введение в машинное обучение от НИУ ВШЭ & Yandex School of Data Analysis. https://www.coursera.org/learn/vvedenie-mashinnoe-obuchenie
4. Пуассоновская регрессия. https://en.wikipedia.org/wiki/Poisson_regression
5. Пуассоновская регрессия. https://towardsdatascience.com/an-illustrated-guide-to-the-poisson-regression-model-50cccba15958
6. Samuel, Arthur L. Some Studies in Machine Learning Using the Game of Checkers // IBM Journal. - 1959. - №3. - http://www.cs.virginia.edu/ evans/greatworks/samuel1959.pdf
7. CRISP-DM. https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining
8. Методологии анализа данных. https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html
9. Основы статистики часть 1. https://stepik.org/course/76/info
10. Основы статистики часть 2. https://stepik.org/course/524/info
11. Normal Equation. http://mlwiki.org/index.php/Normal_Equation
12. Обзор градиентных методов в задачах математической оптимизации. https://habr.com/ru/post/413853/
13. Метод стохастической аппроксимации. https://en.wikipedia.org/wiki/Stochastic_approximation
14. Регрессионный анализ. http://www.machinelearning.ru/wiki/index.php?title=Регрессионный_анализ
15. Эффект переобучения. https://en.wikipedia.org/wiki/Overfitting

16. Деревья решений для решения задач регрессии. https://habr.com/ru/post/116385/

Список иллюстраций

Список таблиц

Стр.