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 объекты (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 данная 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 поездах, 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 [5], 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 задач из T измеренное на основе P, улучшается с приобретением опыта E.

Большинство алгоритмов машинного обучения условно можно разбить на 2 класса: обучение с учителем (supervised learning) и обучение без учителя (unsupervised learning).

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 Y такой, 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 (target function).

Пусть также нам известны:

- $\{x_1, \ldots, x_l\} \subset X$ – известное подмножество объектов;

- $y_i = y(x_i), \ \ \forall\, i = \overline{1,n}$ – известное множество результатов.

Ставится задача найти $a: \ X \to Y$ – искомый алгоритм (decision function).

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 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" (feature data):

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

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: $Y = \mathbb{R}, \ Y = \mathbb{R}^k$;
- ранжирование: Y – конечное упорядоченное множество.

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 неизвестны, поэтому можно говорить лишь о том как данные расположены друг относительно друга, данный метод машинного обучения происходит без участия экспериментатора и применяется для обнаружения внутренних взаимосвязей. Обычно

так решаются задачи кластеризации, понижения размерности, визуализации данных.

Также используются методы обучения с подкреплением (reinforcement learning), когда набор данных дается из некоторого потока, а также рекомендательные системы (recommender systems).

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 $x \in X$:

- $\mathfrak{L}(a, x) = [a(x) \neq y(x)]$ – индикатор ошибки для случая классификации;
- $\mathfrak{L}(a, x) = (a(x) - y(x))^p$ – для случая регрессии.

Замечание: при $p = 1$ для регрессии функция ошибки берется как модуль разности алгоритма и ответа.

На практике для случая регрессии обычно берут $p = 2$ т.к. при $p = 1$ возникает проблема с дифференцированием функции потерь.

Чтобы оценить алгоритм в целом берут деленную на размер сумму функ-

ций потерь, 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 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. Что приводит к большому множеству методов машинного обучения, 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 задач анализа данных. Предполагается алгоритм решения произвольной задачи анализа данных в 5 шагов, причем они могут замыкаться в цикле.

Рисунок 1.1 — Жизненный цикл исследования данных [6]

1. В начале 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 удовлетворяют исследователя, модель внедряется в производственные процессы и эксплуатируется.

Помимо CRISP-DM существуют менее известные стандарты: My own, SEMMA и другие. На сайте [7] публикуются результаты опросов по популярности методологий анализа данных.

1.3 Теория 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 говорится о задаче условной оптимизации, 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 градиентов, 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 $\nabla f(x) = \frac{\partial f(x)}{\partial x_1}\vec{i_1} + \frac{\partial f(x)}{\partial x_2}\vec{i_2} + \cdots + \frac{\partial f(x)}{\partial x_n}\vec{i_n}$. Направление градиента указывает на то направление из точки x, которое имеет наибольшую скорость роста функции из данной точки. Соответственно, $-\nabla f(x)$ 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 большое, то метод градиентного спуска может начать расходится, то есть значение x будет отдалятся от точки экстремума. Существует условие Липшица, 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 $\forall \alpha < \frac{2}{L}$ 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 минимум 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

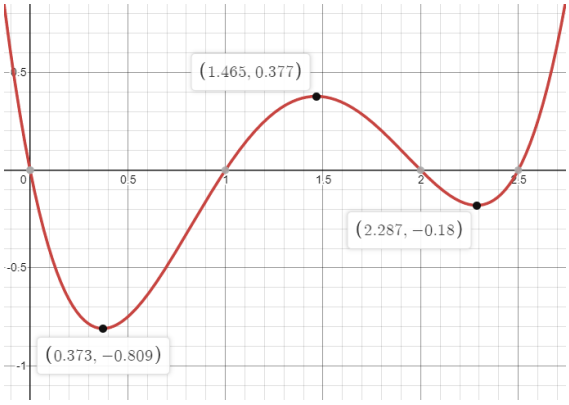

Рисунок 1.2 — Проблема много экстремальности функции

по модулю координатами. Если метод не сойдется к тому же решению, то необходимо сравнить 2 минимума и выбрать наименьший.

При поиска экстремума в овражных функциях градиентный спуск сходится медленно, причем если число аргументов функции велико, то довольно часто можно встретить овражные области. Для решения данной сходимости 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: $\alpha_k = \arg \min\limits_{\alpha \in [0,\infty)} f(x^k - \alpha_{k-1} f'(x_k))$ метод с таким выбором градиентного шага носит называние метод 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 следующей формулой:

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

Дополнительное слагаемое не меняет асимптотической сложности алгоритма, при этом правильном выборе параметров 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 [12]. 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: $\mathcal{L}(\theta_1, \theta_2, \ldots, \theta_m) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$ 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

1.4.1 Линейные модели

Линейная регрессия

Полиномиальная регрессия

Пуассоновская регрессия

Геометрическая регрессия

Проблема переобучения

Пример линейной модели

Эмпирические оценки обобщающей способности
* Эмпирический риск на тестовых данных (Hold-out)
* Скользящий контроль (leave-one-out)

* Кросс-проверка (cross-validation) по N разбиениям

1.4.2 Нелинейные модели

Градиентный бустинг

Решающие деревья

Случайный лес

ГЛАВА 2

ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ ДАННЫХ

2.1 Структура данных

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

Анализ данных будет проведен при помощи языка программирования Python3. Выбор пал на данный язык по нескольким причинам:

- большое количество модулей для анализа данных
- удобство и простота работы с данными в форматах csv, xlsx
- множество встроенных функций и выразительность языка

Определим размеры выборки:

```
print("shape of data frame:", df.shape)
```

```
> shape of data frame: (56, 12)
```

Выведем названия факторов:

```
print(df.columns)
```

```
> Index(['Дата', 'Количество вагонов', 'Макс. число вагонов в сходе',
'Общее количество вагонов', 'Количество сшедших вагонов', 'Скорость',
'Вес', 'Загрузка', 'Стрелочный перевод', 'Кривизна', 'Профиль пути',
'Режим движения'],
dtype='object')
```

Получим первые 5 записей из набора:

№	Дата	Количество вагонов	Макс. число вагонов в сходе	Общее количество вагонов	Количество сшедших вагонов	Скорость	Вес	Загрузка	Стрелочный перевод	Кривизна	Профиль пути	Режим движения
1	2013-01-08	56.0	19.0	58.0	1	57.0	3402.0	0.547101	0	0.000000	0.0007	NaN
2	2013-01-09	60.0	25.0	62.0	1	72.0	4082.0	0.652657	0	0.000000	0.0009	NaN
3	2013-01-10	60.0	4.0	64.0	1	15.0	4420.0	0.734300	0	0.001639	NaN	3.0
4	2013-01-12	66.0	63.0	68.0	21	67.0	5699.0	0.918094	0	0.002326	0.0060	NaN
5	2013-01-19	67.0	34.0	69.0	1	69.0	5854.0	0.932944	0	0.000000	0.0006	2.0

Таблица 2.1 — первые 5 записей в наборе данных

Получим основные статистики по данным с помощью команды `print(df.describe())` (для краткости названия признаков заменены на f1, f2, . . . , f11, признак "Дата"не рассматривается).

	f1	f2	f3	f4	f4	f6	f7	f8	f9	f10	f11
count	54.000000	51.000000	54.000000	56.000000	53.000000	54.000000	54.000000	56.000000	46.000000	44.000000	33.000000
mean	63.870370	37.137255	66.407407	3.875000	49.150943	5126.629630	0.817678	0.107143	0.000806	-0.000384	1.666667
std	9.790342	21.543463	10.053665	6.081455	18.450971	1438.743887	0.243936	0.312094	0.001171	0.005689	0.777282
min	24.000000	2.000000	26.000000	1.000000	9.000000	998.000000	0.179710	0.000000	0.000000	-0.011500	1.000000
25%	60.000000	17.500000	62.500000	1.000000	35.000000	4155.500000	0.690451	0.000000	0.000000	-0.004750	1.000000
50%	66.000000	43.000000	68.000000	1.000000	51.000000	5722.000000	0.925519	0.000000	0.000000	0.000000	1.000000
75%	68.000000	56.500000	71.750000	2.250000	64.000000	6010.250000	0.995586	0.000000	0.001479	0.001875	2.000000
max	96.000000	72.000000	100.000000	26.000000	78.000000	8806.000000	1.076087	1.000000	0.005000	0.010900	3.000000

Таблица 2.2 — основные статистики

Заметим, что в данных есть пропуски, так признак "Режим движения"(f11) содержит только 33 записи. Также много пропусков у признаков "Кривизна"(f9) и "Профиль пути"(f10).

Построим матрицу корреляции признаков:

```
corrmat = df.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True)
```

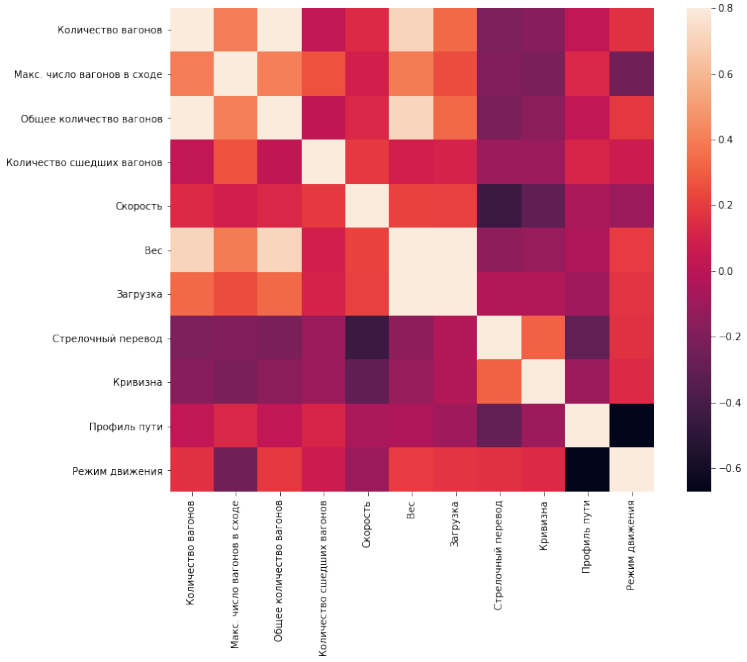

Рисунок 2.1 — Корреляция признаков

Из матрицы видно, что признаки "Количество вогонов"и "Общее число вагонов"имеют сильную корреляцию. Также "Вес"и "Загрузка"сильно коррелируют. Менее сильная корреляция наблюдается у признаков "Вес"и "Общее число вагонов". Также 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 коррелирующие признаки, и, возможно, добавить новые. Решение проблемы мультиколлинеарности смотри в главе "Линейная регрессия".

2.2 Пропуски в данных

Из таблицы 2.2 видно, что в последних четырех признаках присутствуют пропуски в данных.

Существует методы по решению проблемы с пропусками в данных:

- удалить все записи 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.

При решении задачи будут поочередно использованы все 3 метода борьбы с пропусками, предпочтение будет отдаваться тем моделям, у которых будут более лучшие показатели метрик качества.

2.3 Экстремальные значения

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

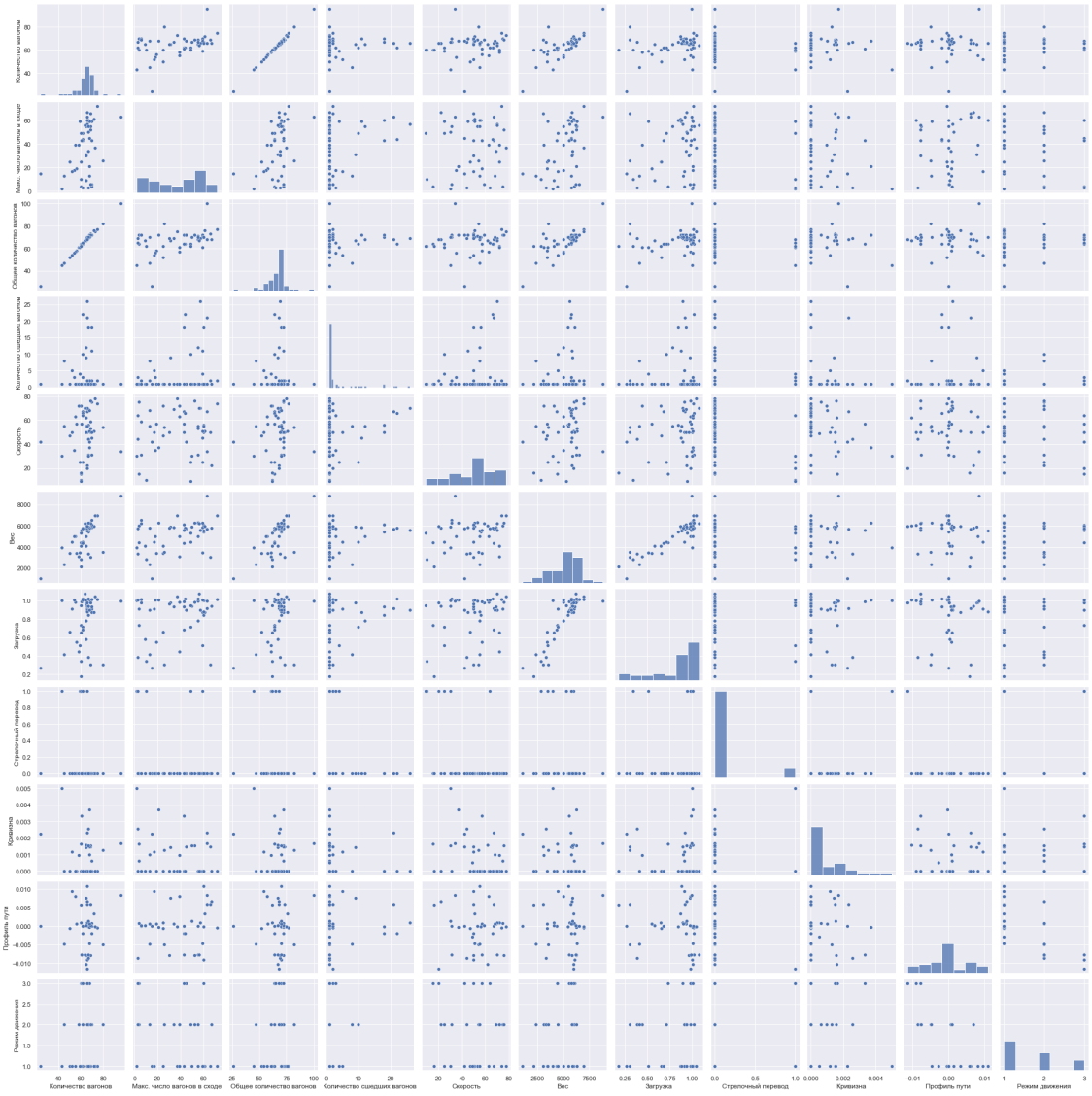

Рисунок 2.2 — Пары признаков

Изучив таблицу 2.2, а также при детальном рассмотрении графиков 2.2 выбросов в данных не обнаружено.

ГЛАВА 3

МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ

3.1 Линейная регрессия

3.1.1 Достоинства и недостатки ЛР для данной задачи

3.2 Пуассоновская регрессия

3.3 Геометрическая регрессия

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. 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
6. CRISP-DM. https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining
7. Методологии анализа данных. https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html
8. Основы статистики часть 1. https://stepik.org/course/76/info
9. Основы статистики часть 2. https://stepik.org/course/524/info
10. Normal Equation. http://mlwiki.org/index.php/Normal_Equation
11. Обзор градиентных методов в задачах математической оптимизации. https://habr.com/ru/post/413853/
12. Метод стохастической аппроксимации. https://en.wikipedia.org/wiki/Stochastic_approximation

Список иллюстраций

Список таблиц

Стр.

Список программных листингов