МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(государственный университет)

ПРОГНОЗИРОВАНИЕ НА БАЗЕ РЕШЕНИЯ НАБОРА ЗАДАЧ КЛАССИФИКАЦИИ С УЧИТЕЛЕМ

Бакалаврская квалификационная работа
студента 376 группы ФУПМ

Луканина Артема Александровича

Научный руководитель:

д.ф.-м.н., профессор Рязанов Владимир Васильевич

Москва 2017

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

Дипломная работа посвящена проблеме восстановления зависимостей по выборкам прецедентов.

В первой части данной работы приводится обзор существующих регрессионных методов, подробно описывается использование линейных моделей в задачах регрессии, рассматриваются методы решения задачи восстановления зависимости коллективами распознающих алгоритмов, особое внимание обращено на алгоритмы вычисления оценок (АВО).

Во второй части работы предлагается метод поиска функциональных зависимостей по выборкам прецедентов с использованием алгоритмов распознавания, основанный на построении набора кодирований зависимой величины по обучающим данным и использовании логико-статистического подхода.

Цели работы:

1. Создание метода восстановления зависимостей на базе коллективов алгоритмов вычисления оценок;
2. Сравнение двух различных способов определения функции близости в АВО на примере двух репозиториев;
3. Сравнение результатов работы созданного метода с результатами, полученными при использовании других известных методов.

2. ОБЗОР СУЩЕСТВУЮЩИХ МЕТОДОВ

2.1. Основные определения

Дано конечное множество объектов (прецедентов), по каждому из которых собраны некоторые данные. Данные об объекте называют также его описанием [8]. Совокупность всех имеющихся описаний объектов называется обучающей выборкой. Требуется по этим частным данным выявить общие зависимости, присущие не только этой конкретной выборке, но вообще всем объектам, в том числе тем, которые ещё не наблюдались.

Объектом называется то, для чего нужно сделать предсказание. Пространство объектов – это множество всех возможных объектов, для которых может потребоваться делать предсказание. Ответом будет называться то, что нужно предсказать.

Будем использовать следующие обозначения: $\mathbf{x}$ – объект, $\mathbb{X}$ – пространство объектов, $y = y(\mathbf{x})$ – ответ на объекте $\mathbf{x}$, $\mathbb{Y}$ – пространство ответов.

Наиболее распространённым способом описания объектов является признаковое описание. Фиксируется совокупность n показателей $(x_1, x_2, \dots, x_n)$, измеряемых у всех прецедентов. Если все n показателей числовые, то признаковые описания представляют собой числовые векторы размерности n.

Центральным понятием машинного обучения является обучающая выборка $\tilde{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$. Это те самые примеры, на основе которых будет строиться общая закономерность. Предсказание будет делаться на основе некоторой модели (алгоритма) $a(\mathbf{x})$, которая представляет из себя функцию из пространства $\mathbb{X}$ в пространство $\mathbb{Y}$. Эта функция должна быть легко реализуема на компьютере, чтобы ее можно было использовать в системах машинного обучения.

Вводится некоторая характеристика качества работы алгоритма – функционал ошибки. $Q(a; \tilde{X})$ – ошибка алгоритма a на выборке $\tilde{X}$. Например, функционал ошибки может быть долей неправильных ответов.

Задача обучения состоит в подборе такого алгоритма a, для которого достигается минимум функционала ошибки. Лучший в этом смысле алгоритм выбирается из некоторого семейства $\mathbb{A}$ алгоритмов. Общая постановка задачи обучения с учителем следующая: для обучающей выборки $\tilde{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$ нужно найти такой алгоритм $a \in \mathbb{A}$, на котором будет достигаться минимум функционала ошибки: $Q(a; \tilde{X}) \to \min_{a \in \mathbb{A}}$.

2.2. Типы задач

Обучением с учителем называются такие задачи, в которых есть и объекты, и истинные ответы на них. И нужно по этим парам восстановить общую зависимость. В зависимости от множества возможных ответов $\mathbb{Y}$, задачи делятся на несколько типов:

1. Задача бинарной классификации.

   В задаче бинарной классификации пространство ответов состоит из двух ответов $\mathbb{Y} = \{0,1\}$. Множество объектов, которые имеют один ответ, называется классом. Говорят, что нужно относить объекты к одному из двух классов, другими словами, классифицировать эти объекты.
2. Задача многоклассовой классификации.

   Классов может быть больше, чем два. В таком случае имеет место задача многоклассовой классификации.
3. Задача регрессии.

   Когда y является вещественной переменной, говорят о задаче регрессии.
4. Задача ранжирования.

   Пример: поиск какой-то информации в интернете. После того, как введен запрос, происходит ранжирование страниц по релевантности их запросу, то есть для каждой страницы оценивается ее релевантность в виде числа, а затем страницы сортируются по убыванию релевантности. Задача состоит в предсказании релевантности для пары (запрос, страница).

Задача обучения без учителя – это такая задача, в которой есть только объекты, а ответов нет. Также бывают «промежуточные» постановки. В случае частичного обучения есть объекты, некоторые из которых с ответами. В случае активного обучения получение ответа обычно очень дорого, поэтому алгоритм должен сначала решить, для каких объектов нужно узнать ответ, чтобы лучше всего обучиться. Рассмотрим несколько примеров постановки задач без учителя:

1. Задача кластеризации.

   Дано множество объектов. Необходимо найти группы похожих объектов. Есть две основные проблемы: не известно количество кластеров и не известны истинные кластеры, которые нужно выделять. Здесь невозможно оценить качество решения. Этим и отличается задача классификации – там тоже нужно делить объекты на группы, но в классификации группы, а точнее классы, фиксированы, и известны примеры объектов из разных групп.

2. Задача визуализации.

   Необходимо нарисовать многомерную (а конкретно, n −мерную) выборку так, чтобы изображение наглядно показывало структуру объектов. Примером задачи визуализации является задача визуализации набора данных MNIST. Этот набор данных был получен в результате оцифровки рукописных начертаний цифр. Каждый скан цифры характеризуется вектором признаков-яркостей отдельных пикселей. Необходимо таким образом отобразить этот набор данных на плоскость, чтобы разные цифры оказались в разных ее областях.

3. Поиск аномалий.

   Необходимо обнаружить, что данный объект не похож на все остальные, то есть является аномальным. При обучении есть примеры только обычных, не аномальных, объектов. А примеров аномальных объектов либо нет вообще, либо настолько мало, что невозможно воспользоваться

классическими методами обучения с учителем (методами бинарной классификации).

2.3. Модели восстановления регрессий

В задаче регрессии пространство ответов $\mathbb{Y} = \mathbb{R}$. Чтобы решить задачу регрессии, необходимо задать:

1. функционал ошибки Q: способ измерения того, хорошо или плохо работает алгоритм на конкретной выборке;
2. семейство алгоритмов $\mathbb{A}$: как выглядит множество алгоритмов, из которых выбирается лучший;
3. метод обучения: как именно выбирается лучший алгоритм из семейства алгоритмов. Рассматривается обучение: $a(\mathbf{x}) = \operatorname{argmin}_{a \in \mathbb{A}} Q(a, \tilde{X})$.

Существует много параметрических и непараметрических методов восстановления регрессии. Рассмотрим основные из них.

2.3.1. Линейная регрессия

2.3.1.1. Линейные модели в задачах регрессии

Далее рассмотрим, как выглядит семейство алгоритмов в случае с линейными моделями [7]. Линейный алгоритм в задачах регрессии выглядит следующим образом:

$$a(\mathbf{x}) = w_0 + \sum_{j=1}^{n} w_j x_j,$$

где w_0 – свободный коэффициент, x_j – признаки, а w_j – их веса.

Если добавить $(n+1)$ −й признак, который на каждом объекте принимает значение 1, линейный алгоритм можно будет записать в более компактной форме:

$$a(\mathbf{x}) = \sum_{j=1}^{n+1} w_j x_j = \langle \mathbf{w}, \mathbf{x} \rangle,$$

где используется обозначение $\langle \mathbf{w}, \mathbf{x} \rangle$ для скалярного произведения двух векторов.

В качестве меры ошибки не может быть выбрано отклонение от прогноза $a(\mathbf{x}) - y$, так как в этом случае минимум функционала не будет достигаться при правильном ответе $a(\mathbf{x}) = y$. Простейший способ – считать модуль отклонения: $|a(\mathbf{x}) - y|$. Но функция модуля не является гладкой функцией, и для оптимизации такого функционала неудобно использовать градиентные методы. Поэтому в качестве меры ошибки часто выбирается квадрат отклонения: $(a(\mathbf{x}) - y)^2$.

Функционал ошибки, именуемый среднеквадратичной ошибкой алгоритма, задается следующим образом:

$$Q(a; \tilde{X}) = \frac{1}{m}\sum_{i=1}^{m}(a(\mathbf{x}_i) - y_i)^2.$$

В случае линейной модели его можно переписать в виде функции (поскольку теперь Q зависит от вектора, а не от функции) ошибок:

$$Q(\mathbf{w}; \tilde{X}) = \frac{1}{m}\sum_{i=1}^{m}(\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2.$$

2.3.1.2. Обучение модели линейной регрессии

Перепишем выражение для качества линейной модели на обучающей выборке:

$$Q(\mathbf{w}; \tilde{X}) = \frac{1}{m}\sum_{i=1}^{m}(\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2 \to \min_{\mathbf{w}}.$$

Прежде, чем будет рассмотрена задача оптимизации этой функции, имеет смысл переписать используемые соотношения в матричной форме. Матрица «объекты – признаки» X составлена из признаковых описаний всех объектов из обучающей выборки, а вектор ответов $\boldsymbol{y}$ составлен из истинных ответов для всех объектов:

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}.$$

Таким образом, в ij −элементе матрицы X записано значение j −го признака на i объекте обучающей выборки. В этом случае среднеквадратичная ошибка может быть переписана в матричном виде:

$$Q(\mathbf{w};\, X) = \frac{1}{m}\|X\mathbf{w} - \boldsymbol{y}\|^2 \to \min_{\mathbf{w}} .$$

2.3.1.3. Аналитический метод решения

Можно найти аналитическое решение задачи минимизации: $\mathbf{w}_* = (X^T X)^{-1} X^T \boldsymbol{y}$. Основные сложности при нахождении решения таким способом:

1. Для нахождения решения необходимо вычислять обратную матрицу. Операция обращения матрицы требует, в случае n признаков, выполнение порядка n^3 операции, и является вычислительно сложной уже в задачах с десятком признаков.
2. Численный способ нахождения обратной матрицы не может быть применен в некоторых случаях (когда матрица плохо обусловлена, т.е. $\|X\| \cdot \|X^{-1}\| \geq 10^3$).

2.3.1.4. Оптимизационный подход к решению

Другой, несколько более удобный, способ найти решение – использовать численные методы оптимизации.

Несложно показать, что среднеквадратическая ошибка – это выпуклая и гладкая функция. Выпуклость гарантирует существование лишь одного минимума, а гладкость – существование вектора градиента в каждой точке. Это позволяет использовать метод градиентного спуска [6].

При использовании метода градиентного спуска необходимо указать начальное приближение. Есть много подходов к тому, как это сделать, в том числе инициализировать случайными числами (не очень большими).

Самый простой способ это сделать – инициализировать значения всех весов равными нулю: $\mathbf{w}^0 = 0$.

На каждой следующей итерации, $t = 1, 2, 3, ...$, из приближения, полученного в предыдущей итерации $\mathbf{w}^{t-1}$, вычитается вектор градиента в соот-

ветствующей точке $\mathbf{w}^{t-1}$, умноженный на некоторый коэффициент η_t, называемый шагом:

$$\mathbf{w}^t = \mathbf{w}^{t-1} - \eta_t \nabla Q(\mathbf{w}^{t-1};\, X).$$

Остановить итерации следует, когда наступает сходимость. Сходимость можно определять по-разному. В данном случае разумно определить сходимость следующим образом: итерации следует завершить, если разница двух последовательных приближений не слишком велика: $\|\mathbf{w}^t - \mathbf{w}^{t-1}\| \leq \varepsilon$.

2.3.1.5. Стохастический градиентный спуск

Необходимо решать задачу минимизации:

$$Q(\mathbf{w};\, X) = \frac{1}{m}\|X\mathbf{w} - \boldsymbol{y}\|^2 \to \min_{\mathbf{w}}\ .$$

В обычном методе градиентного спуска формула для вычисления градиента принимает следующий вид: $\nabla Q(\mathbf{w};\, X) = \frac{2}{m} X^T(X\mathbf{w} - \boldsymbol{y})$. Выражение для j −ой компоненты градиента, таким образом, содержит суммирование по всем объектам обучающей выборки:

$$\frac{\partial Q}{\partial w_j} = \frac{2}{m}\sum_{i=1}^{m} x_{ij}(\langle \mathbf{w}, \mathbf{x}_i\rangle - \ y_i).$$

В этом и состоит основной недостаток метода градиентного спуска – в случае большой выборки даже одна итерация метода градиентного спуска будет производиться долго.

Идея стохастического градиентного спуска основана на том, что в сумме в выражении для j −компоненты градиента i −ое слагаемое указывает то, как нужно поменять вес w_j, чтобы качество увеличилось для i −го объекта выборки. Вся сумма при этом задает, как нужно изменить этот вес, чтобы повысить качество для всех объектов выборки. В стохастическом методе градиентного спуска градиент функции качества вычисляется только на одном случайно выбранном объекте обучающей выборки. Это позволяет обойти вышеупомянутый недостаток обычного градиентного спуска.

Таким образом, алгоритм стохастического градиентного спуска следующий. Сначала выбирается начальное приближение: $\mathbf{w}^0 = 0$. Далее последовательно вычисляются итерации $\mathbf{w}^t$: сначала случайным образом выбирается объект $\mathbf{x}_i$ из обучающей выборки $\tilde{X}$ и вычисляется вектор градиента функции качества на этом объекте, а следующее приближение получается из предыдущего вычитанием умноженного на шаг η_t полученного вектора: $\mathbf{w}^t = \mathbf{w}^{t-1} - \eta_t \nabla Q(\mathbf{w}^{t-1}, \{\mathbf{x}_i\})$. Итерации прекращаются при достижении определенного условия, например: $\|\mathbf{w}^t - \mathbf{w}^{t-1}\| \leq \varepsilon$.

2.3.1.6. Сходимость стохастического градиентного спуска

В обычном градиентном спуске на каждом шаге уменьшается суммарная ошибка на всех элементах обучающей выборки. График сходимости в таком случае обычно получается монотонным. Напротив, в стохастическом методе весовые коэффициенты меняются таким образом, чтобы максимально уменьшить ошибку для одного случайно выбранного объекта. Это приводит к тому, что график сходимости выглядит пилообразным, то есть на каждой конкретной итерации полная ошибка может как увеличиваться, так и уменьшаться. Но в итоге, с ростом номера итерации значение функции уменьшается.

2.3.2. Метод наименьших квадратов

Данный метод уже использовался выше в линейной регрессии. Опишем его подробнее [8].

Задача метода наименьших квадратов состоит в выборе вектора $\mathbf{w}$, минимизирующего ошибку $|X\mathbf{w} - \boldsymbol{y}|^2$. Эта ошибка есть расстояние от вектора $\boldsymbol{y}$ до вектора $X\mathbf{w}$. Вектор $X\mathbf{w}$ лежит в пространстве столбцов матрицы X, так как $X\mathbf{w}$ есть линейная комбинация столбцов этой матрицы с коэффициентами $w_1, \ldots, w_n$. Отыскание решения $\mathbf{w}$ по методу наименьших квадратов эквивалентно задаче отыскания такой точки $\boldsymbol{p} = X\mathbf{w}$, которая лежит ближе всего к $\boldsymbol{y}$ и находится при этом в пространстве столбцов матрицы X. Таким образом, вектор $\boldsymbol{p}$ должен быть проекцией $\boldsymbol{y}$ на пространство столбцов, и вектор не-

вязки $X\mathbf{w} - \boldsymbol{y}$ должен быть ортогонален этому пространству. Ортогональность состоит в том, что каждый вектор в пространстве столбцов есть линейная комбинация столбцов с некоторыми коэффициентами $v_1, \dots, v_n$, то есть это вектор $X\mathbf{v}$. Для всех $\mathbf{v}$ в пространстве $X\mathbf{v}$, эти векторы должны быть перпендикулярны невязке $X\mathbf{w} - \boldsymbol{y}$: $(X\mathbf{v})^T(X\mathbf{w} - \boldsymbol{y}) = \mathbf{v}^T(X^TX\mathbf{w} - X^T\boldsymbol{y}) = 0$. Так как это равенство должно быть справедливо для произвольного вектора $\mathbf{v}$, то $X^TX\mathbf{w} - X^T\boldsymbol{y} = 0$.

Решение по методу наименьших квадратов несовместной системы $X\mathbf{w} = \boldsymbol{y}$, состоящей из m уравнений с n неизвестными, есть уравнение $X^TX\mathbf{w} = X^T\boldsymbol{y}$, которое называется нормальным уравнением. Если столбцы матрицы X линейно независимы, то матрица X^TX обратима и единственное решение системы: $\mathbf{w}_* = (X^TX)^{-1}X^T\boldsymbol{y}$.

2.3.3. Регуляризация

2.3.3.1. Переобучение регрессионных моделей

Если используется слишком сложная модель, а данных недостаточно, чтобы точно определить ее параметры, эта модель легко может получиться переобученной, то есть хорошо описывать обучающую выборку и плохо – тестовую. Одним из способов борьбы с переобучением в линейных моделях является регуляризация [7].

2.3.3.2. L_1-регуляризация и L_2-регуляризация

Есть несколько способов провести регуляризацию:

1. L_2-регуляризатор (ridge-регрессия или гребневая регрессия):

$$Q(\mathbf{w}; X) + \lambda\|\mathbf{w}\|^2 \to \min_{\mathbf{w}},$$

$$\mathbf{w}_* = \arg\min_{\mathbf{w}} \left(\frac{1}{m}\sum_{i=1}^{m}(\langle \mathbf{w}, \mathbf{x}_i\rangle - y_i)^2 + \lambda\sum_{j=1}^{n} w_j^2 \right),$$

2. L_1-регуляризатор (lasso-регрессия или лассо-регрессия):

$$Q(\mathbf{w}; X) + \lambda\|\mathbf{w}\| \to \min_{\mathbf{w}},$$

$$\mathbf{w}_* = \arg\min_{\mathbf{w}} \left(\frac{1}{m} \sum_{i=1}^{m} (\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i)^2 + \lambda \sum_{j=1}^{n} |w_j| \right).$$

Таким образом, при обучении будет учитываться также то, что не следует слишком сильно увеличивать веса признаков.

Введенный выше коэффициент λ, который стоит перед регуляризатором, называется коэффициентом регуляризации. Чем больше λ, тем ниже сложность модели. Например, при очень больших его значениях оптимально просто занулить все веса. В то же время при слишком низких значениях λ высок риск переобучения, то есть модель становится слишком сложной. Поэтому нужно найти некоторое оптимальное значение λ, достаточно большое, чтобы не допустить переобучения, и не очень большое, чтобы уловить закономерности в данных.

Понять различия между L_1 и L_2-регулязаторами можно на модельном примере. Пусть матрица «объекты – признаки» X является единичной матрицей размера $m \times m$. Тогда при решении задачи линейной регрессии использование метода наименьших квадратов без регуляризации $\mathbf{w}_* = \arg\min_{\mathbf{w}} \sum_{i=1}^{m} (w_i - y_i)^2$ дает следующий вектор весов: $w_{*_j} = y_j$. При использовании L_2-регуляризации компоненты вектора весов имеют вид: $w_{*_j} = \frac{y_j}{1+\lambda}$, а при использовании L_1-регуляризатора: $w_{*_j} = \begin{cases} y_j - \frac{\lambda}{2}, & y_j > \frac{\lambda}{2}, \\ y_j + \frac{\lambda}{2}, & y_j < -\frac{\lambda}{2}, \\ 0, & |y_j| \le \frac{\lambda}{2}. \end{cases}$

Таким образом, при использовании L_2-регуляризации зависимость w_{*_j} от y_j линейная (как и у МНК без регуляризации), но компоненты вектора весов ближе расположены к нулю. В случае L_1-регуляризации график зависимости w_{*_j} от y_j выглядит несколько иначе: существует область (размера λ) значений y_j, для которых $w_{*_j} = 0$. То есть L_1-регуляризация позволяет отбирать признаки, а именно: веса признаков, обладающих низкой предсказательной способностью, оказываются равными нулю.

2.3.4. Нелинейная регрессия

Заданы обучающая выборка из m пар $(\mathbf{x}_i, y_i)$ и регрессионная модель $f(\mathbf{x}, \mathbf{w})$. Требуется найти такие значения параметров, которые доставляли бы минимум сумме квадратов регрессионных остатков $S = \sum_{i=1}^{m} r_i^2$, где $r_i = y_i - f(\mathbf{x}_i, \mathbf{w})$, $i = 1, \dots, m$ – остатки.

Для нахождения минимума функции S, приравняем к нулю её первые частные производные по компонентам $\mathbf{w}$ [8]:

$$\frac{\partial S}{\partial w_j} = 2\sum_{i=1}^{m} r_i \frac{\partial r_i}{\partial w_j} = 0 \ (j = 1, \dots, n). \quad (1)$$

Так как функция S в общем случае не имеет единственного минимума, предлагается назначить начальное значение вектора параметров $\mathbf{w}^0$ и приближаться к оптимальному вектору по шагам: $w_j \approx w_j^{k+1} = w_j^k + \Delta w_j$, где k – номер итерации, Δw_j – вектор шага.

На каждом шаге итерации линеаризуем модель с помощью приближения рядом Тейлора относительно параметров $\mathbf{w}^k$:

$$f(\mathbf{x}_i, \mathbf{w}) \approx f(\mathbf{x}_i, \mathbf{w}^k) + \sum_{j=1}^{n} \frac{\partial f(\mathbf{x}_i, \mathbf{w}^k)}{\partial w_j}(w_j - w_j^k) \approx f(\mathbf{x}_i, \mathbf{w}^k) + \sum_{j=1}^{n} \mathcal{J}_{ij}\Delta w_j,$$

где элемент матрицы Якоби $\mathcal{J}_{ij}$ – функция параметра w_j, а значение свободной переменной $\mathbf{x}_i$ фиксировано. В терминах линеаризованной модели $\frac{\partial r_i}{\partial w_j} = -\mathcal{J}_{ij}$ и регрессионные остатки определены как $r_i = \Delta y_i - \sum_{j=1}^{n} \mathcal{J}_{ij}\Delta w_j$, где $\Delta y_i = y_i - f(\mathbf{x}_i, \mathbf{w}^k)$. Подставляя последнее выражение в (1), получаем: $-2\sum_{i=1}^{m} \mathcal{J}_{ij}(\Delta y_j - \sum_{s=1}^{n} \mathcal{J}_{is}\,\Delta w_s) = 0.$

В итоге, получаем систему из n линейных уравнений, которые называются нормальным уравнением $\sum_{i=1}^{m}\sum_{s=1}^{n} \mathcal{J}_{ij}\mathcal{J}_{is}\,\Delta w_s = \sum_{i=1}^{m} \mathcal{J}_{ij}\Delta y_j$ $(j = 1, \dots, n)$. Запишем нормальное уравнение в матричном обозначении как $(\mathcal{J}^T\mathcal{J})\Delta w = \mathcal{J}^T\Delta y$. Для нахождения оптимальных параметров нели-

нейных регрессионных моделей используются метод сопряжённых градиентов [6].

2.3.5. Метод опорных векторов в задачах регрессии

Пусть поставлена задача регрессии. Целью является поиск такой функции $g(\mathbf{x})$, которая бы аппроксимировала обучающую выборку наилучшим образом. Задана функция потерь $\mathcal{L}\big(y, g(\mathbf{x})\big) = \begin{cases} |y - g(\mathbf{x})| - \varepsilon, & |y - g(\mathbf{x})| > \varepsilon \\ 0, & |y - g(\mathbf{x})| \le \varepsilon \end{cases}$, где $\varepsilon > 0$ определяет допустимое отклонение для результата [8]. Считается, что этот параметр задает эксперт.

Будем искать решение задачи регрессии в линейном случае: $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle - w_0$. Функция потерь принимает вид $a(\mathbf{x}_i) = |\langle \mathbf{w}, \mathbf{x}_i \rangle - w_0 - y_i|_\varepsilon$ для каждого вектора $(\mathbf{x}_i, y_i)$, где $|z|_\varepsilon = \max(0, |z| - \varepsilon)$. В таком случае функционал потерь принимает вид: $Q_\varepsilon(a, X) = \sum_{i=1}^{m} |\langle \mathbf{w}, \mathbf{x}_i \rangle - w_0 - y_i|_\varepsilon + \tau \langle \mathbf{w}, \mathbf{w} \rangle^2 \to \min_{\mathbf{w}, w_0}$. Последнее слагаемое удерживает коэффициенты **w** от бесконечного возрастания.

Решение зависит от скалярного произведения объектов, а не от самих объектов. Минимизация в данном случае эквивалентна задаче квадратичного программирования с ограничениями типа неравенств. Покажем это. Положим $C = \frac{1}{2\tau}$, и введем дополнительные переменные $\xi_i^+ = (a(\mathbf{x}_i) - y_i - \varepsilon)_+$, $\xi_i^- = (-a(\mathbf{x}_i) + y_i - \varepsilon)_-$, $i = 1, \dots, m$, значения которых равны потере при завышенном и заниженном ответах соответственно. Теперь можем записать задачу минимизации в виде задачи квадратичного программирования:

$$\begin{cases} \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle^2 + C \sum_{i=1}^{m} (\xi_i^+ + \xi_i^-) \to \min\limits_{\mathbf{w}, w_0, \xi_i^+, \xi_i^-}, \\ \langle \mathbf{w}, \mathbf{x}_i \rangle - w_0 \le y_i + \varepsilon + \xi_i^+, \quad i = 1, \dots, m, \\ \langle \mathbf{w}, \mathbf{x}_i \rangle - w_0 \ge y_i - \varepsilon + \xi_i^-, \quad i = 1, \dots, m, \\ \xi_i^- \ge 0, \quad i = 1, \dots, m, \\ \xi_i^+ \ge 0, \quad i = 1, \dots, m. \end{cases}$$

Решается двойственная задача. Лагранжиан задается через двойственные переменные $\lambda_i^+, \lambda_i^-, i = 1, \dots, m$, а скалярные произведения можно заменить ядром $K(\mathbf{x}_i, \mathbf{x}_j)$. Получим результат:

$$\begin{cases} \sum_{i=1}^{m} \left((\lambda_i^- - \lambda_i^+) y_i - \varepsilon(\lambda_i^+ + \lambda_i^-) \right) - \frac{1}{2} \sum_{i,j=1}^{m} (\lambda_i^- - \lambda_i^+)(\lambda_j^- - \lambda_j^+) K(\mathbf{x}_i, \mathbf{x}_j) \to \max\limits_{\lambda^+, \lambda^-}, \\ 0 \le \lambda_i^+ \le C, \ \ i = 1, \dots, m, \\ 0 \le \lambda_i^- \le C, \ \ i = 1, \dots, m, \\ \sum_{i=1}^{m} (\lambda_i^+ + \lambda_i^-) = 0. \end{cases}$$

Таким образом, все объекты $\mathbf{x}_i$, $i = 1, \dots, m$ делятся на 5 типов:

1. $|a(\mathbf{x}_i) - y_i| < \varepsilon, \ \ \lambda_i^+ = \lambda_i^- = \xi_i^+ = \xi_i^- = 0;$
2. $a(\mathbf{x}_i) = y_i + \varepsilon, \ \ 0 < \lambda_i^+ < C, \ \lambda_i^- = \xi_i^+ = \xi_i^- = 0;$
3. $a(\mathbf{x}_i) = y_i + \varepsilon, \ \ 0 < \lambda_i^- < C, \ \lambda_i^+ = \xi_i^+ = \xi_i^- = 0;$
4. $a(\mathbf{x}_i) > y_i + \varepsilon, \ \ \lambda_i^+ = C, \ \ \lambda_i^- = 0, \ \xi_i^+ = a(\mathbf{x}_i) - y_i - \varepsilon, \ \xi_i^- = 0;$
5. $a(\mathbf{x}_i) < y_i - \varepsilon, \ \ \lambda_i^+ = 0, \ \ \lambda_i^- = C, \ \xi_i^+ = 0, \ \xi_i^- = y_i - a(\mathbf{x}_i) - \varepsilon.$

Объекты 2-5 называются опорными и учитываются при определении вектора весов. На объектах 1-3 потери равны нулю, а на 4 и 5 потери больше нуля. Уравнение регрессии выражается через двойственные переменные: $a(\mathbf{x}) = \sum_{i=1}^{m} (\lambda_i^- - \lambda_i^+) K(\mathbf{x}_i, \mathbf{x}) - w_0$, где параметр w_0 определяется из уравнений: $\langle \mathbf{w}, \mathbf{x}_i \rangle - w_0 = \begin{cases} y_i + \varepsilon, \ \mathbf{x}_i \text{ принадлежит } 2 - \text{ому типу}, \\ y_i - \varepsilon, \ \mathbf{x}_i \text{ принадлежит } 3 - \text{ому типу}. \end{cases}$

Параметр w_0 можно определить из любого уравнения, но чтобы избежать вычислительных ошибок берется медиана полученного множества для w_0. Параметр C подбирается по скользящему контролю, что является трудоёмкой процедурой.

2.3.6. Методы решения задачи восстановления зависимости коллективами распознающих алгоритмов

Имеется выборка $\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, $\mathbf{x} = (x_1, x_2, \dots, x_t)$, $x_j \in \mathbf{M}_j$, $\mathbf{M}_\text{j}$ – множество произвольной природы; $y \in \mathbb{R}$. Обозначим m' число различных значений основного признака в обучающей выборке: $m' = |\{y_i\}_{i=1}^{m}|$.

Общий подход к решению задачи восстановления зависимостей коллективами распознающих алгоритмов состоит из подзадач:

1. формирование разбиений вещественной оси на интервалы;
2. формирование обучающих выборок задач распознавания, выбор классификаторов, их обучение;
3. вычисление оценок параметров модели восстановления зависимости как коллективного решения классификаторов.

Рассмотрим задачу формирования разбиения вещественной оси на интервалы Δ_k, приведем алгоритм решения данной задачи методом динамического программирования.

Строится разбиение $D = \{d_0, d_1, \dots, d_n\}$ области значений основного признака на интервалы $\Delta_1 = (d_0, d_1], \dots, \Delta_n = (d_{n-1}, d_n]$. Без ограничения общности считаем, что объекты упорядочены по возрастанию значения основного признака, тогда $y_1, \dots, y_{m_1} \in \Delta_1$, $y_{m_1+1}, \dots, y_{m_2} \in \Delta_2$, ..., $y_{m_{n-1}+1}, \dots, y_{m_n} \in \Delta_n$. Обозначим $c_k = \frac{\sum_{i=m_{k-1}+1}^{m_k} y_i}{m_k - m_{k-1}}$, $k = 1, \dots, n$.

В задаче динамического программирования используется функционал $Q(\boldsymbol{\Delta}) = \sum_{k=1}^{n} \sum_{i=m_{k-1}+1}^{m_k} (y_i - c_k)^2$, оптимизация проводится по номерам граничных объектов m_k, $k = 1, \dots, n$.

Построение разбиения области значений основного признака осуществляется на основе обучающей выборки и не зависит от алгоритмов распознавания, используемых в дальнейшем.

Рассмотрим модель восстановления зависимости «байесовский корректор» [5].

Возьмем число $2 \leq l \leq n$ и определим N разбиений отрезка R на l интервалов, поставив каждому разбиению в соответствие вектор с

целочисленными компонентами $\mathbf{k}_i = \left(0, k_i^{(1)}, \dots, k_i^{(l-1)}, n\right),\ i = 1, \dots, N,$ $k_i^{(j)} < k_i^{(j+1)} < n$. Каждое разбиение отрезка R определяет разбиение множества $\mathbf{M} = M_1 \times \dots \times M_k$ на l подмножеств $K_1^i, \dots, K_l^i$ (классов): $\mathbf{M} = \bigcup_{t=1}^{l} K_t^i,\ \nu \neq \mu \implies K_\nu^i \cap K_\mu^i = \emptyset$ согласно правилу: объект $\mathbf{x}$ принадлежит классу K_j^i разбиения $\mathbf{K}^i = \{K_1^i, \dots, K_l^i\}$ тогда и только тогда, когда $\mathrm{y} = f(\mathbf{x}) \in \Delta_k$ и $k \in \left(k_{i,j}, k_{i,j+1}\right]$. Каждое разбиение $\mathbf{K}^i$ определяет стандартную задачу распознавания Z_i с l классами. Пусть A_i – некоторый алгоритм решения задачи распознавания Z_i, относящий произвольный объект $\mathbf{x}$ к одному из классов $K_{a_i}^i$, $a_i = 1, \dots, l$.

Считаем декартово произведение $\mathbf{K}^1 \times \dots \times \mathbf{K}^l \times \boldsymbol{\Delta}$ множеством элементарных событий, событие $\left(K_{a_1}^1, \dots, K_{a_N}^N, \Delta_k\right)$ означает: «объект $\mathbf{x}$ отнесен алгоритмом A_1 в класс a_1, ..., алгоритмом A_N в класс a_N, $\mathrm{y} = f(\mathbf{x}) \in \Delta_k$». Вероятность такого события обозначается как $\mathrm{P}\left(K_{a_1}^1, \dots, K_{a_N}^N, \Delta_k\right)$ или $\mathrm{P}(\mathbf{x}, \Delta_k)$.

По формуле Байеса имеем:

$$\mathrm{P}\left(\Delta_k \middle| K_{a_1}^1, \dots, K_{a_N}^N\right) = \frac{\mathrm{P}\left(K_{a_1}^1, \dots, K_{a_N}^N, \Delta_k\right)}{\mathrm{P}\left(K_{a_1}^1, \dots, K_{a_N}^N\right)} = \frac{\mathrm{P}(\Delta_k)}{\mathrm{P}\left(K_{a_1}^1, \dots, K_{a_N}^N\right)} \mathrm{P}\left(K_{a_1}^1, \dots, K_{a_N}^N \middle| \Delta_k\right).$$

Пусть классификаторы статистически независимы, тогда $\mathrm{P}\left(K_{a_1}^1, \dots, K_{a_N}^N \middle| \Delta_k\right) = \prod_{i=1}^{N} \mathrm{P}\left(K_{a_i}^i \middle| \Delta_k\right)$, $\mathrm{P}\left(K_{a_1}^1, \dots, K_{a_N}^N\right) = \prod_{i=1}^{N} \mathrm{P}\left(K_{a_i}^i\right)$, и формулу Байеса можно записать в виде:

$$\mathrm{P}(\Delta_k | \mathbf{x}) = \mathrm{P}\left(\Delta_k \middle| K_{a_1}^1, \dots, K_{a_N}^N\right) = \frac{\mathrm{P}(\Delta_k)}{\prod_{i=1}^{N} \mathrm{P}\left(K_{a_i}^i\right)} \prod_{i=1}^{N} \mathrm{P}\left(K_{a_i}^i \middle| \Delta_k\right). \qquad (2)$$

Байесовским корректором называется функция F: $\left(\mathrm{P}(\Delta_1 | \mathbf{x}), \dots, \mathrm{P}(\Delta_n | \mathbf{x})\right) \to \mathbb{R}$.

«Ответом по среднему» байесовского корректора для объекта $\mathbf{x}$ назовем величину $\bar{y} = \sum_{k=1}^{n} \mathrm{P}(\Delta_k | \mathbf{x}) c_k$.

«Ответом по максимуму» байесовского корректора для объекта **x** назовем величину $\hat{y} = c_k$, где $k = \arg\max_{j=1,\dots,n} \mathrm{P}(\Delta_j|\mathbf{x})$.

Алгоритм восстановления зависимости:

1. формирование разбиений вещественной оси на интервалы Δ_k, $k = 1, \dots, n$;
2. задание разбиений $\mathbf{K}^i$, $i = 1, \dots, N$, формирование обучающих выборок задач распознавания Z_i, $i = 1, \dots, N$, выбор классификаторов A_i, $i = 1, \dots, N$, их обучение;
3. вычисление оценок вероятностей $\mathrm{P}(K_j^i|\Delta_k)$, $\mathrm{P}(\Delta_k)$, $\mathrm{P}(K_j^i)$, $i = 1, \dots, N$, $j = 1, \dots, l$, $k = 1, \dots, n$.

Алгоритм вычисления значения зависимой величины:

1. классификация алгоритмами $A_1, \dots, A_N$ объекта **x**;
2. вычисление значений условных вероятностей $\mathrm{P}(\Delta_k|\mathbf{x})$, $k = 1, \dots, n$ по формулам (2);
3. вычисление «ответа по среднему» $\bar{y}$ или «ответа по максимуму» $\hat{y}$.

Для вычисления параметров модели восстановления зависимости ставится следующая оптимизационная задача:

$$F = \sum_{i=1}^{m} (y_i - \bar{y}_i)^2 \longrightarrow \min_{e_{i,j}^{(k)}, \mathrm{P}(\Delta_k), \mathrm{P}(K_j^i)}$$

при ограничениях:

$$\sum_{a_i=1}^{l} \mathrm{P}(K_{a_i}^i) = 1, \ i = 1, \dots, N;$$

$$\sum_{k=1}^{n} \mathrm{P}(\Delta_k) = 1, \ \mathrm{P}(\Delta_k) > 0, \ k = 1, \dots, n;$$

$$\sum_{k=1}^{n} \mathrm{P}(\Delta_k) e_{i,a_i}^{(k)} = \mathrm{P}(K_{a_i}^i), \ e_{i,a_i}^{(k)} \geq 0, \ i = 1, \dots, N, \ a_i = 1, \dots, l.$$

Аналитическим решением данной задачи являются величины:

$$\mathrm{P}(K_1^i|\Delta_k) = \begin{cases} 0, & i < k, \\ 1, & i = k, \\ \dfrac{\alpha_i}{1 - \sum_{t=1}^{i-1} \alpha_t}, & i > k; \end{cases}$$

$$\mathrm{P}(K_2^i|\Delta_k) = 1 - \mathrm{P}(K_1^i|\Delta_k);$$

$$\mathrm{P}\left(K_1^i\right) = \frac{\alpha_i}{1 - \sum_{t=1}^{i-1} \alpha_t};$$

$$\mathrm{P}\left(K_2^i\right) = 1 - \mathrm{P}\left(K_1^i\right);$$

$$i = 1, \dots, N;\ k = 1, \dots, n.$$

Рассмотрим модель восстановления зависимости «линейный корректор».

Возьмем число $2 \le l \le n$ и определим N разбиений отрезка R на l интервалов, поставив каждому разбиению в соответствие вектор с целочисленными компонентами $\mathbf{k}_i = \left(0, k_i^{(1)}, \dots, k_i^{(l-1)}, n\right)$, $i = 1, \dots, N$, $k_i^{(j)} < k_i^{(j+1)} < n$. Каждое разбиение отрезка R определяет разбиение множества $\mathbf{M} = M_1 \times \dots \times M_k$ на l подмножеств $K_1^i, \dots, K_l^i$ (классов): $\mathbf{M} = \bigcup_{t=1}^{l} K_t^i$, $\nu \ne \mu \implies K_\nu^i \cap K_\mu^i = \emptyset$ согласно правилу: объект $\mathbf{x}$ принадлежит классу K_j^i разбиения $\mathbf{K}^i = \{K_1^i, \dots, K_l^i\}$ тогда и только тогда, когда $\mathrm{y} = f(\mathbf{x}) \in \Delta_k$ и $k \in \left(k_{i,j}, k_{i,j+1}\right]$. Каждое разбиение $\mathbf{K}^i$ определяет стандартную задачу распознавания Z_i с l классами. Пусть A_i – некоторый алгоритм решения задачи распознавания Z_i, относящий произвольный объект $\mathbf{x}$ к одному из классов $K_{a_i}^i$, $a_i = 1, \dots, l$.

Обозначим $u_i^{(j)} = \frac{\sum_{k=k_{i,j+1}}^{k_{i,j+1}} c_k}{k_{i,j+1} - k_{i,j}}$, $i = 1, \dots, N$, $j = 1, \dots, l$. Составим вектор ответов $\mathbf{u}(\mathbf{x}) = \left(1, u_1(\mathbf{x}), \dots, u_N(\mathbf{x})\right)$, $u_i(\mathbf{x}) = u_i^{(j)}$, классификатор A_i отнес объект $\mathbf{x}$ в класс K_j^i.

Линейным корректором называется функция $f\colon\ X \to \mathbb{R}$; $f(\mathbf{x}) = (\mathbf{u}(\mathbf{x}), \mathbf{w})$, где $\mathbf{w}$ – вектор весов, $\mathbf{w} \in \mathbb{R}^{N+1}$.

Вектор весов есть решение оптимизационной задачи $\sum_{i=1}^{m} (y_i - f(\mathbf{x}_i))$.

Алгоритм восстановления зависимости:

1. формирование разбиений вещественной оси на интервалы Δ_k, $k = 1, \dots, n$;

2. задание разбиений $\mathbf{K}^i$, $i = 1, \dots, N$, формирование обучающих выборок задач распознавания Z_i, $i = 1, \dots, N$, выбор классификаторов A_i, $i = 1, \dots, N$, их обучение;
3. вычисление вектора весов оценок $\mathbf{w}$.

Алгоритм вычисления значения зависимой величины:

1. классификация алгоритмами $A_1, \dots, A_N$ объекта $\mathbf{x}$;
2. вычисление вектора ответов $\mathbf{u}(\mathbf{x})$;
3. вычисление значения $f(\mathbf{x})$.

Для обучения корректной модели восстановления зависимости с разбиением области значений основного признака на m' интервалов необходимо обучить $m' - 1$ классификаторов, т.е. число классификаторов сравнимо с количеством объектов обучающей выборки.

3. АЛГОРИТМЫ РАСПОЗНАВАНИЯ, ОСНОВАННЫЕ НА ВЫЧИСЛЕНИИ ОЦЕНОК

Исходной информацией являются описания объектов (ситуаций, предметов, явлений или процессов) в виде векторов значений признаков $\mathbf{x} = (x_1, x_2, \dots, x_n)$, где признаки, характеризуют различные стороны-свойства объекта. Одно из «свойств» $y = y(\mathbf{x})$ объектов (не входящее в состав признаков) считается «основным». Свойство y принимает конечное число значений и для некоторых объектов считается известным. Предполагается, что существует прямая связь между признаками и основным свойством (неизвестная пользователю).

Задача распознавания (прогноза, идентификации, «классификации с учителем») по прецедентам состоит в определении значения свойства $y(\mathbf{x})$ объекта по информации $\{\mathbf{x}_i, y_i\},\ i = 1, 2, \dots, m$ (обучающей или эталонной выборке). Обычно вместо термина «основное свойство объекта» используют термин «класс объекта». Объекты, имеющие равные значения основного свойства считаются принадлежащими одному множеству (образу, классу объектов), и задача распознавания по прецедентам формулируется как задача отнесения объекта к одному из классов.

Пусть некоторое множество объектов является объединением конечного числа непересекающихся подмножеств, именуемых классами: $M = \bigcup_{i=1}^{l} K_i,\ K_i \cap K_j = \emptyset,\ i \neq j$. Данное разбиение известно лишь частично в виде выборки объектов $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$ из данного множества, содержащей представителей всех классов. Для определенности будем считать, что $\mathbf{x}_{m_{i-1}}, \mathbf{x}_{m_{i-1}+1}, \dots, \mathbf{x}_{m_i} \in K_i, m_0 = 0, m_l = m,\ i = 1,2, \dots, l.$

Алгоритмы вычисления оценок (АВО) определяются как последовательное выполнение шести этапов, для каждого из которых имеются различные пути реализации [1, 4]. Ниже будут приведены лишь некоторые основные способы их выполнения.

3.1. Основные определения и этапы алгоритмов вычисления оценок

3.1.1. Задание системы опорных множеств алгоритма

Первым шагом определения АВО является задание множества подсистем признаков, по которым осуществляется сравнение объектов. Пусть Ω_A – некоторая система подмножеств множества $\{1,2,\dots,n\}$, называемая системой опорных множеств алгоритма распознавания A. Элементы $\Omega = \{i_1, i_2, \dots, i_k\} \in \Omega_A$ называются опорными множествами алгоритма. Они определяют номера признаков, по которым сравниваются части эталонных и распознаваемых объектов. Каждому подмножеству $\Omega = \{i_1, i_2, \dots, i_k\}$ можно поставить во взаимно однозначное соответствие булевский вектор $\boldsymbol{\omega} = \{\omega_1, \omega_2, \dots, \omega_n\}$, в котором $\omega_j = 1,\ j = i_1, i_2, \dots, i_k$, а остальные компоненты равны нулю. Значение k находится из решения задачи обучения (оптимизации модели) или задается экспертом. Широко распространенными подходами к выбору Ω_A являются следующие два:

a) $\Omega_A = \{\Omega\colon |\Omega| = k\}$;

b) $\Omega_A = \{\Omega\},\ \Omega \subseteq \{1,2,\dots,n\},\ \Omega \neq \emptyset$.

Второй способ выбора системы опорных множеств, как всевозможных подсистем $\{1,2,\dots,n\}$, является «усреднением» первого и не требует нахождения неизвестных параметров.

3.1.2. Задание функции близости

Пусть фиксировано некоторое опорное множество Ω и соответствующий ему характеристический вектор $\boldsymbol{\omega}$. Фрагмент $x_{i_1}, x_{i_2}, \dots, x_{i_k}$ объекта $\mathbf{x}$, соответствующий всем единичным компонентам вектора $\boldsymbol{\omega}$, называется $\boldsymbol{\omega}$ −частью объекта, и обозначается $\boldsymbol{\omega}\mathbf{x}$. Под функцией близости $B_\Omega(\mathbf{x}_i, \mathbf{x}_j)$ будет пониматься функция от соответствующих $\boldsymbol{\omega}$ −частей сравниваемых объектов, принимающая значение 1 («объекты близки») или 0 («объекты далеки»). Приведем примеры подобных функций:

a) $B_\Omega(\mathbf{x}_\nu, \mathbf{x}_\mu) = \begin{cases} 1, & |x_{\nu i} - x_{\mu i}| \le \varepsilon_i, \ \forall i: \omega_i = 1, \ \boldsymbol{\omega} \leftrightarrow \Omega, \\ 0, & \text{иначе.} \end{cases}$

Здесь $(\varepsilon_1, \varepsilon_2, \dots, \varepsilon_n)$ – неотрицательные параметры, именуемые «точностями измерения признаков».

b) $B_\Omega(\mathbf{x}_\nu, \mathbf{x}_\mu) = \begin{cases} 1, & \sum_{i \in \Omega} |x_{\nu i} - x_{\mu i}| \le \varepsilon, \\ 0, & \text{иначе.} \end{cases}$

Здесь ε также некоторый неотрицательный параметр алгоритма.

3.1.3. Оценка близости объекта x к эталонному объекту $\mathbf{x}_i$ для заданной $\boldsymbol{\omega}$ −части

Данная числовая величина формируется на основе функции близости и, возможно, дополнительных параметров:

a) $\Gamma_\Omega(\mathbf{x}_i, \mathbf{x}) = \mathrm{B}_\Omega(\mathbf{x}_i, \mathbf{x})$.

b) $\Gamma_\Omega(\mathbf{x}_i, \mathbf{x}) = \mathrm{w}_\Omega \mathrm{B}_\Omega(\mathbf{x}_i, \mathbf{x})$, где w_Ω – «вес» опорного множества.

c) $\Gamma_\Omega(\mathbf{x}_i, \mathbf{x}) = \gamma_i\left(\sum_{i:\omega_i=1} p_i\right)\mathrm{B}_\Omega(\mathbf{x}_i, \mathbf{x})$. Здесь γ_i – параметры, характеризующие степень важности объекта $\mathbf{x}_i$ (информативность объекта), а $p_1, p_2, \dots, p_n$ – веса (информативность) признаков.

3.1.4. Оценка объекта x за класс K_j для заданной $\boldsymbol{\omega}$ −части

$\Gamma_j^\Omega(\mathbf{x}) = \frac{1}{m_j - m_{j-1}} \sum_{\mathbf{x}_i \in K_j} \Gamma_\Omega(\mathbf{x}_i, \mathbf{x})$.

3.1.5. Оценка объекта x за класс K_j

a) $\Gamma_j(\mathbf{x}) = \sum_{\Omega \in \Omega_\mathrm{A}} \Gamma_j^\Omega(\mathbf{x})$.

b) $\Gamma_j(\mathbf{x}) = \nu_j \sum_{\Omega \in \Omega_\mathrm{A}} \Gamma_j^\Omega(\mathbf{x})$, где ν_j – «вес» класса K_j. Например, в статистической теории распознавания аналогами параметров ν_j являются априорные вероятности классов, которые характеризуют, насколько часто встречаются объекты различных классов.

3.1.6. Решающее правило

Решающее правило – правило (алгоритм, оператор), позволяющее относить объект по вектору оценок за классы в один из классов, или вырабатывать для объекта «отказ от распознавания». Решающее правило r алгоритма

распознавания A вычисляет для распознаваемого объекта $\mathbf{x}$ булевский вектор $\boldsymbol{\alpha}^A(\mathbf{x}) = \left(\alpha_1^A(\mathbf{x}), \alpha_2^A(\mathbf{x}), \dots, \alpha_l^A(\mathbf{x})\right)$, $\alpha_i^A(\mathbf{x}) \in \{0,1\}$, где $\alpha_i^A(\mathbf{x}) = 1$ обозначает отнесение объекта в класс K_i, $\alpha_i^A(\mathbf{x}) = 0$ – принятие решения: «объект $\mathbf{x}$ не принадлежит классу K_i». По вектору $\alpha_i^A(\mathbf{x}) = 1$ принимается решение об отнесении объекта к одному или нескольким классам. В ситуациях, когда оценки объекта малы за все классы (объект является принципиально новым, аналоги которого отсутствуют в обучающей выборке), или он имеет две или более близкие максимальные оценки за различные классы (объект лежит на границе классов) обычно принимают решение об отказе от распознавания. Простейшее решающее правило – отнесение объекта в класс, за который он имеет максимальную оценку, и отказ от распознавания в случае двух и более максимальных оценок.

$$\alpha_j^A(\mathbf{x}) = \begin{cases} 1, \ \Gamma_j(\mathbf{x}) > \Gamma_i(\mathbf{x}), \ \ i = 1,2,\dots,l, \ \ i \neq j, \\ 0, \qquad\qquad\qquad\qquad\qquad \text{иначе.} \end{cases}$$

Наличие двух или более единиц интерпретируется как «объект, вероятно, принадлежит нескольким классам». Когда бинарный вектор состоит из одних нулей, говорят, что данный объект – выброс, он не похож ни на один из классов, близких его аналогов ранее не наблюдалось.

3.2. Эффективные формулы вычисления оценок

После последовательной подстановки выражений на этапах 2-5 могут быть получены различные общие формулы для вычисления оценок $\Gamma_j(\mathbf{x})$. Например, если выбрать первые примеры реализации различных этапов, будет получена следующая общая формула для вычисления оценок объекта $\mathbf{x}$ за классы K_j, $j = 1,2,\dots,l$:

$$\Gamma_j(\mathbf{x}) = \frac{1}{m_j - m_{j-1}} \sum_{\nu=m_{j-1}+1}^{m_j} \sum_{\Omega \in \Omega_A} \mathrm{B}_\Omega(\mathbf{x}_\nu, \mathbf{x}). \qquad (3)$$

При выборе системы опорных множеств согласно вариантам a) или b) прямое вычисление оценок (3) представляется весьма трудоемким. Действи-

тельно, при вычислении оценок (3) согласно а) требуется mC_n^k вычислений значений функции близости. В действительности необходимость выполнения всех данных вычислений отсутствует, поскольку существуют эффективные комбинаторные формулы вычисления оценок при многих вариантах реализации этапов 2-5 и различных системах опорных множеств.

Теорема. Пусть в модели вычисления оценок используются варианты а) выполнения этапов, тогда справедлива формула

$$\Gamma_j(\mathbf{x}) = \frac{1}{m_j - m_{j-1}} \sum_{\mathbf{x}_i \in K_j} C_{d(\mathbf{x},\mathbf{x}_i)}^k,$$

где $d(\mathbf{x}, \mathbf{x}_i) = |\{\nu: |x_{i\nu} - x_\nu| \leq \varepsilon_\nu,\ \nu = 1,2,\dots,n\}|$.

4. СОЗДАННЫЙ МЕТОД ВОССТАНОВЛЕНИЯ ЗАВИСИМОСТЕЙ

4.1. Описание созданного метода

Создан метод поиска функциональных зависимостей по выборкам прецедентов с использованием алгоритмов распознавания. Он основан на построении набора кодирований зависимой величины по обучающим данным и использовании логико-статистического подхода.

Пусть задана обучающая выборка $\{\mathbf{x}_i, y_i\}$, $i = 0, 1, \dots, m-1$, где $\mathbf{x}_i \in R^n$, $y_i \in R$, $x_{ij} \in M_j$, M_j – множество допустимых описаний j –го признака. Для простоты считаем, что M_j – числовые множества (т.е. признаки кодируются числами). Считаем, что имеет место зависимость $y = f(\mathbf{x})$, которая существует и которую надо определить по данным обучения. Выше был предложен подход к решению данной задачи, основанный на построении различных N задач классификации и объединении результатов распознавания нового объекта с помощью линейного или байесовского корректора. Можно предложить другой вид линейного корректора. Суть данного подхода состоит в следующем. Сначала вводятся N задач классификации по исходной обучающей выборке. Данные задачи соответствуют разбиениям вещественной оси на l множеств по значениям зависимой переменной обучающей выборки. Для каждой задачи находится оптимальный в режиме скользящего контроля алгоритм вычисления оценок (АВО). Каждый алгоритм вычисляет интервал для зависимой величины (т.е. класс). В качестве значения зависимой величины берем выборочное среднее по рассматриваемому интервалу обучающей выборки. Это значение определяется для каждого объекта и каждой задачи (или распознаваемого объекта). Это стандартный (приближенный) вариант сведения регрессионной задачи к задаче классификации с учителем. Здесь надо найти только $l-1$ точку разбиения. Но у нас много разбиений! Значение зависимой величины находится как результат линейной комбинации y_i объектов класса $K^i_{\nu_i}$, $i = 0, 1, \dots, N-1$, при этом классом $K^i_{\nu_i}$ является результат классификации

объекта. Для точек $\mathbf{x}_i$ обучающей выборки (при обучении) это будут, например, выборочные средние соответствующих классов и точки y_i, для распознаваемых объектов – набор точек (например, выборочные средние по классам) и точка y (вычисляемая как их линейная комбинация). Таким образом, мы можем находить оптимальное разбиение интервала зависимой величины, оценивать «информативность» каждого из алгоритмов разбиения.

Рассмотрим данный подход подробнее. Можно поставить в соответствие исходной обучающей выборке некоторые задачи классификации на l классов с учителем $Z^i,\ i = 0, \dots, N-1$, в которых классы определяются как $K_v^j = \left\{\mathbf{x}_i\colon\ y_i \in I_v,\ i = jp, \dots, (j+1)p-1,\ p = \left[\frac{m}{N}\right] + 1\right\}\ (j = 0, \dots, N-1)$, где $I_v\ (v = 0, \dots, l-1)$ – разбиения вещественной оси точками y_t, $t = 0, \dots, m-1$ на l множеств. Пусть для каждой задачи классификации построен алгоритм вычисления оценок. Тогда значение зависимой величины произвольного объекта $\mathbf{x}'$ будем вычислять как $\tilde{y}(\mathbf{x}') = \sum_{i=0}^{N-1} \alpha_i y'^i$, где y'^i – (например) выборочное среднее за класс K_v^j, в который отнесен i –м алгоритмом классификации объект $\mathbf{x}'$. Таким образом, при заданных числах l и N задача состоит в построении АВО для каждой задачи классификации и нахождении наилучших весов $\alpha_i,\ i = 0,1, \dots, N-1$.

Пусть дана контрольная выборка объектов $\{\mathbf{x}'_i, y'_i\},\ i = 0, 1, \dots, q-1$. Тогда значения параметров $\alpha_i,\ i = 0,1, \dots, N-1$ будем вычислять методом наименьших квадратов, решая задачу

$$\sum_{j=0}^{q-1}\left(y'_j - \sum_{i=0}^{N-1}\alpha_i y_j'^i\right)^2 \xrightarrow[\alpha_0,\alpha_1,\dots,\alpha_{N-1}]{} \min. \qquad (4)$$

Здесь y'_j – значение зависимой величины для j –го контрольного объекта, а $y_j'^i$ – результат распознавания i –м алгоритмом. Меняя l и N, можно определять наилучшее решение (значение критерия (4)).

Рассмотрим вопрос о виде АВО. Известно, что существует много различных видов модели АВО. Мы рассмотрим два из них.

В первой модели рассматривается множество всевозможных опорных множеств мощности k и функция близости распознаваемого объекта $\mathbf{x}$ к некоторому эталонному $\mathbf{x}_t$ класса $K_\nu^j,\ j = 0, \dots, N-1,\ \nu = 0, \dots, l-1$ вида

$$B_\Omega(\mathbf{x}, \mathbf{x}_t) = \begin{cases} 1, |\mathrm{x}_\alpha - \mathrm{x}_{t\alpha}| \le \varepsilon_\alpha, \forall \alpha \in \Omega, \\ 0, \quad \text{иначе.} \end{cases}$$

Во второй модели при тех же опорных множествах рассматривалась функция

$$\tilde{B}_\Omega(\mathbf{x}_\alpha, \mathbf{x}, \mathbf{x}_\beta) = \begin{cases} 1, (\mathrm{x}_{\alpha i} \le \mathrm{x}_i \le \mathrm{x}_{\beta i}) \vee (\mathrm{x}_{\beta i} \le \mathrm{x}_i \le \mathrm{x}_{\alpha i}), \forall i \in \Omega, \\ 0, \quad \text{иначе} \end{cases} \forall \mathbf{x}_\alpha, \mathbf{x}_\beta \in K_\nu^j.$$

В первом случае оценка за класс K_ν^j имеет вид

$$\Gamma_\nu^j(\mathbf{x}) = \frac{1}{|K_\nu^j|} \sum_{\mathbf{x}_i \in K_\nu^j} \mathrm{C}^k_{d(\mathbf{x}, \mathbf{x}_i)},$$

где $d(\mathbf{x}, \mathbf{x}_i) = |\{t: |\mathrm{x}_t - \mathrm{x}_{it}| \le \varepsilon_t, t = 1,2, \dots, n\}|$, а во втором случае:

$$\Gamma_\nu^j(\mathbf{x}) = \frac{2}{|K_\nu^j|(|K_\nu^j| - 1)} \sum_{\mathbf{x}_\alpha, \mathbf{x}_\beta \in K_\nu^j, \alpha < \beta} \mathrm{C}^k_{d(\mathbf{x}_\alpha, \mathbf{x}, \mathbf{x}_\beta)},$$

где $d(\mathbf{x}_\alpha, \mathbf{x}, \mathbf{x}_\beta) = |\{t: (\mathrm{x}_{\alpha t} \le \mathrm{x}_t \le \mathrm{x}_{\beta t}) \vee (\mathrm{x}_{\beta t} \le \mathrm{x}_t \le \mathrm{x}_{\alpha t}),\ t = 1,2, \dots, n\}|$. Отметим, что в случае второй модели будет достаточным, если признаки являются порядковыми. Используя первую или вторую модель АВО и метод наименьших квадратов можно эффективно находить требуемый для рассматриваемой задачи предложенный линейный корректор. Несложно показать, что при $N \ge m$ данный подход является корректным (т.е. безошибочным).

Осталось только расписать процесс вычисления оценок в режиме скользящего контроля. Мы предполагаем, что контрольная выборка совпадает с обучающей выборкой. В случае первой модели, надо вычислить матрицу $D_1 = \|d_{ij}\|_{m \times m}$, где $d_{ij} = \mathrm{C}^k_{d(\mathbf{x}_i, \mathbf{x}_j)}$ (значение параметра k задается).

Тогда $\Gamma_\nu^j(\mathbf{x}_i) = \frac{1}{|K_\nu^j|}\sum_{\mathbf{x}_t \in K_\nu^j} d_{it}$, если $\mathbf{x}_i \notin K_\nu^j$ и $\Gamma_\nu^j(\mathbf{x}_i) = \frac{1}{|K_\nu^j|-1}\sum_{\substack{\mathbf{x}_t \in K_\nu^j \\ t \neq i}} d_{it}$, если $\mathbf{x}_i \in K_\nu^j$. В случае второй модели надо вычислить и использовать аналогично при вычислении оценок матрицу $D_2 = \left\| d_{\alpha\gamma\beta} \right\|_{m\times m\times m}$, где $d_{\alpha\gamma\beta} = \mathrm{C}^k_{d(\mathbf{x}_\alpha, \mathbf{x}_\gamma, \mathbf{x}_\beta)}$.

4.2. Алгоритм метода

Работа производилась с двумя репозиториями:

1. "Image Segmentation Data Set",
2. "Abalone Data Set".

Оба репозитория взяты с ресурса "The UCI Machine Learning Repository" (https://archive.ics.uci.edu/ml/datasets.html).

Некоторые сведения об этих репозиториях:

Репозиторий / Описание	Image Segmentation Dataset	Abalone Dataset
Число объектов (прецедентов)	2100	4177
Число признаков	19	8

Табл. 1: Характеристики используемых репозиториев.

Метод запрограммирован на языке Python 2.7. Вся работа производилась в интерактивной оболочке для языка Python, Jupyter Notebook. Перечислим основные шаги алгоритма:

1. Создаем массив, содержащий перемешанную обучающую выборку. Часто данные (как и в нашем случае) в файле записаны в отсортированном виде по какому-нибудь признаку (в нашем случае по основному признаку). Поэтому всегда следует перемешивать выборку прежде, чем производить по ней обучение. В ином случае алгоритм будет показывать плохое качество.
2. Определяем вектор $\boldsymbol{z}$ различных значений основного признака в обучающей выборке и их количество $\mathrm{length}(\boldsymbol{z})$.

3. Определяем столбец $\boldsymbol{Z}$ значений основного признака и среднее по нему (учитывая уже все его значения, а не только различные).
4. Для первой модели необходимо задать точности измерения признаков. Для этого находим минимумы и максимумы значений каждого признака, соответствующие разным значениям основного признака, находим их разницы и задаем точности измерения признаков как средние этих разниц по всем значениям основного признака.
5. Задаем главные гиперпараметры метода:
   - N – число задач классификации по исходной обучающей выборке,
   - l – число классов, на которые разбивается множество значений основного признака,
   - k – мощность опорных множеств, определяющих номера признаков, по которым сравниваются части эталонных и распознаваемых объектов. Для репозитория "Image Segmentation Dataset" $k = 9$, а для репозитория "Abalone Dataset" $k = 4$.
6. Получаем разбиение вещественной оси значениями основного признака обучающей выборки на l подмножеств. Для этого просто высчитываем количество num_z этих значений, которое будет содержаться в каждом множестве. Ранее для этих подмножеств вводилось обозначение I_ν $(\nu = 0, \dots, l-1)$.
7. Разбиваем наш массив на классы, в которых записаны прецеденты для определенной задачи классификации и с определенным значением основного признака. Для этого просто высчитываем количество num_prec_task объектов, которое будет содержаться в каждой задаче классификации. Таким образом, мы делим исходную задачу на N задач классификации, и, объекты в каждой задаче разбиваем еще на l подклассов, в зависимости от того, чему равны их значения основного признака и какому из подмножеств I_ν $(\nu = 0, \dots, l-1)$ они принадлежат. Таким образом, исходная обучающая выборка приобретает вид:

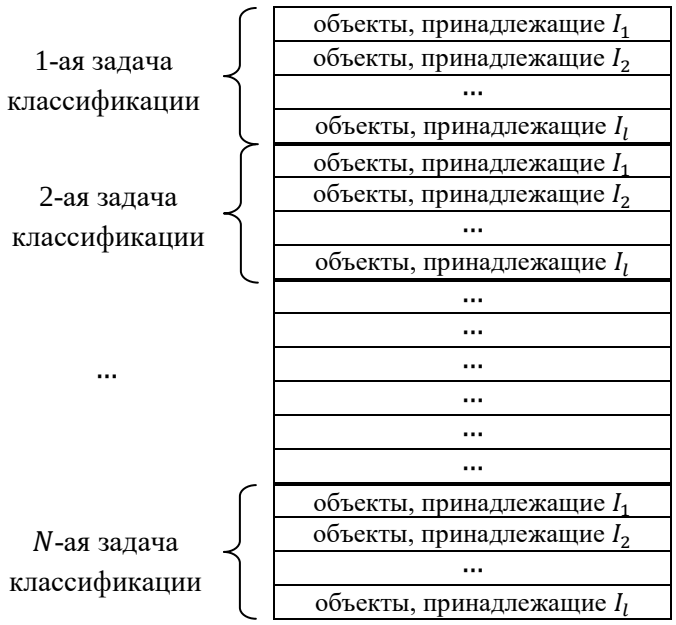

Рис. 1: Разбиение обучающей выборки на классы.

Поскольку исходная выборка заранее перемешивалась, может оказаться так, что в каких-то классах не будет ни одного элемента, т.е. когда в конкретной задаче классификации не нашлось ни одного объекта, чье значение основного признака лежало в конкретном подмножестве I_ν ($\nu = 0, \dots, l-1$). Полученные классы ранее обозначались как K_ν^j ($j = 0, \dots, N-1,\ \nu = 0, \dots l-1$).

8. Определяем все необходимые функции:
   - $d(\mathbf{x}, \mathbf{x}_i)$ (для случая первой модели),
   - $d(\mathbf{x}_\alpha, \mathbf{x}, \mathbf{x}_\beta)$ (для случая второй модели),
   - $\Gamma_\nu^j(\mathbf{x})$ – функцию, возвращающую оценку для объекта $\mathbf{x}$ за класс K_ν^j (для обеих моделей),
   - функцию, возвращающую выборочное среднее за класс K_ν^j,
   - функцию $z(\mathbf{x})$, возвращающую список выборочных средних за каждый класс, в который каждая задача классификации отправила объект $\mathbf{x}$, с дополнительным нулевым элементом, равным единице, для дальнейшего применения стохастического градиентого спуска. Происходит проверка равенства двух или большего числа максимальных

оценок за разные классы и проверка малости оценок за все классы. В первом случае, когда объект похож одинаково на два (или большее количество) класса, берем среднее по ним. Во втором случае, когда оценки малы за все классы (это означает, что объект не похож ни на что из обучающей выборки), в качестве значения его зависимой величины берем среднее по всем значениям основного признака.

9. Применив последнюю функцию, $z(\mathbf{x})$, для каждого прецедента $\mathbf{x}$, получим матрицу размера $m \times (N+1)$. Обозначим ее Z_v^j.
10. Определяем функцию, возвращающую линейную комбинацию столбцов матрицы Z_v^j с весами α_j $(j = 0, \dots, N)$, функцию, возвращающую среднеквадратичную ошибку прогноза.
11. Определяем функцию, реализующую один шаг стохастического градиентного спуска для линейной регрессии. Функция должна принимать матрицу Z_v^j, вектора $\boldsymbol{Z}$ и $\boldsymbol{\alpha}$, число train_ind – индекс строки матрицы Z_v^j, по которому считается изменение весов, а также η – шаг градиентного спуска. Результатом выполнения функции будет вектор обновленных весов.
12. Определяем функцию, реализующую стохастический градиентный спуск для линейной регрессии. Функция принимает на вход следующие аргументы:
    - матрицу Z_v^j,
    - столбец $\boldsymbol{Z}$,
    - $\boldsymbol{\alpha}_{init}$ – вектор начальных весов модели,
    - η – шаг градиентного спуска,
    - max_iter – максимальное число итераций градиентного спуска (по умолчанию 10^6),
    - max_weight_dist – минимальное евклидово расстояние между векторами весов на соседних итерациях градиентного спуска, при котором алгоритм прекращает работу (по умолчанию 10^{-8}).

На каждой итерации в вектор (список) должно записываться текущее значение среднеквадратичной ошибки. Функция должна возвращать вектор весов $\boldsymbol{\alpha}$, а также вектор (список) ошибок.

13. Запускаем 10^6 итераций стохастического градиентного спуска с вектором начальных весов $\boldsymbol{\alpha}_{init}$, состоящим из нулей.
14. Строим график зависимости среднеквадратичной ошибки от номера итерации для 10^6 итераций стохастического градиентного спуска.
15. Выводим вектор весов, к которому сошелся метод.
16. Выводим среднеквадратичную ошибку прогноза на последней итерации.

Данные операции проводим для каждой пары гиперпараметров N и l:

- $N = 2, \dots, N_{\max} - 1$, где $N_{\max} = \left[\frac{m}{2}\right] + 1$,
- $l = 2, \dots, l_{\max} - 1$, где $l_{\max} = \left[\frac{\text{length}(\boldsymbol{z})}{2}\right] + 1$.

При числе задач классификации, равном $N_{\max} - 1$, каждая задача будет работать только с двумя объектами. Аналогично, при числе классов, равном $l_{\max} - 1$, множество значений основного признака разбивается на пары. Делать больше ограничения сверху на эти параметры нет смысла, потому что тогда мы будем работать с единичными объектами в каждой задаче классификации.

Также учитываем, что при разных значениях параметров N или l может быть одни и те же значения num_z и num_prec_task. Поэтому для них мы не проводим описанные выше операции.

В итоге, для обоих репозиториев мы получаем зависимости среднеквадратических ошибок прогноза от двух гиперпараметров N и l. Если взять конкретную пару параметров N и l, при которых среднеквадратичная ошибка прогноза минимальна, взять соответствующий им массив весов $\boldsymbol{\alpha}$, применить функцию $z(\mathbf{x})$ к некоторому прецеденту $\mathbf{x}$, значение основного признака которого необходимо предсказать, и посчитать линейную комби-

нацию матрицы, возвращенной функцией $z(\mathbf{x})$, с весами $\boldsymbol{\alpha}$, получим искомое значение основного признака для объекта $\mathbf{x}$.

4.3. Экспериментальное исследование метода

Сначала приведем основные результаты работы для репозитория "Image Segmentation Dataset". В этом репозитории обучающая выборка состоит из 2100 объектов и для каждого объекта определены значения 19 признаков. Для первой модели мы определяем гиперпараметр k, мощность опорных множеств, определяющих номера признаков, по которым сравниваются части эталонных и распознаваемых объектов, равным 9. Также определяем $N_{\max} = 1051$ и $l_{\max} = 4$.

Отобразим основные полученные графики.

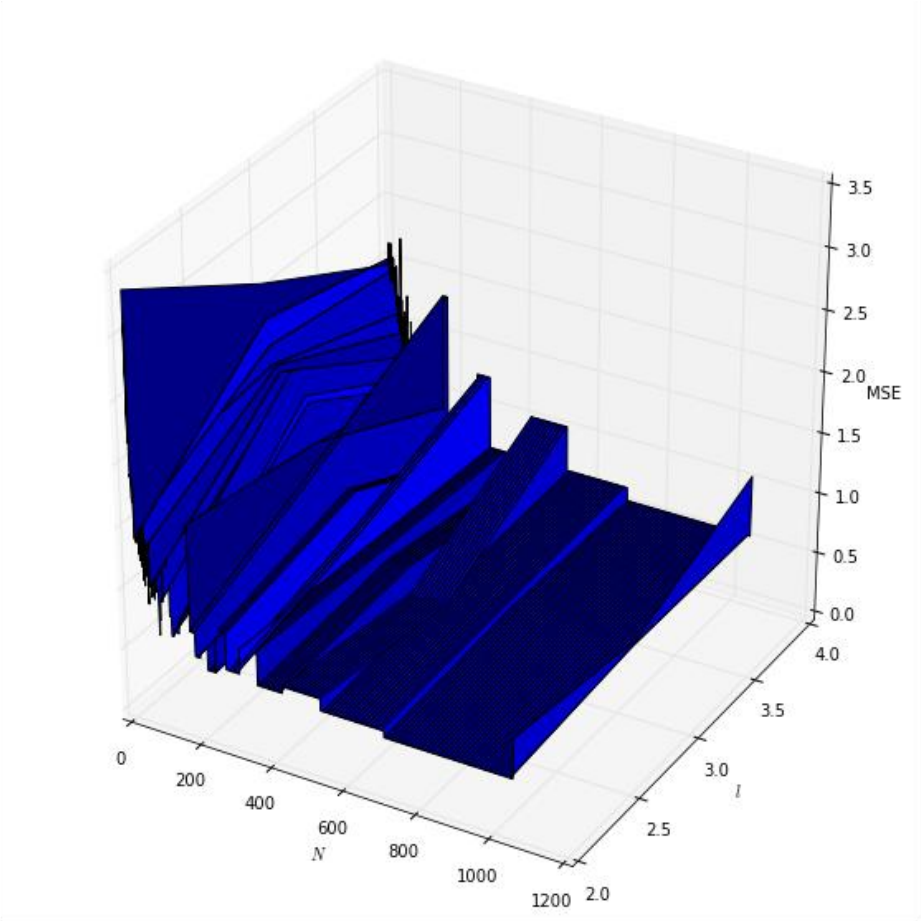

Рис. 2: График зависимости среднеквадратичной ошибки прогноза (MSE) от параметров N и l для первого репозитория в случае первой модели.

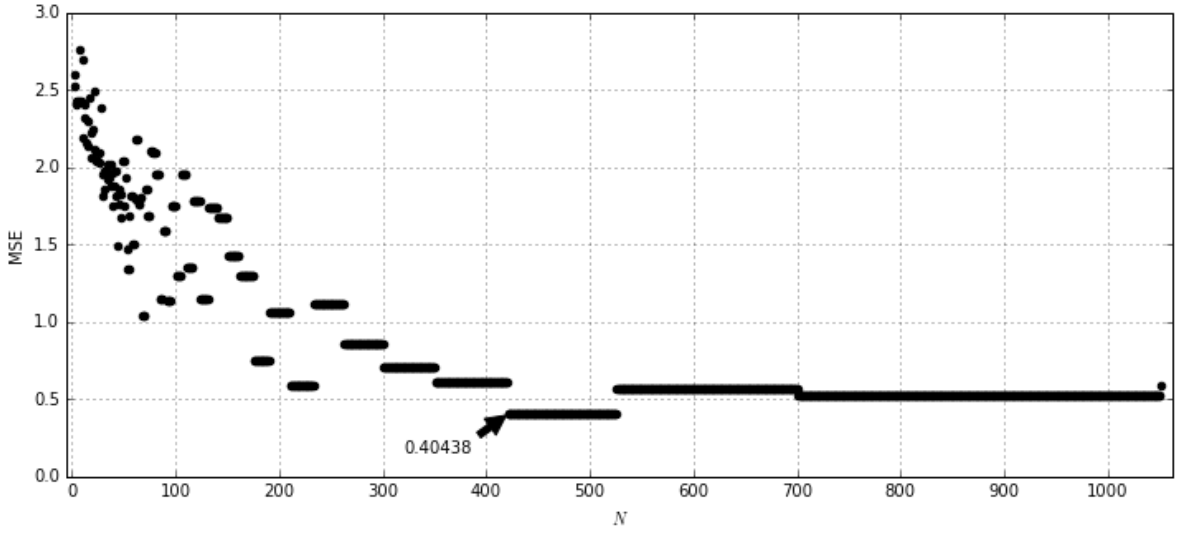

Рис. 3: График зависимости среднеквадратичной ошибки прогноза (MSE) от параметра N при $l = 3$ для первого репозитория в случае первой модели. Минимальное значение MSE в случае первой модели отмечено стрелкой.

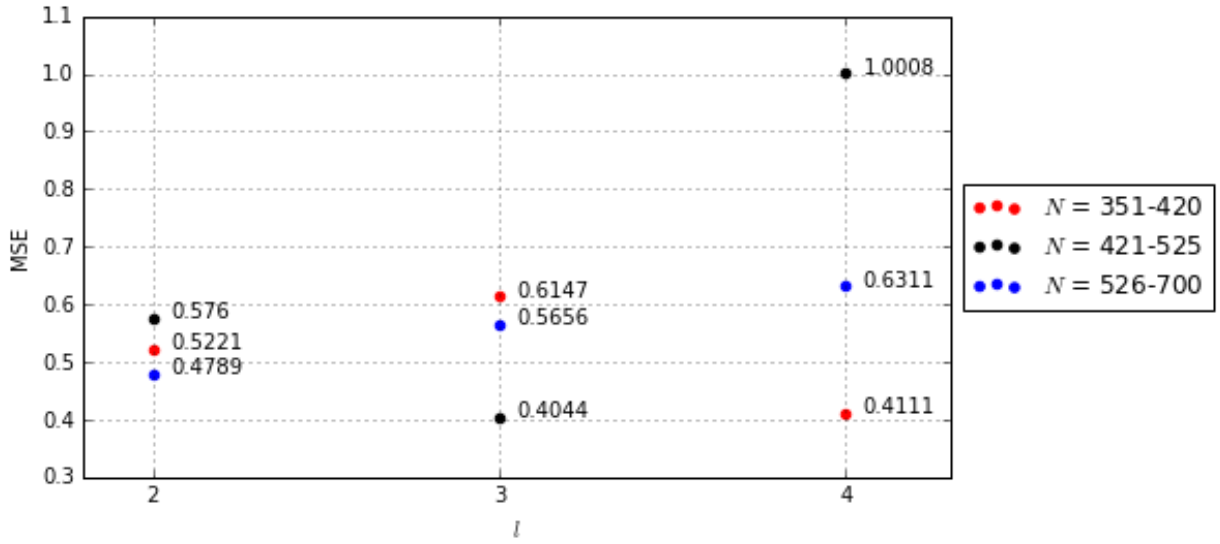

Рис. 4: График зависимости среднеквадратичной ошибки прогноза (MSE) от параметра l при различных значениях параметра N для первого репозитория в случае первой модели.

Таким образом, получили, что для первого репозитория в случае первой модели наименьшее значение среднеквадратичной ошибки прогноза равно 0.404381411103, и оно достигается при $N = 421$ и $l = 3$.

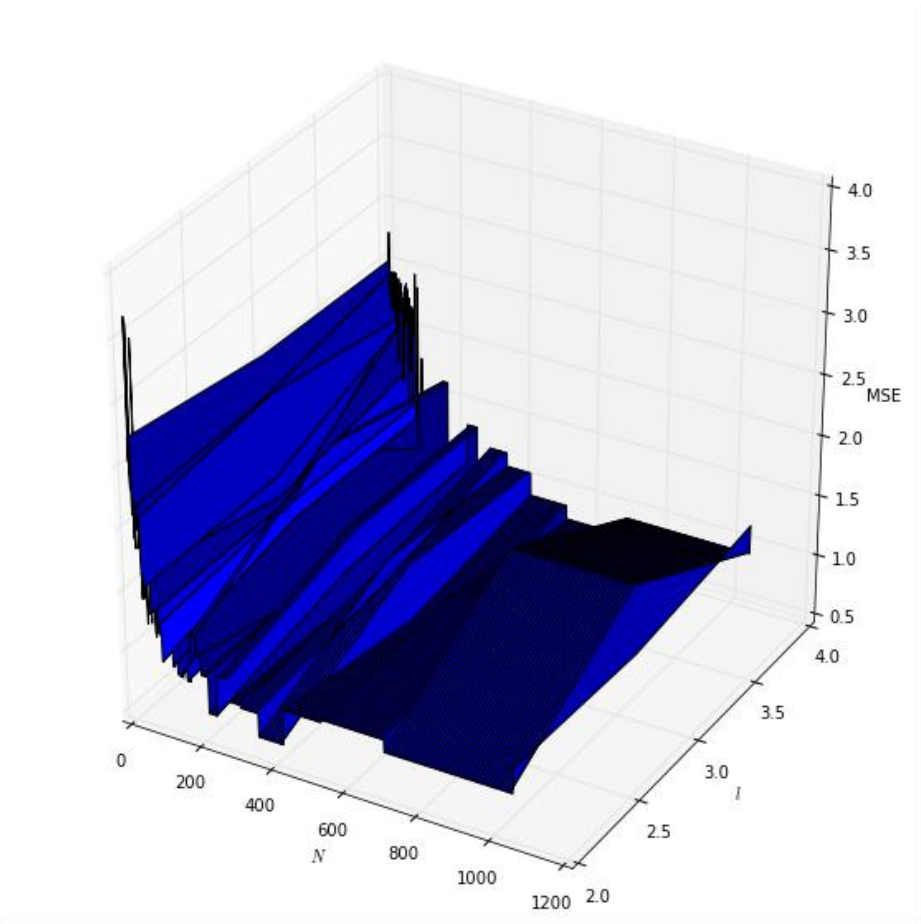

Рис. 5: График зависимости среднеквадратичной ошибки прогноза (MSE) от параметров N и l для первого репозитория в случае второй модели.

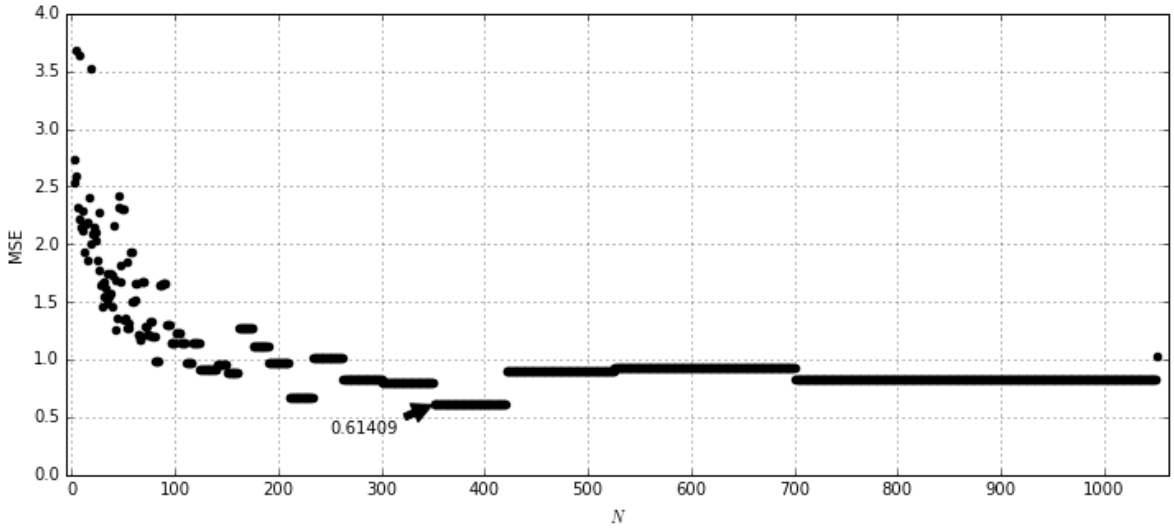

Рис. 6: График зависимости среднеквадратичной ошибки прогноза (MSE) от параметра N при $l = 2$ для первого репозитория в случае второй модели. Минимальное значение MSE в случае второй модели отмечено стрелкой.

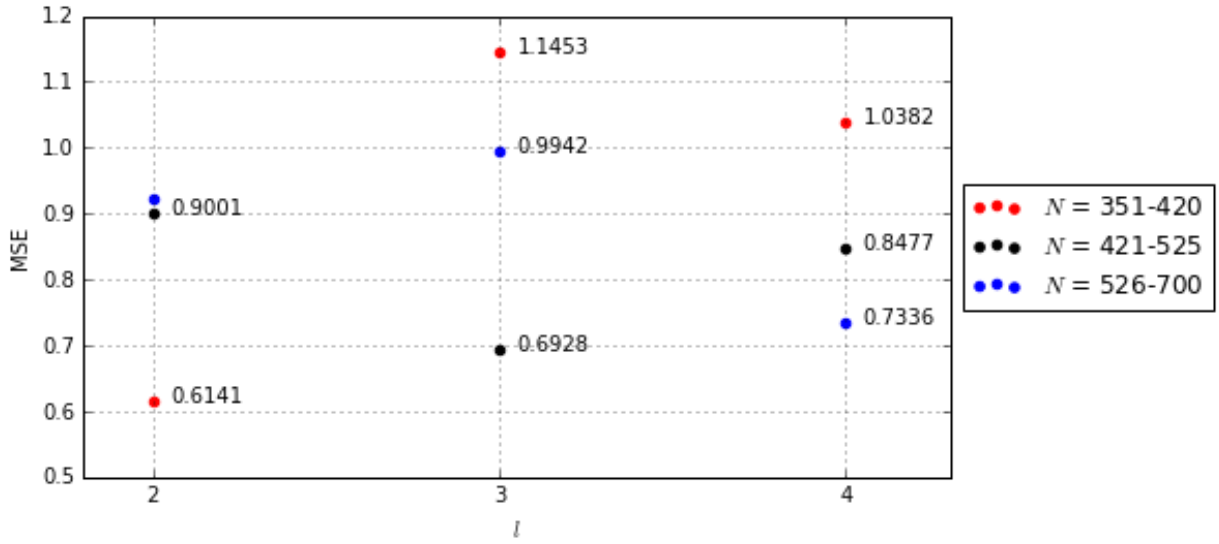

Рис. 7: График зависимости среднеквадратичной ошибки прогноза (MSE) от параметра l при различных значениях параметра N для первого репозитория в случае второй модели.

Таким образом, получили, что для первого репозитория в случае второй модели наименьшее значение среднеквадратичной ошибки прогноза равно 0.614092673183, и оно достигается при $N = 351$ и $l = 2$.

Видим, что первая модель дает меньшую среднеквадратичную ошибку прогноза для данного репозитория.

Теперь приведем основные результаты работы для репозитория "Abalone Dataset". Перечислим основные характеристики этого репозитория:

- $m = 4177$,
- $n = 8$,
- $k = 4$,
- $N_{\max} = 2089$,
- $l_{\max} = 15$.

Отобразим основные полученные графики.

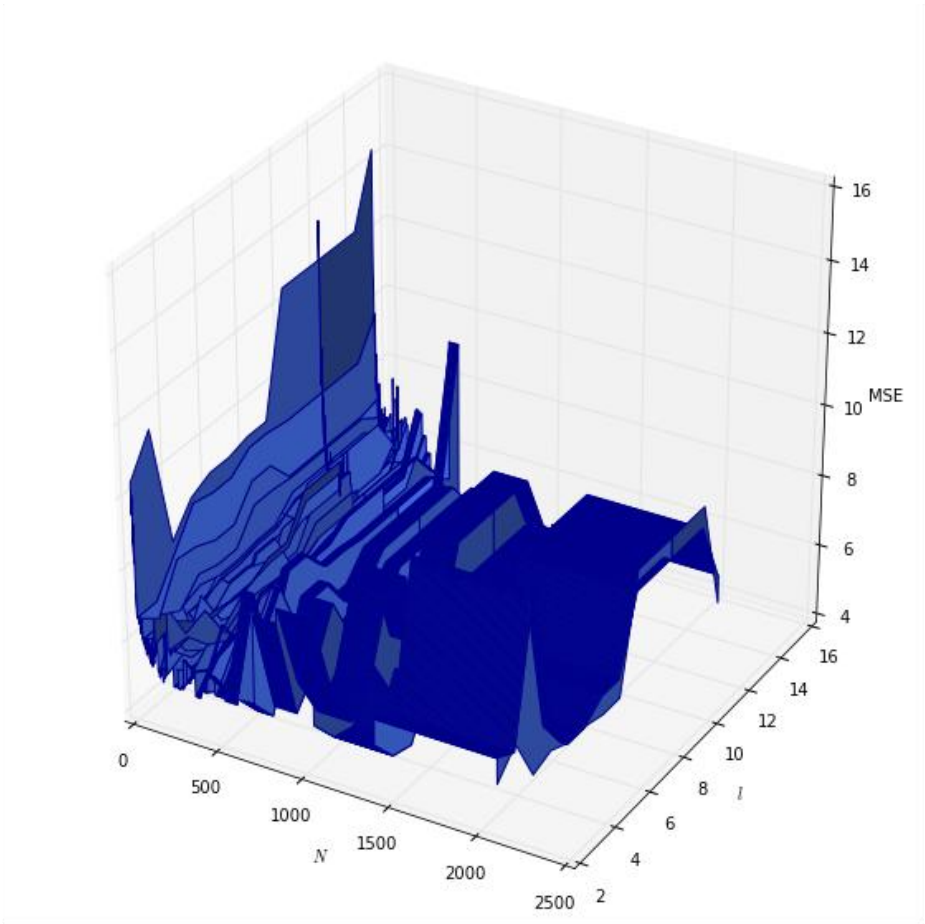

Рис. 8: График зависимости среднеквадратичной ошибки прогноза (MSE) от параметров N и l для второго репозитория в случае первой модели.

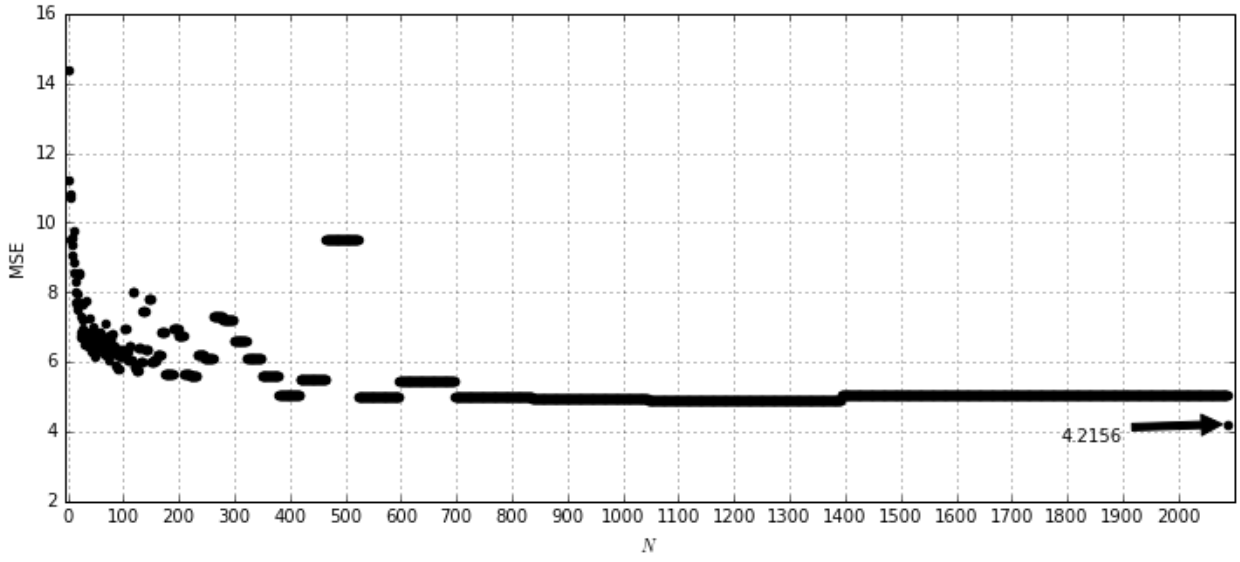

Рис. 9: График зависимости среднеквадратичной ошибки прогноза (MSE) от параметра N при $l = 15$ для второго репозитория в случае первой модели. Минимальное значение MSE в случае первой модели отмечено стрелкой.

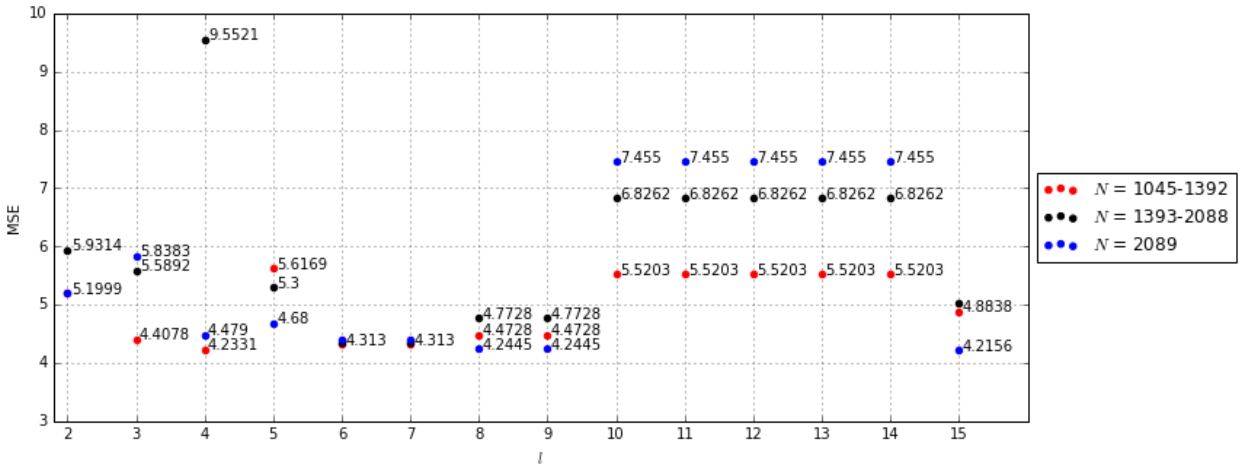

Рис. 10: График зависимости среднеквадратичной ошибки прогноза (MSE) от параметра l при различных значениях параметра N для второго репозитория в случае первой модели.

Таким образом, получили, что для второго репозитория в случае первой модели наименьшее значение среднеквадратичной ошибки прогноза равно 4.215602649507, и оно достигается при $N = 2089$ и $l = 15$.

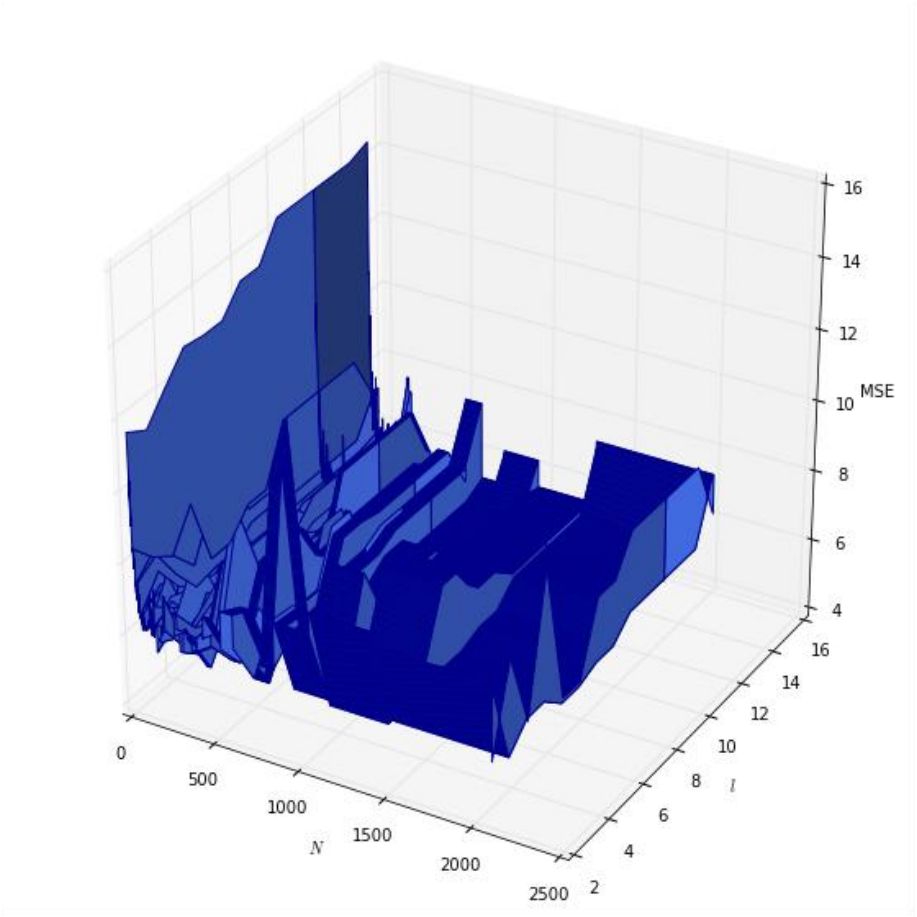

Рис. 11: График зависимости среднеквадратичной ошибки прогноза (MSE) от параметров N и l для второго репозитория в случае второй модели.

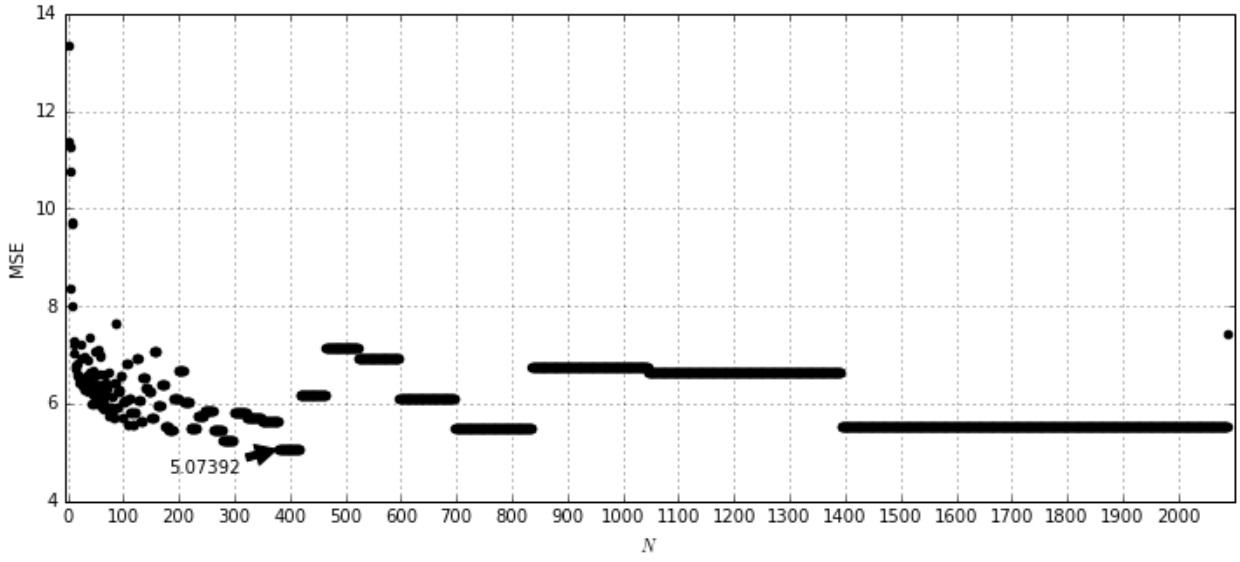

Рис. 12: График зависимости среднеквадратичной ошибки прогноза (MSE) от параметра N при $l = 15$ для второго репозитория в случае второй модели. Минимальное значение MSE в случае первой модели отмечено стрелкой.

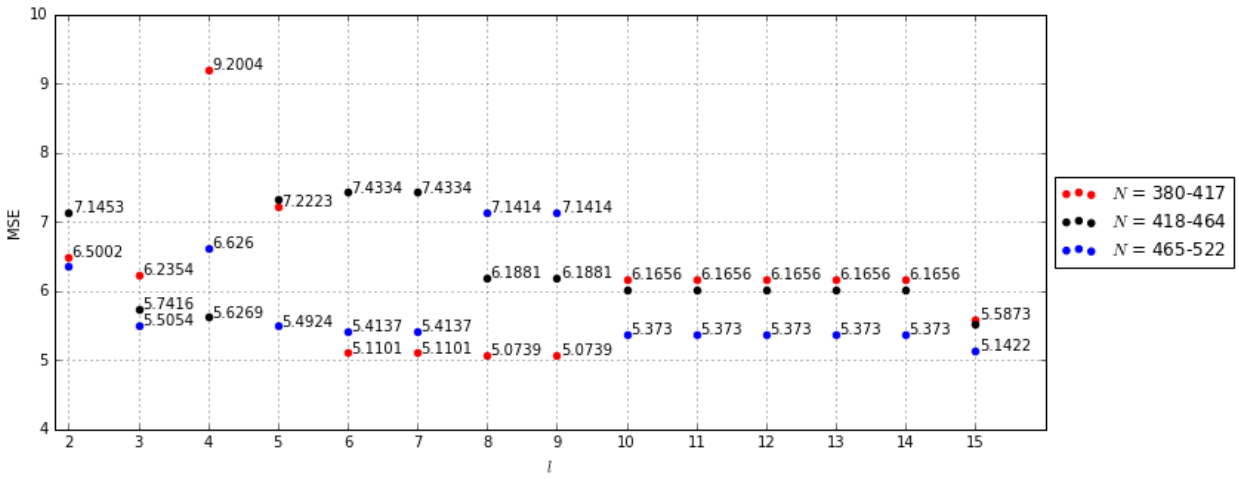

Рис. 13: График зависимости среднеквадратичной ошибки прогноза (MSE) от параметра l при различных значениях параметра N для второго репозитория в случае второй модели.

Таким образом, получили, что для второго репозитория в случае первой модели наименьшее значение среднеквадратичной ошибки прогноза равно 5.073916692033, и оно достигается при $N = 380$ и $l = 8$.

Видим, что первая модель дает меньшую среднеквадратичную ошибку прогноза и для данного репозитория.

Приведем пример работы стохастического градиентного спуска для решения задачи (4). Рассмотрим сначала репозиторий "Image Segmentation Dataset". Для него исполним алгоритм в случае первой модели с значениями гиперпараметров $N = 421$ и $l = 3$, т.е. в случае когда мы получили наименьшее значение среднеквадратичной ошибки прогноза.

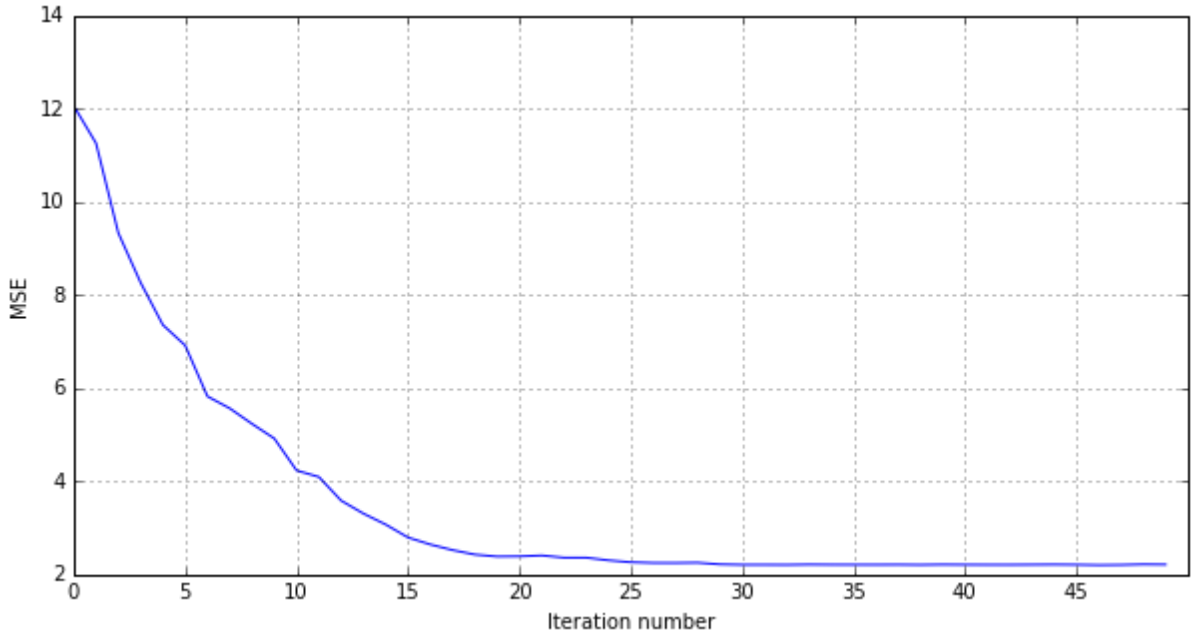

Рис. 14: График зависимости среднеквадратичной ошибки прогноза (MSE) на первых 50 −ти итерациях стохастического градиентного спуска от номера итерации для первого репозитория в случае первой модели.

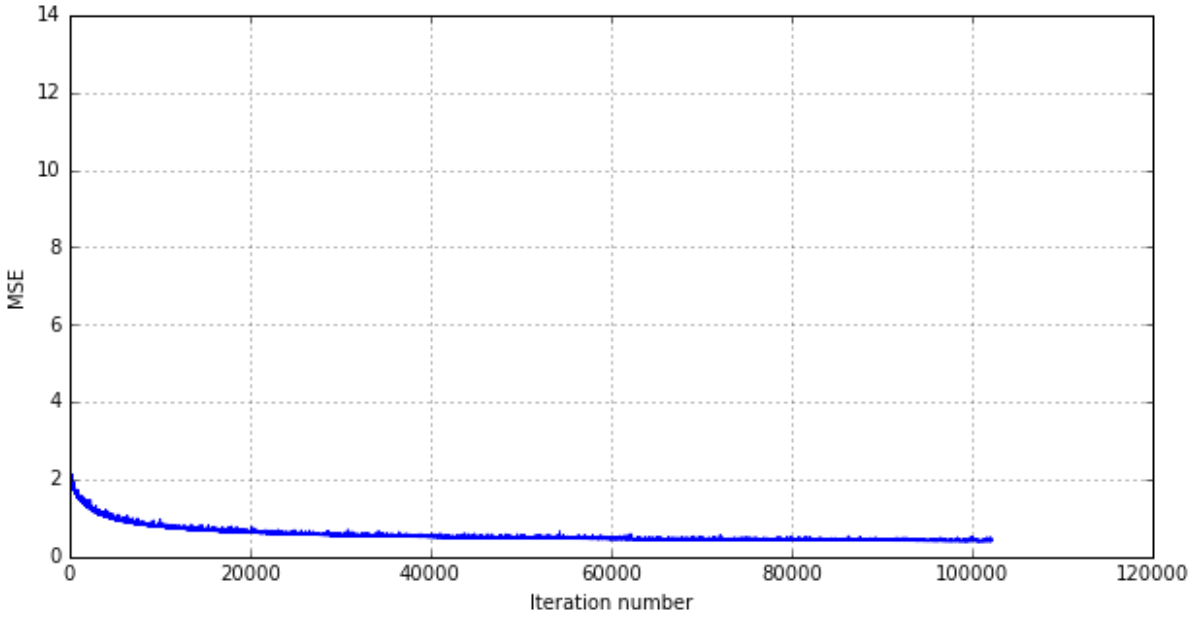

Рис. 15: График зависимости среднеквадратичной ошибки прогноза (MSE) от номера итерации стохастического градиентного спуска для первого репозитория в случае первой модели.

Видно, что алгоритм сходится.

Аналогично рассмотрим репозиторий "Abalone Dataset". Для него исполним алгоритм в случае первой модели с значениями гиперпараметров $N = 2089$ и $l = 15$, т.е. в случае когда мы получили наименьшее значение среднеквадратичной ошибки прогноза.

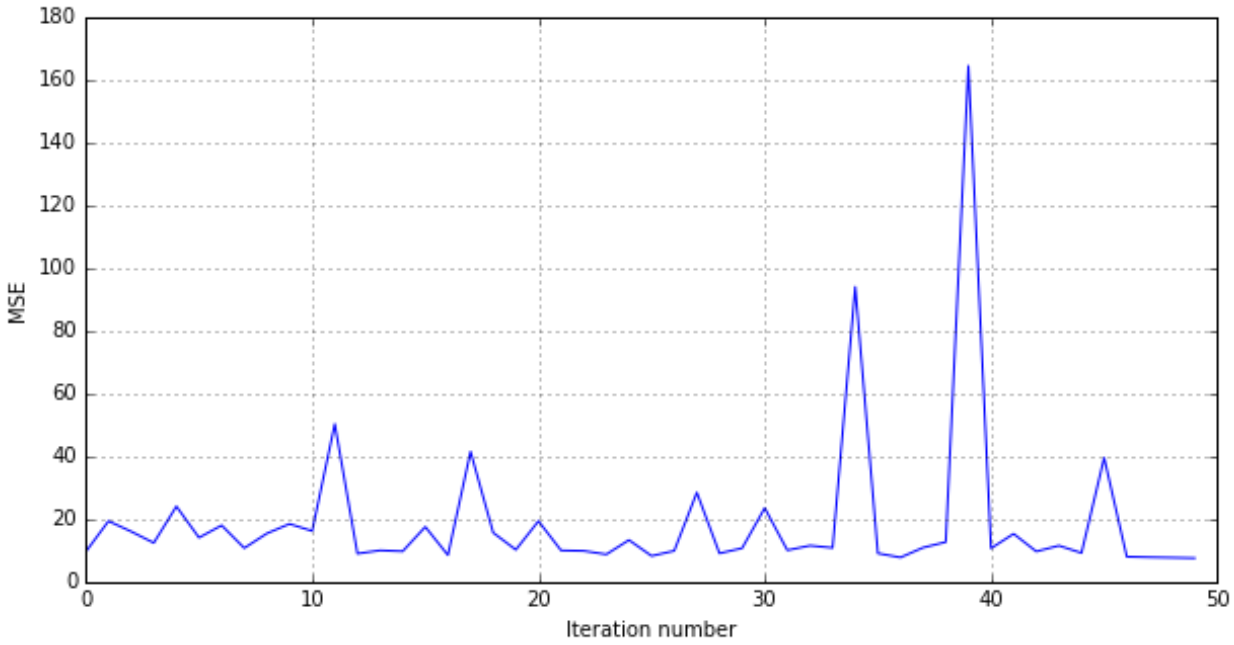

Рис. 16: График зависимости среднеквадратичной ошибки прогноза (MSE) на первых 50 −ти итерациях стохастического градиентного спуска от номера итерации для второго репозитория в случае первой модели.

Видно, что среднеквадратичная ошибка прогноза при выполнении стохастического градиентного спуска необязательно уменьшается на каждой итерации, т.е. функция среднеквадратичной ошибки от номера итерации не является монотонной.

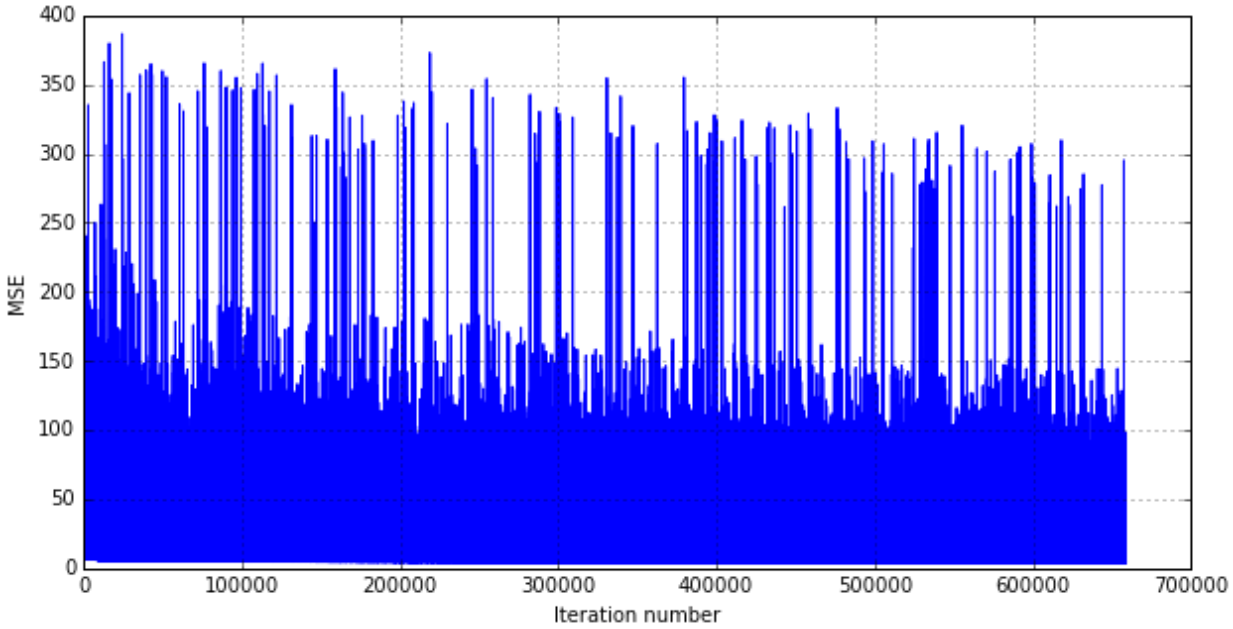

Рис. 17: График зависимости среднеквадратичной ошибки прогноза (MSE) от номера итерации стохастического градиентного спуска для второго репозитория в случае первой модели.

4.4. Сравнение созданного метода с другими известными подходами

Сначала приведем сравнение результатов работы метода для репозитория "Image Segmentation Dataset" с результатами, полученными при использовании других известных методов.

Метод	Среднеквадратичная ошибка
Первая модель	0.40438141110339682
Вторая модель	0.61409267318297744
Linear Regression	1.191162468451379
Random Forest Classifier (n_estimators = 50, max_depth = 3)	0.9776190476190476
Random Forest Regressor (n_estimators = 50, max_depth = 2)	1.0137278134690764
Decision Tree Classifier (max_depth = 5)	0.920952380952381
Decision Tree Classifier (min_samples_leaf = 100)	0.6309523809523809
Decision Tree Regressor (max_depth = 3)	1.04524844809153
Decision Tree Regressor (min_samples_leaf = 200)	0.6484133502710604

Табл. 2: Сравнение результатов работы метода для первого репозитория с результатами, полученными при использовании других известных методов.

Теперь приведем сравнение результатов работы для репозитория "Abalone Dataset" с результатами, полученными при использовании других известных методов.

Метод	Среднеквадратичная ошибка
Первая модель	4.2156026495068701
Вторая модель	5.0739166920331913
Linear Regression	4.909236815818974
Random Forest Classifier (n_estimators = 50, max_depth = 8)	5.945415369882691
Random Forest Regressor (n_estimators = 50, max_depth = 2)	6.213333882046058
Decision Tree Classifier (max_depth = 8)	5.455350730189131
Decision Tree Classifier (min_samples_leaf = 10)	5.964807277950682
Decision Tree Regressor (max_depth = 3)	5.954366194788399
Decision Tree Regressor (min_samples_leaf = 100)	4.9511242372257005

Табл. 3: Сравнение результатов работы метода для второго репозитория с результатами, полученными при использовании других известных методов.

Видим, что на примере двух репозиториев созданный метод работает лучше приведенных методов: с меньшей среднеквадратичной ошибкой прогноза.

5. ЗАКЛЮЧЕНИЕ И ВЫВОДЫ

1. Создан метод восстановления зависимостей на базе коллективов алгоритмов вычисления оценок.
2. Проведено сравнение двух различных способов определения функции близости в АВО на примере двух репозиториев.
3. Проведено сравнение результатов работы созданного метода с результатами, полученными при использовании других известных методов.
4. На основе экспериментального исследования метода становится ясно, что первая модель АВО дает меньшую среднеквадратическую ошибку.

6. СПИСОК ЛИТЕРАТУРЫ

1. Журавлев Ю.И., Избранные научные труды. - М.: Издательство Магистр, 1998. - 420 с.

2. Журавлев Ю.И., Об алгебраическом подходе к решению задач распознавания или классификации // Проблемы кибернетики. М.: Наука, 1978. Вып. 33. С. 5-68.

3. Ryazanov V.V. Regression via Logic Supervised Classification // "Lecture Notes in Computer Science" (LNCS), 7914, 2013, pp. 242–253.

4. Журавлев Ю.И., Рязанов В.В., Сенько О.В., Распознавание: Математические методы. Программная система. Практические применения. - М.: Издательство Фазис, Москва, 2005. - 159 с.

5. Ткачев Ю.И., Методы решения задачи восстановления зависимости коллективами распознающих алгоритмов, Москва, 2013.

6. Жадан В.Г., Методы оптимизации. Ч. II. Численные алгоритмы. - М.: МФТИ, 2015. - 320 с.

7. Coursera.org, Специализация «Машинное обучение и анализ данных», MIPT, Yandex, https://ru.coursera.org/specializations/machine-learning-data-analysis

8. MachineLearning.ru, Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных, http://www.machinelearning.ru