Государственное образовательное учреждение высшего профессионального образования «Московский физико-технический институт (государственный университет)»

Кафедра информатики и вычислительной математики МФТИ

Поиск типичных представителей классов

Выпускная квалификационная работа на степень бакалавра студента 476 группы Трахова Романа Дмитриевича

Научный руководитель: доктор физико-математических наук, профессор Рязанов В.В.

Содержание

1 Введение **3**

2 Постановка задачи **3**

2.1 Основные понятия 3

2.1.1 Обучение с учителем 3

2.1.2 Описание признакового пространства 4

2.1.3 Задача бинарной классификации 4

2.1.4 Оценка качества классификации 5

2.1.5 Стохастический градиентный спуск (SGD) 5

2.1.6 Обучение без учителя 6

2.1.7 Алгоритм кластеризации DBSCAN 7

2.2 Постановка задачи поиска типичных представителей классов 8

2.3 Оценка качества для задачи поиска типичного представителей классов 8

3 Используемые методы **9**

3.1 Метод k ближайших соседей 9

3.2 Метод опорных векторов (SVM) 11

3.3 Метод, основанный на логических закономерностях классов 12

3.4 Kmeans 15

4 Эксперименты на модельных данных **15**

4.1 Набор данных №1 16

4.1.1 Метод ближайших соседей 18

4.1.2 Метод опорных векторов(SVM) 20

4.1.3 Kmeans 22

4.1.4 Логические закономерности классов 23

4.1.5 Сравнение методов 23

4.2 Набор данных №2 24

4.2.1 Метод ближайших соседей 26

4.2.2 Метод опорных векторов(SVM) 27

4.2.3 Kmeans 29

4.2.4 Логические закономерности классов 30

4.2.5 Сравнение методов 31

4.3 Набор данных №3 32

4.3.1 Метод ближайших соседей 32

4.3.2 Метод опорных векторов 32
4.3.3 Kmeans 32
4.3.4 Логические закономерности классов 33
4.3.5 Сравнение алгоритмов 33
4.4 Выводы 33

1 Введение

На сегодняшний день машинное обучение используется во всех областях связанных с большими данными. Это уже не нововведение, а жизненная необходимость: данных стало так много, что человек больше не может их анализировать, прибегая к помощи устаревших простых инструментов анализа. Машинный анализ дает новые необъятные возможности для выявления закономерностей, трендов, зависимостей, теперь мы можем узнать такие невероятные вещи, о которых могли только догадываться несколько десятилетий назад. Машинное обучение расширяет границы познаний в физике, математике, социологии и, пожалуй, во всех известных науках. Данная работа рассматривает область классификации данных.

Классификация данных – задача определения, к какому классу из известных относится новый объект. Существует большое количество примеров применения алгоритмов классификации, такие как: задачи медицинской диагностики, предсказание месторождений полезных ископаемых, оценка кредитоспособности заемщиков, то есть случаи, где решения принимаются на основании принадлежности данных к одному из выделенных классов. Благодаря этим технологиям жизнь может стать проще, а принимаемые решения- точнее и дешевле.

2 Постановка задачи

2.1 Основные понятия

2.1.1 Обучение с учителем

Обучение с учителем — один из разделов машинного обучения, изучающий широкий класс задач, в которых известно множество объектов (ситуаций) и множество возможных ответов (откликов, реакций) и существует некоторая зависимость между ситуациями и откликом. Задачи обучения с учителем делятся на 2 больших класса задач: классификации и регрессии. В задаче классификации множество ответов конечно.(количество меток классов), а в задаче регрессии ответы являются действительными числами или векторами. К известным задачам обучения с учителем относят задачи медицинской диагностики, кредитного

скоринга, предсказания наличия полезных ископаемых, потребительского спроса, временных рядов и многие другие.

Будем рассматривать множество объектов $\mathcal{X}$,размерности $\mathcal{N}$, заданных на признаковом пространстве $\mathcal{D}$, множество меток $\mathcal{Y} = \{0, 1\}$, и целевую функцию $y^* : \mathcal{X} \to \mathcal{Y}$, значения которой известны на конечном наборе $\mathcal{X}_{train} = X_1, X_2, .., X_n \in \mathcal{X}$ и равны $\mathcal{Y}_{train} = y_1, y_2, ..., y_n \in \mathcal{Y}$ соответственно.

Совокупность $\{\mathcal{X}_{train}, \mathcal{Y}_{train}\}$ называется обучающей выборкой.

Остальные $N_{test} = N - N_{train}$ элементов, для которых неизвестно значение целевой функции образуют тестовую выборку $\mathcal{X}_{test}$

Задача обучения с учителем состоит в том, чтобы по обучающей выборке научиться восстанавливать зависимость y^*, т.е. построить такое отображение $\mathcal{X} \to \mathcal{Y}$, которое приближает целевую функцию y^*, как на обучающей выборке, так и на тестовой.

2.1.2 Описание признакового пространства

Признак - это такое отображение $f : X \to D_f$, где D_f - множество допустимых значений данного признака. Если заданы признаки $f_1, \ldots, f_n$, то вектор $\mathbf{x} = (f_1(x), \ldots, f_n(x))$ называется признаковым описанием объекта $x \in X$. Каждый объект описывается своим набором признаков. При этом множество $X = D_{f_1} \times \cdots \times D_{f_n}$ называют признаковым пространством.

В зависимости от множества D_f признаки делятся на следующие типы:
бинарный признак: $D_f = \{0, 1\}$;
номинальный признак: D_f - заданное ограниченное и конечное множество;
порядковый признак: D_f — конечное упорядоченное множество;
количественный признак: D_f — множество действительных чисел .

2.1.3 Задача бинарной классификации

Классификация — один из разделов машинного обучения, посвященный решению задачи на множестве объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Классовая принадлежность остальных объектов не известна.

В задаче бинарной классификации имеется 2 класса и целевая переменная y принимает 2 значения 0 и 1.

$$y_k = \begin{cases} 1, & \text{если объект } \boldsymbol{x_k} \text{ принадлежит классу 1} \\ 0, & \text{если объект } \boldsymbol{x_k} \text{ принадлежит классу 0} \end{cases}.$$

2.1.4 Оценка качества классификации

Для оценки алгоритмов используют разные функционалы качества. В данной работе будет использована метрика AUC, равная площади под ROC кривой. Пусть имеется вектор ответов y и вектор предсказанный классификатором $\widetilde{y} \in [0,1]^n$, равный вероятности принадлежности к классу 1. Тогда:

$$AUC(\widetilde{y}, y) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} I[y_i < y_j] I'[\widetilde{y}_i < \widetilde{y}_j]}{\sum_{i=1}^{n}\sum_{j=1}^{n} I[y_i < y_j]} \in [0,1],$$

где

$$I[y_i < y_j] = \begin{cases} 1, & \text{если } y_i < y_j \\ 0, & \text{иначе} \end{cases}.$$

$$I'[y_i < y_j] = \begin{cases} 1, & \text{если } y_i < y_j \\ 0.5, & \text{если } y_i = y_j \\ 0, & \text{если } y_i > y_j \end{cases}.$$

Чем ближе значение AUC к единице — тем выше качество классификации.

Часто для более объективной оценки качества, выборка разбивается на k непересекающихся частей (фолдов), далее алгоритм обучается на k-1 частях выборки и оценивается качество на оставшейся одной. Далее полученные k метрик качества усредняются. Такой подход называется кроссвалидацией.

2.1.5 Стохастический градиентный спуск (SGD)

Решается задача минимизации функционала:

$$\boldsymbol{Q(w;x)} = \frac{1}{l}||Xw - y|| \to \min_{w} \tag{1}$$

Для определения шага обычного градиентного спуска требуется вычисление градиента: $\nabla Q(W, x) = \frac{2}{l} X^T (Xw - y)$. Выражение для j-ой компоненты градиента, таким образом, содержит суммирование по всем объектам обучающей выборки:

$$\frac{\partial Q}{\partial w_j} = \frac{2}{l} \sum_{i=1}^{l} x_i^j (\langle w, x_i \rangle - y_i) \tag{2}$$

В случае большой выборки вычисление даже одной итерации затруднительно. Идея стохастического градиентного спуска состоит в 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 уменьшить количество итераций. Таким образом, алгоритм стохастического градиентного спуска следующий. Сначала выбирается начальное приближение: $w_0 = 0$. Далее последовательно вычисляются итерации w^t: сначала случайным образом выбирается объект x_i из обучающей выборки X и вычисляется вектор градиента функции качества на этом объекте, а следующее приближение получается из предыдущего вычитанием умноженного на шаг n_t полученного вектора:

$$w^t = w^{t-1} - n_t \nabla Q(w^{t-1}, \{x_i\}) \tag{3}$$

Итерации прекращаются, когда выполняется некоторый критерий, например : $w||^t - w^{t-1}|| < \varepsilon$.

2.1.6 Обучение без учителя

Обучение без учителя — один из разделов машинного обучения, изучающий широкий класс задач обработки данных, в которых известны только описания множества объектов (обучающей выборки), и требуется обнаружить внутренние взаимосвязи, зависимости, закономерности, существующие между объектами. К задачам обучения без учителя относят задачи кластеризации, поиска аномалий, визуализации и др.

Кластерный анализ — задача разбиения заданной выборки объектов (ситуаций) на непересекающиеся подмножества, называемые кластера-

ми, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались.

2.1.7 Алгоритм кластеризации DBSCAN

DBSCAN (Density-based spatial clustering of applications with noise) - плотностный метрический алгоритм кластеризации, распознающий кластеры произвольной формы.

Основная идея DBSCAN - представление объектов кластера в виде группы точек в метрическом пространстве. Кроме того, две точки в таком графе соединены в случае, если расстояние между ними меньше заданного расстояния по метрике. Если вблизи определенного объекта недостаточно точек, то он распознается как выброс.

Входные данные алгоритма: множество объектов X с известной метрической функцией расстояния $\rho(x_1, x_2)$

Выходные данные: вектор ответов $\mathcal{Y}$, где

$$y_k = \begin{cases} k, & \text{k} \in \overline{1, K}, \text{ если объект попал в кластер k} \\ -1, & \text{если объект не попал ни в один кластер} \end{cases}.$$

Введем два параметра:

ϵ - минимальное расстояние, на котором учитываются объекты

L_{min} - минимальное число объектов для образования кластера

Описание алгоритма:

Для каждого объекта x выборки X:

1.Находим соседние объекты в ϵ-окрестности объекта x

2.Сравниваем количество соседних объектов с L_{min}, определяя, является ли x ядровым объектом (относится к какому-либо кластеру)

3.Если объект p является ядровым:

-создаём новый кластер и запускаем поиск в ширину из данного объекта x, находя все объекты кластера.

Если объект x не является ядровым:

-отмечаем его как выброс.

Алгоритм не требует на вход заданное число кластеров и находит заранее неизвестное число кластеров произвольной формы.Также алгоритм хорошо справляется с зашумленными данными, выделяя выбросы в отдельную категорию объектов. Очень часто этот алгоритм используется как раз для того, чтобы выделить и отсеять шум.

2.2 Постановка задачи поиска типичных представителей классов

На практике не всегда требуется классифицировать все объекты тестовой выборки. Иногда достаточно предоставить только самых "типичных"представителей от каждого класса. Эти объекты могут, как содержаться, так и не содержаться на тестовой и обучающей выборке.

Ставится задача поиска такой универсальной функции $y : \mathcal{X} \to \mathcal{Y}$, где $\mathcal{X}$ - объединение обучающей и тестовой выборок, а $\mathcal{Y}$ - вектор типичных представителей классов, $\mathcal{Y} \in \mathcal{R}^2$ для задачи бинарной классификации.

2.3 Оценка качества для задачи поиска типичного представителей классов

Будем рассматривать задачу на множестве количественных признаков. Если у нас имеются категориальные или биномиальные признаки, то известными методами их можно преобразовать в количественные (вещественные). Опишем формально какой объект мы считаем типичным. Для этого введем расстояние между объектами x и $\widetilde{x}$ в метрическом пространстве, как квадрат евклидова расстояния:

$$\rho(\widetilde{x}, x) = \sum_{i=1}^{D} (x_i - \widetilde{x}_i)^2$$

Также введем функционал типичности объекта $\widetilde{x}$:

$$L(\widetilde{x}, K, p) = \sum_{i=1}^{K} (\rho(x, \widetilde{x}) * (K - i + 1) * I(x, \widetilde{x})),$$

где:

$$I(\widetilde{x}, x_k) = \begin{cases} 1, & \text{если классы объектов } \widetilde{x} \text{ и } \boldsymbol{x_k} \text{ совпадают} \\ p, p > 1, & \text{если классы объектов } \widetilde{x} \text{ и } \boldsymbol{x_k} \text{ не совпадают} \end{cases}.$$

K - количество значимых ближайших объектов

p - штраф за попадание объекта не своего класса

Суммирование происходит по ближайшим K объектам к объекту $\widetilde{x}$. Это допущение объясняется важностью только ближайших K соседей,

при чем K не должно превышать N/2 для задачи бинарной классификации. Домножение на коэффициент $(K - i + 1)$ нормирует вклад i объекта в функцию потерь(чем ближе объект, тем больший вклад он вносит).Таким образом данный функционал сильно штрафует, если наблюдение является выбросом (находится далеко от других объектов обучающей выборки), также функционал штрафует, если в ближайших K объектах присутствуют объекты не своего класса.

Теперь введем метрику типичности для объекта $\widetilde{x}$:

$$M(\widetilde{x}) = \exp(-AL(\widetilde{x}, K, p) \in [0; 1],$$

где A -нормировочный коэффициент. Чем ближе эта метрика к 1, тем более типичным является объект для своего класса.

Значения параметров K и p выбираются в зависимости от задачи и важности каждого из штрафов. (По умолчанию будет считать $K = N/10$, а $p = 10$)

3 Используемые методы

Обычно размерность обучающей выборки очень большая, и, количество данных растет экспоненциально с ростом размерности пространства (Проклятие размерности). Поэтому минимизировать функционал L возможно только стохастическими методами.

В этой главе предложены методы поиска типичных представителей классов, основанные на решении задач классификации и кластеризации.

Мы также подразумеваем, что выполнена гипотеза компактности данных, т.е. выполнено предположение о том, что схожие объекты гораздо чаще лежат в одном классе, чем в разных; или, другими словами, что классы образуют компактно локализованные подмножества в пространстве объектов. Это также означает, что граница между классами имеет достаточно простую форму.

3.1 Метод k ближайших соседей

Метод k ближайших соседей является одним из наиболее простых алгоритмов классификации. Он требует вычисления расстояния между объектами. Существует много различных метрик.Объект x относится

к классу y_k, если большинство ближайших к нему объектов по метрике принадлежат классу k. В задачах с 2 классами число k выбирается нечетным для избежания возникновения неопределенности.

Алгоритм нахождения типичных представителей классов:

На пером шаге алгоритмом DBSCAN отсеиваем все выбросы (объекты не попавшие ни в один кластер)

Далее для каждого объекта вычисляются ближайшие по метрике k соседей.

Следующим шагом выбираются самые надежные элементы, те у которых все k или хотя бы k-1 сосед принадлежит тому же классу, что и сам элемент. Если таких не имеется, то условие ослабляется на 1 соседа. После этой операции мы получаем две подвыборки наиболее типичных элементов для каждого из классов. Чтобы получить по 1 элементу мы усредняем все признаки для каждой из выборок.

Реализация метода на языке python3:

```
def knn_typical(k,metric):
   knn = KNeighborsClassifier(n_neighbors=k,metric=metric).fit(X,Y)
   X['knn'] = X['target'].apply(lambda x: 0)
   for i in tqdm_notebook(range(X.shape[0])):
      if knn.predict_proba([X.iloc[i].values])[0][0] == 1:
         X['knn'].values[i] = 1
      elif knn.predict_proba([X.iloc[i].values])[0][1] == 1:
         X['knn'].values[i] = 2
    if sum(X[X['knn']==1]) == 0 or sum(X[X['knn']==2]) ==0 :
      neibours(k-1,metric)
    elif:
      y=[X[X['knn']==1][column].mean(),X[X['knn']==2][column].mean()]
      print('Typical element of 1 class')
      for column in X.drop(['knn'],axis=1):
         print(y[0])
      print('Typical element of 2 class')
      for column in X.drop(['knn'],axis=1):
         print(y[1])
    return y
```

3.2 Метод опорных векторов (SVM)

Метод опорных векторов является классификатором с "максимальным зазором так как главным свойством этого метода является непрерывное уменьшение эмпирической ошибки классификации и увеличение зазора.Рассмотри общую постановку задачи с M классами. Для каждого класса заводится свой вектор весов $\boldsymbol{w_k}, k = \overline{1, M}$. Получаем матрицу весов $\boldsymbol{W} = ||\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_M}||$. Для классификации объекта $\boldsymbol{x}$ вычисляется произведение $\boldsymbol{W}^T\boldsymbol{x}$ - вектор оценок. Выбирается тот класс, чья оценка самая высокая.

Функция потерь L_i на одном объекте выглядит следующим образом:

$$\boldsymbol{L_k} = \sum_{j \neq y_k} \max\left(0, \boldsymbol{x_k}^T \boldsymbol{w_k} - \boldsymbol{x_k}^T \boldsymbol{w_{y_k}} + \Delta\right) \tag{4}$$

Также есть смысл использовать регуляризацию

$$R(W) = \sum_{i=1}^{D} \sum_{j=1}^{M} W_{i,j}^2 \tag{5}$$

Итоговая функция потерь может быть записана как

$$\boldsymbol{L} = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda R(W)$$

Или более полно:

$$L = \frac{1}{N} \sum_{k=1}^{N} \sum_{j \neq y_k} \max\left(0, \boldsymbol{x_k}^T \boldsymbol{w_k} - \boldsymbol{x_k}^T \boldsymbol{w_{y_k}} + \Delta\right) + \lambda \sum_{i} \sum_{j} W_{i,j}^2 \tag{6}$$

Остается найти минимум по весам данной функции. Оптимизация проводится стохастическим градиентным спуском. Производная функции потерь по весу верного класса выражается в виде:

$$\nabla_{\boldsymbol{w_{y_k}}} L_k = -\left(\sum_{j \neq y_k} \mathbf{1}(\boldsymbol{x_k}^T \boldsymbol{w_j} - \boldsymbol{x_k}^T \boldsymbol{w_{y_k}} + \Delta > 0)\right) \boldsymbol{x_k} \tag{7}$$

где $\mathbf{1}$ - функция-индикатор, равная единице, если условие верно, и нулю, если условие не верно. По остальным весам ($j \neq y_k$):

$$\nabla_{\boldsymbol{w_j}} L_k = \mathbf{1}(\boldsymbol{x_k}^T \boldsymbol{w_j} - \boldsymbol{x_k}^T \boldsymbol{w_{y_k}} + \Delta > 0) \boldsymbol{x_k} \tag{8}$$

После построения разделяющих плоскостей, находим расстояние для каждого элемента выборки до разделяющей плоскости (margin). Выбираем элементы, находящиеся далеко от разделяющей плоскости. Такие элементы назовем надежными. Типичными будем считать элементы, которые являются надежными и принадлежат каждому из классов соответственно (Таким образом, мы убираем пограничные элементы, которые могут быть ошибочными) Чтобы получить по одному представителю от каждого класса, усредним все признаки типичных элементов.

Реализация метода на языке python:

```
def svm_typical(metric):
   clf = svm.SVC(kernel='linear').fit(X, Y)
   y = clf.decision_function(X)
   w_norm = np.linalg.norm(clf.coef_)
   dist = y / w_norm
   X['y'] = Y
   X['dist'] = dist
    print('Typical element of 1 class:')
    for column in X.drop(['dist','y'], axis=1):
    print(column, X[(X['dist']>=np.percentile(X[(X['y']==1) &
        (X['dist'] > 0)] ['dist'].values, 50, axis=0)) &
        X['y']==1][column].mean())
    print('Typical element of 2 class:')
    for column in X.drop(['dist','y'], axis=1):
    print(column, X[(X['dist']>=np.percentile(X[(X['y']==1) &
        (X['dist'] > 0)] ['dist'].values, 50, axis=0)) &
        X['y']==1][column].mean())
    return [X[(X['dist']>=np.percentile(X[(X['y']==1) & (X['dist'] >
        0)] ['dist'].values, 50, axis=0)) & X['y']==1][column],
        X[(X['dist']>=np.percentile(X[(X['y']==1) & (X['dist'] > 0)]
        ['dist'].values, 50, axis=0)) & X['y']==1][column].mean()]
```

3.3 Метод, основанный на логических закономерностях классов

Методы, использующие логические закономерности классов легко интерпретируемы, т.е. описываются простой формулой и она понятна для экспертов из этой области. Формально нельзя дать определение интер-

претируемости. В большей части задач для интерпретируемости необходимо, но не достаточно, зависимость предиката $\phi(x)$ от небольшого числа признаков K. Иногда даже вводятся ограничение сверху на K(например $\leq K \leq 10$). Также вводятся ограничения на функциональный вид правил. Здесь будут рассмотрены в виде функций - коньюнкии, иногда применяются также шары, дизъюнкции и др.

Алгоритм становится понятным людям, если его можно представить в виде: «если $\phi(x)$, то относительно объекта x принять решение y». Такие правила называются логическими: $\phi(x) = \prod_{j \in \omega} \left[f_j(x) \leq = \geq a_j \right]$,

где ω — подмножество признаков мощности не более K, a_j — пороговое значение признака. Количественные и порядковые признаки сравниваются знаками сравнения $\leq$ или $\geq$; бинарные и номинальные — только равенство.

Рассмотрим методы поиска закономерностей вида если $A_1(x_1)\&A_2(x_2)\&...\&A_n(x_k)$ то $x \in K_i$, здесь $A_1, A_2, .., A_n$ – одноместные предикаты «простейшего вида», зависящие от одного какого-либо признака. Возможно, что условия данного вида выполняются если не на всех объектах обучающей выборки из некоторого класса, то по крайней мере на многих эталонах класса.

Создадим класс LZK, покрывающий обучающую выборку простейшими предикатами. Реализуем методы на языке python:

```
class LZK

    def fit(self, data, y):
        self._U = [tuple(x) for x in data]
        self._A = list(range(len(data[0])))
        self._D = list(set(y))
        self._rules = []

        for d in self._D:
            X = [u for i, u in enumerate(self._U) if y[i] == d]
            Y = [u for i, u in enumerate(self._U) if y[i] != d]
            lower_approximation = set(X) - set(Y)
            upper_approximation = set(X)

            # Find the certain rules:
            covering = self._LEM2(lower_approximation)
            certain = [(1, self._coverage(C), C, d) for C in covering]
            self._rules.extend(certain)
```

```
        # Find the possible rules:
        if lower_approximation != upper_approximation:
            covering = self._LEM2(upper_approximation)
            possible = [(self._accuracy(C, X), self._coverage(C),
                C, d) for C in covering]
            self._rules.extend(set(possible) - set(certain))

    self._rules = sorted(self._rules, reverse=True)
    self._majority_class = Counter(y).most_common()[0][0]
```

Также реализуем методы классификации новых объектов по созданным правилам и печати всех решающих правил.

```
def print_rules(self, attr_names=None, class_name='d', min_acc=0,
    min_cov=0):
        for (acc, cov, conditions, decision) in self._rules:
            if acc * 100 >= min_acc and cov * 100 >= min_cov:
                print("Rule: ({}, {}) <-".format(class_name,
                    decision), end=' ')
                for i, (a, v) in enumerate(conditions):
                    comma = "" if i == 0 else "\b, "
                    if attr_names is None:
                        print((comma + "({}, {})").format(a, v), end='
                            ')
                    else:
                        print((comma + "({},
                            {})").format(attr_names[a], v), end=' ')
                print("[Acc. {0:.1f}, Cov. {1:.1f}]".format(100 *
                    acc, 100 * cov))

    def predict(self, X, method='lers'):
        if not type(X[0]) is list:
            if method == 'first_fit':
                return self._first_fit(X)
            elif method == 'lers':
                return self._lers_classification(X)

        else:
            if method == 'first_fit':
```

```
            return [self._first_fit(x) for x in X]
        elif method == 'lers':
            return [self._lers_classification(x) for x in X]

    print("Invalid classification method.")
    return []
```

В результате работы метода fit, вся обучающая выборка покрывается n-мерными параллелепипедами. Типичным элементом x_i класса K_i является центр наибольшего параллелепипеда класса K_i (с наибольшим числом элементов класса K_i)

3.4 Kmeans

Задачу поиска типичных представителей классов можно решать не только используя алгоритмы классификации, но и с помощью алгоритмов кластеризации.

Алгоритм kmeans заключается в том, что на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике.

Типичными элементами классов являются центры кластеров, полученных в результате кластеризации.

Реализация на языке python:

```
def kmeans_typical():
   km = KMeans(n_clusters=2).fit(X,y)
   print('Typical element of 1 class:')
   for i in zip(X.columns, km.cluster_centers_[0]):
      print(i)
    print('Typical element of 2 class:')
   for i in zip(X.columns, km.cluster_centers_[0]):
      print(i)
```

4 Эксперименты на модельных данных

В данной части описывается ряд экспериментов на модельных данных для методов вычисления типичных представителей классов, опи-

санных в предыдущей части. Все эксперименты проводятся на датасетах, состоящих из 2 признаков и 2 классов. Это сделано для упрощения визуализации полученных результатов.

4.1 Набор данных №1

Сгенерируем набор данных X из двух классов, полученных из нормальных распределений $N(1,4;1)$ и $N(-1.4;1)$.

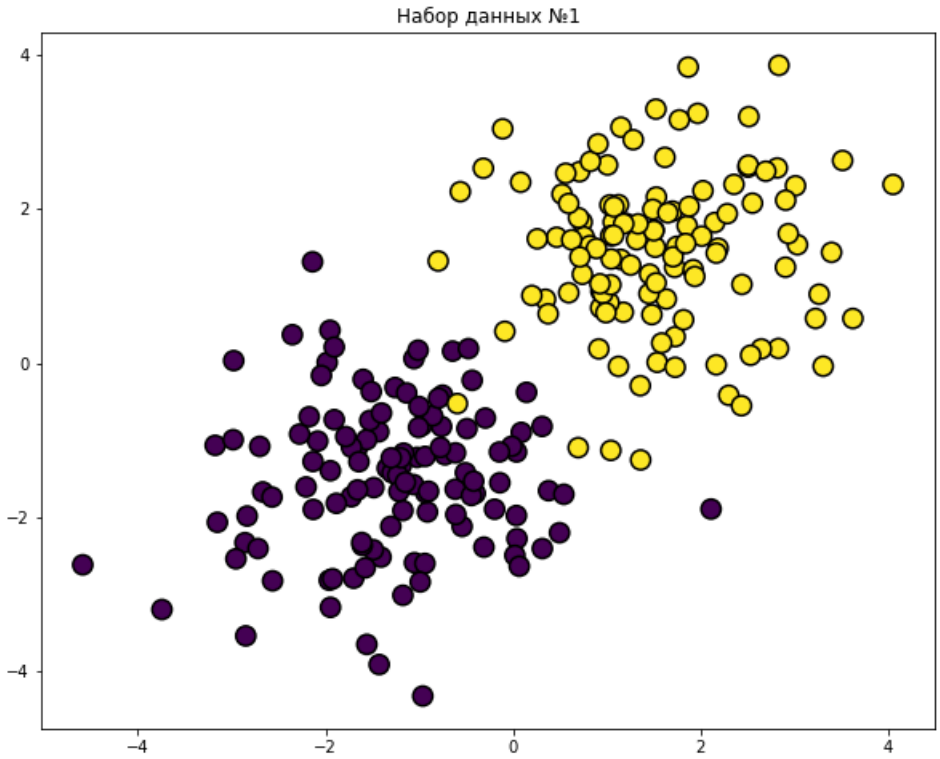

Рис. 1: Данные набора 1

Выполним плотностную кластеризацию алгоритмом DBSCAN, чтобы детектировать выбросы.

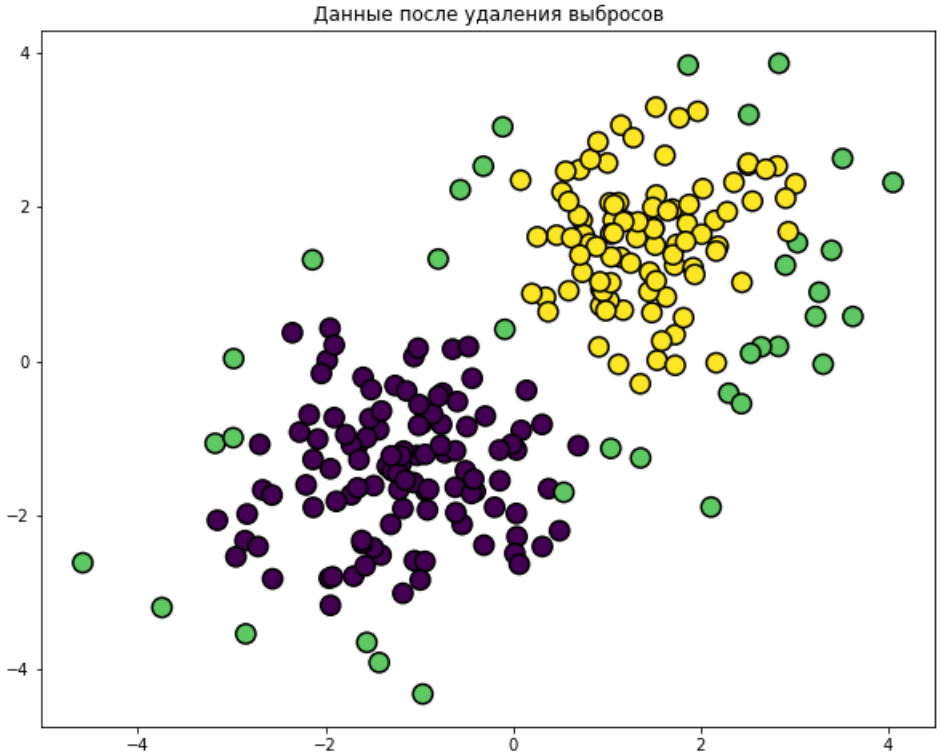

Рис. 2: DBSCAN

Видно, что DBSCAN хорошо справляется с задачей выделения шума. Далее во всех методах мы не будем обучаться на зашумленных данных, а просто удалим их на стадии обучения.

4.1.1 Метод ближайших соседей

Найдем надежные элементы методом ближайших соседей с числом k = 10.

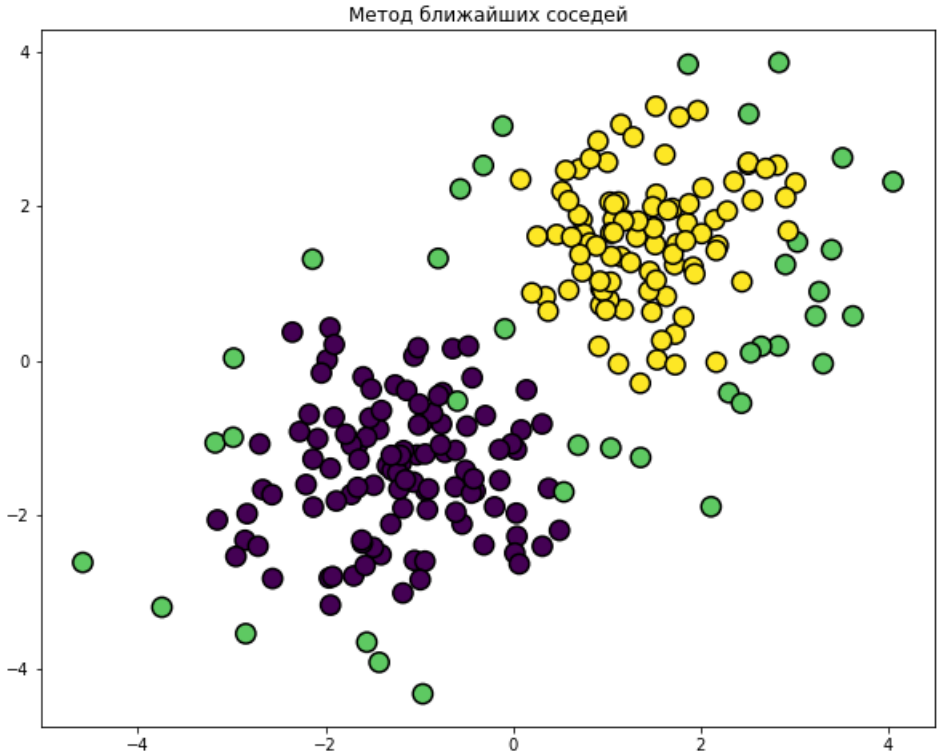

Рис. 3: knn 1

Видно, что метод ближайших соседей детектировал ненадежные элементы, которые находятся на границе двух классов. (те, у которых число ближайших соседей меньше 9)

Типичные представители классов, полученные усреднением по всем надежным элементам алгоритме k ближайших соседей:

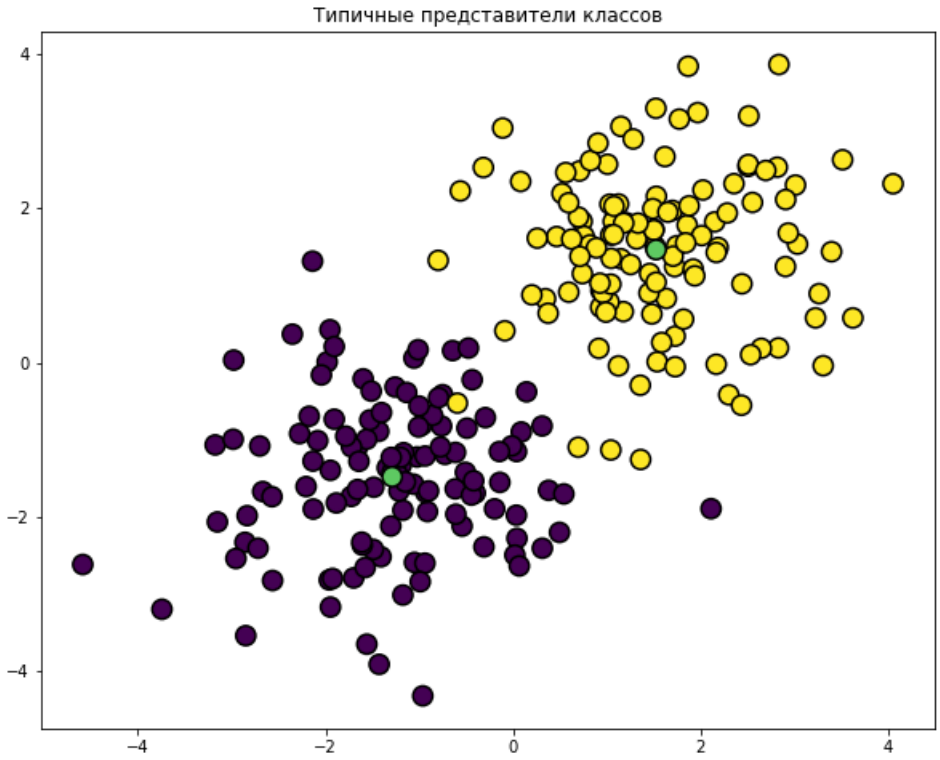

Рис. 4: knn 2

Функционал ошибки для нулевого класса равен: $L_0(x) = 153.009$ и для первого: $L_0(x) = 148.868$. Метрики $M_0(x) = p_0 = 0.86$ и $M_1(x) = p_1 = 0.87$ соответственно.

4.1.2 Метод опорных векторов(SVM)

Надежные элементы полученные алгоритмом SVM:

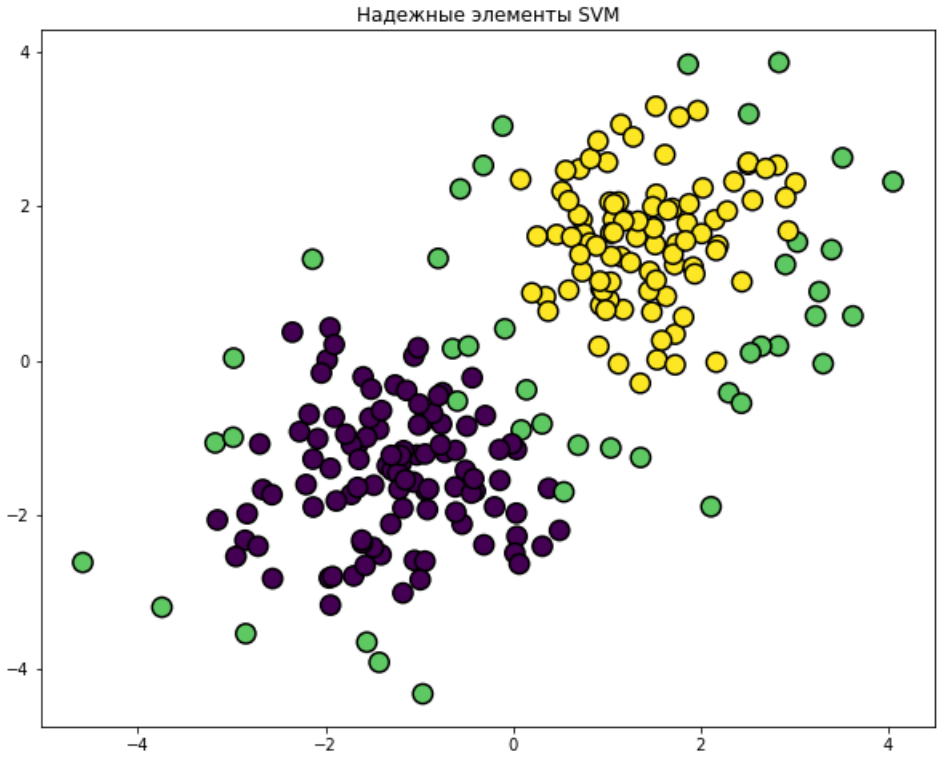

Рис. 5: svm 1

Можно заметить, что метод основанный на опорных векторах отмечает больше объектов как ненадежные.

Типичные представители классов для алгоритма SVM:

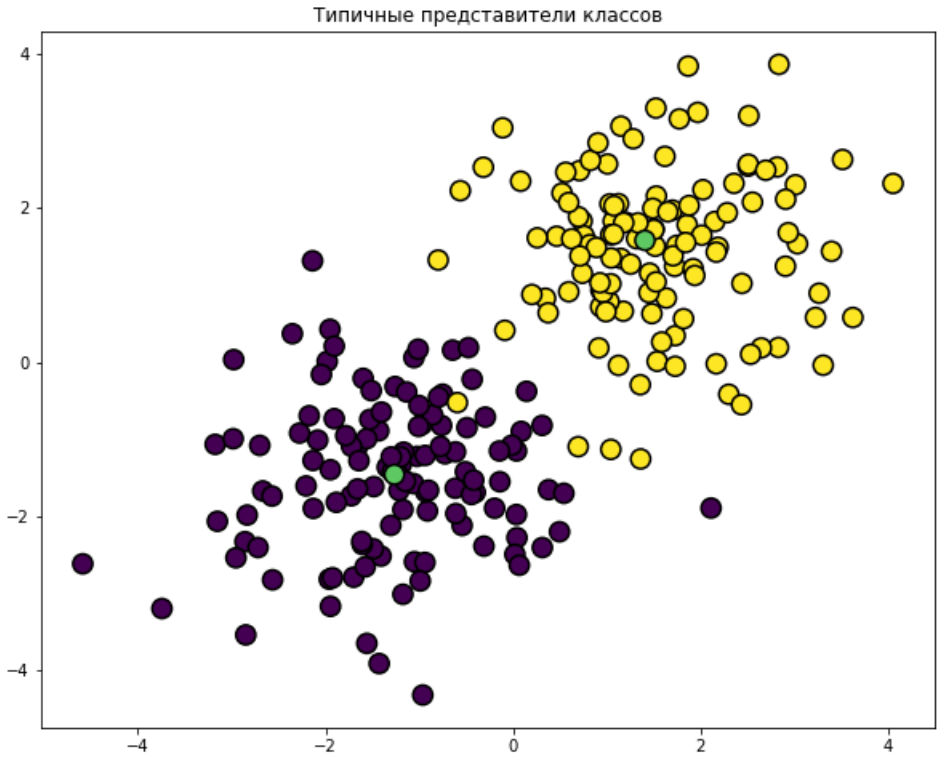

Рис. 6: svm 2

Посчитаем функционал качества L на типичных элементов, полученных алгоритмом SVM: $L_0 = 159.211$, $L_1 = 150.142$. Метрики $M_0(x) = p_0 = 0.846$ и $M_1(x) = p_1 = 0.868$ соответственно. Данный алгоритм отработал хуже, т.к. выбросил слишком много граничных, но значимых элементов.

4.1.3 Kmeans

Рассмотрим типичных представителей найденных из центров кластеров,полученных алгоритмом Kmeans:

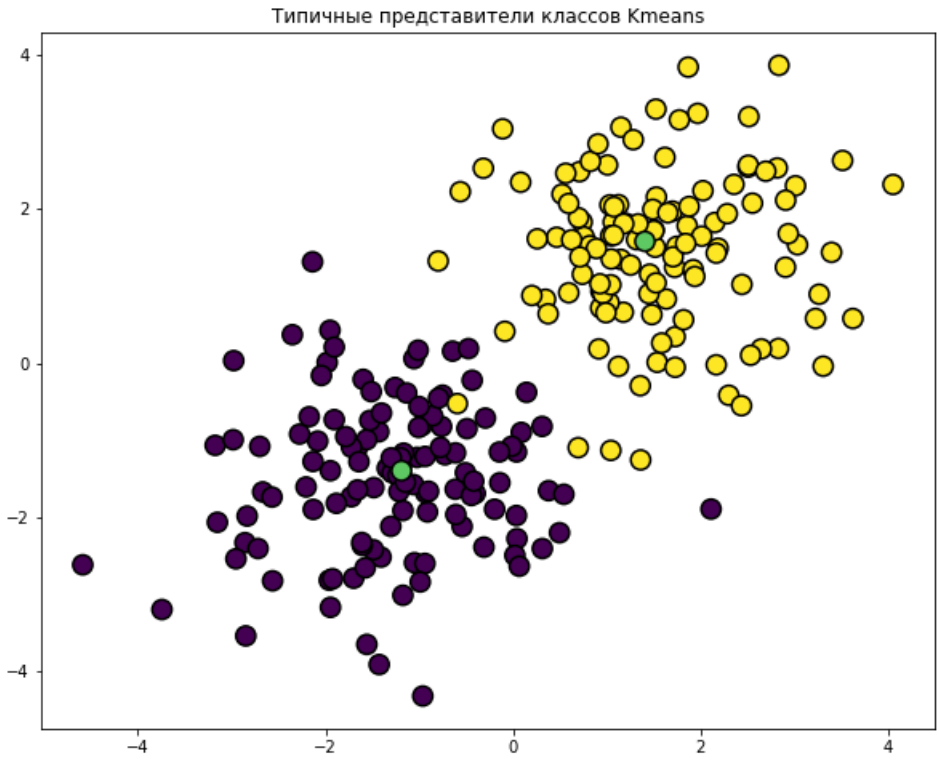

Рис. 7: kmeans

Вычислим функционалы ошибок и метрики на обоих классах: $L_0 = 152.097$, $L_1 = 147.432$ Метрики $M_0(x) = p_0 = 0.863$ и $M_1(x) = p_1 = 0.875$ соответственно. Данный алгоритм хорошо справляется с кластеризацией на этих данных, т.к. данные получены из нормально распределения.

4.1.4 Логические закономерности классов

Вычислим типичных представителей классов, полученных из метода логических закономерностей классов: Вычислим функционалы ошибок

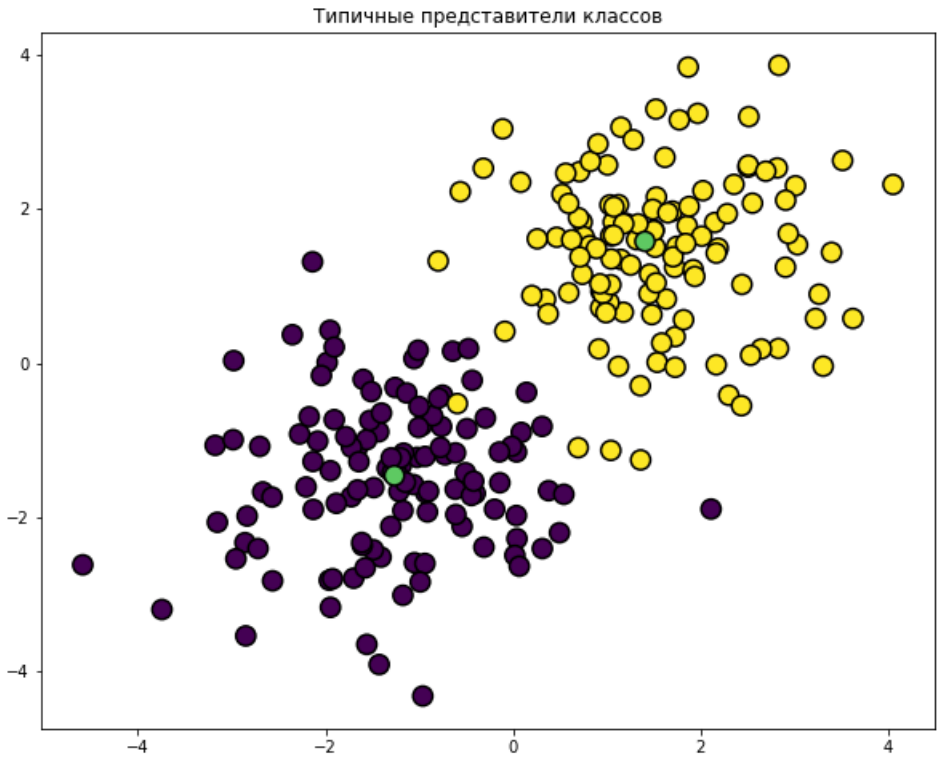

Рис. 8: lzk 1

и метрик на этих элементах:$L_0 = 154.013$, $L_1 = 149.434$ Метрики $M_0(x) = p_0 = 0.861$ и $M_1(x) = p_1 = 0.87$ соответственно.

4.1.5 Сравнение методов

Сравним полученные результаты. Из-за простой формы кластеров с этой задачей лучше всего справился алгоритм KMeans.

Таблица 1: Качество алгоритмов

Алгоритм	p_0	p_1
KNN	0.86	0.87
SVM	0.846	0.868
Kmeans	0.863	0.875
LZK	0.861	0.87

где p_0 и p_1 значение метрик для каждого из классов соответственно.

4.2 Набор данных №2

Сгенерируем набор данных более сложной формы X из двух классов и также имеющий 2 признака, используя библиотеку Python:

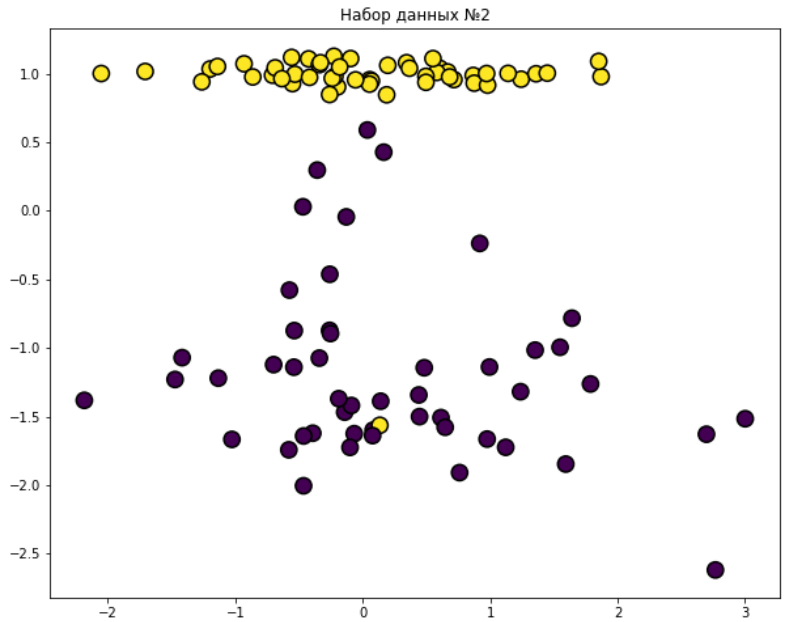

Рис. 9: Данные набора 2

Выполним плотностную кластеризацию алгоритмом DBSCAN и детектируем выбросы:

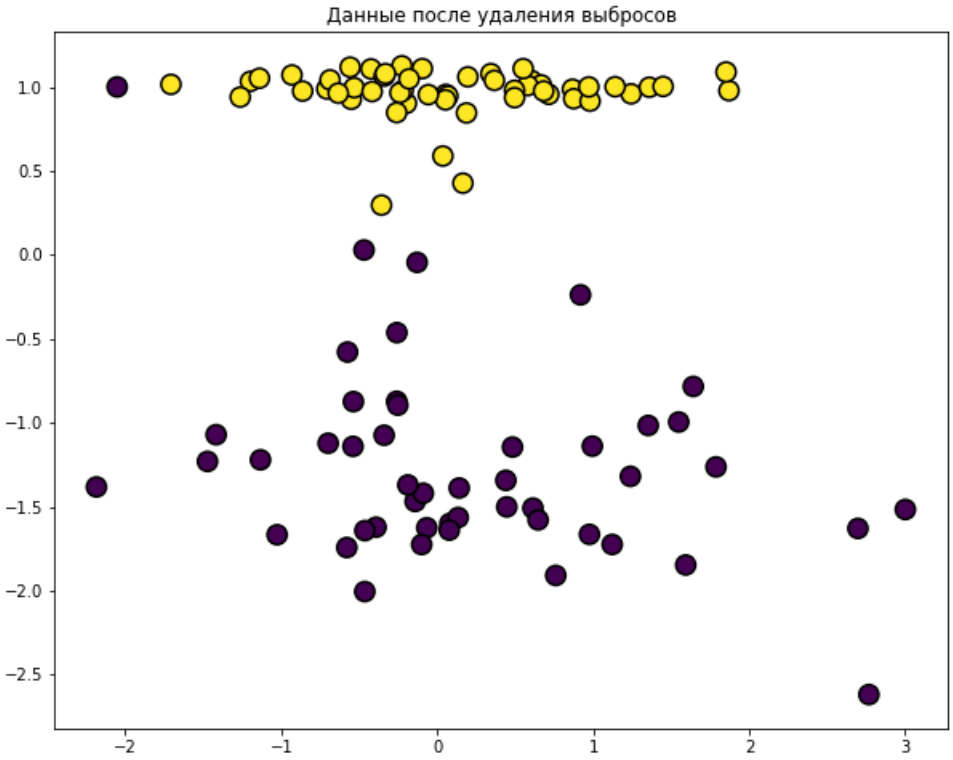

Рис. 10: DBSCAN

Кластеризация определила часть выбросов.

4.2.1 Метод ближайших соседей

Определим надежные элементы методом ближайших соседей с количеством соседей k = 9: В данном алгоритме метод ближайших соседей

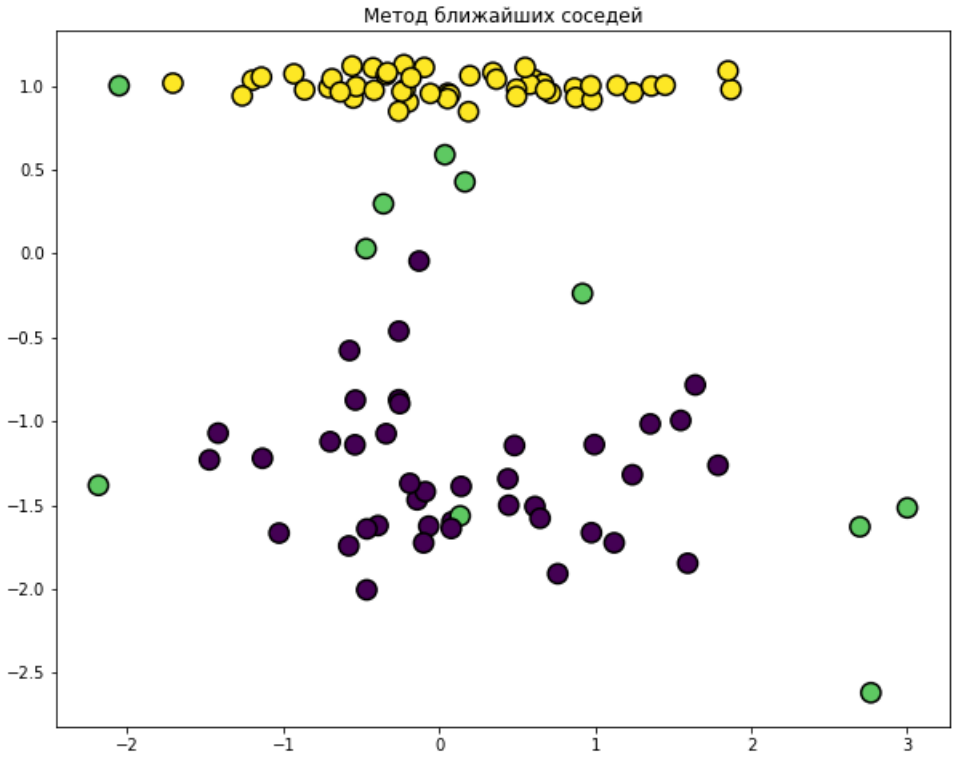

Рис. 11: knn 1

не относит большое количество объектов к множеству ненадежных элементов.

Типичные представители классов, полученные усреднением по всем надежным элементам: Качество для классов 0 и 1: $L_0 = 43.688$, $L_1 = 19.941$ Метрики $M_0(x) = p_0 = 0.835$ и $M_1(x) = p_1 = 0.869$ соответственно.

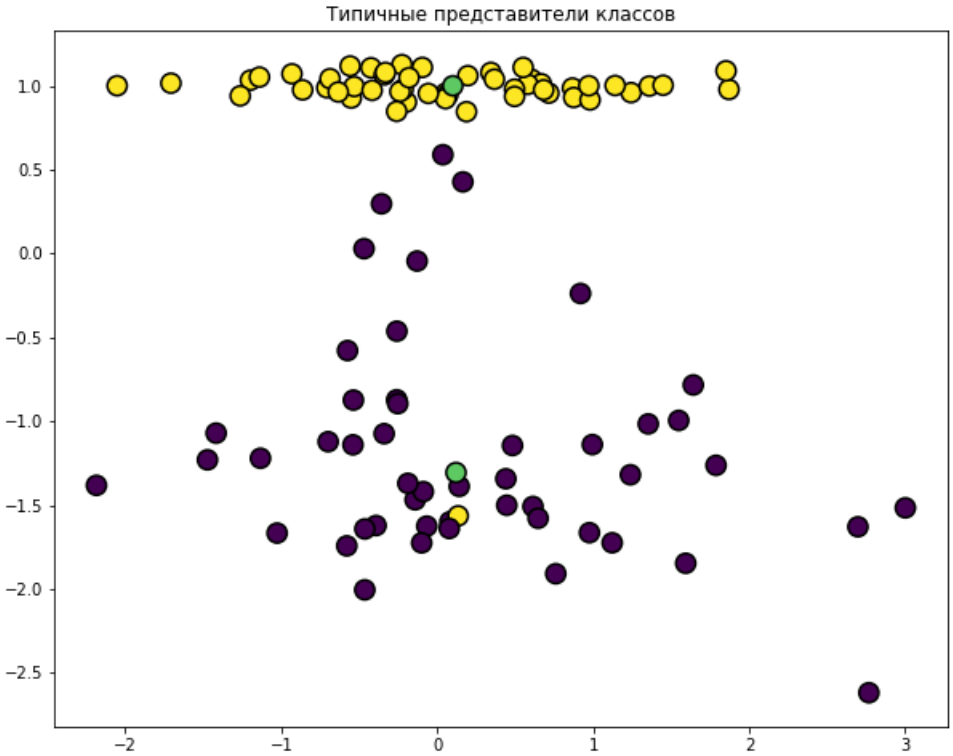

Рис. 12: knn 2

4.2.2 Метод опорных векторов(SVM)

Надежные элементы полученные алгоритмом SVM: На данном объеме данных метод опорных векторов четко строит границу между классами и выбрасывает разумное количество ненадежных элементов. Значение функционалов ошибок равно : $L_0 = 41.394$, $L_1 = 17.813$ для класса 0 и 1 соответственно. Метрики $M_0(x) = p_0 = 0.855$ и $M_1(x) = p_1 = 0.91$

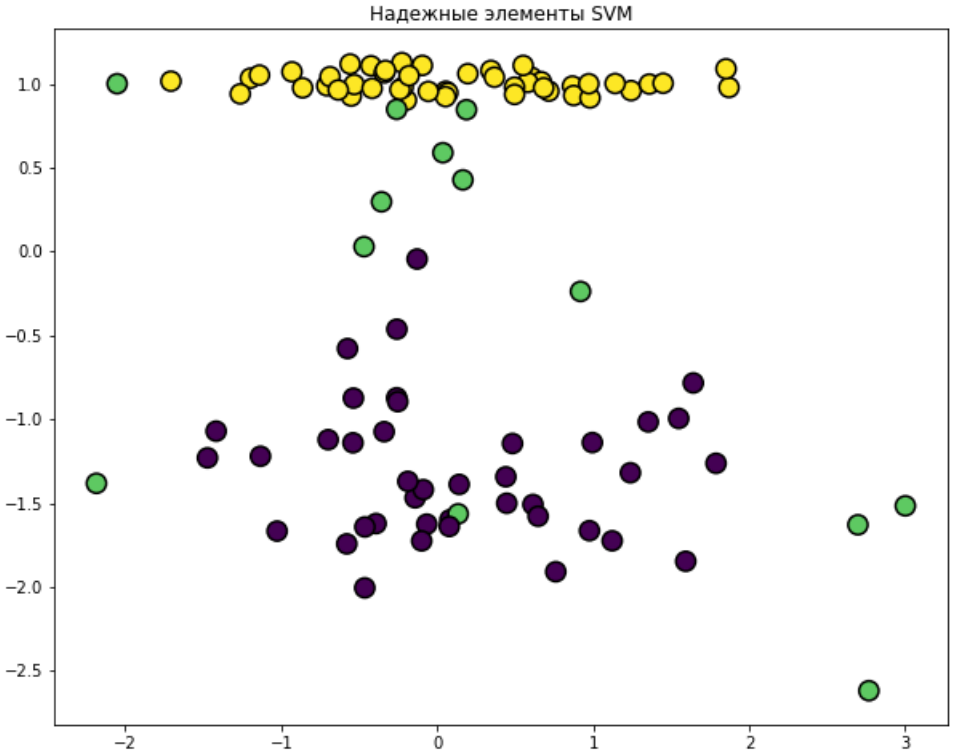

Рис. 13: svm 1

Типичные представители классов для алгоритма SVM:

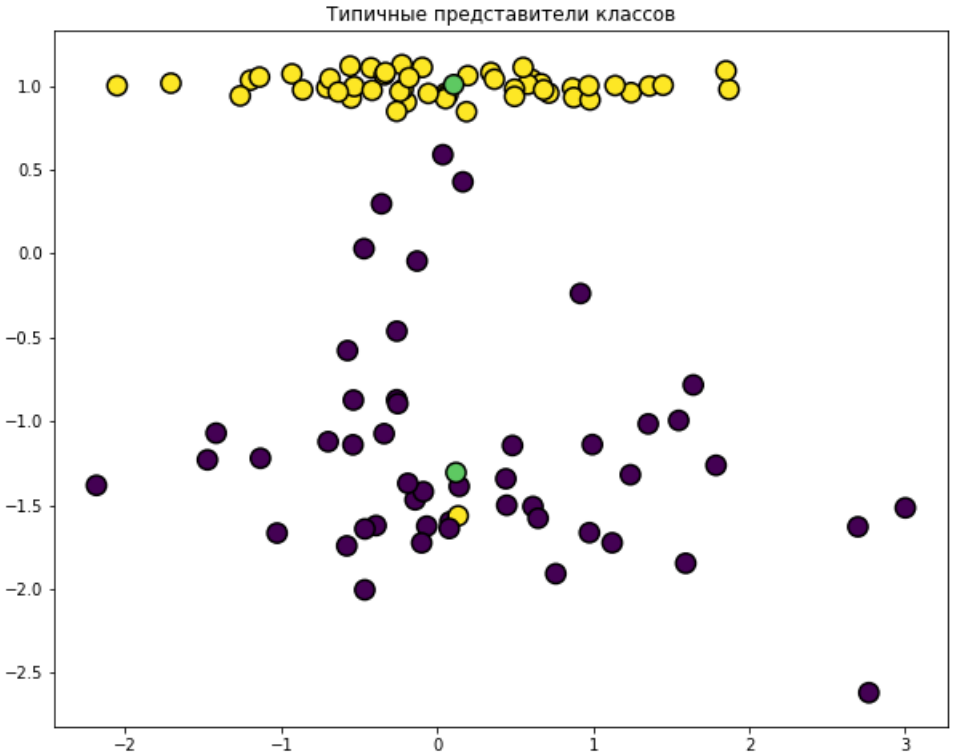

Рис. 14: svm 2

4.2.3 Kmeans

Типичные представители полученные алгоритмом Kmeans: Из-за необычной формы кластеров алгоритм плохо плохо справляется с задачей кластеризации.Качество: $L_0 = 43.237$, $L_1 = 23.491$ для класса 0 и 1 соответственно. Метрики $M_0(x) = p_0 = 0.839$ и $M_1(x) = p_1 = 0.808$

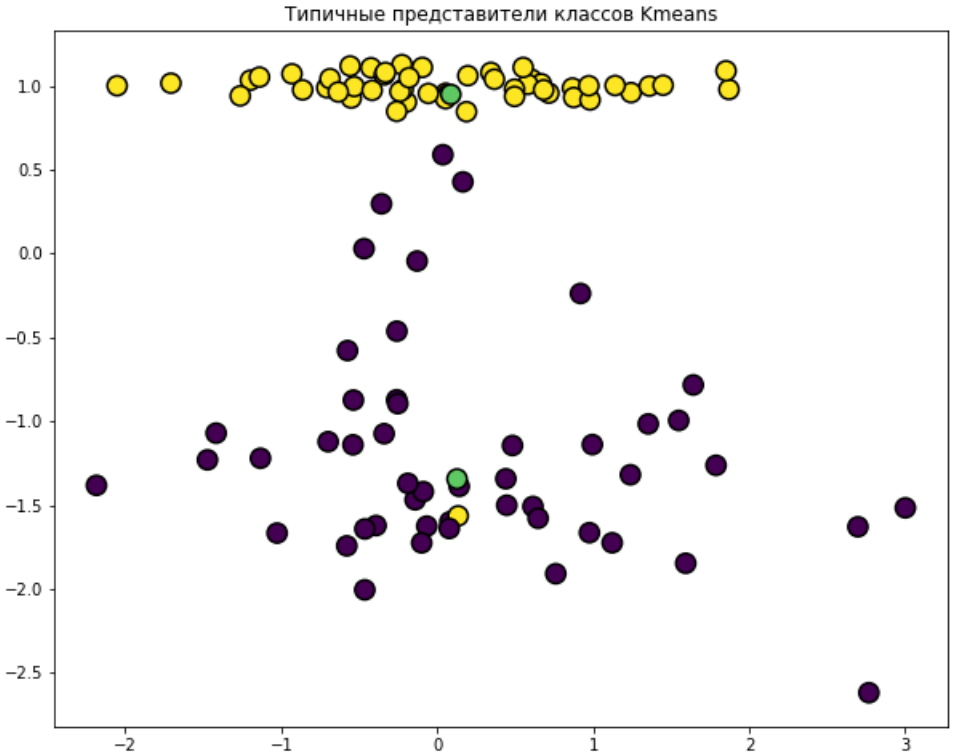

Рис. 15: kmeans

4.2.4 Логические закономерности классов

Типичные представители классов:
Рассчитаем качество для этого метода: $L_0 = 41.394$, $L_1 = 20.198$ для класса 0 и 1 соответственно. Метрики $M_0(x) = p_0 = 0.846$ и $M_1(x) = p_1 = 0.864$

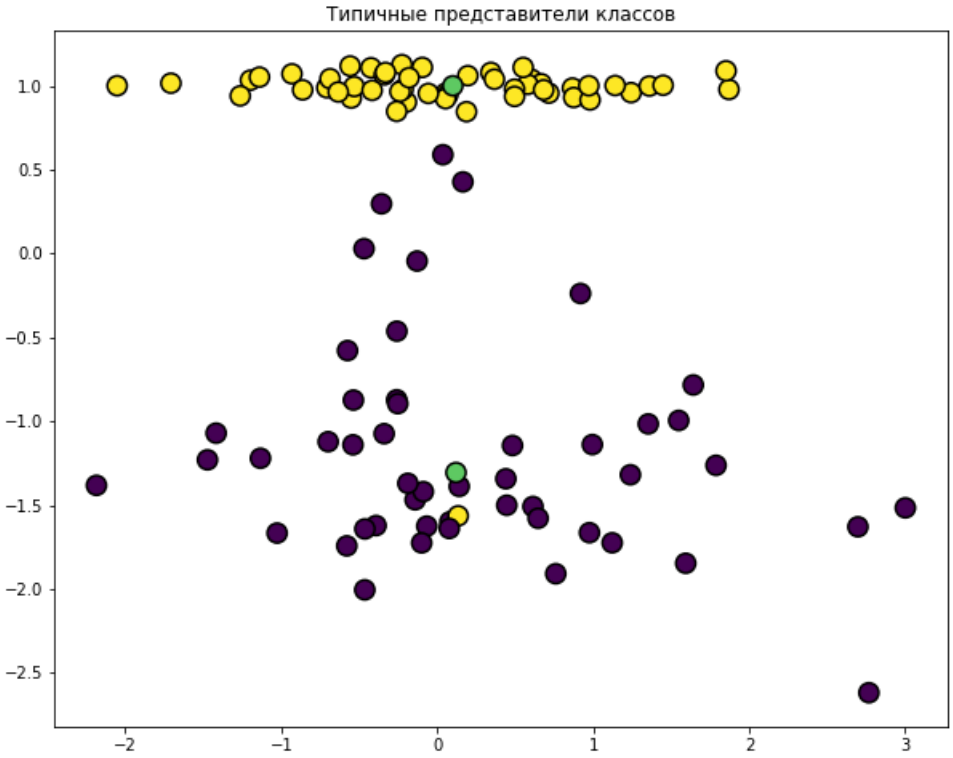

Рис. 16: lzk 2

4.2.5 Сравнение методов

С данной задачей лучше всего справился метод опорных векторов, т.к. форма кластеров легко разделяется прямой.

Таблица 2: Качество алгоритмов

Алгоритм	p_0	p_1
KNN	0.835	0.869
SVM	0.855	0.91
Kmeans	0.839	0.в808
LZK	0.846	0.864

4.3 Набор данных №3

Рассмотрим датасет $weather_AUS$ с историей осадков 49 городов Австралии. Предлагается определить типичный день когда идет дождь и типичный день без осадков. Данные содержат 24 признака и целевую переменную. Разделим данные на тестовую и обучающую выборку методом out_of_time.

В данном примере, помимо значения функционала L и метрик p, вычислим метрику AUC, которая отображает качество классификации.

4.3.1 Метод ближайших соседей

Рассмотрим метод ближайших соседей. Значение метрики auc на тестовой выборке 0.73562. $L_0 = 90.1233$, $L_1 = 92.9343$ для класса 0 и 1 соответственно. Метрики $M_0(x) = p_0 = 0.867$ и $M_1(x) = p_1 = 0.856$. Данный алгоритм плохо справляется с задачей классификации, и поэтому недостаточно успешно находит типичных представителей классов.

4.3.2 Метод опорных векторов

Значение качества классификатора на тестовой выборке: auc = 0.7156. И функционала ошибки L : $L_0 = 92.8376$ для первого класса и $L_1 = 93.2848$ для второго. Метрики $M_0(x) = p_0 = 0.856$ и $M_1(x) = p_1 = 0.854$. SVM справляется с классификацией еще хуже, чем knn.

4.3.3 Kmeans

Значение качества классификатора на тестовой выборке: auc = 0.50124.Функционал ошибки L : $L_0 = 92.6843$ для первого класса и $L_1 = 93.1085$ для второго. Метрики $M_0(x) = p_0 = 0.857$ и $M_1(x) = p_1 = 0.855$.

Таблица 3: Качество алгоритмов

Алгоритм	p_0	p_1
KNN	0.867	0.856
SVM	0.856	0.854
Kmeans	0.857	0.855
LZK	0.877	0.875

4.3.4 Логические закономерности классов

Данный алгоритм хорошо справляется с классификацией, т.к. решающие правила простые, auc = 0.79 Функционалы ошибок L: $L_0 = 87.5871$ для первого класса и $L_1 = 88.2541$ для второго. Метрики $M_0(x) = p_0 = 0.877$ и $M_1(x) = p_1 = 0.864$.

4.3.5 Сравнение алгоритмов

С данной задачей лучше всего справился метод логических закономерностей, т.к. данные имеют много признаков и интерпретируются простыми решающими правилами.(Таблица 3)

4.4 Выводы

На данный момент не существует общепринятого стандартного набора алгоритмов для решения задач поиска типичных представителей классов. На разных данных разные алгоритмы показывали лучшее качество. В настоящей работе был проведен обзор существующих подходов, были реализованы и представлены новые методы и сравнены эффективности их работы на реальных данных. Проведена связь алгоритмов классификации с алгоритмами поиска типичных представителей классов. Мы убедились, что не всегда существуют прямая корреляция между этими двумя задачами.

Стоит отметить, что в экспериментах участвовали алгоритмы по отдельности и не рассматривались объединения алгоритмов из разных семейств в композиции. Хотя известно, что подобные техники позволяют сильно улучшать итоговое качество работы систем машинного обучения, даже если базовые алгоритмы показывали не очень высокое качество. [6]

Список литературы

[1] Загоруйко Н.Г. Методы распознавания и их применение, 2003

[2] Рязанов В.В. Логические закономерности классов

[3] Ткачев Ю.И., Методы решения задачи восстановления зависимости коллективами распознающих алгоритмов

[4] Coursera.org, Специализация «Машинное обучение и анализ данных», MIPT, Yandex, https://ru.coursera.org/specializations/machine-learning-data-analysis, 2015

[5] Орпанен И.С. Плотностный алгоритм кластеризации (DBSCAN) project.org/ru/DBSCAN, 2016

[6] Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных machinelearning.ru