# РЕФЕРАТ

Магистерская диссертация содержит 54 страницы, 35 рисунков, 2 приложения, 12 источников.

МОРСКИЕ ВОЛНОВОДЫ, ПРОФИЛЬ СКОРОСТИ ЗВУА, МАШИННОЕ ОБУЧЕНИЕ, РАЗЛОЖЕНИЕ ПО БАЗИСУ, ГЛАВНЫЕ КОМПОНЕНТЫ, АЛГОРИТМЫ КЛАССИФИКАЦИИ.

В магистерской диссертации были исследованы алгоритмы сжатия информации на основе методов машинного обучения.

Во введении рассказывается об актуальности, теоретической и практической значимости. Перечислены цели и задачи, а также указаны объекты и предметы исследования.

В основной части, речь идёт об объекте исследования, дана краткая теоретическая справка по методам машинного обучения, в частности о методах снижения размерности и классификации, рассказано о методах интерпретации моделей машинного обучения.

В практической части приведены графики исходных профилей ВРСЗ для Баренцева моря и их аппроксимации. Приведены графики точности классификации при снижении размерности и без профиля ВРСЗ, приводится сравнение между собой методов сжатия данных и их влияние на точность классификации профилей ВРСЗ по сезонам и месяцам.

В заключении происходит подведение итогов и составление вывода по проделанной работе.

СОДЕРЖАНИЕ

# ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

– ВРСЗ - вертикальное распределение скорости звука,

– PCA - principal component analysis,

– AE - autoencoder,

– K-SVD - singular value decomposition,

– MSE - mean squared error (среднеквадратическое отклонение),

– X - выборка профилей ВРСЗ.

# ВВЕДЕНИЕ

Акустика океана – раздел гидроакустики, изучающий распространение звуковых волн в океане. Звуковые волны - единственный вид излучения, который способен распространяться в океане на расстояния в сотни и тысячи километров. Поэтому акустика океана играет важную роль в процессе освоения и изучения океана. На использовании звуковых волн основаны подводная локация и связь, обнаружение рыбных косяков, изучение биологического состава глубоководных звукорассеивающих слоёв, ветрового волнения, неоднородностей водной толщи и подводного грунта.

Целью работы является изучение методов аппроксимации вертикального распределения скорости звука (ВРСЗ) в морях и океанах, изучение профиля ВРСЗ с целью выделения наиболее важной информации о сезоне.

Задачи работы:

- Написание программы для получения из базы профилей обучающей и тестовой выборки ВРСЗ в важных районах мирового океана.
- Применение методов сжатия размерности, с целью уменьшения пространства параметров, описывающих профили ВРСЗ.
- Применение алгоритмов классификации с целью выявления из профиля ВРСЗ наиболее важной информации о сезоне.

# ОСНОВНАЯ ЧАСТЬ

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Обзор объекта исследования

В однородной и изотропной среде акустические волны распространяются прямолинейно и с постоянной скоростью, не зависящей от направления распространения.

В океане среднее значение скорости звука обычно принимается равным 1500 м/с. В зависимости от температуры, давления, глубины, солености, эта величина может, изменяться на несколько десятков м/с в ту или другую сторону.

Для расчета скорости звука в океане используется формула Вильсона, предложенная им в 1960 году. Формула Вильсона принята Национальным центром сбора океанографических данных (NODC) США для машинной обработки гидрологической информации.

Формула Вильсона имеет следующий вид:

$$c(\mathrm{S},\mathrm{T},\mathrm{P}) = c_0 + \Delta c_T + \Delta c_S + \Delta c_P + \Delta c_{STP},$$

$$c_0 = 1449.14,$$

$$\Delta c_T = 4.5721T - 4.4532\cdot10^{-2}T^2 - 2.6045\cdot10^{-4}T^3 + 7.9851\cdot10^{-6}T^4,$$

$$\Delta c_S = 1.39799(S-35) + 1.69202\cdot10^{-3}(S-35)^2,$$

$$\Delta c_P = 1.63432P + 1.06768\cdot10^{-3}P^2 + 3.73403\cdot10^{-6}P^3 - 3.6332\cdot10^{-8}P^4,$$

$$\Delta c_{STP} = (S-35)(-1.1244\cdot10^{-2}T + 7.7711\cdot10^{-7}T^2 + 7.85344\cdot10^{-4}P -$$

$$-1.3458\cdot10^{-5}P^2 + 3.2203\cdot10^{-7}PT + 1.6101\cdot10^{-8}T^2P) +$$

$$+P(-1.8974\cdot10^{-3}T + 7.6287\cdot10^{-5}T^2 + 4.6176\cdot10^{-7}T^3) +$$

$$+P^2(-2.6301\cdot10^{-5}T + 1.9302\cdot10^{-7}T^2) + P^3(-2.0831\cdot10^{-7}T).$$

Где $c(\mathrm{S},\mathrm{T},\mathrm{P})$ - скорость звука, м/с; T – температура, °C; S – соленость, промилле; P – гидростатическое давление, МПа.

В СССР по этой формуле в 1965 г. Были составлены и изданы гидрографической службой ВМФ «Таблицы для расчета скорости звука в морской воде» [1].

В каждом районе океана температура, соленость, давление зависят, в первую очередь, от глубины. Таким образом скорости звука, определяемая формулой Вильсона, можно рассматривать также как зависимости от глубины. Кривая $c(z)$ называется вертикальным распределением или профилем скорости звука.

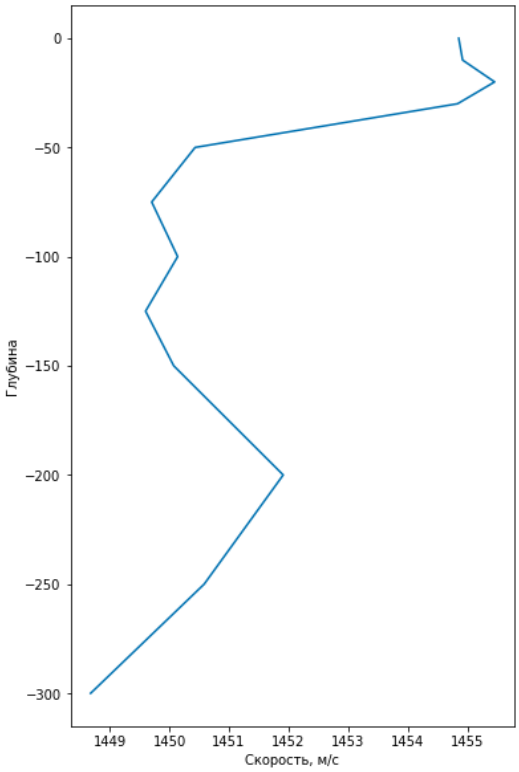


Рис. 1.1 Профиль ВРСЗ

Вид профиля скорости звука определяется типом термической структуры вод океана. Очень существенную роль играет глубина места в океане, так как эффекты, связанные с давлением, проявляются на больших глубинах.

## 1.2 Машинное обучение

Машинное обучение, это процесс, в ходе которого система обрабатывает большое число примеров, выявляет закономерности и использует их, чтобы прогнозировать характеристики на новых данных. Большую часть задач машинного обучения можно разделить на обучение с учителем (supervised learning) и обучение без учителя (unsupervised learning). Под «учителем» здесь понимается сама идея вмешательства человека в обработку данных. При обучении с учителем у нас есть данные, на основании которых нужно что-то

предсказать, и некоторые гипотезы. При обучении без учителя у нас есть только данные, свойства которых мы и хотим найти.

В данной работе использовано как обучение с учителем, так и обучение без учителя.

### 1.2.1 Снижение размерности

В данной работе задача снижения размерности осуществляется алгоритмами машинного обучения без учителя. Требуется каждый вектор профиля ВРСЗ перевести вектор меньшей размерности, т.е. для каждого профиля ВРСЗ $x^i \in \mathbb{R}^n$, $i = 1..N$, где $N$ это количество профилей в выборке, найти соответствующий ей кодированный вектор $c^i \in \mathbb{R}^k$. Если k меньше n, то для хранения кодированных точек потребуется меньше памяти, чем для исходных. Ставится задача найти функцию кодирования $f(x) = c$ и функцию декодирования $x \approx g(f(x))$ минимизирующую среднеквадратическое отклонение.

$$\|x - g(f(x))\| \xrightarrow[w]{} \min, \text{ где}$$

$w$ - параметры алгоритма сжатия, подобранные по обучающей выборке. В работе рассматриваются следующие алгоритмы сжатия:

- PCA (principal component analysis, метод главных компонент).
- K-means(K средних)
- K-SVD (обобщение метода K-means с использование SVD разложения)
- Autoencoder (автокодировщик)

Опишем первые метод снижения размерности для выборки ВРСЗ - PCA.

Теорема (Дж. Форсайт). Для любой вещественной матрицы $A$ существуют две вещественные ортогональные матрицы $U$ и $V$ такие, что

$$U^T A V = \Lambda$$

Более того, можно выбрать $U$ и $V$ так, чтобы диагональные элементы $\Lambda$ имели вид

$$\lambda_1 \geq \lambda_2 \geq .... \geq \lambda_r = ... = \lambda_n, \text{ где}$$

$r$ - ранг матрицы $A$.

Метод главных компонентов (PCA) представляет собой статистическую процедуру, которая использует ортогональное преобразование для преобразования набора наблюдений возможных коррелированных переменных в набор значений линейно некоррелированных переменных, называемых главными компонентами. Число главных компонентов меньше или равно меньшему числу исходных переменных или количеству наблюдений. Это преобразование определяется таким образом, что первый главный компонент имеет наибольшую возможную дисперсию (то есть учитывает, как можно большую часть изменчивости данных), и каждый последующий компонент, в свою очередь, имеет наибольшую дисперсию, возможную при ограничении, что он ортогонален предыдущим компонентам. Полученные векторы представляют собой некоррелированный ортогональный базисный набор. PCA - чувствителен к относительному масштабированию исходных переменных.

PCA в основном используется в качестве инструмента для анализа разведочных данных и для создания прогнозирующих моделей. PCA может быть выполнено путем декомпозиции собственной ковариационной (или корреляционной) матрицы данных или сингулярной декомпозиции матрицы данных, как правило, после среднего центрирования и нормализации матрицы данных для каждого атрибута. Результаты применения PCA в данной работе используются для разложения профиля ВРСЗ по главным компонентам, количество которых меньше размерности профиля скорости звука. [мой диплом]

Один из способов нахождения главных компонент, это использовать сингулярное разложение. Пусть $X$ - обучающий набор профилей ВРСЗ. Предварительно центрируем выборку ВРСЗ при помощи вычета среднего вектора по всем профилям из каждого профиля.

$$X^0 = X - \bar{X}, \text{ где}$$

$\bar{X}$ - усредненный по всем координатам профиль ВРСЗ. Требуется найти собственные векторы матрицы ковариационной матрицы:

$$A = \frac{1}{n} X^{0T} X^{0},$$

$$(A - \lambda_k E) v_k = 0.$$

Воспользуемся сингулярным разложением матрицы $X^0$:

$$X^0 = U \sum V^T, \text{ где}$$

$U$, $V$ - ортогональные матрицы, $\sum$ - диагональная. Отсюда нетрудно видеть, что главные компоненты это столбцы ортогональной матрицы $U$.

Найдя все главные компоненты для тренировочного набора профилей ВРСЗ, возьмем профиль, который не участвовал в обучении (поиске главных компонент) и разложим по базису из $k \le n$ главных компонент.

Теперь опишем алгоритм основанный на кластеризации выборки ВРСЗ - K-means. В машинном обучении метод k-means является методом кластеризации данных (присвоение каждому элементу выборки метку из конечного числа кластеров заранее неизвестных). На вход алгоритму подадим обучающие профили ВРСЗ с заранее известным набором кластеров. Алгоритм попытается разбить выборку профилей ВРСЗ на кластеры таким образом, чтобы дисперсия в каждом кластере была минимальна. Идея алгоритма заключается в том, что на каждой итерации перевычисляется центр каждого кластера, полученного на предыдущем шаге. Затем каждый профиль ВРСЗ отнесем к кластеру, центр которого к нему ближе всего по выбранной метрике (например, евклидовой). Алгоритм завершается, когда на какой-то итерации не происходит изменения кластеров.

Несмотря на это, можно провести параллели между алгоритмом K-means и PCA. Если метод PCA пытается представить данные в виде суммы главных компонент, то метод k-means напротив, пытается представить каждую точку данных в пространстве, используя один центр кластера. Разбивая n-мерное пространство выборки ВРСЗ на k кластеров, каждый вектор ВРСЗ кодируется в k-мерном пространстве по принципу one-hot-encoding, то есть кодируется вектором из k элементов, в котором на $i$-ом месте стоит единица, а все

остальные нули, где $i$ – номер кластера к которому принадлежит данный профиль ВРСЗ [3].

Рассмотрим теперь обобщение метода K-Means – K-SVD. Данный метод является еще одним способом получить разреженное представления данных. Это частный случай метода под названием dictionary learning. Он позволяет эффективно находить словарь (набор базисных векторов ВРСЗ из обучающей выборки) с помощью SVD разложения и является своего рода обобщением метода k-means. В отличии от k-means, K-SVD позволяет получать кодированный вектор с заранее заданным количеством ненулевых элементов (в k-means только один не нулевой элемент). В методе K-SVD решается следующая задача оптимизации:

$\min_{X,D}\{\|X - DV\|_F^2\}$, при условии $\|v_i\|_0 \le T_0$, для $\forall i \in 1..N$, где

$X \in \mathbb{R}^{n\times N}$ - обучающий набор профилей ВРСЗ, $D \in \mathbb{R}^{n\times k}$ – базисные профили, $V \in \mathbb{R}^{k\times N}$ - кодированные профили ВРСЗ, $F$ - норма Фробениуса (корень из суммы квадратов всех элементов матрицы), $\|.\|_0$ - $l_0$ норма вектора (количество ненулевых элементов в векторе), $T_0$ - максимальное количество ненулевых элементов в кодированном векторе ВРСЗ.

Данная задача решается итеративным поиском при помощи SVD разложения, более подробный алгоритм изложен в [4]

K-SVD является обобщением метода k-means, так как при $T_0 = 1$, K-SVD эквивалентен методу k-means. Восстановленный вектор по методу K-SVD уже является не центром одного кластера, как это было в k-means, а их линейной комбинацией [3].

Рассмотрим теперь метод сжатия основанный на нейронных сетях. Как уже говорилось ранее, для сжатия размерности данных требуется найти такую функцию кодирования $f(x) = c$, $x \in \mathbb{R}^n$, $c \in R^k$ $(k \le n)$ и функцию декодирования $x \approx g(f(x))$, которая наилучшим образом (с точки зрения заданной метрики качества) будет приближать исходный вектор по сжатому отображению.

Теорема Цыбенко. Искусственная нейронная сеть прямой связи с одним скрытым слоем может аппроксимировать любую непрерывную функцию многих переменных с любой точностью. Условиями являются: достаточное количество нейронов скрытого слоя [2].

Данная теорема утверждает, что мы можем аппроксимировать функцию $f(x)$ и $g(c)$ с любой точностью.

Для данного типа задач существует специальная архитектура нейронной сети под названием autoencoder (автокодировщик). Она состоит из двух частей: кодирование и декодирование.

На рис. 1.2 приведена архитектура автокодировщика с одним скрытым слоем кодирования и одним скрытым слоям декодирования.

Методом обратного распространения ошибки, нейронную сеть можно обучить кодировать данные в пространство меньшей размерности (нужно всего лишь минимизировать среднеквадратическое отклонение между исходными векторами и векторами на выходе нейронной сети). Выбирая количество нейронов на скрытом слое С, будем задавать размерность пространства, в которое перейду профили ВРСЗ.

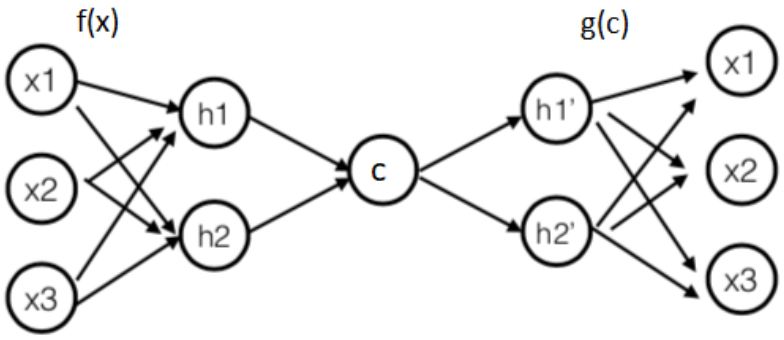


Рис. 1.2 Архитектура autoencoder (автокодировщик)

После обучения нейронной сети, обрезав вторую часть, которая отвечает за декодирование данных, и оставив только часть, отвечающую за функцию кодирования $f(x)$, будем подавать на вход сети профили ВРСЗ и на выходе будем получать закодированные вектора размерности скрытого слоя С. Так как у нас остался обученный декодер $g(c)$, мы в любой момент можем

восстановить (с потерей точности конечно) профили ВРСЗ прогнав их через сеть декодера [3].

### 1.2.2 Классификация

Классификация — вид алгоритмов машинного обучения с учителем. Имеется множество объектов (профилей), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

В данной работе осуществляется классификация профилей ВРСЗ на зимний и летний сезон, а также отдельно по каждому месяцу. Цель классификации – исследовать на сколько хорошо разделяются профили по сезонам и по месяцам, выявить какая глубина наиболее важна для определения месяца и сезона. Также при помощи алгоритмов классификации исследовано какой из алгоритмов снижения размерности сохраняет наибольшую информацию о сезоне. Для алгоритмов сжатия размерности, которые используют базисные профили, с помощью алгоритма классификации исследовать какой базисный профиль несет наибольшую информацию о сезоне и месяце.

Классификация бывает двух видов: бинарная и многоклассовая. В случае классификации на зимний/летний сезон - это бинарная классификация (2 класса), а классификация по месяцам - это многоклассовая классификация (12 классов). Задачу многоклассовой классификации можно свести к решению нескольких бинарных задач. Для этого существуют следующие основные стратегии выбора решения [5]:

- «Один против всех» (one-against-all). Для решения задачи строится $n$ классификаторов таким образом, что каждый класс $a_j$ сопоставляется с остальными ($n$-1) классами, т.е. в каждом из $j$ случаев выбор осуществляется из двух вариантов: «класс $a_j$» и «не класс $a_j$». Итоговое

решение по всем классам принимается по схеме «победитель забирает всё» (winner takes all) - победителем считается класс, имеющий максимальное значение целевой функции (вероятности).

- «Каждый против каждого» (one-against-one). Классификаторы строятся для каждой пары классов для того, чтобы можно было однозначно разделить любые два класса из множества *A*. Количество классификаторов в этом случае равно $n(n\text{-}1)/2$. После подачи на входы каждого из обученных классификаторов тестового образца получаем ответы, содержащие информацию о его принадлежности одному из двух классов, участвовавших в обучении. К полученному множеству ответов применяется схема мажоритарного голосования и класс, выбранный большинством классификаторов, принимается как итоговое решение.

Существует не мало оценок качества классификации. В данной работе будет использоваться самая простая метрика точности классификации – точность (accuracy)

$$\text{Accuracy} = \frac{P}{N}, \text{ где}$$

$P$ – количество правильно классифицированных профилей ВРСЗ. $N$ – объем выборки ВРСЗ поданной алгоритму для классификации.

В работе использованы следующие алгоритмы классификации:

- Логистическая регрессия
- Случайный лес
- Градиентный бустинг над решающими деревьями

Логистическая регрессия (Logistic regression) — метод построения линейного классификатора, позволяющий оценивать апостериорные вероятности принадлежности объектов классам, т.е. предсказывает вероятность события по его признакам. В случае данной работы, это предсказать к какому сезоны/месяцу относится профиль ВРСЗ по его координатам скорости звука. Пусть $y$ - целевая переменная, отвечающая за то к какому сезону/месяцу принадлежит профиль ВРСЗ. Для простоты будем

рассматривать бинарную классификацию на зимний и летний сезон. Алгоритм классификации по месяцам легко обобщается методом «Один против всех» или «Каждый против каждого». Пусть $y = 0$ для зимнего профиля ВРСЗ и $y = 1$ для летнего. Имеется множество координат скорости звука вектора ВРСЗ $x_1, x_2, ..., x_n$ на основе которых требуется принять решение к какому сезону принадлежит вектор.

Рассмотрим вероятность того, что выбранный профиль относится к летнему сезону ( $y = 1$):

$$P(y = 1 | x) = f(z), \text{ где}$$

$z = \Theta^T x = \Theta_1 x_1 + ... + \Theta_n x_n . \Theta_1 ... \Theta_n -$ параметры модели (коэффициенты регрессии), $f(z) -$ логистическая функция или как ее называют чаще - сигмоида.

$$f(z) = \frac{1}{1 + e^{-z}}$$

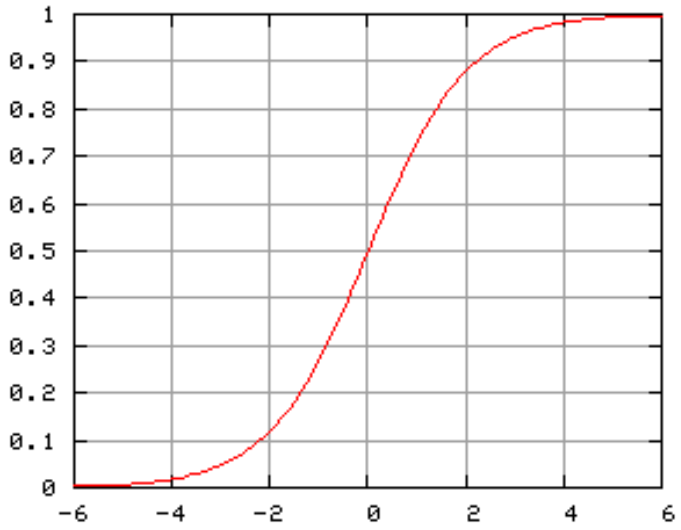


Рис. 1.3 Сигмоида

Так как $f(z) -$ вероятность принадлежности профиля ВРСЗ к летнему сезону, то вероятность принадлежности к зимнему равна $1 - f(z)$.

Более кратко, это можно объединить в одну формулу

$$P(y \mid x) = f(z)^{y}(1 - f(z))^{1-y}.$$

Для подборки параметров $\Theta_1 ... \Theta_n$ составим обучающую выборку из профилей ВРСЗ, в которую обязательно должны входить как летние так и зимние профили. Составим пары $(X_i, y_i)$, множество таких пар составит обучающую выборку для классификатора. Для обучения модели (поиск параметров $\Theta_1 ... \Theta_n$) воспользуемся методом максимального правдоподобия:

$$\hat{\Theta} = \arg\max_{\Theta} L(\Theta) = \arg\max_{\Theta} \prod_{i=1}^{N} P(y = y^i \mid x = x^i)$$

Максимизация такой функции эквивалента максимизации логарифма этой функции

$$\log L(\Theta) = \sum_{i=1}^{N} \log P(y = y^i \mid x = x^i) = \sum_{i=1}^{N} y^i \log f(\Theta^T x^i) + (1 - y^i) \log(1 - f(\Theta^T x^i))$$

Обычно, для максимизации такой функции, используют метод градиентного спуска [6]:

$$\Theta := \Theta - \alpha \vec{\nabla} \log L(\Theta) = \Theta - \alpha \sum_{i=1}^{N} (y^i - f(z)) x^i, \alpha > 0.$$

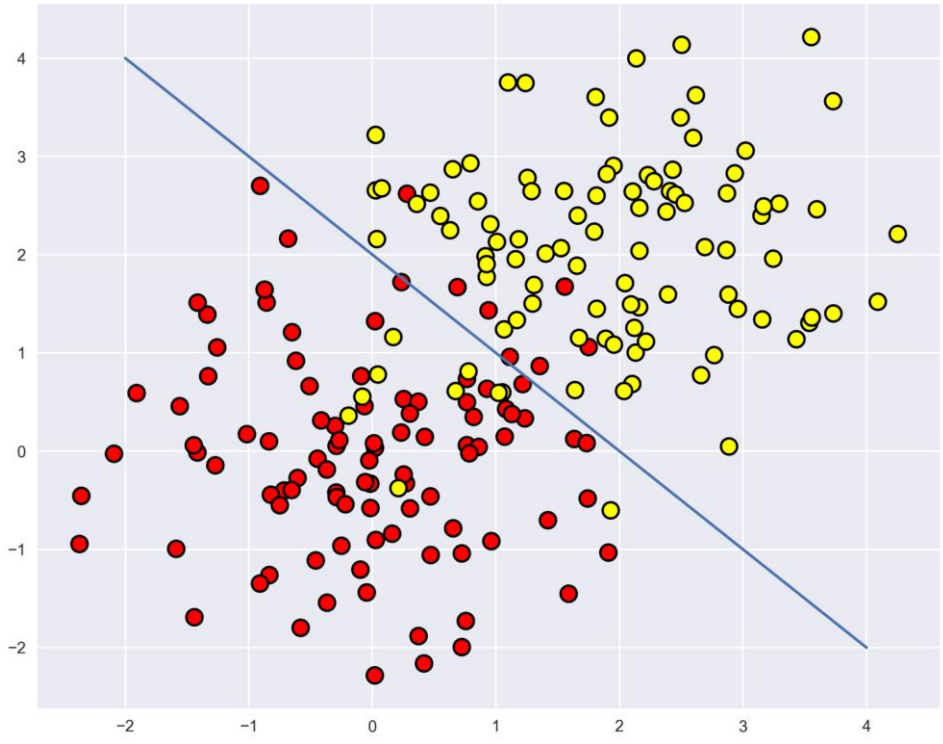

Рис. 1.4. Пример разделения двумерной выборки логистической регрессией

Случайный лес - это алгоритм машинного обучения для задач классификации и регрессии. Случайный лес состоит из множества более простых алгоритмов под названием решающие деревья. В задаче регрессии ответы всех решающих деревьев усредняются, а в задаче классификации принимается решение голосованием по большинству.

Дерево решений как алгоритм машинного обучения – по сути объединение логических правил вида «значение признака» $a$ меньше (больше) $a_0$ и «значение признака» $b$ меньше (больше) $b_0$ и т.д. по всем признакам. Если выполняется один набор условий, то это класс $i$, если другой набор, то класс $j$. Огромное преимущество деревьев решений в том, что они легко интерпретируемы, понятны человеку. Например, в случае с выборкой ВРСЗ, дерево будет сравнивать значение координат скорости с пороговыми значениями и в зависимости от этого принимать решение к какому сезону/месяцу отнести данный профиль.

Алгоритм построения решающего дерева.

Возьмем выборку ВРСЗ и найдем ее наилучшее разбиение на две части по одной из координат скорости звука $R_1(i,t) = \{x \mid x_j < t\}$ и $R_2(j,t) = \{x \mid x_j \geq t\}$ с точки зрения заранее заданного функционала качества. Найдя наилучшее значения $i$ и $t$ создадим корневую вершину дерева. Все профили ВРСЗ разобьются на две части и пойдут либо в левое поддерево, либо в правое.

Для каждой из разделенных подвыборок ВРСЗ повторим процедуру рекурсивно поставив дочерние вершины до корневой, и так далее. В каждой вершине необходимо проверить выполнение условия останова. В построенном дереве, каждому листу ставится свой ответ. Профилю ВРСЗ который прошел все условия до данного листа ставится соответствующая данному листу метка

класса. В случае классификации, это класс, который попал больше всего в данный лист при обучении.

При построении дерева требуется задать функционал качества на основе которого осуществляется разбиение выборки. Чаще всего используются один из следующий функционалов качества.

Ошибка классификации:

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x_i, y_i) \in \mathbb{R}} [y_i \neq c]$$

Оптимальным предсказанием в данном функционале качества будет самый популярный класс.

Критерий Джини:

$$H(R) = \min_{\sum_k c_k = 1} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} \sum_{k=1}^{K} (c_k - [y_i = k])^2.$$

Такой критерий выдает в вершине не один класс, а распределение на всех классах

$$c = (c_1 \ldots c_k), \sum_{k=1}^{K} c_k = 1.$$

Тогда оптимальный вектор вероятностей состоит из долей классов

$$c_* = (p_1 \ldots p_k).$$

Отсюда

$$H(R) = \sum_{k=1}^{K} p_k (1 - p_k).$$

Энтропийный критерий:

$$H(R) = \min_{\sum_k c_k = 1} \left(-\frac{1}{|R|} \sum_{(x_i y_i) \in R} \sum_{k=1}^{K} [y_i = k] \log c_k\right)$$

При выполнении условий $\sum_{k=1}^{K} c_k = 1$, можно показать, что данный критерий будет представлять собой энтропию распределения классов:

$$H(R) = -\sum_{k=1}^{K} p_k \log p_k.$$

Из теории вероятностей известно, что энтропия ограничена снизу нулем, причем минимум достигается на вырожденных распределениях ( $p_i = 1, p_j = 0$ для $i \neq j$ ).

Максимальное же значение энтропия принимает для равномерного распределения. Отсюда видно, что энтропийный критерий отдает предпочтение более «вырожденным» распределениям классов в вершине.

Как уже говорилось ранее, для остановки обучения используется критерий останова. Это может быть, как:

- Ограничение глубины дерева.
- Ограничение на число объектов в листе.
- Ограничение на максимальное количество листьев.
- Все объекты в листе относятся к одному классу.
- Требование, что функционал качества улучшался на $s$ процентов.

К сожалению алгоритм решающих деревьев очень склонен к переобучению (дает хорошее качество на тренировочных данных и плохое на данных которые модель не видела), если объединить много решающих деревьев в ансамбль, обученных как на разных подвыборках данных, так и на подвыборках признаков в одну модель, получим модель под названием случайный лес. Плюсами такой модели будет: высокая точность предсказания,

устойчивость к переобучению (модели случайного леса почти никогда не переобучаются), не чувствителен к выбросам. Так же минусами полученной модели является: сложность интерпретации предсказания, работает хуже линейных методов при разреженных данных, не умеет экстраполировать.

Алгоритм построения случайного леса:

- Пусть количество объектов в обучающей выборке $N$, а количество признаков $n$.
- Выбрать $L$ как число отдельных моделей.
- Для каждой модели $l$, выбрать количество признаков $d < D$ и обучающую подвыборку размера $z < N$.
- Обучить каждую модель.
- Объединить результаты отдельных моделей мажоритарным голосованием (для классификации) или усреднением (для регрессии).

Бустинг — это техника построения ансамблей моделей, в которой предсказатели построены не независимо, а последовательно. Как правило используется бустинг над решающими деревьями, то есть строится последовательно много не глубоких (как правило глубиной от 2 до 8) решающих деревьев, каждое из которых исправляет ошибки предыдущей модели.

Алгоритм построения градиентного бустинга.

Пусть $a_M(x)$ – сумма базовых алгоритмов:

$$a_M(x) = \sum_{m=1}^{M} b_m(x).$$

Пусть у нас имеется начальный базовый алгоритм $b_0(x)$. Он может быть самым простым, например, для любого профиля ВРСЗ пусть он говорит, что он относится к зимнему сезону. Тогда $b_0(x) = 0$. Допустим уже построили $M-1$ базовых алгоритмов:

$$a_{M-1}(x) = \sum_{m=1}^{M-1} b_m(x)$$

Ставится задача построить такой базовый алгоритм $b_M$, который будет минимизировать следующую функцию ошибки:

$$\sum_{i=1}^{N} L(y_i, a_{M-1}(x_i) + b_M(x_i)) \xrightarrow[b]{} \min$$

В случае задачи регрессии, функция $L$ будет, например, среднеквадратическим отклонением, а при задачи классификации обычно минимизируют функцию энтропии.

Можно минимизировать сразу данную функцию ошибки, либо разбить задачу на две части: минимизировать сначала более простую функцию:

$$F(s) = \sum_{i=1}^{N} L(y_i, a_{M-1}(x_i) + s_i) \xrightarrow[s]{} \min .$$

Отсюда:

$$s = -\vec{\nabla} F = (-L'(y_1, a_{M-1}(x_1)), ....., -L'(y_N, a_{M-1}(x_N))) .$$

Теперь вся информация про $L$ содержится в сдвигах $s$.

Можем обучить новый базовый алгоритм $b_m(x)$, как алгоритм регрессии на сдвигах $s_i$:

$$b_M(x) = \arg\min_b \frac{1}{N} \sum_{i=1}^{N} (b(x_i) - s_i)^2 .$$

Градиентный бустинг по сути осуществляет градиентный спуск в пространстве базовых алгоритмов.

Плюсами градиентного бустинга является высокая точность предсказаний, но к сожалению, в отличии от случайного леса он может переобучаться. Чаще всего в качестве базовых алгоритмов используют не глубокое решающее дерево. Интерпретация ответов алгоритмов градиентного

бустинга так же очень сложна. Для интерпретации ответов модели градиентного бустинга используются специальные программные пакеты. В данной работе использовался программный пакет shap values.

### 1.2.3 Интерпретация предсказаний модели

Очень важной задачей в сфере работы с алгоритмами машинного обучения помимо построения модели способной делать качественные предсказания -это интерпретировать результат предсказания такой модели. В случае классификации профилей ВРСЗ, важно не только знать к какому из сезонов алгоритм машинного обучения отнес данный профиль, но и почему он так сделал, ориентируясь на какие координаты скорости, какие величины каких координат характерны для зимнего сезона, а какие характерны для летнего.

Если с линейными моделями всё более-менее понятно, чем больше абсолютное значение коэффициента при координате скорости профиля ВРСЗ, тем данная координата важнее. Объяснить важность координат скорости для случайного леса и градиентного бустинга заметно сложнее.

В большинстве библиотек градиентного бустинга есть встроенные оценки важности фичей (координат ВРСЗ в данном случае):

- Gain – показывает относительный вклад каждой координаты в модель. Делаем проход по каждому дереву и смотрим какая координата в каждом узле приводит к его разбиению и на сколько снижается неопределенность модели согласно выбранной метрике. Для каждой координаты суммируется ее вклад по всем деревьям.
- Cover - показывает количество наблюдений для каждой координаты.
- Frequency – показывает, как часто данная кооралдината встречается в узлах дерева, то есть считается суммарное количество разбиений дерева на узлы для каждой координаты в каждом дереве.

Главная проблема таких подходов то, что непонятно как именно каждая координата влияет на ответ алгоритма. То-есть какие значения координат будут характерны для летнего/зимнего сезона. Для данного исследования на помощь приходит библиотека shap (по имени американского математика Шепли и названа библиотека).

Вектор Шепли — принцип оптимальности распределения выигрыша между игроками в задачах теории кооперативных игр. Представляет собой распределение, в котором выигрыш каждого игрока равен его среднему вкладу в благосостояние тотальной коалиции при определенном механизме её формирования.

Для оценки важности координат векторов ВРСЗ при классификации, происходит оценка предсказаний модели с и без данной координаты. В теории игр значения Шепли задают распределение выигрыша игроков в командной игре в соответствии с их вкладом. Соблюдается следующий принцип: при равном вкладе в игру двух игроков – каждый из них получает одинаковую награду, если игрок не сделал никакого вклада, такой игрок награду не получает. При проведении одним игроков двух и более игр – его награды суммируются.

Пусть каждая координата профиля ВРСЗ это игрок, а выигрыш - это итоговое предсказание модели. Всего игр столько, сколько профилей в выборке. Тогда формула для расчета значения Шепли для $i$-ой координаты:

$$\phi_i(p) = \sum_{S \subset N/\{i\}} \frac{|\mathrm{S}|!(\mathrm{n}-|\mathrm{S}|-1)!}{n!}(p(S \cup \{i\}) - p(\mathrm{S})), \text{ где}$$

$p(S \cup \{i\})$ - предсказание модели с использованием $i$-ой координаты скорости.

$p(\mathrm{S})$-предсказание модели без $i$-ой координаты скорости.

$n$ - размерность (количество координат в векторе).

$S$ - набор координат без $i$ координаты.

Значение Шэпли для $i$-ой координаты профиля ВРСЗ рассчитывается для каждого профиля из выборки по всем возможным комбинациям отсутствия $i$-ой координаты включая отсутствие всех координат. Полученные значения

суммируются по модулю для получения важности $i$-ой координаты профиля ВРСЗ. К сожалению такие операции очень затратны для больших объемов данных, поэтому используются различные алгоритмы оптимизации [7].

# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1 Сравнение алгоритмов сжатия с точки зрения MSE

Исследуем выше перечисленными алгоритмы машинного обучения. Возьмем выборку из 150 профилей ВРСЗ Баренцева моря. Сначала исследуем алгоритмы сжатия с точки зрения среднеквадратическое отклонения. Исходный профиль ВРСЗ для Баренцева моря имеет 12 координат скорости звука, т.е. его профиль лежит в $\mathbb{R}^{12}$.

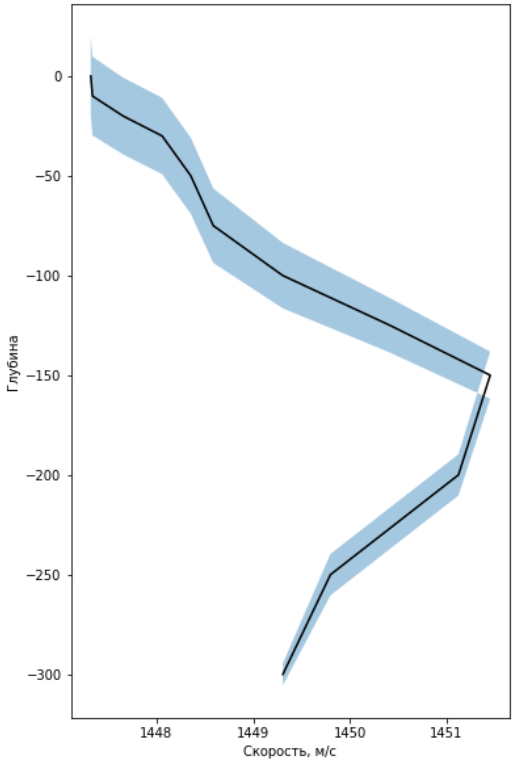


Рис. 2.1 Усредненный профиль ВРСЗ Баренцево море

На рис. 2.1 изображен усредненный профиль ВРСЗ и +- 3 стандартных отклонения. Разделим выборку на обучающую и тестовую. Возьмем 112 профилей для обучений и 38 отложим для проверки качества. Применив алгоритм сжатия PCA, получим набор базисных профилей (главных компонент).

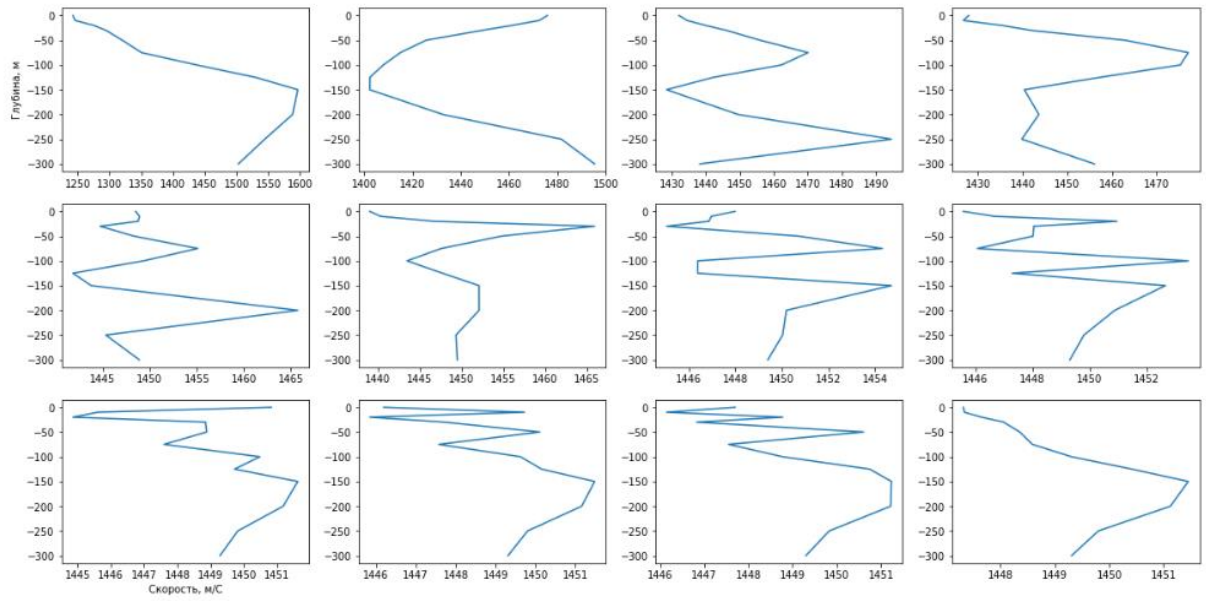


Рис. 2.2 Главные компоненты разложения профиля ВРСЗ для Баренцева моря

Рассмотрим разложение по 1, 3 и 5 компонентам. На рис. 2.3 изображен исходный профиль ВРСЗ из отложенной выборки, и он же разложенный по 1,3 и 5 компонентам.

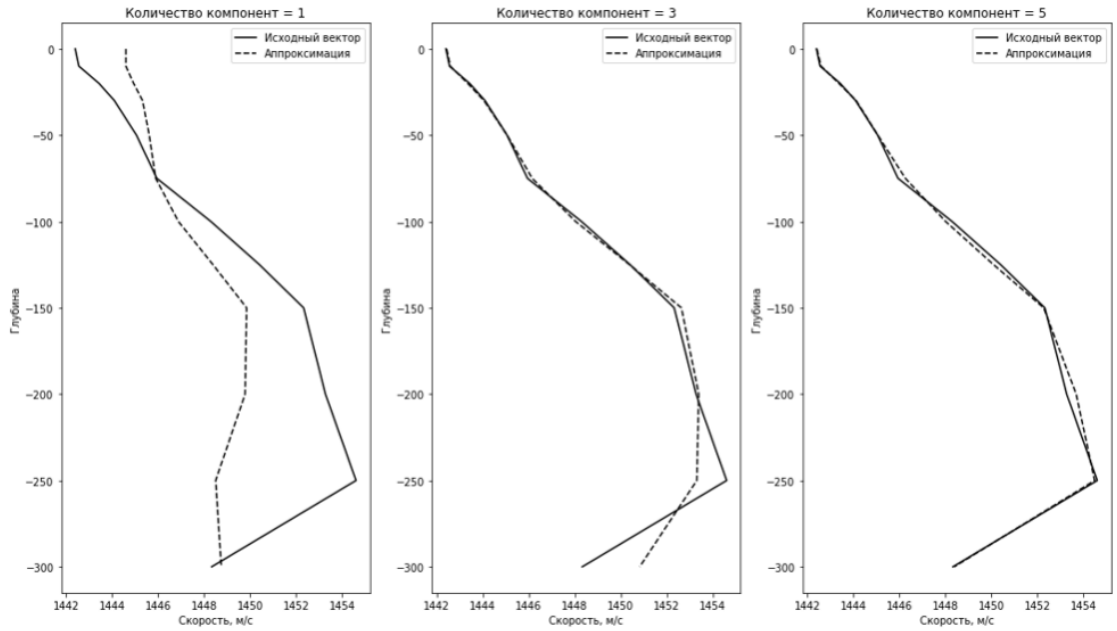


Рис. 2.3 Исходный и аппроксимированный профиль методом PCA

Сравнивая исходный профиль и аппроксимированный, видно, что приемлемое качества достигается уже при разложении по 5 главным компонентам.

Рассмотрим теперь алгоритм сжатия K-means.

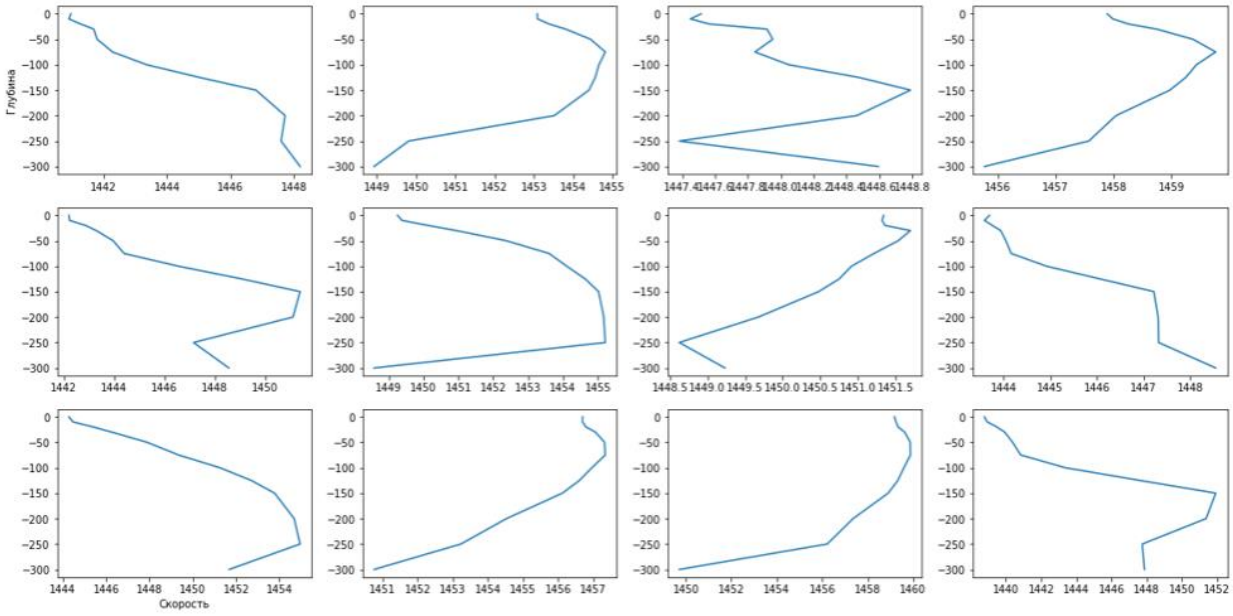


Рис. 2.4 Примеры центра кластеров

На рис. 2.4 изображены 12 профилей, являющихся центрами 12 кластеров.

Разобьем выборку на разное количество кластеров и попробуем аппроксимировать профиль ВРСЗ ближайшим центром кластера.

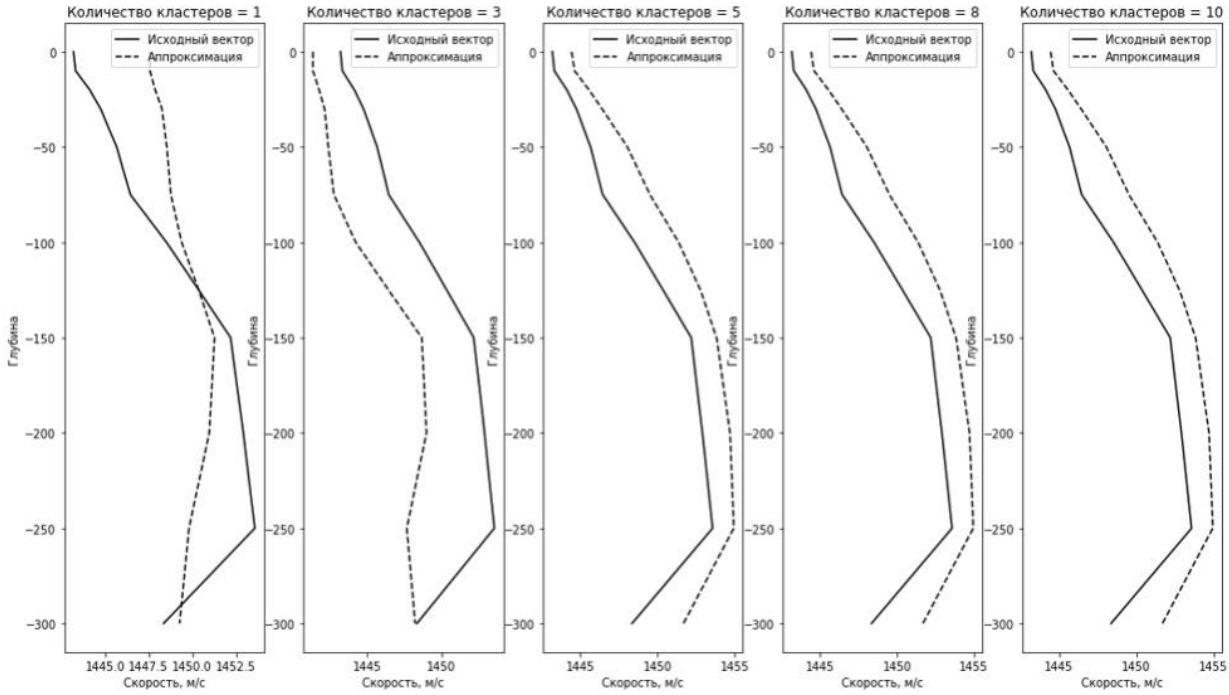


Рис. 2.5 Исходный и аппроксимированный профиль методом k-means

Как и ожидалось, данный алгоритм справился гораздо хуже, чем PCA.

Рассмотрим теперь метод K-SVD. Построим базисные профили для данного метода.

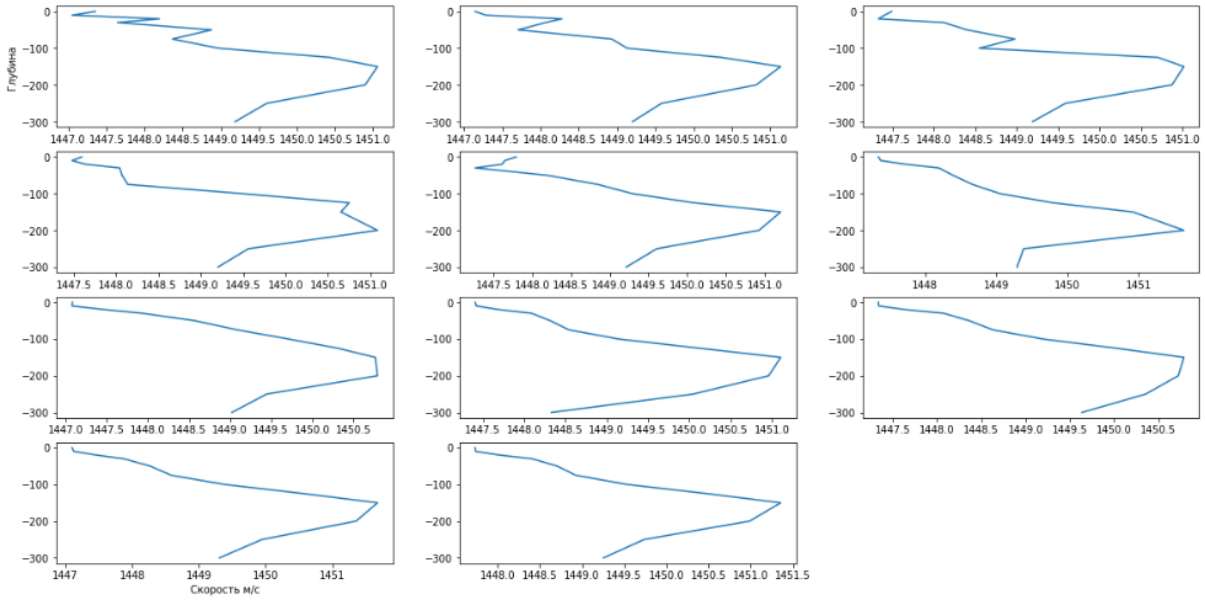


Рис. 2.6 Базисные профили для метода k-svd

Рассмотрим теперь построения исходного и аппроксимированного профиля ВРСЗ при разной степени разреженности кодирования.

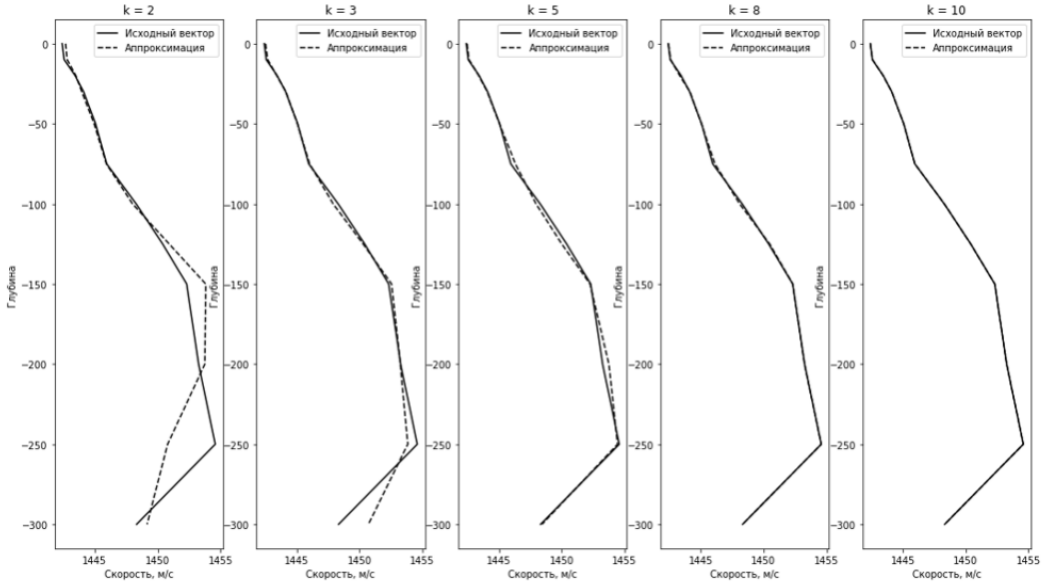


Рис. 2.7 Кодирование без нулевых компонент

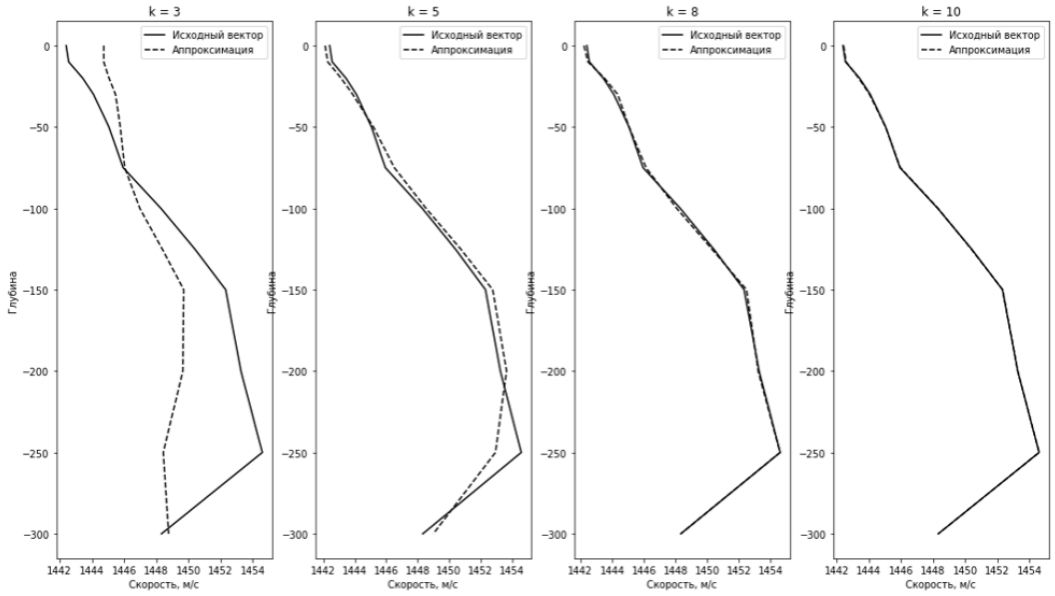


Рис. 2.8 Кодирование с двумя нулевыми компонентами

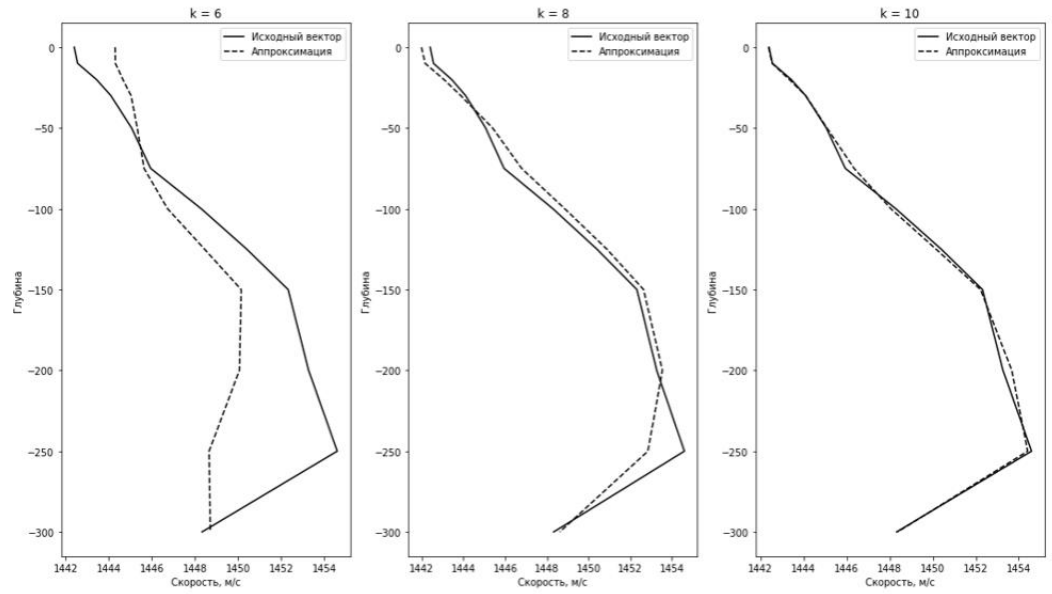


Рис. 2.9 Кодирование с 5 нулевыми компонентами

Данный алгоритм справился гораздо лучше, чем k-means, для сравнения с PCA далее будет приведен график среднеквадратичной ошибки (MSE).

Рассмотрим теперь архитектуру нейронной сети для сжатия размерности профиля ВРСЗ - автокодировщик (autoencoder). Воспользуемся библиотекой для нейронных сетей Keras. Экспериментами была выявлена наилучшая архитектура сети с одним скрытым слоем кодирования и декодирования.

Функция активации на скрытом слое - softplus, на слое кодирования – линейная функция активации.

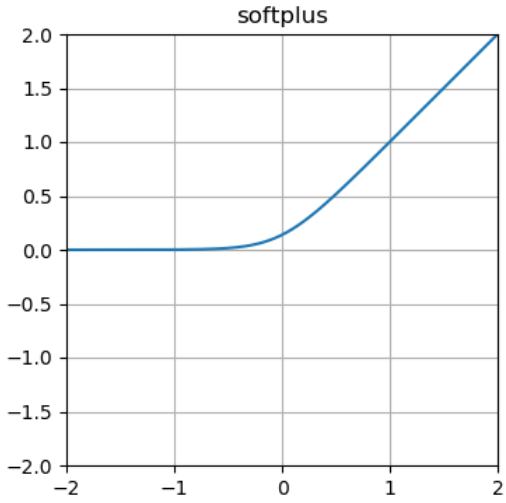


Рис 2.10 Функция активации softplus

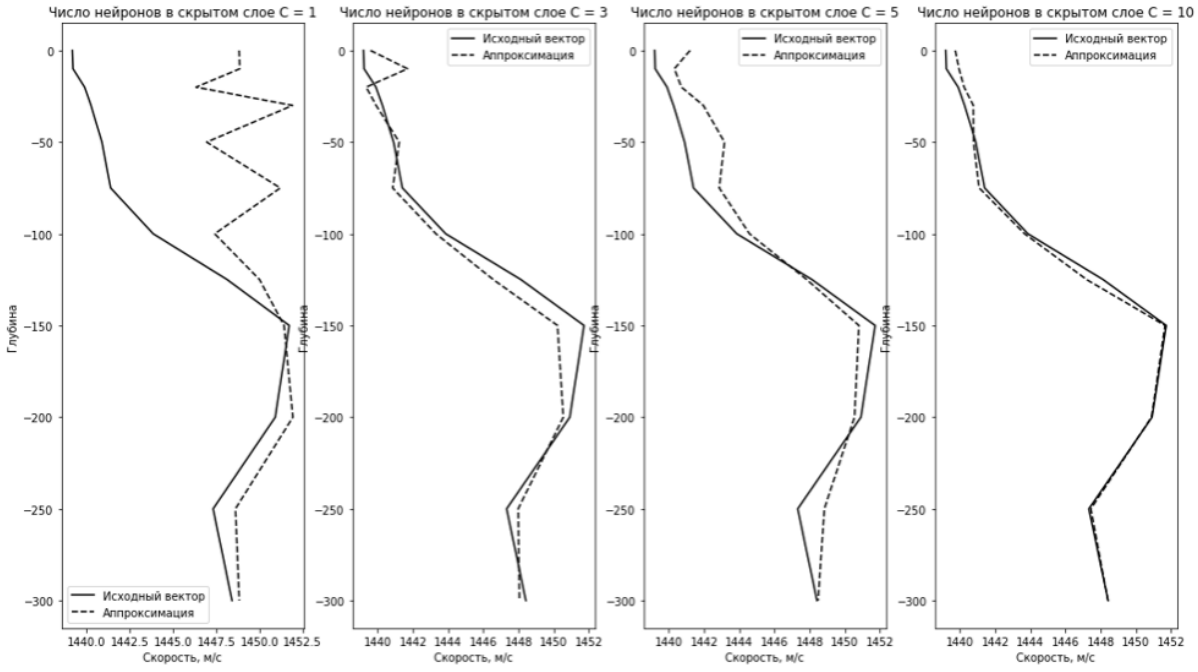


Рис. 2.11 Исходный и аппроксимированный профиль нейронной сетью.

Сравним теперь все 4 алгоритма сжатия по качеству среднеквадратичного отклонения.

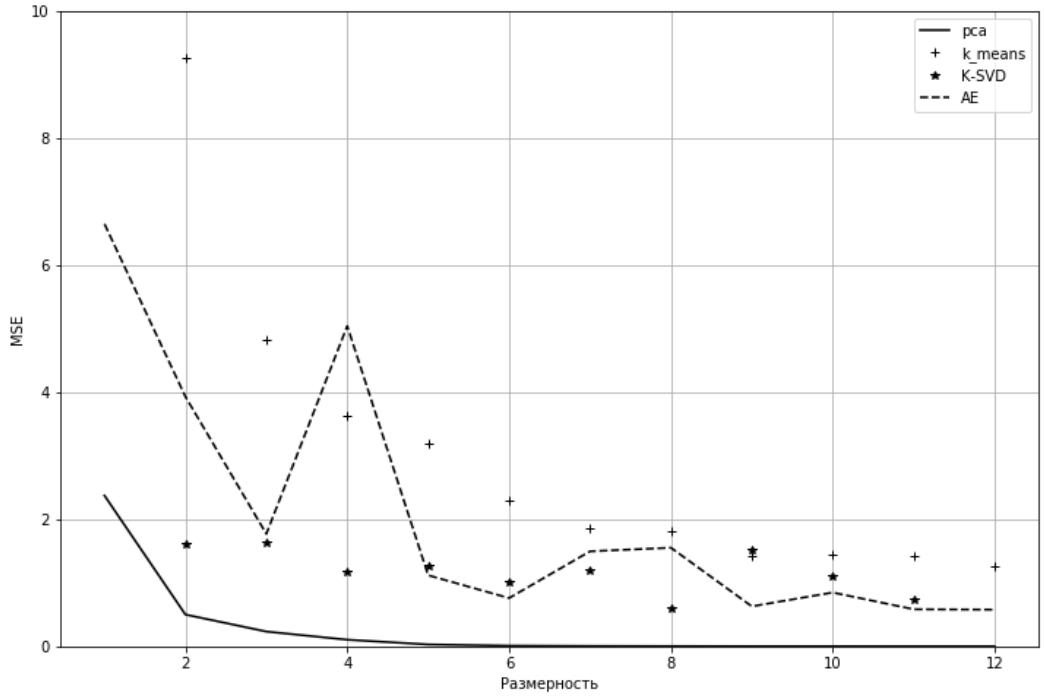


Рис. 2.12 Сравнение алгоритмов сжатия

Как видно из графика 2.10 зависимости среднеквадратического отклонения исходного профиля от аппроксимированного, наилучшим алгоритмом сжатbя с точки зрения MSE – является метод PCA. Сравним теперь методы сжатия с точки зрения сохранения информации о сезоне.

## 2.2 Сравнение алгоритмов сжатия с точки зрения сохранения информации о сезоне.

В разделе 2.1 данной работы, для сжатия профиля с качеством среднеквадратического отклонения, была взята выборка профилей ВРСЗ за 2-ой месяц (февраль). В данном разделе возьмем выборку для классификации на зимний (12, 1 и 2 месяцы) и летний (6, 7 и 8 месяцы) сезоны. А также для классификации по месяцам возьмем все 12 месяцев.

На рис. 2.11 Представлен график профиля ВРСЗ усредненный по выборке из зимних и летних профилей, а также +- 3 стандартных отклонения.

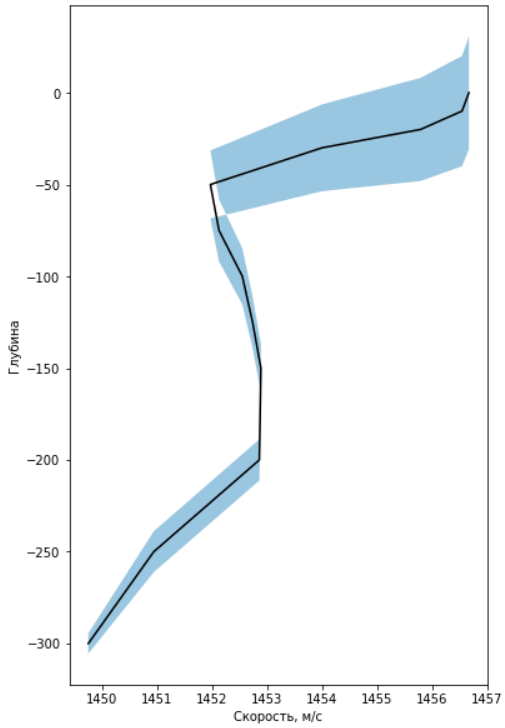


Рис. 2.13 усредненный профиль для летнего и зимнего сезона

Проверим классификатор логистическая регрессия. Исследуем сначал а классификацию без алгоритмов сжатия. Точность на отложенной выборке ( accuracy) составила 99.5%, что говорит об очень хорошей линейной разделим ости зимних и летних профилей. Исследуем теперь важность координат для п ринятия решения классификатора с помощью пакета shap.


Рис. 2.14 Важность глубин (shap values) для классификации алгоритмом логи стическая регрессия

Наибольшую важность для классификации профилей ВРСЗ на зимний/летний составили координаты на поверхности и на глубине -10 и - 30 метров. Причем для зимнего профиля на поверхности характерны меньшее значения скорости чем для верхнего, а на глубине -30 метров – наоборот. Сравним теперь алгори тмы сжатия с точки зрения качества классификации:

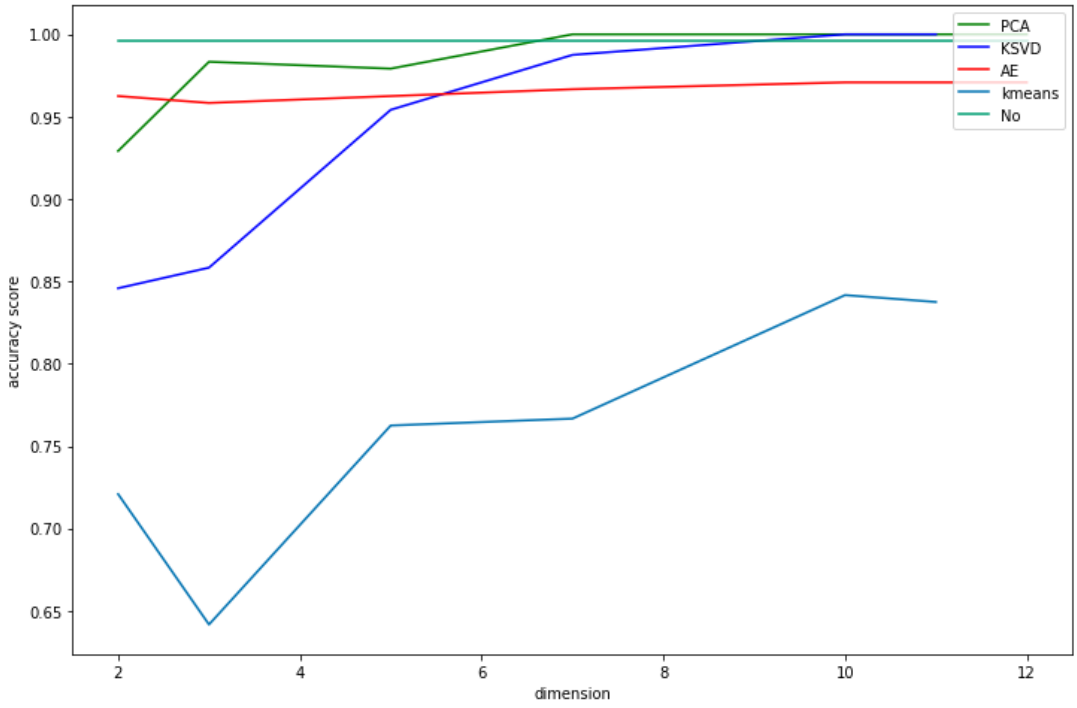


Рис. 2.15 Сравнение алгоритмов сжатия с точки зрения качества классификации алгоритмом линейная регрессия

При сжатии данных методом PCA, точность классификации уже при размерности сжатого вектора 7, превысила точность классификации на исходном векторе ВРСЗ. Исследуем влияние базисных компонент в методах сжатия PCA и K-SVD

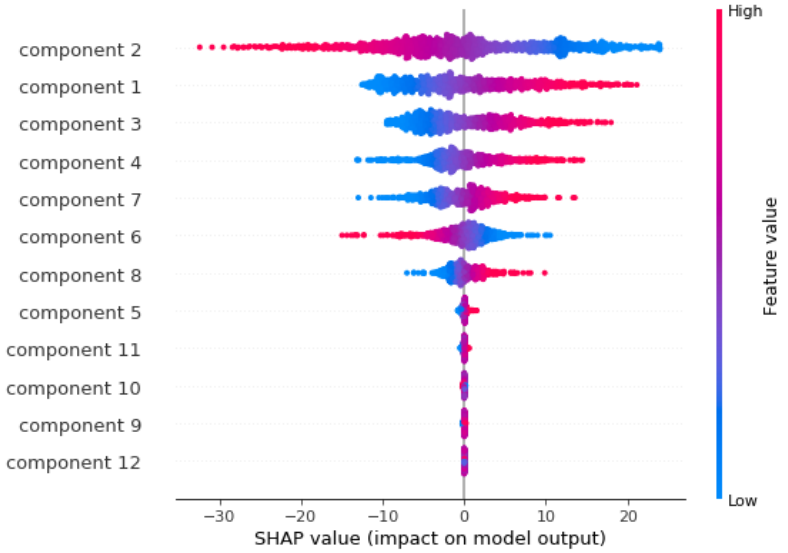


Рис. 2.16 Важность главных компонент (PCA) для классификации

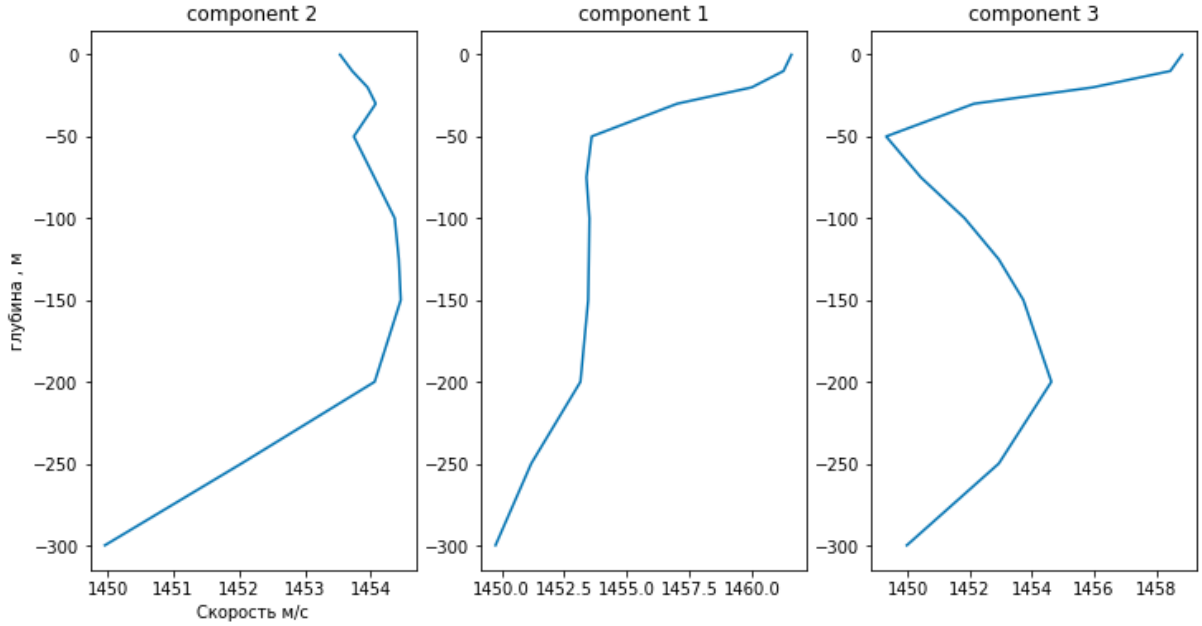


Рис. 2.17 Три наиболее важных главных компоненты PCA при классификации

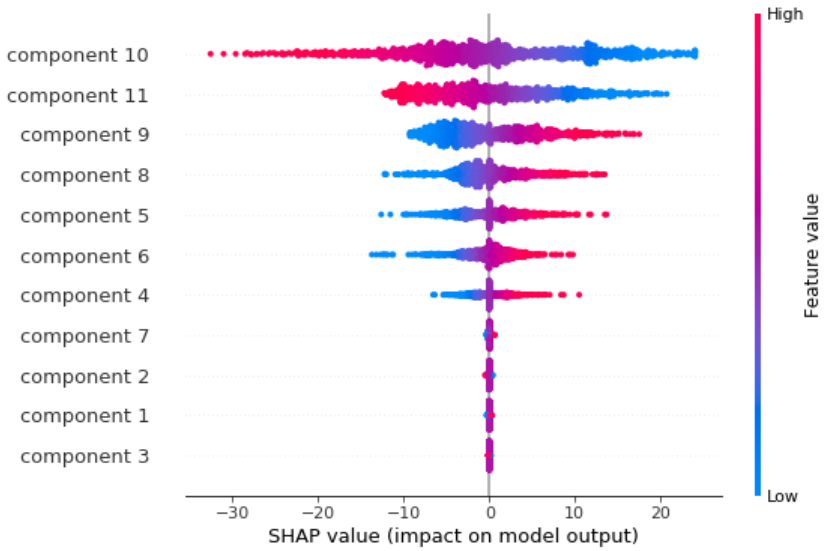


Рис. 2.18 Важность компонент разложения K-SVD

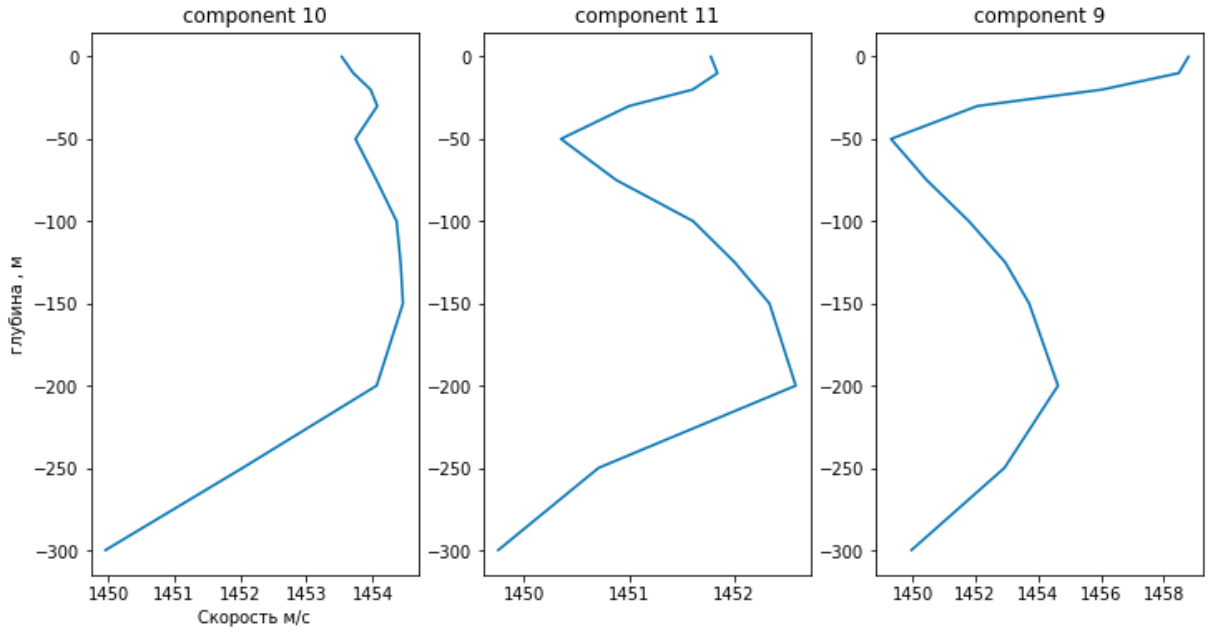


Рис. 2.19 Три наиболее важных профиля в алгоритме K-SVD для классификации

Как видно из графиков базисных компонент, хоть и нумерация у них разная, но профиль, который важнее всего для определения сезона имеет одинаковую форму в обоих методах.

Исследуем также алгоритмы классификации, основанные на деревьях решений – случайный лес и градиентный бустинг над решающими деревьями.

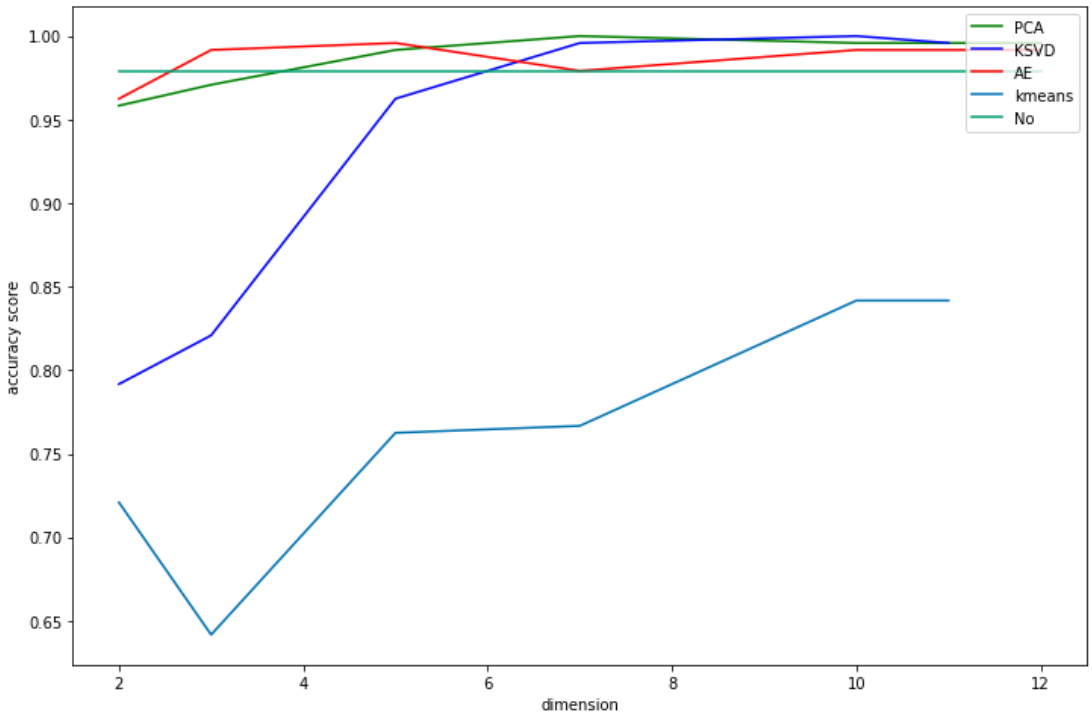


Рис. 2.20 Сравнение алгоритмов сжатия с точки зрения качества классификации алгоритмом случайный лес

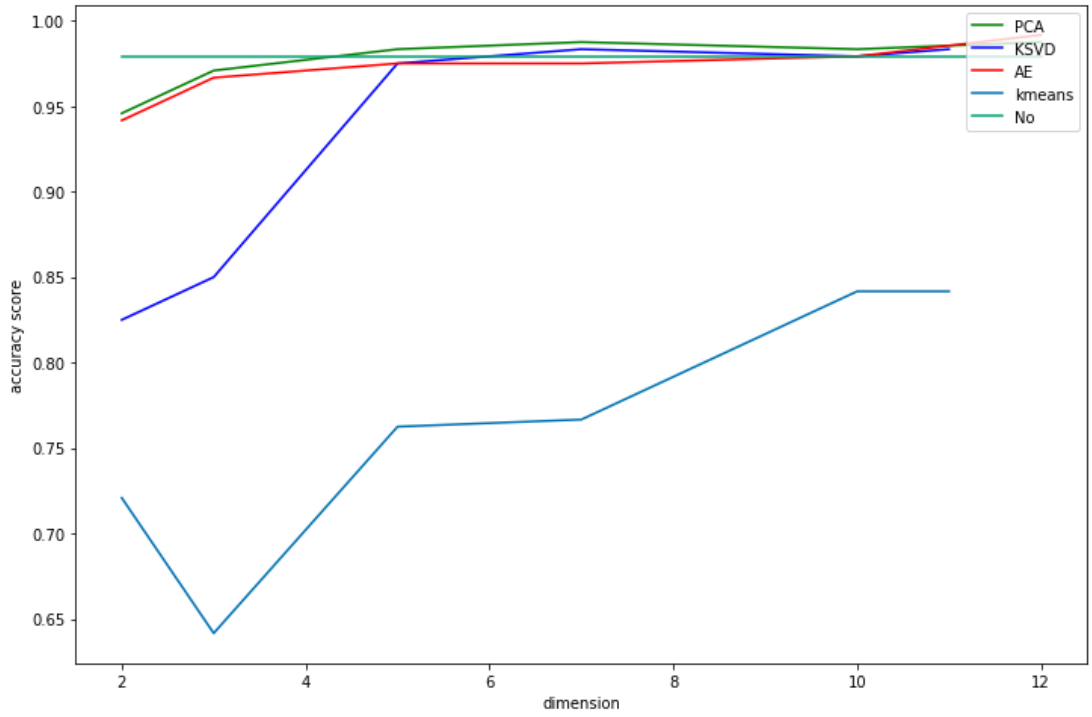


Рис. 2.21 Сравнение алгоритмов сжатия с точки зрения качества классификации алгоритмом градиентный бустинг

Из графиков точности классификации видно, что все алгоритмы сжатия данных, кроме k-means дают, начиная с размерности 7 точность лучшею, чем

классификация на исходных не сжатых данных. Это говорит о том, что данные зашумлены.

Применим к профилям ВРСЗ алгоритм визуализации umap, который переводит выборку данных из пространства высокой размерности на двумерную плоскость пытаясь сохранить соотношение расстояний между векторами. Данный алгоритм не участвовал в исследовании по средствам того, что может визуализировать только обучающую выборку и не обобщается на новые данные.

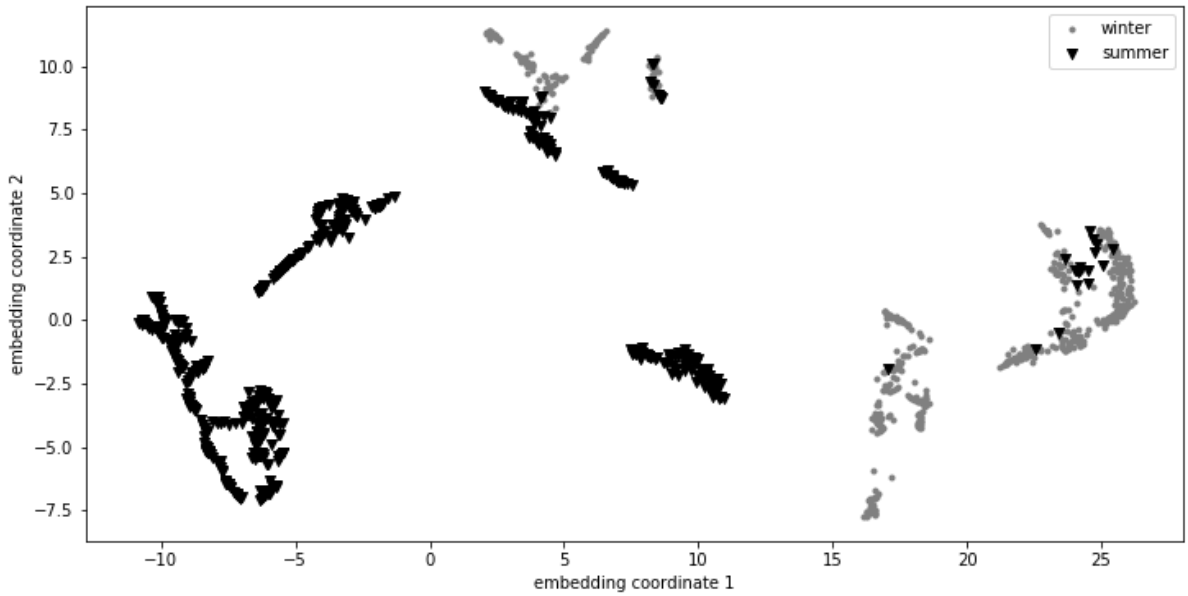


Рис. 2.22 Визуализация методом umap

Проведем аналогичные исследования для классификации по месяцам.

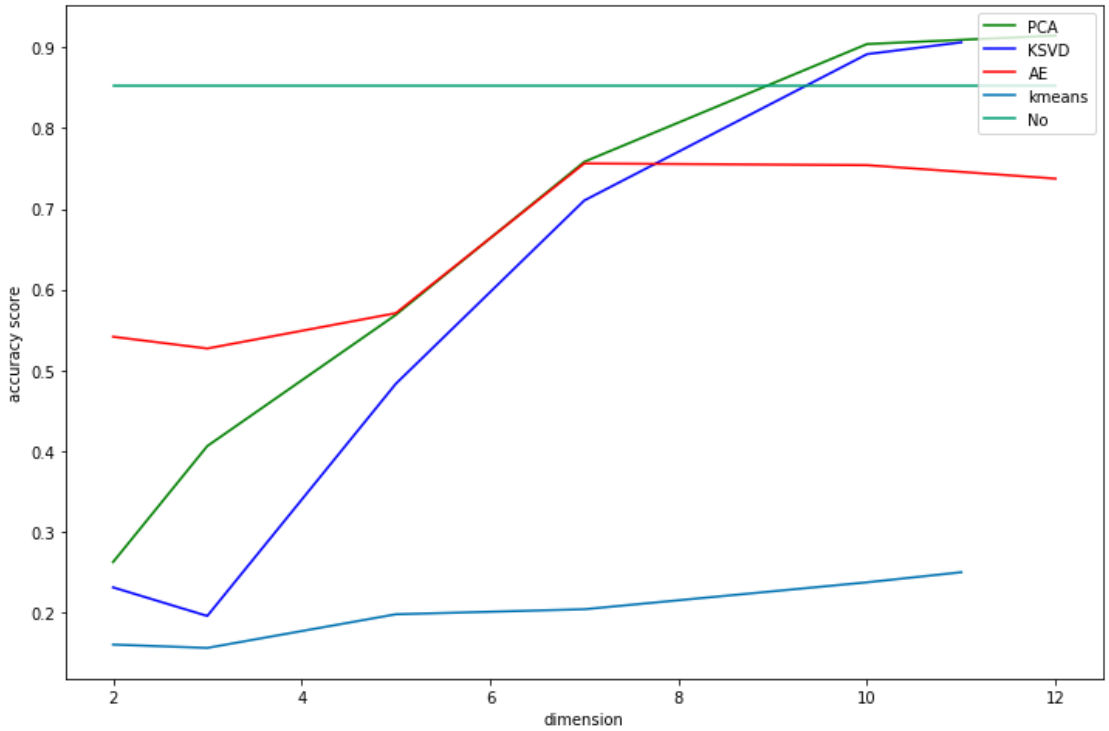


Рис. 2.23 Сравнение алгоритмов сжатия с точки зрения качества классификации по месяцам алгоритмом логистическая регрессия

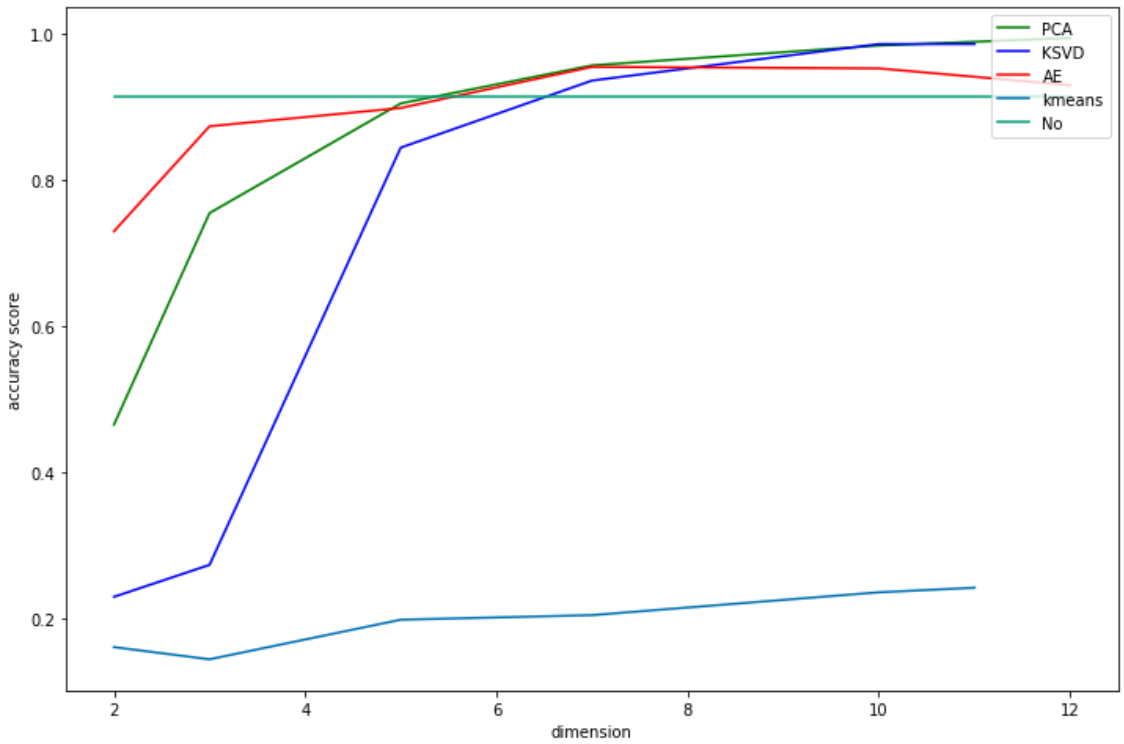


Рис. 2.24 Сравнение алгоритмов сжатия с точки зрения качества классификации по месяцам алгоритмом случайный лес

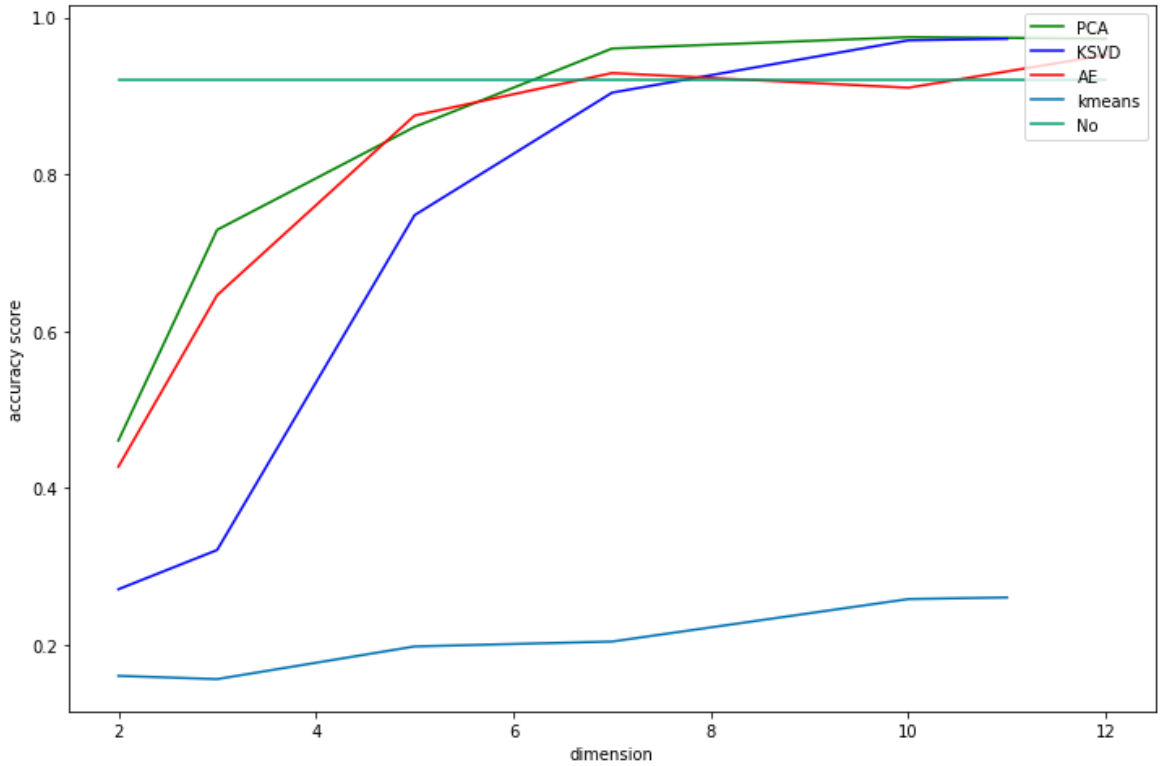


Рис. 2.25 Сравнение алгоритмов сжатия с точки зрения качества классификации по месяцам алгоритмом градиентный бустинг

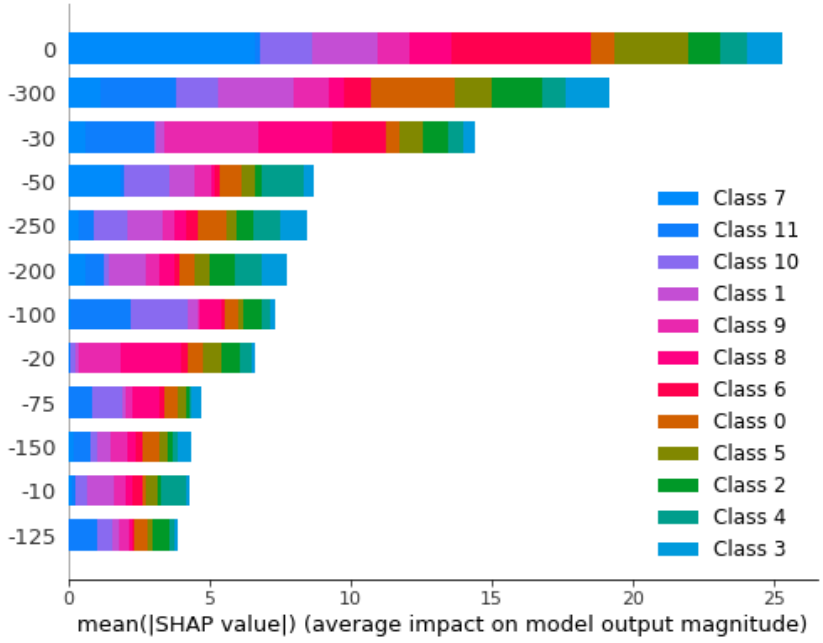


Рис. 2.26 Важность глубин (shap values) для алгоритма градиентный бустинг

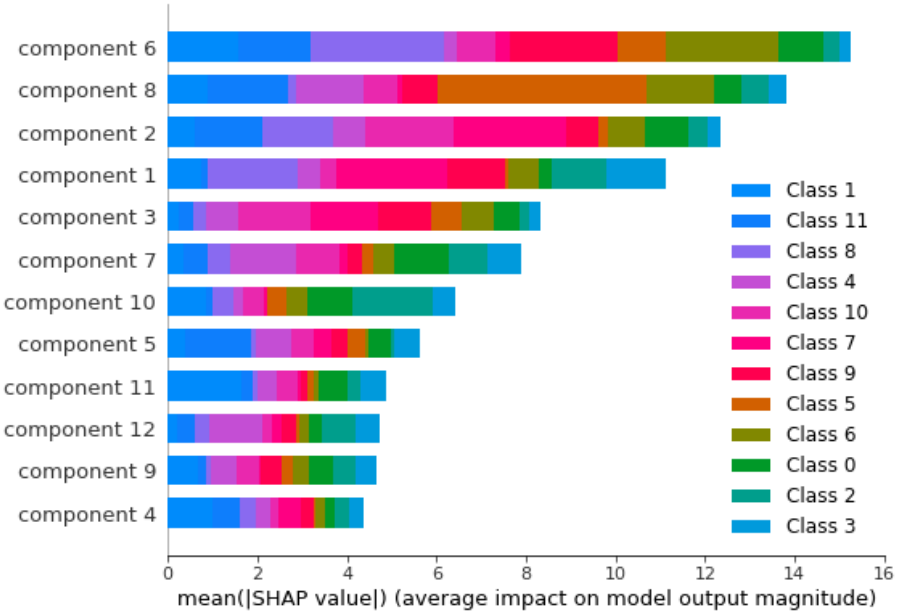


Рис. 2.27 Важность главных компонент (PCA) для классификации алгоритмом градиентный бустинг

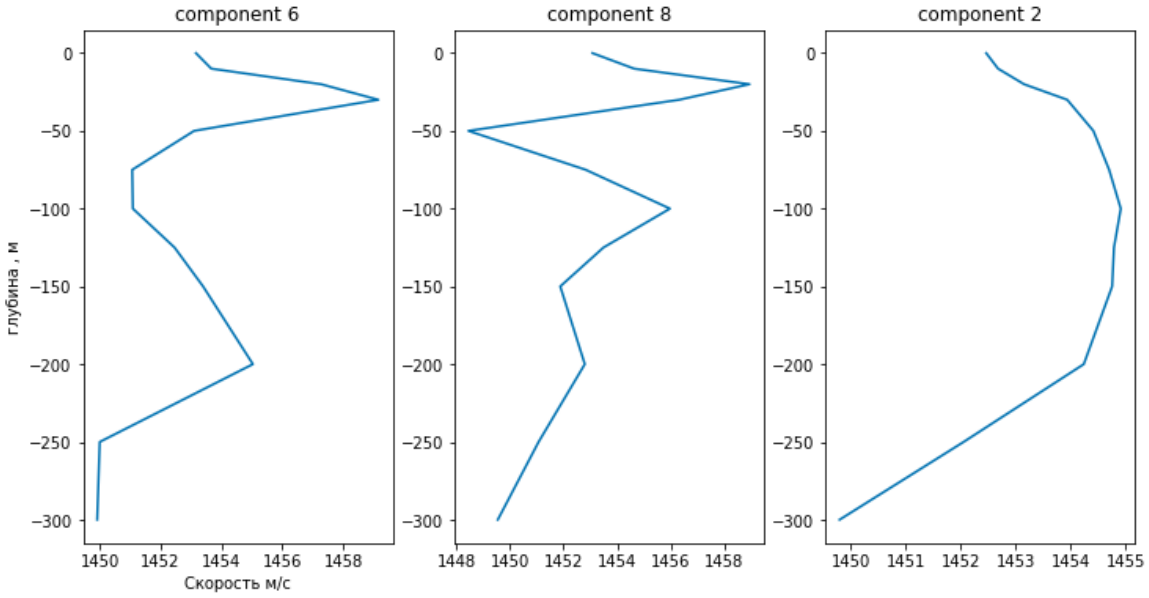


Рис. 2.28 Три наиболее важных базисных PCA при классификации

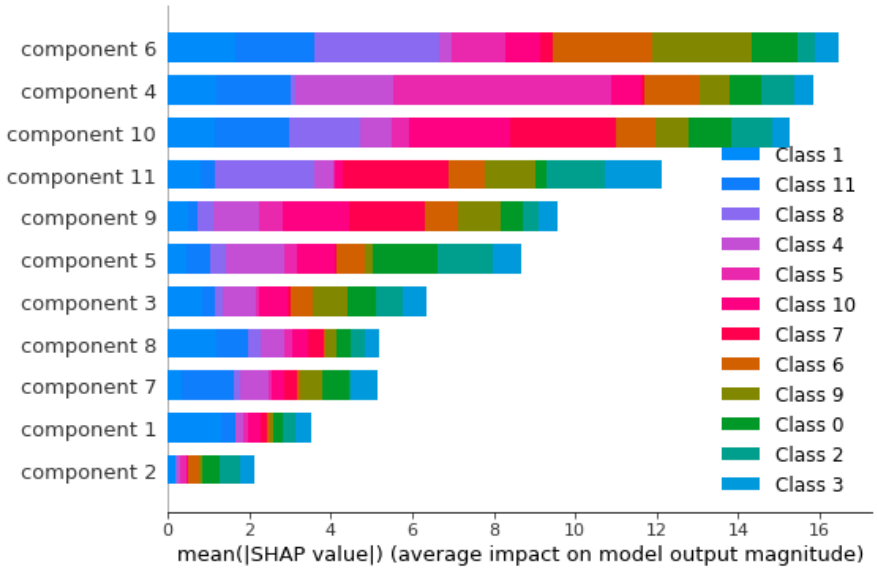


Рис. 2.28 Важность компонент разложения K-SVD для классификации градиентным бустингом

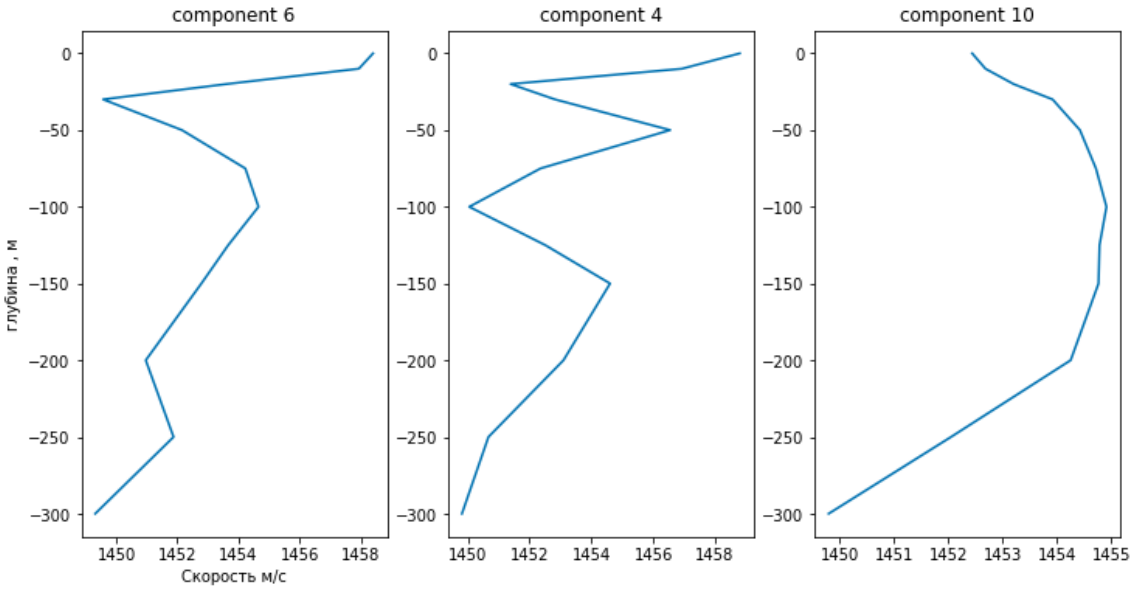


Рис. 2.30 Три наиболее важных базисных PCA при классификации

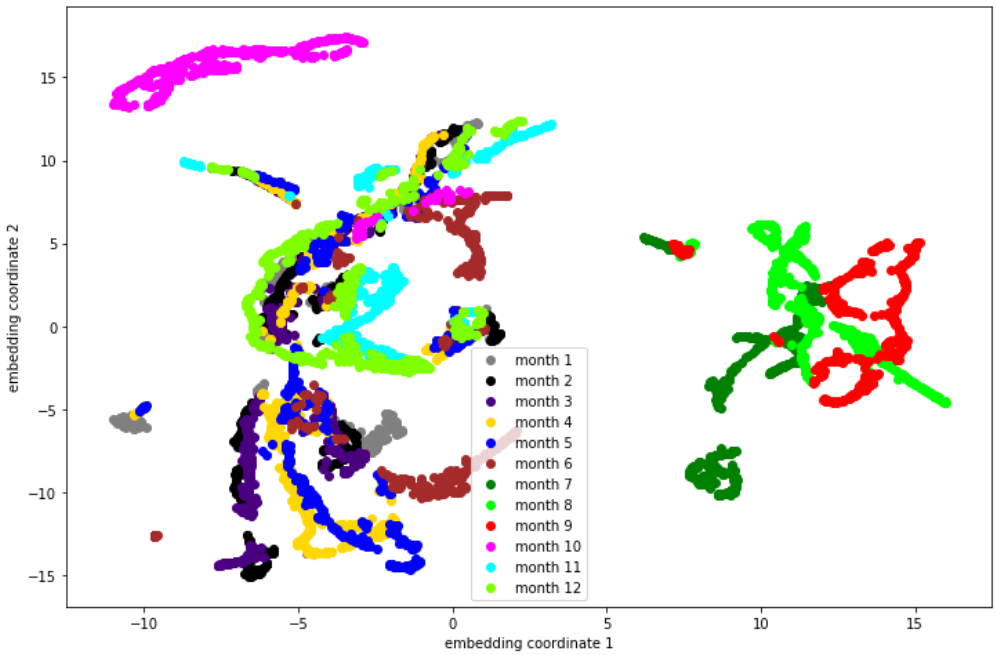


Рис. 2.31 Визуализация методом umap по месяцам

Из графиков точности классификации (2.21-2.23) видно, что точность многоклассовой классификации по месяцам снизилась по сравнению с классификацией по сезонам. Линейная модель логистической регрессии практически не пригодна для данной задачи (дает точность классификации accuracy - 0.85). Как и при классификации по сезонам, при классификации по месяцам алгоритмы сжатия размерности добавили точности классификации, особенно метод PCA. Алгоритм K-Means оказался самым слабым способом сжатия размерности в обоих задачах.

# ЗАКЛЮЧЕНИЕ

В работе проведены исследования профилей ВРСЗ Баренцева моря. Применены методы машинного обучения такие как сжатие размерности и классификация. Были сравнены между собой алгоритмы сжатия данных с оценкой качества среднеквадратического отклонения исходного профиля от аппроксимированного. Затем исследованы эти же методы с точки зрения сохранения информации о сезоне профиля. Выявлены наиболее важные координаты глубины для принятия решения о сезонности профиля и конкретного месяца. Исследованы базисные компоненты разложения профиля, выявлены какие из них несут наибольшую информацию о сезонности профиля ВРСЗ. Среди всех исследуемых алгоритмов сжатия, лучше всего подходят для данной задачи алгоритмы PCA и K-SVD. Алгоритм k-means по своей структуре сам является очень слабым методам сжатия и кластеризации, в данной работе он себя никак не проявил. Что касается нейронных сетей, алгоритм автокодировщика показал результат немного хуже по сравнению с PCA и K-SVD, это связано со слабой архитектурой и с относительно маленькой обучающей выборкой (всего 112 профилей для обучения).

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Андреева И.Б. Физические основы распространения скорости звука в океане. Гидрометеоиздат Ленинград 1975.
2. Cybenkot G. Approximation by Superpositions of a Sigmoidal Function. Math. Control Signals Systems (1989) 2:303-314.
3. Захаров В.О. Применение методов машинного обучения для хранения и анализа гидрологической информации. ВМСППС 2019.
4. Michal Aharon, Michael Elad, Alfred M. Bruckstein. K-SVD and its Non-Negative Variant for Dictionary Design. Department of Computer Science Technion—Israel Institute of Technology Technion City, Haifa 32000, Israel.
5. Chih-Wei Hsu, Chih-Jen Lin. A Comparison of Methods for Multi-class Support Vector Machines. Department of Computer Science and Information Engineering National Taiwan University Taipei 106, Taiwan.
6. Романенко А.В. Логистическая регрессия. Сибирский федеральный университет.
7. Scott M.L. Su-In Lee. A Unified Approach to Interpreting Model Predictions. Department of Genome Sciences University of Washington Seattle, WA 98105.
8. Michael Biancoa, Peter Gerstof. Dictionary learning of sound speed profiles. Scripps Institution of Oceanography, University of California San Diego, La Jolla, California 92093–0238, USA.
9. Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep learning. The MIT Press Cambridge, Massachusetts London, England.
10. Andreas Müller, Sarah Guido. Introduction to Machine Learning with Python. O'Reilly Media; 1 edition (October 21, 2016).
11. Захаров В.О. Некоторые методы снижения размерности для вертикального разреза скорости звука в океане // Моделирование и анализ данных 2019.

12. Захаров В.О. Аппроксимация вертикального распределения скорости звука в морских волноводах. Гагаринские чтения – 2018: XLIV.

# ПРИЛОЖЕНИЯ

Приложение 1

Программный код алгоритмов сжатия с метрикой MSE

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import StandardScaler
from keras.layers import Input, Dense, Dropout
from sklearn.metrics import mean_squared_error
from keras.models import Model
from mylibrary import plot_ssp, creatMas, my_cross_val_score
with open('{0}/CZ.txt'.format(ocean), 'r') as f:
    line = f.readlines()
CZ=creatMas(line)
with open('{0}/zs.txt'.format(ocean), 'r') as f:
    line = f.readlines()
    line = line[:-1]
zs = [-int(a) for a in line]
N_cord=len(zs)
def AEncoder(input_dim, latent_dim):
    hidden_layer = int(input_dim * 0.8)
    inputs = Input(shape=(input_dim,))
    inp_layar1 = Dense(hidden_layer,
activation='softplus',kernel_initializer='he_normal')(inputs)
```

```
    encoded = Dense(hidden_layer,activation='linear',
kernel_initializer='he_normal')(inp_layar1)
    dec_lay = Dense(latent_dim, activation='softplus',
kernel_initializer='he_normal')(encoded)
    #dec_lay = Dropout(0.1)(dec_lay)
    decoded = Dense(input_dim,activation='linear',
kernel_initializer='he_normal')(dec_lay)
    autoencoder = Model(inputs, decoded)
    encoder = Model(inputs, encoded)
    autoencoder = Model(inputs, decoded)
    autoencoder.compile(optimizer='adam', loss='mse')
    return autoencoder, encoder
kf = KFold(n_splits=5)
scaler = StandardScaler()
MSEmean = []
std_mean = []
Dim = list(range(1, N_cord+1))
for dim in Dim:
    autoencoder, encoder = AEncoder(N_cord, dim)
    MSE = []
    for train, test in kf.split(CZ):
        CZ_train, CZ_test = CZ[train], CZ[test]
        CZ_train_std = scaler.fit_transform(CZ_train)
```

```
        CZ_test_std = scaler.transform(CZ_test)
        autoencoder.fit(CZ_train_std, CZ_train_std, epochs=200,batch_size=64, shuffle=True, verbose=0)
        predict = autoencoder.predict(CZ_test_std)
        predict = scaler.inverse_transform(predict)
        mse = mean_squared_error(CZ_test, predict)
        MSE.append(mse)
    MSEmean.append(np.mean(MSE))
    std_mean.append(np.std(MSE))
MSEmean = np.array(MSEmean)
std_mean = np.array(std_mean)
CV, CV1, CV2 = [], [], []
X = range(1,N_cord+1)
for i in range(1,N_cord+1):
    CV.append(np.mean(my_cross_val_score(CZ, n_comp = i, model = 'pca')))
    CV1.append(np.mean(my_cross_val_score(CZ, n_comp = i, model = 'k_means')))
CV2 = [np.mean(my_cross_val_score(CZ, n_comp = i, model = 'k_svd')) for i in range(2,N_cord)]
```

Приложение 2

Программный код алгоритмов сжатия с метрикой качества классификации

```
def researcher_pca(CLF,X_train, Y_train, X_test, Y_test, n_comp=[3,5,7,10], **kwargs):
    scores = []
    for n in n_comp:
        dec = PCA(n_components=n)
        clf = CLF(**kwargs)
        pipe = Pipeline([('dec', dec), ('clf', clf)])
        pipe.fit(X_train, Y_train)
        predict = pipe.predict(X_test)
        score = accuracy_score(Y_test, predict)
        scores.append(score)
    return scores
def researcher_kmeans(CLF,X_train, Y_train, X_test, Y_test, n_comp=[3,5,7,10], **kwargs):
    scores = []
    for n in n_comp:
        kmeans = KMeans(n_clusters = n)
        kmeans.fit(X_train)
        test_red = np.zeros((len(X_test), n))
        train_red = np.zeros((len(X_train), n))
        centers_test=kmeans.predict(X_test)
```

```
        centers_train=kmeans.predict(X_train)
        for i in range(len(X_train)):
            train_red[i,centers_train[i]]=1
        for i in range(len(X_test)):
            test_red[i,centers_test[i]]=1
        clf = CLF(**kwargs)
        clf.fit(train_red, Y_train)
        predict = clf.predict(test_red)
        score = accuracy_score(Y_test, predict)
        scores.append(score)
    return scores
def researcher_ksvd(CLF, X_train, Y_train, X_test, Y_test, k=2,
n_comp=[3,5,7,10], **kwargs):
    scores = []
    for n in n_comp:
        ksvd = ApproximateKSVD(n_components=n,
transform_n_nonzero_coefs=max(n-k, 1))
        meantr = np.mean(X_train,axis=0)
        ksvd.fit(X_train - meantr).components_
        gamma_train = ksvd.transform(X_train - meantr)
        gamma_test = ksvd.transform(X_test - meantr)
        clf = CLF(**kwargs)
        clf.fit(gamma_train, Y_train)
```

```
        predict = clf.predict(gamma_test)
        score = accuracy_score(Y_test, predict)
        scores.append(score)
    return scores
def researcher_ae(CLF, X_train, Y_train, X_test, Y_test,
n_units=[3,5,7,10],epochs=750, **kwargs):
    scores = []
    scaler = StandardScaler()
    X_train_std = scaler.fit_transform(X_train)
    X_test_std = scaler.transform(X_test)
    N_cord = X_train.shape[1]
    for n in n_units:
        autoencoder_std, encoder_std = AEncoder(N_cord, n)
        autoencoder_std.fit(X_train_std, X_train_std,
                epochs=epochs,
                batch_size=64,
                shuffle=True, verbose=0)
        test_embedding = encoder_std.predict(X_test_std)
        train_embedding = encoder_std.predict(X_train_std)
        clf = CLF(**kwargs)
        clf.fit(train_embedding, Y_train)
        predict = clf.predict(test_embedding)
        score = accuracy_score(Y_test, predict)
```

```
        scores.append(score)
    return scores
score_pca = researcher_pca(LogisticRegression, X_train, y_train, X_test, y_test, C=20, n_comp=[2,3,5,7,10,12])
score_kmeans = researcher_kmeans(LogisticRegression, X_train.values, y_train, X_test.values, y_test,C=20, n_comp=[2,3,5,7,10,11])
score_ksvd = researcher_ksvd(LogisticRegression, X_train.values, y_train, X_test.values, y_test, C=20, n_comp=[2,3,5,7,10,11])
score_ae = researcher_ae(LogisticRegression, X_train.values, y_train, X_test.values, y_test, epochs=1200, n_units=[2,3,5,7,10,12], C=20)
plt.figure(figsize=(12,8))
plt.plot([2,3,5,7,10,12], score_pca, 'g', [2,3,5,7,10,11], score_ksvd, 'b',
        [2,3,5,7,10,12], score_ae, 'r', [2,3,5,7,10,11], score_kmeans, [2,12], [xgb_acc,xgb_acc])
plt.legend(('PCA', 'KSVD', 'AE', 'kmeans', 'No'),
          loc='upper right')
plt.xlabel('dimension')
plt.ylabel('accuracy score')
plt.show()
```

Код проекта и данные ВРС3 можно найти в репозитории GitHub:

https://github.com/WadimZakharov/decompos_SSP