Санкт-Петербургский государственный университет

ПОПКОВ Александр Александрович

Выпускная квалификационная работа

ПРИМЕНЕНИЕ МЕТОДА ЧАСТИЧНОГО ОБУЧЕНИЯ CO-TRAINING В ЗАДАЧЕ КЛАССИФИКАЦИИ КЛИЕНТСКОЙ БАЗЫ

Направление 38.03.01 «Экономика»

Основная образовательная программа бакалавриата «Экономика»

Профиль: математические и статистические методы в экономике

Научный руководитель: д.ф.- м.н.,
профессор ХОВАНОВ Николай Васильевич

Рецензент: начальник отдела систем
анализа и прогнозирования
ПАО «Газпром нефть»
ЧЕРНИЦЫН Иван Геннадьевич

Санкт-Петербург
2018

СОДЕРЖАНИЕ

Введение

Тенденции всеобщей цифровизации, развития Интернета вещей[1] и автоматизации большинства однотипных рутинных процессов в различных сферах социально-экономической деятельности привели к тому, что сегодня многие компании накапливают свои базы данных, собирая как можно больше статистики о собственном бизнесе. Кроме процесса глобальной автоматизации это также обусловлено тем, что в XXI веке умение эффективного управления внутренними и внешними данными на предприятии становится в один ряд приоритетных задач. Благодаря этому система поддержки принятия решений фирмы усложняется, стремясь максимально быстро выявить изменение паттернов и трендов потребительского поведения. В свою очередь для этих целей всё более расширяется применение моделей статистического обучения с целью всестороннего анализа данных.

С точки зрения системы поддержки принятия решений одной из распространенных задач на практике является разбиение объектов на классы (задача классификации). Однако в экономической сфере велика доля неопределённости. Это выражается тем, что, в первую очередь, в сфере бизнеса и экономики в целом массив имеющихся данных, используемых для моделирования, содержит лишь небольшую часть так называемой размеченной информации – данных, в которых присутствует некоторая целевая переменная. В первую очередь, это объясняется тем, что затраты на разметку всех данных являются слишком велики для компаний, работающих с большими объемами данных. Кроме того, моделируемых целевых показателей может просто не существовать вовсе, и они разрабатываются в процессе становления той или иной корпоративной технологии. Для подобных задач с недостатком размеченных данных помимо нейросетевых подходов сегодня развиваются методы «полуавтоматического обучения» или «частичного обучения»[2], например, анализ данных методов проводится в работах [24, 22, 31].

В 1998 году Avrim Blum и Tom Mitchel из Университета Карнеги – Меллон (Carnegie Mellon University) опубликовали работу под названием «Combining Labeled and Unlabeled Data with Co-Training», в которой описан эффективный метод кооперативного обучения моделей классификации в условиях дефицита размеченной информации и наличия неструктурированных данных. Изначально метод был применен для классификации веб-страниц в интернете.

[1] IoT: Internet of Things, концепция создания умных сетей между устройствами, бытовыми и промышленными предметами, [35]
[2] SSL: Semi-Supervised Learning tasks, [8]

Соответственно, в статье [14] была показана его работоспособность на подобной прикладной задаче.

В экономической же практике зачастую необходимо анализировать ситуацию «здесь и сейчас», когда существует значимый дефицит времени, за которое необходимо получать нетривиальные оценки текущего состояния среды бизнеса, его целевых покупателей и осуществлять управленческие решения в максимально короткие сроки. В таком случае на помощь бизнесу зачастую приходят методы прикладной статистики и анализа данных. В представленной работе исследуется возможность применения указанного метода кооперативного обучения в практической экономической задаче – оценке текущего поведения клиентов, а также выносятся предлагаемые модификации метода.

Таким образом, целью представленной работы является оценка потенциальной дополнительной выгоды от удержания нелояльных клиентов путем создания и применения прикладной программы моделирования и мониторинга степени лояльности клиентов методом частичного обучения Co-Training, которая может стать полезным инструментом в корпоративной рекомендательной системе.

Следовательно, объект исследования данной работы в рамках практической задачи – группа потребителей товаров и услуг на автозаправочных станциях.

Предметом исследования выступает потребительское поведение данных клиентов. Они могут потреблять как основной продукт автозаправочных станций – топливо разных марок, так и дополнительные опции в рамках обслуживания их автомобилей, а также сопутствующие товары и услуги – продукция общепита, бытовая химия и автомобильные товары.

Основная гипотеза, поставленная в начале изучения указанной проблематики: в условиях дефицита информации и наличия неструктурированных данных анализируемый в работе метод позволяет дать значимые оценки вероятностей поведенческой классификации клиентов автозаправочных станций с качеством предиктора выше случайного гадания, и, соответственно, измеримую потенциальную выгоду от повышения лояльности этих клиентов.

Реализация указанной цели потребовала решения следующих задач:

1. Изучить общепринятые постановки задач прикладной статистики.
2. Прояснить особенности решения каждого класса задач статистического обучения.
3. Выяснить текущую проработанность сферы задач SSL.
4. Раскрыть концепцию метода Co-Training в рамках задач SSL.

5. Разработать программу решения задач с помощью метода Co-Training.
6. Построить ряд моделей в рамках метода Co-Training для анализа потребительского поведения, определить из них наилучшую по качеству и выявить значимые паттерны в процессе моделирования.
7. Оценить конкретный экономический эффект от повышения степени лояльности клиентов.

Необходимость реализации указанных цели и задач обусловила структуру и логику работы. Представленная работа состоит из введения, трёх глав, списка использованных источников и приложения.

Во введении тема работы актуализируется с учетом текущих потребностей коммерческих организаций с целью максимально возможного извлечения полезной информации из истории поведения потребителей. Первая глава посвящена исследованию теоретических аспектов моделей частичного обучения без отрыва от других задач прикладной статистики, а также рассмотрению вопроса оценки качества моделирования. Во второй главе детально разбирается метод кооперативного обучения Co-Training, проводится его критический анализ, вносятся предложения по его модификации и представляются варианты метода для прикладного использования. В третьей главе исследуемый метод реализуется на широко распространённом объектно-ориентированном интерпретируемом языке программирования Python 3, ставится конкретная бизнес-задача и переводится в рамки задач статистического обучения, после чего анализируемая модель настраивается и применяется в формализованной задаче. Приложение содержит листинг программы, реализующей метод, рассматриваемый в данной работе, а также пример её работы с использованием удобного графического интерфейса пользователя.

Глава 1: Введение в модели частичного обучения

§1.1 Постановка задач обучения с учителем и без учителя

Традиционно принято различать 2 различных типа задач статистического обучения: обучение с учителем (supervised learning) и обучение без учителя (unsupervised learning) [9].

Рассмотрим общую постановку задачи обучения с учителем. Предположим, производится анализ N объектов. Каждый объект может быть описан рядом характеристик (средний доход, частота посещения торгового центра, модальный товар или услуга определенной категории и т.п.), которые принято называть признаковым описанием объектов. Виды признаков объектов бывают различными: они могут быть вещественными, бинарными, категориальными – принадлежащими некоторому неупорядоченному множеству, ординарными – принадлежащие упорядоченному множеству, а также принимать определенные подмножества значений из некоторого общего универсального множества. Кроме того, каждый объект имеет известную метку – в самом простом виде – принадлежность к той или иной целевой категории (например, посещает ли кинотеатр или нет, степень доверия для выдачи кредита и т.п.).

Тогда пусть $\{(x_i, y_i)\}, i := 1, \dots, n$– пары «объект-метка», $x_i \in \mathcal{X}$ – множество векторов с признаковым описанием объектов, $y_i \in \mathcal{Y}$ – метки объектов. Таким образом, можно рассмотреть $(n \times d)$-матрицу признакового описания объектов, где $d -$ количество признаков объектов. Предполагается, что пары (x_i, y_i) – н.о.р. – независимо одинаково распределены на $\mathcal{X} \times \mathcal{Y}$. Данное предположение является очень строгим, однако практика показывает, что предположение возможно опускать и получать значимые результаты моделирования.

Задача заключается в том, что необходимо найти некоторое отображение:

$$a: x \to y \qquad (1.1.1)$$

В случае, если $\mathcal{Y} = \mathcal{R}^d$, то рассматривается задача регрессии. Иначе, если метки объектов принимают дискретные множества значений, то ставится задача классификации. Для задач классификации ответы или классы могут быть формализованы как:

1) $Y = \{-1, +1\}$ – в случае наличия двух классов в данных;

2) $Y = \{1, \dots M\}$ – M непересекающихся классов;

3) $Y = \{0,1\}^M$ – на M классов с возможностью их пересечения;

Однако если множество признакового описания исследуемых объектов не содержит целевой переменной, то перед исследователем встаёт задача обучения без учителя. Тогда пусть $X = \{x_1, \dots, x_n\}$ – набор векторов в многомерном пространстве, которые также являются признаковым описанием объектов, где $x_i \in [n] \coloneqq \{1, \dots, n\}$. Как правило, делается предположение, что представленные объекты – н.о.р. из общего распределения $\mathcal{X}$. Также удобно определить матрицу $(n \times d)$, где по строкам расположено описание всех свойств каждой исследуемой единицы в имеющемся пространстве объектов. Цель обучения без учителя – выявить структуру данных, по природе которой возможно делать последующие выводы, извлекать аномалии. Примерами обучения без учителя могу служить кластерный анализ, оценка плотности распределения объектов и снижение размерности.

Оба типа представленных задач подчиняются общим методологическим принципам моделирования, которые могут быть выражены следующим образом:

1. Понимание конкретной проблемы;
2. Формальная постановка задачи;
3. Использование априорной информации;
4. Анализ имеющихся эмпирических данных для решения задачи;
5. Разработка и создание признакового описания объектов;
6. Создание обучающей и контрольной выборок;
7. Отбор показателей оценки качества модели;
8. Предварительная обработка эмпирических данных;
9. Выбор множества возможных моделей и их построение;
10. Оценивание качества моделей;
11. Извлечение выводов для разрешения поставленной проблемы;

На международном уровне указанные принципы в более сокращённом виде носят название CRISP-DM[3] или «Межотраслевой стандартный процесс для исследования данных» [10]. Далее все рассуждения будут касаться непосредственно задач классификации.

[3] CRISP-DM: Cross-Industry Standard Process for Data Mining

§1.2 Оценка качества предсказательных моделей

Построение любой модели (1.1.1) помимо отбора наиболее информативных значимых признаков сопровождается надлежащем контролем качества. Показатели качества традиционно определяют через так называемую функцию эмпирического риска [4] или функцию потерь, которая демонстрирует среднюю ошибку текущей модели по выборке данных. Чтобы выбрать наилучшую модель в смысле минимума эмпирического риска, который определяется как мера отклонения ответов модели от правильных ответов, необходимо найти экстремум функционала качества:

$$a = argminQ(a,\ X^m), a \in A \qquad (1.2.1)$$

Запись (1.2.1) означает, что для неизвестного закона распределения вероятностей $P(x)$ решение задачи (1.2.1) обусловлено выбором такого правила $a \in A$, которое могло бы доставить минимум среднего риска по эмпирическим данным (эмпирического риска, потерь) выбранному функционалу качества $Q(a,\ X^m)$, где средний риск по эмпирическим данным есть:

$$I(a) = \int Q(a,\ X^m)P(x)dx \qquad (1.2.1a)$$

Простейшие виды функции потерь – средняя абсолютная ошибка MAE и средняя квадратичная ошибка MSE:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|a(x_i) - y_i| \qquad (1.2.2)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(a(x_i) - y_i)^2 \qquad (1.2.3)$$

Кроме того, довольно распространенной оценкой качества является функция логарифмических потерь *Log Loss*:

$$logloss = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}log(p_{i,j}) \qquad (1.2.4.a)$$

где N – число наблюдений, M – число меток класса, log – в данном случае натуральный логарифм, $y_{i,j}$ = 1, если наблюдение i классе j и $y_{i,j}$ = 0 в противном случае, а $p_{i,j}$ является предсказанной вероятностью, что наблюдение i принадлежит классу j.

Для бинарной классификации (1.2.4.a) принимает вид:

$$logloss_{binary} = \frac{1}{N}\sum_{i=1}^{N}(\ y_i{}^{+}\log(p_i{}^{+})\ +\ (1 - y_i{}^{+})\log(1 - p_i{}^{+})\ ) \qquad (1.2.4b)$$

Идеальная модель будет иметь значение логарифмических потерь, равное нулю. Потери увеличиваются по мере отклонения предсказанных вероятностей от фактического класса объекта. Визуально кривая Log Loss выглядит следующим образом

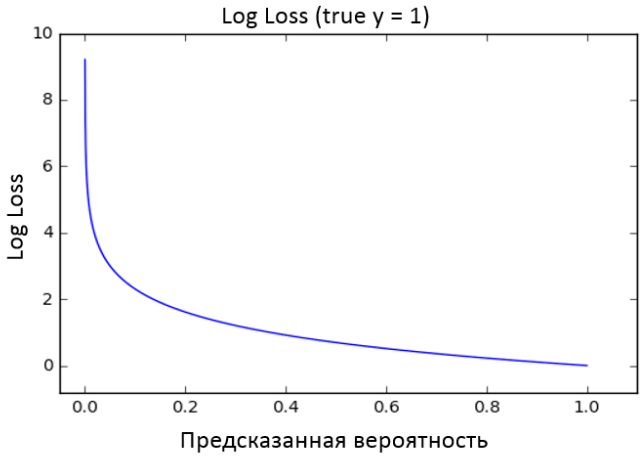

Рисунок 1: Поведение логарифма ошибки в зависимости от предсказанной вероятности модели

Рассчитано автором по: библиотека Python «scikit-learn».

На приведенном графике в условиях тестового примера показан диапазон возможных значений логарифма потерь при истинном наблюдении (true y=1). Если прогнозируемая вероятность приближается к 1, логарифмическая потеря медленно уменьшается, если прогнозируемая вероятность снижается, логарифмическая потеря быстро возрастает. Таким образом, логарифмические потери учитывают оба типа ошибок. Особенно учитываются те предсказания, которые являются уверенными по вероятности, но распознаны неверно, то есть чем больше вероятность принадлежности объекта к определенном классу при условии, что на самом деле объект находится в другом классе, тем существенней увеличивается показатель логарифмических потерь.

Одно из конструктивных объяснений данного критерия состоит в том, что минимизация логарифмической потери сводится к минимизации расхождения Кульбака – Лейблера[4] [8] между функцией, которая используется для моделирования, параметры которой необходимо оптимизировать, и истинной функцией, которая генерирует исследуемые данные, применяемые для обучения модели.

[4] Сокращённо - РКЛ.

Покажем, что функция *Log Loss* может быть строго определена через РКЛ:

] $p(x,y)$ – процесс, который генерирует данные, $D_{KL}(p \parallel a)$ – расстояние Кульбака - Лейблера a относительно p[5] . Необходимо подобрать такой параметр θ функции $a(y|x,\theta)$, чтобы $p(x,y), a(y|x,\theta)$ были похожи друг на друга в смысле РКЛ, тогда:

$$D_{KL}(p \parallel a) \to extr \qquad (1.2.5)$$

В общем случае: $D_{KL}(p \parallel a) = \int(x,y)\, p(x,y) \log\left[\frac{p(x,y)}{a(x,y)}\right] =$

$$= \int(x,y)\, p(x,y) \log[p(x,y)] - \int(x,y)\, p(x,y) \log[a(x,y)] \qquad (1.2.5a)$$

Т.к. a – дает оценки классификатора y, используя x, а не всё распределение x, то:

$a(x,y) = p(x)a(y|x)$, тогда из (1.2.5a):

$D_{KL}(p \parallel a)$ = $\int(x,y)\, p(x,y) \log[p(x,y)]$ -$\int(x,y)\, p(x,y) \log[p(x)]$ –

$$- \int(x,y)\, p(x,y) \log[a(y|x)] \qquad (1.2.5b)$$

Можно заметить, что 1 и 2 части выражения (1.2.5b) не зависят от функции a, параметр которой подбирается, тогда:

$\int(x,y)\, p(x,y) \log[p(x,y)]$ -$\int(x,y)\, p(x,y) \log[p(x)] = \mathrm{C} = const$

И из (2.2.5b): $D_{KL}(p \parallel a)$ = C - $\int(x,y)\, p(x,y) \log[a(y|x)]$

Следовательно, задача сводится к:

$$- \int(x,y)\, p(x,y) \log[a(y|x)] \to extr \qquad (1.2.5c)$$

В дискретном случае (1.2.5c):

$-\sum_i \log[a(y_i|x_i,\theta)\,] = -\sum_i p_i \log[a(x_i,\theta)]$, что аналогично (1.2.4.a) с точностью до обратного знака и нормировочного коэффициента в зависимости от числа наблюдений. Стоит отметить, что на практике Log Loss берётся также с обратным знаком, как было показано из (1.2.5). Таким образом, данный показатель учитывает неопределенность прогноза модели, исходя из того, насколько он отличается от фактической метки анализируемого объекта. Это дает более

[5] Общепринятого определения $D_{KL}(p \parallel a)$ нет, в данной работе РКЛ понимается указанным в тексте образом.

тонкий взгляд на качество модели в отличие от MAE и MSE. Кроме того, MAE и MSE также имеют недостатки:

- MAE – не дифференцируема в 0;
- MSE – имеет высокую чувствительность к выбросам;

Тем не менее, у описанных оценок есть общий существенный недостаток – они не позволяют судить о конструктивном качестве прогноза. Под конструктивным качеством прогноза в данном случае понимается факт того, насколько процентов мы далеки от действительности согласно валидационной или контрольной выборке по модели, с которой работаем. Более того, отсутствие теоретико-вероятностного распределения параметров также должно увеличивать скептицизм исследователя даже на контрольной выборке. Данное заключение очень важно для формирования задачи в рамках производственного предприятия, чтобы каждый сотрудник понимал, насколько тем или иным прогнозам можно доверять для осуществления управленческих решений. Проблема оценка качества результатов моделирования продолжает оставаться открытой.

Наиболее простым и объяснимым показателем качества является доля верно предсказанных значений по тестовой выборке данных:

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N}[a(x_i) = y_i] \qquad (1.2.6)$$

Однако данный показатель также не является надёжным, если данные не сбалансированы по меткам классов, что очень часто встречается на практике. Проиллюстрируем недостаток $Accuracy$ на примере:

Допустим, поставлена задача по созданию классификатора, оценивающего безопасность транзакций в банке. Службой по борьбе с мошенничеством в банке установлено, что небезопасные транзакции либо блокируются, либо поступают дополнительные запросы на обслуживание клиентов по телефону. Даны обучающая выборка с N транзакциями и тестовая выборка с 1000 примерами по транзакциям. Из тестовой выборки: 951 пример – безопасные транзакции, 49 примеров – выявленные случаи мошенничества. Тогда если классификатор обнаружил 10 фактов мошенничества, а остальные 990 отметил как безопасные, из которых, 39 неверно размеченные, то качество модели будет:

(951 + 10)/1000 =0,961 = 96,1%, что кажется превосходным результатом для модели. Тем не менее, модель не выявила и 50% мошеннических транзакций, что не позволяет оценивать её как эффективную.

Более информативный критерий качества модели базируется на так называемой матрице ошибок (confusion matrix) [20], которая учитывает соотношения четырёх типов предсказаний:

- Верно положительные (True positive, TP):

$$TP = \sum_{i=1}^{N} [a(x_i) = +1 | y_i = +1];$$

- Верно отрицательные (True negative, TN):

$$TN = \sum_{i=1}^{N} [a(x_i) = -1 | y_i = -1];$$

- Ложно положительные (False positive, FP):

$$FP = \sum_{i=1}^{N} [a(x_i) = +1 | y_i = -1];$$

- Ложно отрицательные (False negative, FN)

$$FN = \sum_{i=1}^{N} [a(x_i) = -1 | y_i = +1];$$

Схематично понятия индикаторов TP, FP, TN, FN могут быть проиллюстрированы следующим примером:

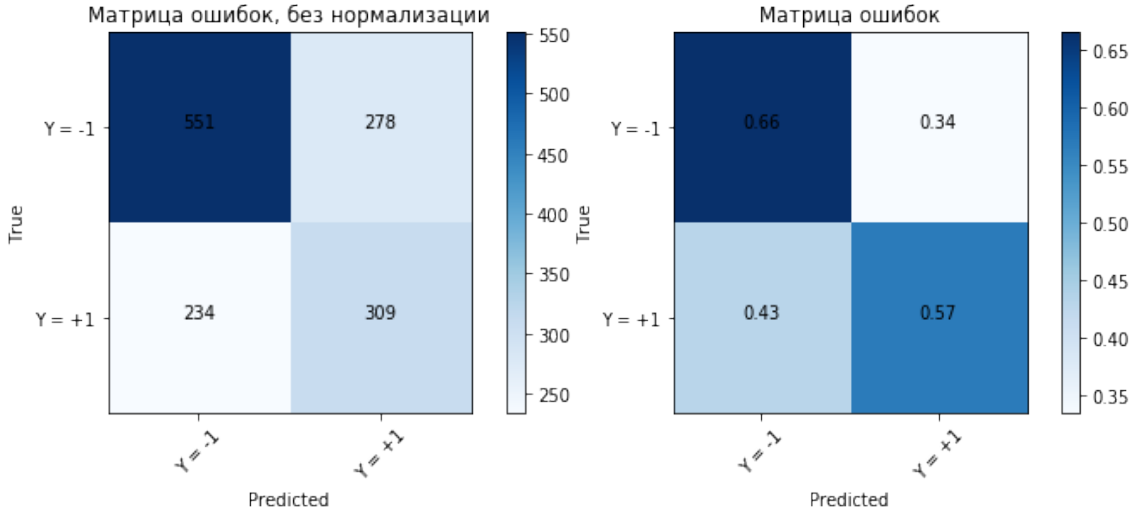

Рисунок 2: Матрица ошибок модели в абсолютных и относительных значениях

Рассчитано автором по: библиотеки Python «scikit-learn», «matplotlib».

На основе представленной матрицы ошибок возможно вычисление следующих полезных соотношений для анализа представляемой информации:

Точность, Precision – доля правильно положительно классифицированных объектов среди всех положительно классифицированных объектов на основе предиктора:

$$Precision = \frac{TP}{TP+FP} \quad (1.2.7)$$

Полнота, Recall – доля правильно положительно классифицированных объектов среди всех положительны объектов выборки:

$$Recall = \frac{TP}{TP+FN} \quad (1.2.8)$$

Очевидно, что в процессе моделирования необходимо учитывать оба представленных показателя. Общепринятым решением проблемы агрегации данных показателей является взвешенный индикатор между (1.2.7) и (1.2.8), в литературе [8, 20] называемой F-мерой (F-score, F-measure):

$$F - score = \beta \frac{Precision*Recall}{Precision+Recall} \quad (1.2.9)$$

Коэффициент β в представленном индикаторе качества может быть адаптирован под решение той или иной специфической задачи, где управленческим структурам важнее либо иметь высокую точность получаемых прогнозных данных, либо их максимально возможную полноту. Канонической формой (1.2.9) является среднее геометрическое, при β = 2. Смысл использования среднего геометрического основан на том, что данный подход учитывает возможные ситуации, когда один из составляющих показателей достаточно велик, а другой близок к нулю. Тогда F-мера в целом также будет приближаться к нулю. Данный показатель является одним из самых распространённых в задачах классификации. Тем не менее, как уже было отмечено, основной его недостаток – неоднозначность выбора β, влияющего на общий результат оценки работы модели.

Таким образом, в решении задач типа обучения с учителем существует неоднозначность оценки моделей, подбираемых для решения поставленных проблем. Во многом выбор индикатора качества или набора индикаторов может зависеть от баланса между сложностью и точностью предоставляемого результата, а также от конкретных особенностей разрешаемого вопроса. В силу огромного числа показателей качества, предлагаемых в рамках построения моделей, в данной работе анализ базируется преимущественно на показателях (1.2.7 – 1.2.9).

§1.3 Введение в задачи частичного обучения

Наряду с представленными типами задач обучения с привлечением учителем и без него особняком стоит тип задач с частичным привлечением учителя или частичного обучения[6]. В этих задачах изначально в дополнение к данным без целевой переменной исследователь получает некоторую информацию о метках возможных классов объектов, но не обязательно обо всех объектах. Зачастую дополнительная информация представляет собой небольшое число известных меток классов объектов.

Общий вид задачи выглядит следующим образом:

$X = (x_i), i = 1{:}\, n -$ имеющиеся данные по анализируемым объектам в виде $(n \times d)$ матрицы признакового описания объектов. X может быть разделён на $X_l \coloneqq (x_1, \dots, x_l) -$ н.о.р., размеченные данные, имеющие целевую переменную Y_l, другая часть данных $X_u \coloneqq (x_{l+1}, \dots, x_{l+u}) -$ н.о.р., не имеет меток класса. Необходимо построить максимально качественное отображение, используя X_u. Построение отображения основывается на минимизации функции эмпирического риска (1.1.1), подбираемой исследователем. В данной интерпретации задачи частичного обучения находятся ближе к типу задач обучения с учителем. Условно в задачах частичного обучения методы могут разделяться на два основных типа – непосредственно классификация и оценка плотностей и разделение предполагаемых смесей объектов.

Проблема, связанная с современным пониманием задач частичного обучения, была поставлена советским математиком и одним из создателей статистической теории восстановления зависимостей по эмпирическим данным В. Н. Вапником в виде понятия «трансдуктивного обучения» [4]. В его постановке даются наборы так называемых размеченных и неразмеченных данных. Идея трансдуктивного обучения состоит в том, чтобы осуществить предсказания только по неразмеченному набору данных. Теория восстановления зависимостей не лишена критики. В частности, в [5] К.В. Воронцовым приводится конструктивная критика теории, а именно отсутствие принятия во внимание частных особенностей распределения объектов в пространстве, восстанавливаемой зависимости (специфических свойств), а также метода обучения. В контрасте с трансдуктивным обучением выступает так называемое «индуктивное обучение», где главной целью является построение функции, определенной на всём возможном пространстве объектов $\mathcal{X}$. Необходимо отметить,

[6] Semi – Supervised Learning, SSL

что в данной работе разделяются понятия трансдуктивного обучения и стандартной задачи SSL по природе близкой к обучению с использованием прецедентов.

Принято считать [9], что самые ранние идеи использования неразмеченной и неструктурированной информации в задачах классификации связаны с понятием «самообучения» (self-learning, self-training). Данный метод частичного обучения является методом-обёрткой, который использует метод решения задач обучения с учителем определенное число итераций. Вначале подбирается оптимальное решающее правило (1.1.1) на данных с известными целевыми переменными X_l. На каждой итерации алгоритм осуществляет предсказание классов на неразмеченных данных и берет часть предсказаний в качестве истинных данных для последующей подгонки модели на новой выборке. Идею можно видеть в таких работах как [13, 28]. Естественно, работоспособность данного метода только немного зависит от самой «обёртки»[7], совместный эффект с оберточным алгоритмом происходит только в результате встроенной модели обучения с учителем или обучения по прецедентам [9].

Первоначальный виток развития методов частичного обучения приходится на время введения в статистический анализ линейного дискриминанта Р. Фишером в 1936 г. и канонического дискриминантного анализа соответственно. Однако существенный интерес к задачам обучения с частичным привлечением учителя возрос только к концу XX века, в 1990-х гг. с распространением Интернета. Большинство первоначальных задач применения этих методов состояло в естественной обработке языка и классификации информации в форме текстов [9]. Одна из классических работ в этой сфере и принадлежит группе ранее упомянутых учёных Avrim Blum и Tom Mitchel [14].

Любое моделирование предполагает под собой наличие ряда предположений и допущений, которые позволяют упростить генерируемый процесс и получить значимые практические результаты. Тип задач SSL также базируется на ряде подобных предположений. Обобщенно данные предположения говорят, что информация о $p(x)$ – плотности распределения объектов, значения целевой функции которых неизвестны, должна быть релевантной и влиять на условную плотность $p(y|x)$. Основные предположения об имеющихся данных раскрываются в следующем параграфе.

[7] От англ. «Wrapper methods», Wrap – «заворачивать, оборачивать»

§1.4 Основные предположения в задачах частичного обучения

Набор следующих предположений лежит в основе построения любой модели частичного обучения. Без представленных ниже предположений невозможно провести анализ результатов моделирования и дать адекватное объяснение работоспособности или неработоспособности выбранной исследователем модели [9].

1. Предположение гладкости:

Если изучаемые объекты: x_1, x_2 находятся относительно близко с точки зрения одной из мер в области высокой плотности данных, т.е. принадлежат одному кластеру с точки зрения решения задачи кластеризации, тогда, вероятно, их целевые переменные будут идентичны или близки друг к другу. С другой стороны, если x_1, x_2 находятся на относительно большом расстоянии друг от друга в месте низкой плотности, их целевые переменные не обязательно будут аналогичны или близки. Предположение гладкости относится к задачам и регрессии, и классификации.

2. Предположение кластеризованности:

Каждый представленный класс в данных стремится сформировать единый кластер в признаковом пространстве объектов. Тогда неразмеченная информация может помочь в поиске более точной разделяющей границы между несколькими кластерами. Таким образом, если точки признакового пространства находятся в одном кластере, с большой вероятностью они могут иметь одинаковые классовые метки. Необходимо понимать, что данное предположение не обязательно подразумевает наличие компактных множеств-кластеров в признаковом пространстве.

С точки зрения понятия плотности предположение кластеризованности говорит, что разделяющая классы граница должна быть в области низкой плотности.

Предположение кластеризованности может быть также представлено как частный случай предположения гладкости, учитывая, что кластеры, часто определяемые как множество скопленных в пространстве точек, которые имеют между собой небольшое расстояние, что может быть только в областях с высокой плотностью.

3. Предположение об избыточности данных:

Связанное с предыдущими, предположение о избыточности признаков позволяет формировать целый ряд методов в задачах с частичным привлечением учителя. Суть указанного предположения состоит в том, что данные высокой размерности в смысле большого числа признаков исследуемых объектов могут быть переведены в множество с более низкой размерностью без существенной потери качества.

Известно, что статистические методы и алгоритмы обучения имеют общую проблему – так называемое «проклятие размерности» [23]. Данное понятие связано с тем фактом, что решение таких статистических задач, как оценка плотности зависит от числа измерений и количества примеров для обучения по экспоненциальному закону. Кроме этого возникает связанные проблема наличия взаимных корреляций признаков изучаемых объектов. Однако в связанном пространстве более низкой размерности эту проблему возможно обходить, например, применяя метод главных компонент PCA. В целом метод главных компонент позволяет образовать новое связанное признаковое пространство с диагональной ковариационной матрицей, следовательно, без возможных корреляций между признаками.

Отмеченные предположения о гладкости, кластеризованности и многообразии данных необходимо учитывать, чтобы построить наиболее адекватное отображение с максимально возможным качеством для того или иного типа задачи – классификации или регрессии в рамках восстановления зависимостей по имеющей эмпирической информации.

Таким образом, рассмотренная задача SSL близка к задаче обучения с учителем. Её отличие от стандартной классификационной модели – помимо множества размеченных н.о.р. данных $D_l = \{(x_i, y_i) | i = 1, .. n\}, D_l \in D -$ присутствует подмножество н.о.р. неразмеченных данных $D_u = \{x_{n+j} | j = 1, .. m\}, D_u \in D$. В подобных задачах наиболее важными ситуациями являются те, при которых $m \ll n$ – когда известно небольшое число классов объектов и другие методы прогнозирования признаются неэффективными или экономическими затратными. Несмотря на то, что задачи с частичным привлечением учителя были впервые поставлены относительно давно, их использование и распространение началось только в конце XX в., прямо сейчас эти методы находят свое приложение в различных сферах деятельности, где необходим быстрый анализ значительного объема информации в том числе и экономической области.

Глава 2: Метод Co-training в задаче частичного обучения

§2.1 Метод Co-training в задаче частичного обучения

Первая работа, в которой был полностью сформулирован метод кооперативного обучения или «Co-Training»-метод задачи SSL, принадлежит двум американским исследователям: Avrim Blum и Tom Mitchel. В 1998 году они опубликовали статью «Combining Labeled and Unlabeled Data with Co-Training» [14], в которой также представили результаты оценки качества разработанного алгоритма классификации.

Главное предположение, которые выдвинули авторы состоит в следующем:

Предположение метода Co-Training: (2.1.1)

«Если возможно разделить всё признаковое пространство объектов на два слабо зависимых друг от друга множества, то каждого представления для объектов может быть достаточно для эффективного обучения классификаторов в модели» [14].

Данное предположение (2.1.1) целесообразно перевести в понятия необходимого и достаточного условия эффективной работы предложенного метода.

Необходимое условие Co-Training: (2.1.2)

Признаковое пространство объектов разделимо и может быть разбито на два слабо зависимых друг от друга подпространства: $X = X_1 \times X_2$, $Cov(X_1, X_2) \to diag\{p_1, \dots, p_n\}$, причем $k \geq 2, k -$ общее число признаков.

Достаточное условие Co-Training: (2.1.3)

Каждое признаковое подпространство объектов X_1, X_2 должно оставаться информативным и включать достаточно размеченных данных и признаков, определяющих в каждом классификаторе решающее правило (1.1.1)

Формализация поставленной проблемы выглядит следующим образом:

Пусть поставлена задача бинарной классификации объектов: $Y = \{-1, +1\}$. Представим пространство исследуемых образцов как $X = X_1 \times X_2$, где $X_1, X_2 - 2$ различных «взгляда»[8] на объекты, 2 набора признаков. Если x – пара (x_1, x_2), то $x_1, x_2 -$ различные «поля зрения» на x, Предполагается, что выполнены условия (2.1.2), (2.1.3).

Введём $\Theta_j -$ пространство ответов классификаторов h_j , $j = \{1,2\}$. Элементы $\theta = (\theta_1, \theta_2) \in \Theta = \Theta_1 \times \Theta_2$ – общие представления о классах X. Если классификаторы h_j согласовывают свои предсказания: $\theta_1(x_1) = \theta_2(x_2)$, то можно записывать, что: $\theta(x) = \theta_1(x_1)$. Тогда если $A \subset X$, можно сказать, что представление о классах объектов $\theta = (\theta_1, \theta_2)$ совместимо с A, если $\theta_1(x_1) = \theta_2(x_2)$ $\forall\, x = (x_1, x_2) \in A$.

Необходимо найти решающие правила (1.1.1) , выбирая некоторое $A \subset X$ на каждой итерации с помощью классификаторов h_j, $j = \{1,2\}$ по следующему алгоритму, представленному в виде псевдокода:

Вход:

1. → Небольшой массив размеченных данных L
2. → Остальной массив неразмеченных данных U
3. → Два списка признаков изучаемых объектов V_1 и V_2, разделяющих пространство X
4. → Задать модели классификации h_1, h_2

Для i-й итерации в i=(1:k):

5. → Использовать L для обучения модели h_1 с набором признаков V_1
6. → Использовать L для обучения модели h_2 с набором признаков V_2
7. → Оценить классы объектов в U с помощью h_1, h_2
8. → Выбрать уверенно классифицированные объекты A, добавить в L, удалить из U;

(2.1.4)

Каждое новое предсказание алгоритмов, согласованное со всеми независимыми предикторами, добавляется в размеченную выборку объектов (labeled data), классифицированные объекты на итерации удаляются из неразмеченной выборки (unlabeled data) и процесс обучения повторяется до k-й итерации включительно, корректируя модель.

[8] Оригинальное определение – views [13]

Таким образом возможно обучать 2, 3..., n-независимых алгоритмов, нацеленных на одну и ту же задачу классификации, однако использующих слабо зависимые признаковые описания объектов.

Авторы использовали данные по классификации тематики web-страниц. Используя выдвинутое изначально предположение (2.1.1), они разделили на два множества признаки, содержащиеся в самом тексте и признаки, содержащиеся в гиперссылках web-страниц. Сфера первоначального применения метода была в первую очередь обусловлена наличием значимого числа данных, на которых возможен предложенный анализ. На сегодня объем данных в экономической сфере существенно вырос, многие компании стараются хранить всю информацию по своим клиентам, их истории покупок и посещений; большинство банков разрабатывает скоринговые модели по классификации клиентов для анализа их потребительского поведения, в том числе и кредитного скоринга.

Тем не менее, необходимо подчеркнуть, что предложенный метод классификации (2.1.3) не является панацеей решения всех возникающих проблем, которые возможно формализовать, и, как многие модели и методы прикладной статистики и машинного обучения, имеет собственные рамки применения, основанные на допущениях, а также свои преимущества и недостатки. Метод кооперативного обучения расширяет спектр инструментов сегодняшней аналитики, позволяя исследователю получить многосторонний взгляд на ту или иную поставленную проблему. Критический анализ представленного метода изложен в следующем параграфе.

§2.2 Преимущества и недостатки метода Co-training

Основные преимущества метода Co-training состоят в следующем:

- Данный метод в рамках задач SSL является относительно простым в исполнении и способен повысить качество классификации на больших данных с ограниченным числом обучающих примеров.
- В работе [14], показано, что при задании оптимального числа итераций возможно значимое улучшение качества модели по сравнению с обычным моделированием по прецедентам и, соответственно, единоразовым предсказанием классов объектов.
- Представленный метод – метод-обёртка и подразумевает применение внутренних моделей в качестве функций классификации, поэтому он является очень гибким и может использовать все возможные алгоритмы, поддающиеся формализации в рамках этого метода. Например, можно использовать такие модели, как наивный байесовский классификатор или логистическую регрессию.
- С экономической точки зрения метод способен дать новую полезную информацию об исследуемых объектах без существенного вовлечения средств на дополнительную разметку данных, что необходимо для нормальной работы других моделей. Например, затраты компании на обзвон всех её клиентов или вовлечение в интернет-опрос с целью узнать, в какое время они предпочитают осуществлять покупки, могли бы быть необоснованными.

Помимо заявленных преимуществ метода, необходимо отметить ряд недостатков и открытых вопросов:

- При проверке необходимого условия (2.1.2) неизвестно, насколько незначительны должны быть элементы ковариационной матрицы $Cov(X_1, X_2)$, находящиеся вне её диагонали.
- Проверка достаточного условия (2.1.3) может быть осуществлена только после вычислительных экспериментов на основе итоговой оценки качества работы алгоритма.
- Оптимальное число итераций k необходимо определять индивидуально для каждой задачи, настраивая его, ориентируясь по показателям качества.
- Не определено, какой объем размеченных объектов для обучения относительно неразмеченных данных был бы достаточен для эффективной работы алгоритма.

- Остаётся неясным, как правильно необходимо формировать подмножества из множества признакового описания объектов. Для начала изучения оптимального разбиения признаков можно использовать априорную информацию о них.
- Базовый критерий согласованности предсказаний классов объектов является относительно жестким, т.к. предсказывается сразу вероятный класс для каждого включенного классификатора.
- Кроме того, создание n-классификаторов на основе n-подмножеств признакового описания объектов ограничено возможностью разбиения условно независимых подмножеств этих признаков.

Отмеченные в работе проблемы также поднимаются исследователями. В статьях [19, 29, 30, 32] авторами анализируется возможность модификации метода Co-training и предлагаются собственные нововведения, эффективность которых измеряется на реальных данных.

В качестве идей по модификации метода и повышении его оцениваемого качества, можно предложить:

1) Смягчить условие уверенной классификации до согласованной вероятности принадлежности объекта к одному из представленных классов и отбирать уверенно классифицированные объекты на основе разделяющего критерия $\boldsymbol{\delta} = (0, 1)$.

2) Использовать алгоритм разбиения признаков с целью автоматической минимизации их зависимости в каждом из подмножеств.

3) Добавить возможность выбора критерия сходимости: с заданием числа итераций и с условием полной разметки объектов и итерационным регулированием разделяющего критерия вероятности $\boldsymbol{\delta}$ из 1) на шаг $\boldsymbol{\Delta}$ для соблюдения условия сходимости алгоритма.

4) Проверять качество суммарной классификации на каждом шаге с целью поиска и разметки только значимых множеств объектов $\boldsymbol{A}$ из (2.1.4) во избежание добавления шума в обучающую выборку и регулирование её от переобучения.

5) Внедрить в алгоритм проверку достаточного условия (2.1.3).

6) Добавить элемент обучения с подкреплением[9] - введение и корректировка на каждой итерации коэффициентов значимости вклада в общий прогноз применяемых внутри метода моделей.

[9] «Reinforcement learning – «обучение с подкреплением» – один из способов машинного обучения, в котором создаваемая среда взаимодействует с агентом (моделью), получает дополнительные сигналы для повышения качества моделирования [11].

§2.3 Предлагаемые варианты применения метода Co-training

В данной работе в рамках метода Co-training используются 3 общепринятые модели прикладной статистики, применяемые во многих статьях, например, в [18, 24, 29] по разработке и оценке предлагаемых подходов к решению задач: наивный байесовский классификатор, логистическая регрессия и ансамблевый алгоритм случайного леса.

Модель наивного байесовского классификатора использует классическую формулу условной вероятности:

$$P(x|y) = \frac{P(x,y)}{P(y)} \qquad (2.3.1)$$

Основное строгое предположение, которое используется при моделировании наивным байесовским классификатором – объекты исследования X имеют независимые признаки [2]. Тогда совместная вероятность из (2.3.1) $P(x, y) = P(y|x)P(x)$.

Именно из-за этого предположения классификатор называют «наивным», т.к. на практике не существует набора признаков, имеющих полную независимую природу происхождения. Данный классификатор удобно использовать в качестве некоторой базы сравнения относительно других моделей классификации.

$P(x|y)$ называется апостериорной вероятностью – вероятностью принадлежности объекта к тому или иному классу при условии наличия информации – некоторой статистики об объектах.

$P(y|x)$ – вероятность получения текущих данных при условии фиксированных параметров модели.

$P(x)$ – априорная вероятность – вероятность того, что известно заранее по имеющимся данным. $P(y)$ является некоторым нормировочным коэффициентом.

Несмотря на кажущиеся чрезмерно упрощенными предположения, наивный байесовский классификатор неплохо работает во многих реальных ситуациях, отлично классифицируя, например, потоки писем электронной почты. Этот классификатор требует небольшого объема обучающих данных для оценки необходимых параметров. Разделение распределений условных объектов класса означает, что каждое распределение может быть независимо оценено как одномерное распределение. Это в свою очередь помогает облегчить проблемы, вытекающие из проклятия размерности.

Применение наивного байесовского классификатора может быть легко проиллюстрировано на следующем примере:

Предположим, что имеется информация о 1000 автомобилях. Акцент на ценах автомобиля в данном примере не производится. По опросам пользователей автомобилей этих моделей выявлено, что есть авто: высокой комфортности, средней комфортности, низкой комфортности. Известно крайне небольшое число информации по автомобилям: просматривается ли авто каждый день в поисковых запросах в Интернете, основная фабрика находится в Германии, в качестве двигателя установлен бензиновый ДВС.

Естественным образом опросы по автомобилям позволяют сформировать следующие классы с имеющейся информацией:

Таблица 1: Модельные данные для наивного байесовского классификатора

Класс авто по опросам	Ежедневно в поисковых запросах	Основная фабрика в Германии	Бензиновый ДВС	Итого
Комфортный	250	280	200	300
Средний комфорт	400	300	350	500
Низкий комфорт	50	0	180	200
Итого	700	580	730	1000

Допустим, мы хотим оценить комфортность этого автомобиля: популярного – ежедневно встречается в поисковых запросах, производимого в Германии и двигатель которого не бензиновый - по имеющимся данным. Вопрос, который мы задаём, какова вероятность нашего автомобиля быть высоко комфортным на протяжении всей его эксплуатации:

Применяя формулу (2.3.1), получаем следующее:

$$P\left(\text{Класс авто} - \text{Комфортный} \,\middle|\, \begin{matrix}\text{Ежедневно в Интернет} - \text{запросах},\\ \text{Производится в Германии},\\ \text{Не бензиновый двигатель}\end{matrix}\right) =$$

$[P(\text{Ежедневно в Интернет} - \text{запросах}|\text{Комфортный}) *$

$P(\text{Производится в Германии}|\text{Комфортный}) *$

$P(\text{Не бензиновый двигатель}|\text{Комфортный}) *$

$P(\text{Комфортный})]/$

$[P(\text{Ежедневно в Интернет} - \text{запросах}) *$

$P(\text{Производится в Германии}) *$

$P(\text{Не бензиновый двигатель})]$

Используя сокращения, получим:

$$P(class = comf \mid Int.search, Germany, AltEngine) =$$

$$= \left(\frac{250}{300} \cdot \frac{280}{300} \cdot \frac{(300-200)}{300} \cdot \frac{(300)}{1000}\right)/\left(\frac{700}{1000} \cdot \frac{580}{1000} \cdot \frac{(1000-730)}{1000}\right) = 0{,}71$$

Таким образом, вероятность принадлежности исследуемого автомобиля к классу повышенной комфортности составляет 0,71 в условиях наивного байесовского классификатора.

На практике наивный байесовский классификатор является одним из самых быстрых алгоритмов по оценке принадлежности объектов к классам и, как было показано, прост в реализации. Однако основным его недостатком является низкое качество классификации из-за строгих предпосылок независимости признаков и, как уже было сказано, он используется в качестве базы сравнения других моделей.

Вторая модель, используемая в исследовании – логистическая регрессия, является линейным классификатором и позволяет также дать оценки вероятностей принадлежности объектов к одному из заранее известных классов.

Как и другие формы регрессионного анализа, логистическая регрессия использует одну или несколько предикторных переменных, которые могут быть непрерывными или категориальными. Однако, в отличие от обычной линейной регрессии, логистическая регрессия используется для прогнозирования зависимых переменных, которые принимают ограниченное число некоторых категорий (рассматривая зависимую переменную в биномиальном случае как результат испытания Бернулли). Учитывая это различие, допущения линейной регрессии нарушаются. В частности, остатки не могут быть нормально распределены. Кроме того, линейная регрессия может делать бессмысленные прогнозы для бинарной зависимой переменной. Необходим способ преобразования двоичной переменной в непрерывную, которая может принимать любое действительное значение (отрицательное или положительное). Для этого биномиальная логистическая регрессия сначала берет коэффициенты события, происходящего на разных уровнях каждой независимой переменной, затем берет отношение шансов, а затем берет логарифм этого отношения, чтобы создать непрерывный критерий в качестве преобразованной версии зависимой переменной.

Условимся, что рассматриваем бинарную классификацию, где $Y = \{-1, +1\}$. Модель логистической регрессии использует так называемую функцию активации – сигмоиду, которая включает в себя латентную переменную $z = (x_i, w_i)$ – скалярное произведение признакового описания объектов на вектор весов:

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad (2.3.2)$$

Для того, чтобы обучить данную модель логистической регрессии, необходимо настроить вектор весов $w_i = (w_1, w_2, \dots, w_n)$ по обучающей выборке X, минимизировав соответствующий функционал эмпирического риска:

$$\frac{1}{n}\sum_{i=1}^{n} \log[\,1 + \exp(-y_i(w_1 x_{i1} + w_2 x_{i2}))] + 0.5 \cdot C\left||w|\right|^2 \to \min;\ \ w_1, w_1 \quad (2.3.3)$$

Для решения данного функционала (2.3.3) можно воспользоваться градиентным шагом, обновляя веса w_1, w_2 соответственно с шагом k:

$$w_1 \coloneqq w_1 + k\frac{1}{n}\sum_{i=1}^{n} y_i\, x_{i1}\left(1 - \frac{1}{1+\exp\left(-y_i(w_1 x_{i1} + w_2 x_{i2})\right)}\right) - kCw_1$$
$$w_2 \coloneqq w_2 + k\frac{1}{n}\sum_{i=1}^{n} y_i\, x_{i2}\left(1 - \frac{1}{1+\exp\left(-y_i(w_1 x_{i1} + w_2 x_{i2})\right)}\right) - kCw_1$$

(2.3.4)

$C\left||w|\right|^2$ является дополнительной константой регуляризации L2 модели или регуляризацией по Тихонову [6], применяющейся при корректировках для предотвращения переобучения и нахождения приближенного решения задачи, т.к. в большинстве случаев причина переобучения модели является в завышении весовых коэффициентов подобных моделей [2].

Настроив соответствующие веса, получаем обученную сигмоиду – функцию активации, определяющую вероятность принадлежности объекта к классу:

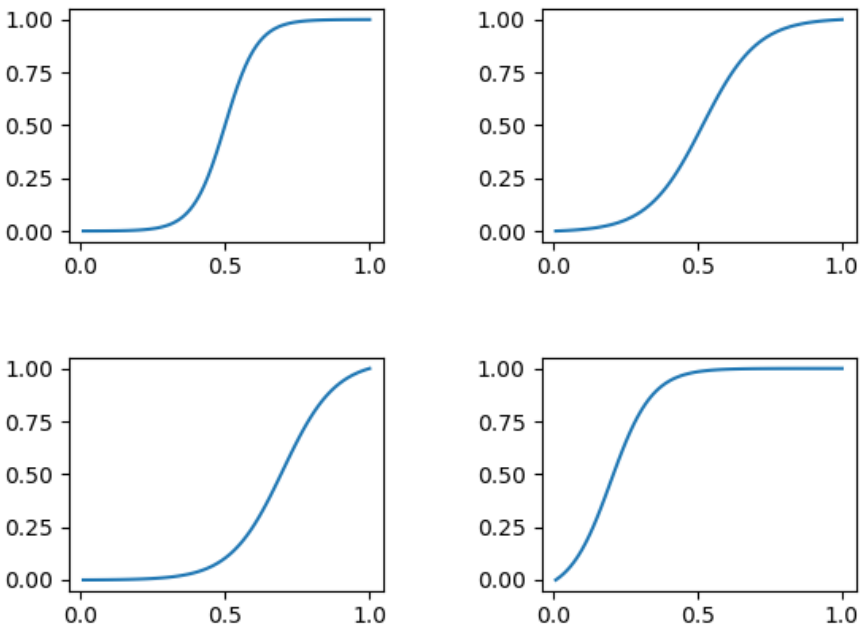

Рисунок 3: Примеры моделируемых сигмоид в логистической регрессии

Рассчитано автором по: библиотеки Python «scikit-learn», «matplotlib».

Современное обобщение функции активации для мультиклассификации носит название Softmax - функции [8, 16]. Основным преимуществом использования Softmax является диапазон выходных вероятностей. Значения принимаются в диапазоне от 0 до 1, а сумма всех вероятностей составляет единицу. Если функция Softmax используется для мультиклассификационной модели, то она возвращает вероятности каждого класса:

$$P(y = j|x) = \sigma(z_j) = \frac{e^{z_j}}{\sum_{j=0}^{k} e^{z_j}}, \text{j} = 0,1,2, \dots \text{k}; \qquad (2.3.5)$$

Запись (2.3.5) показывает, что Softmax-функция вычисляет экспоненциальную мощность заданного входного значения и сумму экспоненциальной мощности всех значений на входе. Тогда отношение экспоненты входного значения и суммы экспоненциальных значений является выходом функции Softmax.

Третья модель, включённая в работу, является относительно новой сфере прикладного анализа, однако справляется со множеством задач в области обучения по прецедентам – случайный лес. Алгоритм случайного леса был предложен в 2001 году исследователями Leo Breiman, Adele Cutler [15]. Он заключается в применении комитета решающий деревьев.

Методов построения решающих деревьев на сегодня известно достаточно много. Самые распространённые из них: CART – Classification and Regression Trees, ID3 - Iterative Dichotomiser 3, C4.5, C 5.0 [8].

Дерево решений или решающее дерево, решающий пень - простое представление для классификации объектов. Для этого предполагается, что все входные функции имеют конечные дискретные области, и есть одна целевая функция, называемая «классификация». Каждый элемент области данной классификации называется классом. Деревом решений или деревом классификации является такое дерево, в котором каждый внутренний (не листовой) узел помечен функцией ввода. Ветви, идущие от узла с меткой входной функции, помечены каждым из возможных значений целевого или выходного признака, или же ветвь приводит к подчиненному узлу решения с другой функцией ввода. Каждый лист дерева помечен классом или распределением вероятности по классам.

Дерево можно обучить, разделив исходный набор данных на подмножества на основе определения значимости признака. Этот процесс повторяется на каждом производном подмножестве рекурсивным образом, называемом рекурсивным разбиением.

Алгоритмы построения деревьев решений обычно работают сверху вниз, начиная с родительского главного узла, выбирая переменную на каждом шаге, которая наилучшим образом разбивает набор элементов [8]. Различные алгоритмы используют разные показатели для измерения наилучшего разбиения. Обычно эти показатели измеряют однородность целевой переменной внутри подмножеств.

ID3 (Iterative Dichotomiser 3) был разработан в 1986 году Россом Квинланом. Алгоритм создает ветвящееся дерево, находя для каждого узла (т. е. «жадным» образом) категориальный признак, который даст наибольший прирост информации для категориальных целей. Деревья создаются до максимального размера, а затем обычно применяется шаг обрезки деревьев, чтобы улучшить способность дерева обобщать данные.

C4.5 основан на алгоритме ID3. В нём снято ограничение, что объекты должны быть категориальными, динамически определяя дискретный атрибут (на основе числовых переменных), который разбивает непрерывное значение атрибута на дискретный набор интервалов. C4.5 преобразует обученные деревья (т. е. вывод алгоритма ID3) в наборы правил «если-то». Затем оценивается точность каждого правила для определения порядка их применения. Обрезка дерева или прунинг (pruning) выполняется путем удаления предварительного условия Правила, если точность правила улучшается без него.

C5.0 – последняя версия алгоритма, использует меньше памяти и создает меньшие наборы правил, чем C4.5 и на сегодня является одним из самых точных в ряду алгоритмов построения деревьев.

CART (деревья классификации и регрессии) похож на C4.5, но он отличается тем, что поддерживает числовые целевые переменные (регрессию) и не вычисляет наборы правил.

CART строит бинарные деревья, используя функцию и порог, которые дают наибольший прирост информации на каждом узле.

Одним из распространённых показателей эффективного разбиения входного множества данных для построения дерева носит название Gini Impurity [8]. Используемый алгоритмом CART, данный показатель является мерой того, как часто случайно выбранный элемент из набора имеющихся данных был бы неправильно помечен, если бы он был помечен случайным образом в соответствии с распределением меток в подмножестве.

Предположим, имеется j – классов, $i \in \{1,2,\dots,j\}$, p_i – доля размеченных объектов классом i в представленных данных. Gini Impurity можно вычислить путем суммирования вероятностей p_i, умноженных на $\sum_{k\neq i} p_k = 1 - p_i$ – ошибки. Показатель достигает минимального значения – нуля, когда все случаи в узле составляют один класс:

$$I_G(p) = \sum_{i=1}^{j} p_i \sum_{k\neq i}^{j} p_k = \sum_{i=1}^{j} p_i\,(1-p_i) = \sum_{i=1}^{j}(p_i - {p_i}^2) = \\ = \sum_{i=1}^{j} p_i - \sum_{i=1}^{j} {p_i}^2 = 1 - \sum_{i=1}^{j} {p_i}^2 \qquad (2.3.6)$$

Также распространённым критерием формирования дерева является энтропийный критерий – энтропия Шеннона:

$$I_S = -\sum_{i=1}^{j} p_i \log(p_i) \qquad (2.3.7)$$

Энтропия Шеннона или информационная энтропия согласно теории информации, является общей мерой неопределенности, «хаоса» в текущем узле дерева, распределяющим объекты на соответствующие классы [7].

Для иллюстрации механизма работы показателей эффективного построения дерева используем энтропию Шеннона на следующем примере. Допустим, имеется информация о 2 классах объектов и некоторый описывающий их признак X:

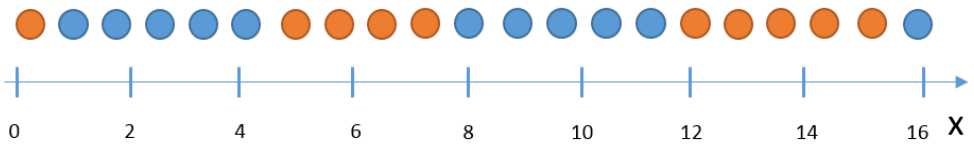

Рисунок 4: Пример работы критерия построения дерева (1)

Изначально по всему множеству объектов показатель энтропии равен:

$$I_{S,0} = -\left(\frac{11}{21}\right)\cdot \ln\left(\frac{11}{21}\right) - \left(\frac{10}{21}\right)\cdot \ln\left(\frac{10}{21}\right) \approx 0{,}69$$

По набору признаков «жадным» перебором ищется та точка, при которой значение энтропии было бы минимальным, т.е. разбиение было наиболее эффективным на текущей итерации. Возьмем для примера условие разбиения множества объектов: $X_1 \leq 8$, тогда:

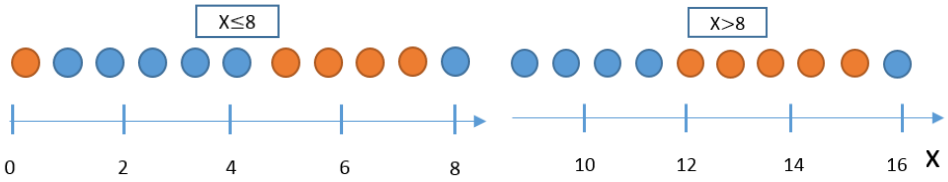

Рисунок 5: Пример работы критерия построения дерева (2)

И энтропия по двум подмножествам, разделённым узлом $X_1 \leq 8$, будет:

$$I_{S,1\,(1)} = -\left(\frac{6}{11}\right) \cdot \ln\left(\frac{6}{11}\right) - \left(\frac{5}{11}\right) \cdot \ln\left(\frac{5}{11}\right) \approx 0{,}68$$

$$I_{S,1(2)} = -\left(\frac{5}{10}\right) \cdot \ln\left(\frac{5}{10}\right) - \left(\frac{5}{10}\right) \cdot \ln\left(\frac{5}{10}\right) \approx 0{,}69$$

Цифры в индексе показателя энтропии вне скобок отсылают к разделяющей границе, в скобках указывают нумерацию множества, по которому оценивается уровень энтропии. Как видно, существенного прироста информации, т.е. снижения показателя энтропии не произошло. Однако дальнейшее разбиение по данному признаку:

$$X_{1,1} \leq 4, X_{1,2} \leq 11: I_{S,1,1(1)} \approx 0{,}45, I_{S,1,1(2)} \approx 0{,}50, I_{S,1,2(3)} = 0, I_{S,1,2(4)} \approx 0{,}45$$

– позволило достичь существенного прироста информации в классификации объектов по предоставленным данным. Дерево решений строится именно на таком подходе, создавая соответствующие узлы, оптимальные на каждой итерации и достигая определённого критерия останова – по число узлов дерева, уровню информационного критерия и т.п.

При использовании же ансамбля решающих деревьев – случайного леса, используется генерация деревьев по следующему алгоритму (пример алгоритма CART):

Вход:

1. → Множество из M – объектов с k - признаками

Операции:

2. → Создать из M– выборки подвыборку данных размером N с повторением;
3. → Построить решающее дерево с числом признаков $\approx \sqrt{k}$;
4. → Опционально: Подобрать оптимальное число деревьев, ориентируясь на ошибке по тестовой выборке;
5. → Итоговыми оценками классификатора будут вектора ответов для каждого объекта, подлежащего классификации. Простое или взвешенное голосование; комитета деревьев даёт вероятности принадлежности объекта к классу.

На примере классификации ирисов Фишера [8] – известной задачи в теории машинного обучения, пространство объектов с помощью дерева решений разбивается на дискретные области, пример которых можно видеть на следующем рисунке:

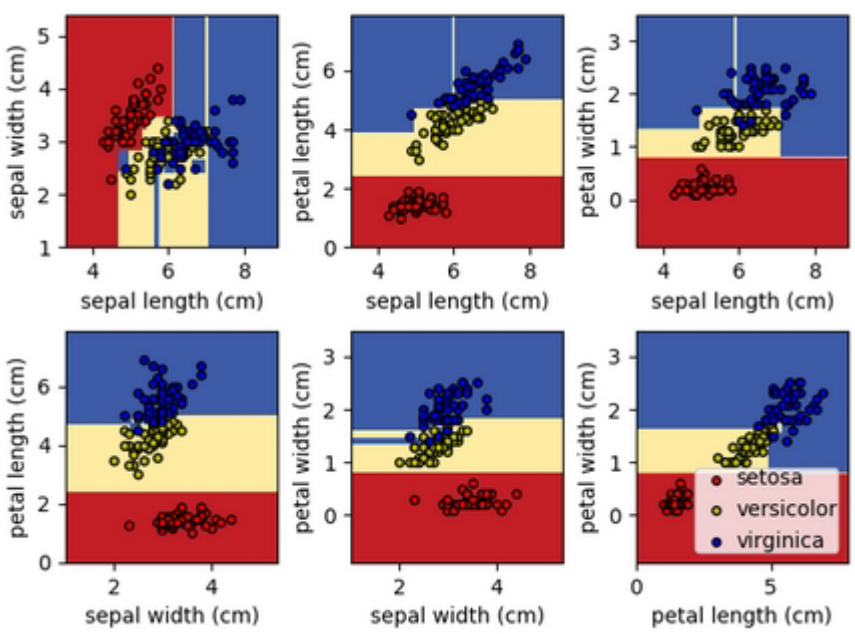

Рисунок 6: Разбиение пространства объектов на основе 2 признаков

Источник: http://scikit-learn.org

Таким образом, в рамках метода Co-training (2.1.4) с использованием данных моделей возможно рассмотреть следующие случаи:

1. 2 наивных байесовских классификатора;
2. 2 логистических регрессии;
3. 2 случайных леса;
4. 3 Комбинации из трёх предложенных моделей.

В дополнение важно добавить алгоритм обучения по прецедентам для сравнения показателей качества. Кроме того, необходимо отметить, что случай использования логистической регрессии в качестве внутренних моделей метода Co-training и использование предложенной модификации 4) представляет собой специфический случай модели искусственной нейронной сети [8], в которой 2 модели взаимно улучшают качество друг друга, имея соответствующие функции активации в виде сигмоид:

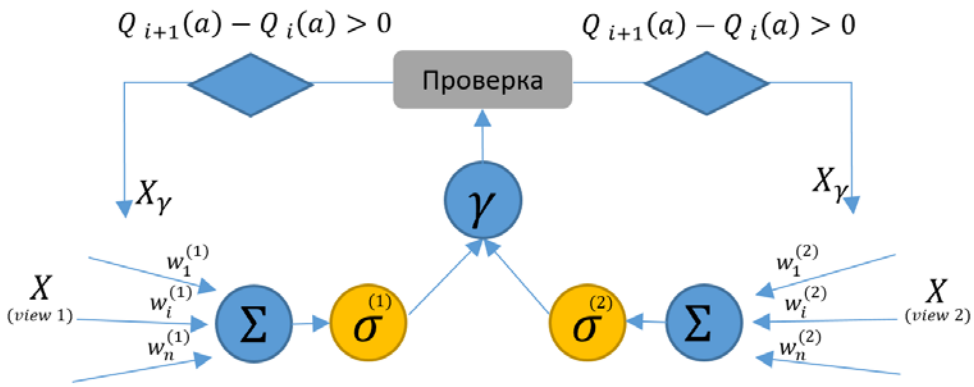

Рисунок 7: Предложенная модификация Co-Training с логистической регрессией, повторяющая элементы работы искусственной нейронной сети модели Ф. Розенблатта [12]. Составлено автором.

На Рис.6 Σ – так называемый сумматор, вычисляющий итоговое значение произведения весов w_i на значения X_i на входе, γ – набор согласованных объектов для классификации, проверка – предложенная модификация 4) представленного метода.

Следуя рекомендованным этапам процесса моделирования по стандарту CRISP-DM [10], применение предложенного метода Co-training может быть реализовано согласно следующей методологии:

1. Априорное разбиение множества признаков на 2 подмножества;
2. Корректировка априорного разбиения подмножеств с целью минимизации их зависимости внутри каждого подмножества;
3. Отбор релевантных признаков:
   - ⇨ На основе статистических критериев;

- ⇨ На основе значимости признаков при разбиении в модели случайного леса;

4. Формирование обучающей и тестовой выборок;
5. Выбор критериев оценки качества моделирования;
6. Построение предложенных моделей;
7. Отбор наилучшей модели из построенных;
8. Итоговая классификация множества объектов;
9. Последующий экономический анализ на основе полученных результатов.

Таким образом, рассмотренный в теоретической части работы метод частичного обучения с учителем является одной из перспективных разработок в области статистического восстановления существующих зависимостей, что, например, показано в работе [14]. Важно принимать во внимание предположения гладкости, кластеризованности, а также избыточности данных для всех задач с частичным обучением. При рассмотрении метода Co-training необходимо задавать внутренние модели – классификаторы для работы с предоставляемой информацией по анализу данных. Кроме этого, важно учитывать основные предположения авторов представленного метода и ряд недостатков, которые могут повлиять на желаемый результат относительно тестируемых гипотез в каждом конкретном случае. Соответственно, учёт всех представленных факторов позволяет осуществлять моделирование в рамках пространственной структуры данных и использовать оценки для экономического анализа субъекта и лица, принимающего решения, в пользу которого он проводится.

Глава 3: Оценка объема лояльных клиентов

§3.1 Описание бизнес-задачи и её постановка в рамках машинного обучения

Во время прохождения производственной практики в компании ООО «Газпромнефть-Центр» в направлении информационно-аналитических методов повышения эффективности деятельности коллегами из блока маркетинга была предложена задача по разработке автоматизированной модели оценки поведения клиентов, участвующих в бонусной программе, с точки зрения предсказания степени их лояльности к посещению автозаправочных станций компании. Исследовательский инструментарий для решения данной задачи включил в себя объектно-ориентированный интерпретируемый язык программирования Python 3, а также библиотеки scikit-klearn и statmodels в качестве основы построения моделей [34, 35].

Таким образом, рассмотрим ситуацию, когда менеджменту компании необходимо осуществить оценку потенциального дохода от стимулирования продаж собственных продуктов – объемов топлива и сопутствующих товаров и услуг. Чтобы путем маркетинговых акций повлиять на поведение клиента и, соответственно, увеличивать объемы прокачки топлива по станциям, необходимо знать оценку числа клиентов, потенциально способных увеличить объемы продаж топлива в компании по причине вероятного посещения ими иных заправок. Данное поведение клиентов принято называть «свитчерским». Следовательно, «свитчеры» (от англ. switch – «переключать(ся)») – это нелояльные клиенты, часть потребляемых объемов топлива которых закупается у конкурентов компании. Поэтому рассмотрим задачу оценки степени лояльности клиентов, сфокусировавшись на определении свитчеров в общем объеме клиентов, пользующихся картами программы лояльности, которые позволяют присваивать транзакции конкретному потребителю.

Поставленную задачу могут решать структурные подразделения, носящие довольно разные наименования: аналитики клиентского актива, исследования и разработок (R&D) в области работы с потребителями, анализа данных (Data Science, Data Factory) и т.п. Решение поставленной задачи основывается на двух типах информации: данных из транзакционной базы, а также статистике из блока маркетинга, полученной в результате работы с клиентами посредством выборочных опросов. Количество клиентов автозаправок по программе лояльности, и, соответственно, векторов их признакового описания составляет 4 582 063, из которых 10894 клиента имеют метки класса по лояльности на основе выборочных опросов (1 – нелояльный клиент, свитчер, 0 – лояльный).

Таким образом, изначально были даны следующие показатели транзакционной базы за первые 3 квартала 2017 года:

- Транзакция Розница Номер Карты Бонусной;
- Транзакция Розница Номер Чека Порядковый;
- Транзакция Розница Дата Операции;
- Транзакция Розница Время Операции;
- Транзакция Розница Количество;
- Транзакции Розница Объем Литр;
- Транзакции Розница Выручка Брутто С Налогом Руб;
- Регион Наименование;
- Объект Управления Свойство Тип Расположения АЗС;
- Номенклатурная Группа Наименование;

На основе этих полей предварительно было создано 2 набора признаков, составляющих уникальные профили: частотно временной и территориальный профиль, а также клиентский или потребительский профиль, данные по которым по априорным предположениям генерируются условно независимыми процессами в соответствии с логикой моделирования с помощью метода Co-Training:

Частотно-временной и территориальный профиль:

days_per_check – отношение всего рассматриваемого периода к числу чеков клиента;
modal_check_time_hour – модальный час заправки клиента;
modal_buy_region – модальный регион покупок клиента;
exist_in_other_regions – покупает ли клиент топливо или СТиУ[10] в других регионах;
modal_type_azs – модальный тип (свойство) заправки клиента;
use_other_azs_type – использует ли клиент АЗС другого типа (свойства);
modal_day_of_week – модальный день заправки клиента;
azs_unique_quantity – уникальное число АЗС, на которых побывал клиент за весь период;
day_type_weekend – 1 - заправляется чаще в выходные, 0 - заправляется чаще в будни;
unique_azs_per_day – число уникальных АЗС, на которых заправлялся клиент по отношению к сумме дней, когда заправлялся клиент;

[10] СТиУ – сопутствующие товары и услуги

Потребительский профиль:

total_check – общее число чеков клиента за период;
total_fuel_value – общая сумма купленного топлива за весь период;
avg_check_per_day – среднее число чеков, деленное на весь период;
avg_sum_per_check – средняя сумма на 1 чек клиента;
modal_sum_per_check – модальная сумма за чек;
modal_fuel_sum_per_check – модальная сумма объемов топлива за чек;
avg_fuel_sum_per_check – среднее количество объемов топлива на 1 чек;
buy_stiu – покупал ли вообще клиент СТиУ или нет за свою транзакционную историю;
modal_buy_stiu – модальный тип СТиУ клиента, если покупал СТиУ;
modal_fuel_type – модальный тип закупки топлива клиентом;
exist_other_fuel_type – покупал ли клиент не модальный тип топлива;
num_stiu_per_num_fuel – сумма числа покупок СТиУ (в чеках), деленная на число покупок топлива (в чеках);
sum_fact_fuel_per_day – склонность к ежедневной заправке клиента – сумма фактов заправок клиента (в чеках), деленная на весь период в днях;
sum_fact_stiu_per_day – склонность клиента к ежедневной покупке СТиУ – сумма фактов покупки СТиУ (в чеках), деленная на весь период в днях;
volume_92_per_check – средний объем закупок 92 топлива за чек;
volume_95_per_check – средний объем закупок 95 топлива за чек;
volume_95_brand_per_check – средний объем закупок 95 бренд топлива за чек;
volume_98_per_check – средний объем закупок 98 топлива за чек;
volume_98_brand_per_check – средний объем закупок 98 бренд топлива за чек;

Соответственно, в рамках задачи статистического обучения, необходимо найти такое решающее правило (1.1.1) для каждого клиентского профиля, чтобы максимизировать функционал качества (1.2.1). Решением задачи будет являться массив вероятностной оценки принадлежности клиента к группе свитчеров, которые потенциально могут увеличивать объемы продаж нефтепродуктов. Для этого на основе опросов из размеченных по лояльности 10894 клиентам формируется обучающая и валидационная выборка. Из практических соображений и на основе общепринятых эвристик целесообразно использовать 80% данных подобного формата на обучение и 20% ― на тестирование качества модели.

В целом общий объём клиентов, склонных к свитчерскому, нелояльному поведению можно оценивать либо точечно – фиксированным значением в результате работы модели частичного обучения, либо интервально – на основе предсказанных вероятностей принадлежности клиента к одному из классов, выбирая соответствующую границу вероятности. Мы будем акцентировать внимание на точечной оценке, основанной на классификационном качестве модели.

§3.2 Алгоритм решения задачи и выбор оптимальной модели частичного обучения

Таким образом, первоначальное априорное разбиение сгенерированных клиентских признаков позволяет сформировать 2 подмножества для решения задачи, условно называемые в процессе исследования feature_view_1 – частотно-временной и территориальный профиль и feature_view_2 – потребительский профиль.

Второй этап решения задачи – корректировка априорного разбиения признаков. Необходимо нивелировать взаимосвязь клиентских характеристик в рамках каждого признакового представления клиентов. Построение корреляционных матриц, совмещённых с тепловыми картами, наглядно показывает признаки, обязательные к удалению:

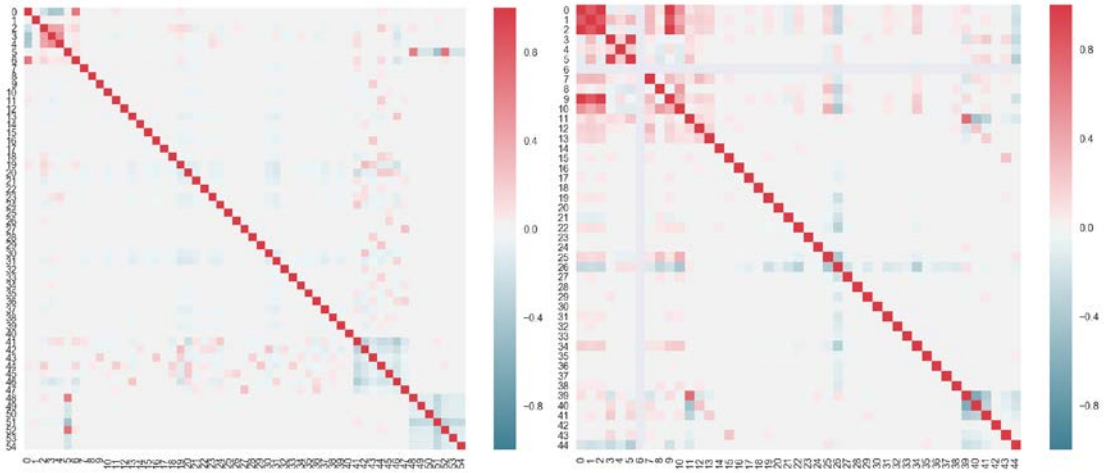

Рисунок 8: Корреляционные матрицы признаков с визуализацией на тепловых картах, слева направо частотно-временной и территориальный, потребительский профили

Рассчитано автором по: библиотеки Python «scikit-learn», «matplotlib», «seaborn».

Таким образом, из 1 подмножества признаков, согласно размеченным индексам, необходимо извлечь day_type_weekend, azs_unique_quantity, unique_azs_per_day, а из 2 подмножества признаков: 'total_fuel_value, total_check, sum_fact_fuel_per_day, avg_fuel_sum_per_check, sum_fact_stiu_per_day, modal_fuel_type_бензин 92, modal_fuel_type_бензин 95, modal_buy_stiu_нет, buy_stiu. В таком случае можно принять, что поставленное необходимое условие метода Co-Training (2.1.2) выполняется.

Для оценки значимости признаков в двух подмножествах проведём ступенчатый анализ, используя стандартные эконометрические тесты, а также относительно новый показатель значимости признаков, введённый Leo Breiman [14]. Значимость признаков с точки

зрения построения деревьев рассчитывается на основе ошибки OOB – Out of Bag error [7], которая является оценкой внутренних алгоритмов-деревьев случайного леса как доля верно классифицированных объектов. Тогда важность признаков – это разница между получающейся OOB с признаком и без него при построении дерева, т.е. вклад, который создает тот или иной признак для настройки соответствующего классификатора модели Значения важности признаков нормируются:

$$FI^T = \frac{\sum_{i \in B} I(y_i = \tilde{y}_i)}{B} - \frac{\sum_{i \in B} I(y_i = \widetilde{y_{i,pj}})}{B} \qquad (3.2.1)$$

где: FI^T – важность признака в дереве, B-число узлов, $\tilde{y}_i$- предсказание класса перед удалением признака, а $\widetilde{y_{i,pj}}$ – предсказание класса после удаления признака. Тогда нормированная важность признака x по всему лесу из N деревьев равняется:

$$FI(x) = \frac{\sum_{T=1}^{N} FI^T(x)}{\frac{\sigma}{\sqrt{N}}} \qquad (3.2.2)$$

Исходя из (3.2.1 - 3.2.2), важности, значимости признаков относительно логики построения случайного леса следующие:

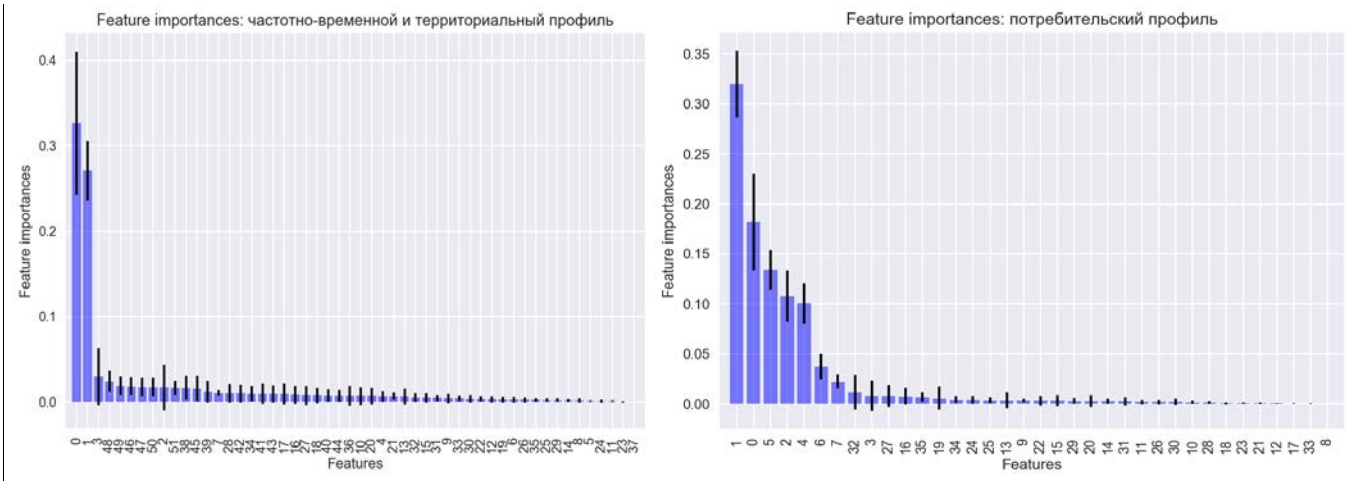

Рисунок 9: Показатели важности для двух подмножеств признаков с учётом 3σ

Рассчитано автором по: библиотеки Python «scikit-learn», «matplotlib», «seaborn».

Следовательно, по критерию важности признаков, высоко информативными с точки зрения классификации являются: для 1 подмножества days_per_check, modal_check_time_hour и низко информативными modal_day_of_week_пт, modal_day_of_week_сб, modal_day_of_week_вт, modal_day_of_week_пн, modal_day_of_week_ср, modal_day_of_week_чт. Для 2 подмножества значимыми являются

avg_sum_per_check, avg_check_per_day, volume_92_per_check, modal_fuel_sum_per_check, num_stiu_per_num_fuel, менее информативными volume_95_per_check, volume_95_brand_per_check.

Однако классические эконометрические тесты показывают[11], что на 95% доверительном уровне для признаков modal_day_of_week_пт, modal_day_of_week_сб, modal_day_of_week_вт, modal_day_of_week_пн, modal_day_of_week_ср, modal_day_of_week_чт в 1 подмножестве и volume_95_per_check, volume_95_brand_per_check во 2 подмножестве не отвергается гипотеза о равенстве коэффициентов при переменных нулю в уравнении латентной переменной логистической регрессии. Тест отношения правдоподобия показывает значимость в целом представленных моделей.

Два различных теста информативности признаков не полностью согласовывают свои результаты, т.к. тест значимости признаков с точки зрения случайного леса имеет иную природу, основанную на теории информации и нацеленную на повышение прироста объясняющей способности на каждом узле разбиения деревьев. Поэтому с точки зрения согласованности данных критериев для увеличения скорости вычислительных процедур в модели целесообразно использовать значимые признаки по двум тестам в совокупности.

Формирование обучающей и тестовой выборок осуществляется в соотношении 80%/20%, как было описано ранее. Тогда, с учётом предложенной модификации 4) алгоритм (2.1.4) примет вид:

Вход:

1. → Небольшой массив размеченных данных L
2. → Остальной массив неразмеченных данных U
3. → Два списка признаков изучаемых объектов V_1 и V_2, разделяющих пространство X
4. → Задать модели классификации h_1, h_2
5. → **Оценить качество начальной модели: $\mathbf{Q}(t_0)$**

Для i-й итерации в i=(1:k):

6. → Использовать L для обучения модели h_1 с набором признаков V_1
7. → Использовать L для обучения модели h_2 с набором признаков V_2
8. → Оценить классы объектов в U с помощью h_1, h_2
9. → Выбрать уверенно классифицированные объекты A
10. → **Оценить качество на итерации: $\mathbf{Q}(t_i)$**
11. → **$\mathbf{Q}(t_i) - \mathbf{Q}(t_{i-1}) > 0$, то добавить A в L, удалить из U;**

[11] См. Прил. 1

Важно отметить, что во избежание переобучения, 20% данных в качестве валидационной выборки не участвуют в процессе итерационного частичного обучения. Для оценки качества предложенных вариантов моделей в рамках метода Co-Training был выбран показатель F-меры, описанный в предыдущей главе как среднее гармоническое между важными характеристиками точности и полноты формируемой модели. Однако заранее неизвестно, какое количество итераций кооперативного обучения необходимо брать, поэтому был проведён анализ зависимости роста качества предсказаний от числа итераций частичного обучения.

Следующие результаты показывают зависимость качества от числа итерация метода Co-Training при данных условиях:

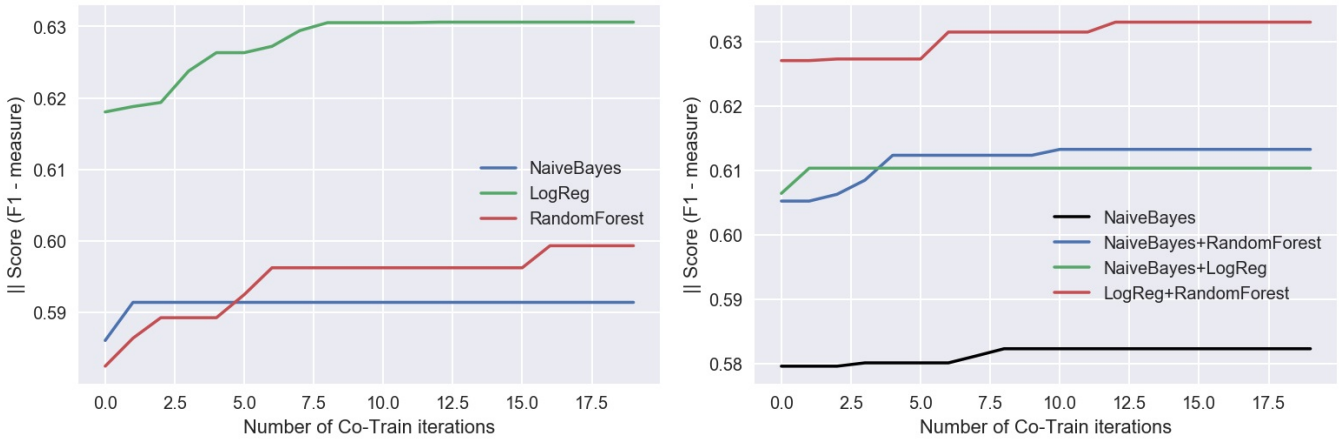

Рисунок 10: Зависимость качества прогноза (F-мера) от числа итераций в Co-Training – методе, слева – использованы отдельные модели в качестве ядра метода Co-Training, справа – их комбинации

Рассчитано автором по: библиотеки Python «scikit-learn», «matplotlib», «seaborn».

Выявлено, что модели типа случайного леса, а также наивный байесовский классификатор склонны к улучшению точности – показателя Precision, однако существенно проигрывают в значении полноты классифицируемых клиентов на контрольной выборке. По сравнению с этими классификаторами, логистическая регрессия показывает более значимую полноту классификации – Recall, однако проигрывает случайному лесу и наивной байесовской модели в точности.

На графике (Рис. 10) видно, что число итераций в рамках метода Co-Training способно увеличивать качество модели. В условиях представленной задачи высокую динамику роста

качества показывает логистическая регрессия в качестве ядра алгоритма, однако наибольшее качество по F-мере было получено за счет комбинации моделей. В двух наилучших моделях присутствует логистическая регрессия и модель для компенсации низкого показателя точности при проверке качества – случайный лес. Оптимальное число итераций находится в интервале от 5 до 10.

Следующая таблица показывает итоговые показатели качества анализируемых моделей:

Таблица 2: Сравнение качества классификаторов (по 1 классу), CT – метод Co-Training, жирным выделены топ-3 оценки качества

	Precision	Recall	F1-score	Total F1-score (Class: 0,1)
NaiveBayes	0,53	0,34	0,41	0,59
LogReg	**0,62**	0,36	**0,46**	**0,63**
RandomForest (500 est)	**0,61**	0,32	0,42	0,61
CT:NaiveBayes	0,55	0,24	0,33	0,58
CT:LogReg	0,52	**0,57**	**0,55**	**0,63**
CT:RandomForest (500est)	0,53	0,28	0,37	0,59
CT:NaiveBayes+RandomForest	0,55	0,29	0,38	0,60
CT:NaiveBayes+LogReg	0,52	**0,37**	0,43	0,60
CT:LogReg+RandomForest	**0,55**	**0,42**	**0,48**	**0,63**

Наглядно видно, что при наличии экстремальных условий недостатка обучающих примеров метод Co-Training добивается улучшения качества классификации. Кроме этого, в комбинации различных моделей в 2/3 случаях получено более высокое качество, чем при моделировании с одинаковыми моделями в обёртке метода Co-Training.

Таким образом, для моделирования лояльности клиентов могут быть взяты: логистическая регрессия в обертке метода Co-Training, а также комбинация логистической регрессии и случайного леса. В рамках оценки объема лояльных клиентов бизнес-задачи для менеджмента выберем вариант «CT: LogReg» – комбинацию двух логистических регрессий, т.к. по 2 параметрам классификатор относительно превосходит остальные варианты метода CT.

Результаты моделирования показывают, что 41,21% клиентской базы с вероятностью более 50% можно считать свитчерами, что, по точечной оценке, в номинальном выражении

составляет 1 888 695 человека. Тогда, на базе разработанной модели, при условии стоимости тонны нефтепродуктов в среднем 43 000 руб., плотности нефтепродуктов 760 кг/м3 и возможности получения от каждого клиента добавочных в среднем 9-10 литров за условный период, в зависимости от их охвата, получение следующих дополнительных объемов топлива и финансовых поступлений соответственно может быть оценено следующим образом:

Таблица 3: Распределение оценки финансового результата за условный период от дополнительных маркетинговых мероприятий в зависимости от охвата потенциальных свитчеров программы лояльности

Охват свитчеров, чел.	Потенциальные объемы топлива, л.	Потенциальная масса топлива, тонн.	Финансовая оценка, тыс. руб.
5%	850 131,85	646,10	27 782,31
10%	1 700 263,70	1 292,20	55 564,62
15%	2 550 395,55	1 938,30	83 346,93
20%	3 400 527,40	2 584,40	111 129,24
25%	4 250 659,25	3 230,50	138 911,54
30%	5 100 791,10	3 876,60	166 693,85
35%	5 950 922,95	4 522,70	194 476,16
40%	6 801 054,80	5 168,80	222 258,47
45%	7 651 186,65	5 814,90	250 040,78
50%	8 501 318,50	6 461,00	277 823,09
55%	9 351 450,35	7 107,10	305 605,40
60%	10 201 582,20	7 753,20	333 387,71
65%	11 051 714,05	8 399,30	361 170,02
70%	11 901 845,90	9 045,40	388 952,32
75%	12 751 977,75	9 691,50	416 734,63
80%	13 602 109,60	10 337,60	444 516,94
85%	14 452 241,45	10 983,70	472 299,25
90%	15 302 373,30	11 629,80	500 081,56

Последующий анализ получения дополнительных потенциальных продаж объемов топлива и роста финансового результата соответственно может лежать в области исследования методов моделирования в динамике, в региональном разрезе или разрезе по ряду местных локаций в регионах, форматам станций, привязкой к внешней информации по инфраструктуре и положению конкурентов с целью роста точности оценки свитчеров и дополнительного финансового результата от планируемых маркетинговых акций.

§3.3 Выводы

Таким образом, показано, что метод частичного обучения Co-Training действительно способен осуществлять значимую классификационную оценку клиентов при условии ограниченности размеченных данных на несбалансированных выборках. По результатам исследования поставленную в работе гипотезу нельзя отвергать.

Выявлено, что метод Co-Training с комбинацией различных внутренних моделей повышает качество предсказания. Так, модели, хорошо оценивающие объекты по показателю полноты действительно могут работать вместе с моделями, имеющими высокую точность предсказания, что способно нивелировать или значимо снизить их общие недостатки за счет составления композиции.

Показано, что количество итераций в Co-Training имеет предел с точки зрения повышения качества. Таким образом, для каждой задачи возможно нахождение своего числа итераций, которые позволяют с помощью дополнительной информации улучшить качество классификации.

Кроме того, рассмотрена неоднозначность выбора критериев качества моделирования и опасность применения единой метрики качества модели, в особенности на выборочных данных с несбалансированным числом классов.

Данный метод частичного обучения Co-Training и представленная методология работы с ним на практическом примере могут быть использованы в качестве инструмента в системе оценке поведенческих особенностей клиентов, предсказания их действий или общих тенденций с точки зрения принадлежности к заранее определённым классам в рамках задачи классификации.

Дальнейший анализ самого метода частичного обучения и его производных представляется автором как ряд исследований в области факторного анализа его гиперпараметров, оценки изменения результатов прогноза в зависимости от вариации объёма размеченной информации, числа классификаторов в модели случайного леса и использования иных моделей в рамках представленного метода, а также реализации новых методик оценки выполнения условия работы метода (2.1.1). Кроме того, представляется возможным оценка значимости применения дополнительной обработки данных с помощью различных подходов, связанных с семплированием на данных с несбалансированным количеством классов.

Заключение

В представленной выпускной квалификационной работе:

- Предложена и проанализирована модификация метода частичного обучения Co-Training, которая позволяет эффективнее использовать дополнительную информацию об объектах с целью их исследования в рамках задачи классификации.
- Реализована программа представленного метода на основе статьи разработчиков университета Карнеги-Меллон [14] с его дополнительными модификациями и выбором условий остановки алгоритма на языке Python 3, которая может быть использована во многих системах поддержки принятия решений и работы с клиентами в компаниях.
- Проведены эксперименты на реальных данных нефтяной компании ПАО «Газпром нефть» с условием экстремальной нехватки обучающих данных для моделирования. Полученные результаты экспериментов показали, что нельзя отвергать поставленную в работе гипотезу, а также доказали работоспособность метода в области повышения классификационного качества.
- Получены значимые по качеству результаты оценки лояльности клиентов с точки зрения их дополнительных закупок топлива у конкурентов компании, что доказывает возможную практическую применимость представленного метода в более широком ключе возникающих задач, чем это было представлено его авторами ранее.

Таким образом, в условиях дефицита информации и наличия неструктурированных данных анализируемый в работе метод позволяет не только дать значимые оценки вероятностей поведенческой классификации клиентов автозаправочных станций с качеством предиктора выше случайного гадания, но и повысить эту уверенность анализом работы метода Co-Training на итерациях, тем самым повысить точность прогнозов в области оценки потребительского рынка в целом.

Список использованных источников:

1. Айвазян, С.А. Классификация многомерных наблюдений/ С.А. Айвазян [и др.] // М.: Статистика, 1974.
2. Айвазян, С.А. Прикладная статистика: Классификация и снижение размерности / С.А. Айвазян [и др.] // М.: Финансы и статистика, 1989.
3. Айвазян, С.А. Основы моделирования и первичная обработка данных / С.А. Айвазян [и др.] // М.: Финансы и статистика, 1989.
4. Вапник, В.Н., Червоненкис, А.Я. Теория распознавания образов (статистические проблемы обучения) / В.Н. Вапник, А.Я Червоненкис //М.: Наука. 1974.
5. Воронцов, К. В. Обзор современных исследований по проблеме качества обучения алгоритмов/ К. В. Воронцов // Таврический вестник информатики и математики. – 2004. –№ 1. – С. 5–24
6. Тихонов, А. Н., О некорректных задачах линейной алгебры и устойчивом методе их решения/ А. Н. Тихонов // Докл. АН СССР, 1965, т. 163, № 3, - С. 591—594.
7. Шеннон, К. Работы по теории информации и кибернетике/ К. Шеннон // М.: Изд. иностр. лит., 2002.
8. Bishop, C. Pattern recognition and Machine Learning. Springer Science/ C. Bishop // Business Media, LLC. 2006.
9. Olivier C., Bernhard, S., Alexander, Z, Semi-Supervised Learning (Adaptive Computation and Machine Learning series)/ C. Olivier, S. Bernhard, Z. Alexander, // Cambridge, Mass.: MIT Press, 2006.
10. Óscar M., Gonzalo M., Javier S. A Data Mining & Knowledge Discovery Process Model. In Data Mining and Knowledge Discovery in Real Life Applications/ M. Óscar, M. Gonzalo, S. Javier // Book edited by: Julio Ponce and Adem Karahoca, pp. 438-453, I-Tech, 2009.
11. Quinlan, J.R. Induction of Decision Trees/ J.R. Quinlan // Machine Learning 1: 81-106, 1986.
12. Rosenblatt, F. Principles of Neurodynamic: Perceptrons and the Theory of Brain Mechanisms/ F. Rosenblatt // Washington, DC: Spartan. 1962.
13. Agrawala, A. K. Learning with a probabilistic teacher/ A. K. Agrawala // IEEE Transactions on Information Theory.1970. 16, pp. 373–379.
14. Blum, A., Mitchell, T. Combining Labeled and Unlabeled Data with Co-Training/ A. Blum, T. Mitchell // Proceedings of the Eleventh Annual Conference on Computational Learning theory.1998. COLT, 92-100.
15. Breiman L. Random forests/ L. Breiman // Machine Learning. 2001. Vol. 45(1), pp. 5–32.
16. de Brebisson, A., Vincent, P. An exploration of Softmax alternatives belonging to the spherical loss family/ A. de Brebisson, P. Vincent // MILA, D´epartement dInformatique et de Recherche Op´erationnelle, University of Montr´eal. ICLR. 2016.
17. Cohen, J. A coefficient of agreement for nominal scales/ J. Cohen // Educational and Psychological Measurement. 20(1):37-46.1960.
18. Chao D., Zu Guo, M. A new co-training-style random forest for computer aided diagnosis/ D. Chao, M. Zu Guo // Journal of Intelligent Information Systems. Vol. 36(3), pp 253–281.
19. Du, J., Ling, C. X., Zhou, Z.-H. When does co-training work in real data?/ J. Du, , C. X. Ling, Z.-H Zhou // IEEE Transactions on Knowledge and Data Engineering, 2011. 23(5). pp. 788-799.

20. Fawcett, T. An Introduction to ROC Analysis/ T. Fawcett, // Pattern Recognition Letters. 2006. 27 (8), pp 861–874.

21. Tsoumakas G., Katakis, I. Multi-Label Classification: An Overview/ G. Tsoumakas, I. Katakis, // International Journal of Data Warehousing & Mining. 2007. 3(3), pp. 1-10.

22. Huang, KY. A heuristic approach to classifying labeled/unlabeled data sets/ KY. Huang // The Journal of the Operational Research Society.2012. Vol. 63, No. 9 pp. 1248-1257.

23. Hughes, G.F. On the mean accuracy of statistical pattern recognizers/ G.F. Hughes // IEEE Transactions on Information Theory.1968. 14 (1), pp. 55–63.

24. Kennedy, B. Namee M. , Delany, SJ. Using semi-supervised classifiers for credit scoring/ B. Kennedy, M. Namee, SJ. Delany, // Journal of the Operational Research Society. 2013. Vol. 64, pp. 513-529.

25. Mika S., Fisher Discriminant Analysis with Kernels. Neural Networks for Signal Processing/ S. Mika // IX, 1999: Proceedings of the 1999 IEEE Signal Processing Society Workshop, IEEE. 1999. pp. 41-48.

26. Nigam, K., Ghani, R. Analyzing the effectiveness and applicability of co-training/ K. Nigam, R. Ghani // In: Proceedings of the 9th International Conference on Information and Knowledge Management.2000.

27. Shen, D., Zhang, J., Su, J., Zhou, G., Tan, C.-L. A collaborative ability measurement for co-training/ D. Shen, J. Zhang, J. Su, G. Zhou, C.-L. Tan, // IX, 1999: Proceedings of the 1999 IEEE Signal Processing Society Workshop 2005.

28. Scudder, H. J. Probability of error of some adaptive pattern-recognition machines/ H. J. Scudder // IEEE. Transactions on Information Theory. 1965. 11, pp. 363–371.

29. Wang, W., Zhou, Z.-H. Co-training with insufficient views/ W. Wang, Z.-H. Zhou // Proceedings of the 5th Asian Conference on Machine Learning, Canberra, Australia. JMLR: W&CP 29. 2013. pp. 467-482.

30. Wang, W., Zhou, Z.-H. A new analysis of co-training/ W. Wang, , Z.-H. Zhou // In: Proceedings of the 27th International Conference on Machine Learning. Haifa, Israel, 2010, pp. 1135-1142.

31. Ya, X., Justin, S. D., Art B. Owen. Empirical stationary correlations for semi-supervised learning on graphs/ X. Ya, S. D. Justin, B. Owen Art. // The Annals of Applied Statistics.2010. Vol. 4, No. 2 pp. 589-614.

32. Zhou, Z.-H., Zhan, D.-C., Q. Yang. Semi-supervised learning with very few labeled training examples/ Z.-H. Zhou, D.-C. Zhan, Q. Yang // In: Proceedings of the 22nd AAAI Conference on Artificial Intelligence, Vancouver, Canada. 2007. pp. 675-680.

33. Center for Machine Learning and Intelligent Systems: UCI Machine Learning Repository [Электронный ресурс]. URL: https://archive.ics.uci.edu/ml/. (Дата обращения: 11.11.2017).

34. Scikit-Learn. Machine Learning in Python. [Электронный ресурс]. URL: http://scikit-learn.org/stable/index.html#. (Дата обращения: 10.09.2017).

35. Statmodels. Statistics in Python. [Электронный ресурс]. URL: http://www.statsmodels.org/stable/index.html. (Дата обращения: 10.09.2017).

36. What is the internet of things? [Электронный ресурс]. URL: https://www.theguardian.com/technology/2015/may/06/what-is-the-internet-of-things-google. (Дата обращения: 1.08.2017).

Приложение

1. Результаты эконометрического моделирования: проверка статистической значимости клиентских признаков по модели логистической регрессии.

   Feature view 1:

```
Optimization terminated successfully.
         Current function value: 0.660456
         Iterations 4
                         Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.017
Dependent Variable: SWITCHER         AIC:              9074.1743
Date:               2018-02-23 21:12 BIC:              9135.6702
No. Observations:   6856             Log-Likelihood:   -4528.1
Df Model:           8                LL-Null:          -4604.1
Df Residuals:       6847             LLR p-value:      7.4559e-29
Converged:          1.0000           Scale:            1.0000
No. Iterations:     4.0000
----------------------------------------------------------------
          Coef.    Std.Err.      z       P>|z|     [0.05     0.95]
----------------------------------------------------------------
const    -0.4256     0.0250   -17.0381   0.0000   -0.4667   -0.3845
x1        0.2895     0.0249    11.6465   0.0000    0.2486    0.3303
x2        0.0657     0.0253     2.6035   0.0092    0.0242    0.1073
x3       -0.0298     0.0319    -0.9356   0.3495   -0.0822    0.0226
x4       -0.0244     0.0324    -0.7530   0.4515   -0.0778    0.0289
x5       -0.0225     0.0324    -0.6944   0.4874   -0.0759    0.0308
x6       -0.0624     0.0326    -1.9131   0.0557   -0.1161   -0.0088
x7       -0.0610     0.0417    -1.4644   0.1431   -0.1295    0.0075
x8       -0.0308     0.0341    -0.9039   0.3660   -0.0869    0.0253
================================================================
```

Feature view 2:

```
Optimization terminated successfully.
         Current function value: 0.663069
         Iterations 5
                         Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.013
Dependent Variable: SWITCHER         AIC:              9108.0029
Date:               2018-02-23 21:12 BIC:              9162.6660
No. Observations:   6856             Log-Likelihood:   -4546.0
Df Model:           7                LL-Null:          -4604.1
Df Residuals:       6848             LLR p-value:      4.7688e-22
Converged:          1.0000           Scale:            1.0000
No. Iterations:     5.0000
----------------------------------------------------------------
          Coef.    Std.Err.      z       P>|z|     [0.05     0.95]
----------------------------------------------------------------
const    -0.4270     0.0250   -17.1133   0.0000   -0.4680   -0.3860
x1       -0.0504     0.0253    -1.9945   0.0461   -0.0920   -0.0088
x2       -0.1937     0.0281    -6.8903   0.0000   -0.2400   -0.1475
x3       -0.0535     0.0256    -2.0922   0.0364   -0.0956   -0.0114
x4       -0.1045     0.0259    -4.0393   0.0001   -0.1470   -0.0619
x5        0.1342     0.0311     4.3165   0.0000    0.0830    0.1853
x6        0.0034     0.0268     0.1264   0.8994   -0.0407    0.0475
x7       -0.0099     0.0274    -0.3608   0.7182   -0.0549    0.0352
================================================================
```

2. Листинг программы, реализующей метод Co-Training на языке Python 3

Необходимые библиотеки для работы: re, math, random, copy, pandas, numpy, scipy, sklearn.

Листинг программы:

```
#Функция проверки качества моделирования на итерации
def CoTrain_Classifier_Test(dataframe_test, X1, X2, clf1, clf2, y_true = None):

    if y_true is None:
        # Предобработка данных
        X1_test_ = dataframe_test[X1]
        X2_test_ = dataframe_test[X2]
        X1_test = np.asarray(X1_test_).reshape(len(X1_test_), len(X1))
        X2_test = np.asarray(X2_test_).reshape(len(X2_test_), len(X2))

        proba_clf1_test = clf1.predict_proba(X1_test)
        proba_clf2_test = clf2.predict_proba(X2_test)
        aggr_proba_test = (proba_clf1_test + proba_clf2_test)/2

        clss_test = []
        for i in range(0, len(aggr_proba_test)):

            # Максимум вероятности из предсказанных вероятностей
            if aggr_proba_test[i][1] > aggr_proba_test[i][0]:
                clss_test.append(1)
            else:
                clss_test.append(0)

        return clss_test  #Вернуть только предсказанные значения модели

    elif y_true is not None:
        # Предобработка данных
        X1_test_ = dataframe_test[X1]
        X2_test_ = dataframe_test[X2]
        X1_test = np.asarray(X1_test_).reshape(len(X1_test_), len(X1))
        X2_test = np.asarray(X2_test_).reshape(len(X2_test_), len(X2))

        proba_clf1_test = clf1.predict_proba(X1_test)
        proba_clf2_test = clf2.predict_proba(X2_test)
        aggr_proba_test = proba_clf1_test * proba_clf2_test

        clss_test = []
        for i in range(0, len(aggr_proba_test)):
```

```
            # Максимум вероятности из предсказанных вероятностей
            if aggr_proba_test[i][1] > aggr_proba_test[i][0]:
                clss_test.append(1)
            else:
                clss_test.append(0)

        f_measure_it = sk.metrics.f1_score(y_true, clss_test, average=weighted)
        return f_measure_it  #Вернуть качество модели

#Co-Training метод
def CoTrain_Classifier(dataframe, X1, X2, y,
                       clf_1, clf_2=None,
                       method =convergency,
                       conv_mark = aggr_proba_max,
                       th_step = 0.001,
                       co_train_k=5,
                       aggr_proba_th = 0.3,
                       p_const = 100,
                       n = None, p = None,
                       u_marker=999,
                       objs_keys=[],
                       dataframe_test=None,
                       y_data_test = None):

    #////////////Параметры модели частичного обучения Co-Training//////////
    #Для работы алгоритма необходимы библиотеки: pandas, numpy, sklearn
    #////////////////////////////////////////////////////////////////////////////////////
    #dataframe (pandas) - таблица данных для обучения
    #X1 (list), str - список полей для 1 обучения
    #X2 (list), str - список полей для 2 обучения
    #y (pandas) - вектор разметки объектов
    #clf_1, clf_2 - классификаторы в модели, по умолчанию clf_2 = clf_1
    #method: подметод классификации:
        #>k_iters - число итераций задано
            #>co_train_k - заданное число итераций
        #>convergency - до полной сходимости
            #>conv_mark: разметка для конечной сходимости, если method =
        convergency
                #>aggr_proba_max - единоразовая разметка с максимумом возможной
        кооперативной вероятности
                #>proba_th_min - с постепенной минимизацией критерия уверенной
        классификации
                    #>th_step - шаг снижения вероятности при несходимости на
        итерации, по умолчанию 0.001
    #aggr_proba_th - уровень уверенной кооперативной классификации объектов
    #n, p -число уверенных разметок за итерацию
```

```
#u_marker – псевдоразметка объектов в y для классификации, по умолчанию
    999
#objs_keys (pandas) – привязка к дополнительным номинальным меткам
    объектов для БД, по умолчанию None
#dataframe_test, y_true – таблица тестовых данных и соответствующий вектор
    классов, по умолчанию отключено


#Задание классификаторов для обучения
clf1 = clf_1
if clf_2 is None:

    clf2 = copy.copy(clf1)
else:
    clf2 = clf_2
#Поиск индексов размеченных и неразмеченных объектов, по умолчанию
    u_marker=999 в y
U = [i for i, y_i in enumerate(y) if y_i == u_marker]
L = [i for i, y_i in enumerate(y) if y_i != u_marker]

y_tmp_ = y.iloc[L].copy()

df_X1_L_tmp_ = dataframe[X1].iloc[L].copy()
df_X2_L_tmp_ = dataframe[X2].iloc[L].copy()

df_X1_U_tmp_ = dataframe[X1].iloc[U].copy()
df_X2_U_tmp_ = dataframe[X2].iloc[U].copy()

y_tmp = np.asarray(y_tmp_).reshape(len(y_tmp_), 1)
y_tmp_prob = copy.copy(y_tmp)

df_X1_L_tmp = np.asarray(df_X1_L_tmp_).reshape(len(df_X1_L_tmp_), len(X1))
df_X2_L_tmp = np.asarray(df_X2_L_tmp_).reshape(len(df_X2_L_tmp_), len(X2))

df_X1_U_tmp = np.asarray(df_X1_U_tmp_).reshape(len(df_X1_U_tmp_), len(X1))
df_X2_U_tmp = np.asarray(df_X2_U_tmp_).reshape(len(df_X1_U_tmp_), len(X2))

#Привязка к дополнительным номинальным меткам объектов для БД, по
    умолчанию None
if len(objs_keys) == 0:
    print(objs_keys = None, Ключи для БД не заданы)

else:
    print(objs_keys, Ключи для БД заданы)
    objs_keys_L_tmp_ = objs_keys.iloc[L].copy()
    objs_keys_U_tmp_ = objs_keys.iloc[U].copy()
```

```
    objs_keys_L_tmp = np.asarray(objs_keys_L_tmp_).reshape(len(objs_keys_L_tmp_), 1)
    objs_keys_U_tmp = np.asarray(objs_keys_U_tmp_).reshape(len(objs_keys_U_tmp_), 1)

if method ==convergency:
    print(method = , method)
    cotrain_iters = 0
    print(40*/)
    print(40*/)
    unlabeled_rows = len(df_X1_U_tmp) #Общее число неразмеченных строк
    unlabeled_rows_hist = [unlabeled_rows+1]        #Для проверки сходимости
     алгоритма

    while unlabeled_rows > 0:
        cotrain_iters += 1
        unlabeled_rows_hist.append(unlabeled_rows)
        conv_indicator = unlabeled_rows_hist[cotrain_iters-1] -
     unlabeled_rows_hist[cotrain_iters]
        print(conv indi = , conv_indicator)
        print(aggr_proba_th = , aggr_proba_th)

        if conv_indicator > 0 :

            print(40 * /,₩n,40 * /)
            print(Co-train iters = , cotrain_iters)
            print(40 * /,₩n,40 * /)



            # Обучение
            clf1.fit(df_X1_L_tmp, y_tmp.ravel())
            clf2.fit(df_X2_L_tmp, y_tmp.ravel())

            model_qual_t0 = CoTrain_Classifier_Test(dataframe_test, X1, X2, clf1, clf2,
     y_data_test)

            # Оценка вероятностей классов
            proba_clf1 = clf1.predict_proba(df_X1_U_tmp)
            proba_clf2 = clf2.predict_proba(df_X2_U_tmp)
            unmkd_tmp = len(df_X1_U_tmp)

            aggr_proba = (proba_clf1 + proba_clf2)/2

            pos_aggr_prob_ind = np.asarray(np.where(aggr_proba[ :, 1] >
     aggr_proba_th)).ravel()
```

```
        neg_aggr_prob_ind = np.asarray(np.where(aggr_proba[ :, 0] >
aggr_proba_th)).ravel()

        randomize_p = np.arange(len(pos_aggr_prob_ind))
        randomize_n = np.arange(len(neg_aggr_prob_ind))

        np.random.shuffle(randomize_p)
        np.random.shuffle(randomize_n)
        pos_aggr_prob_ind = pos_aggr_prob_ind[randomize_p]
        neg_aggr_prob_ind = neg_aggr_prob_ind[randomize_n]

        neg_aggr_prob_ind= neg_aggr_prob_ind[:len(pos_aggr_prob_ind)]


        conf_examples = np.concatenate((pos_aggr_prob_ind,  neg_aggr_prob_ind))

        y_clss = []        # размеченный класс
        y_clss_proba = []  # вероятность принадлежности к 1 классу


        unmkd = unmkd_tmp - len(conf_examples)
        print(Разметка данных)
        print(Неразмеченных объектов= , unmkd)

        # добавление переменных по меткам классов
        df_X1_L_tmp_ch = copy.copy(np.vstack([df_X1_L_tmp,
df_X1_U_tmp[conf_examples]]))
        df_X2_L_tmp_ch = copy.copy(np.vstack([df_X2_L_tmp,
df_X2_U_tmp[conf_examples]]))

        y_clss = np.asarray(y_clss)
        y_clss = y_clss.reshape(len(y_clss), 1)
        y_clss_proba = np.asarray(y_clss_proba)
        y_clss_proba = y_clss_proba.reshape(len(y_clss_proba), 1)

        y_clss_p = np.ones( shape =(len(pos_aggr_prob_ind),1), dtype = int)
        y_clss_n = np.zeros( shape =(len(neg_aggr_prob_ind),1), dtype = int)

        #итоговые массивы для добавления
        y_clss = np.vstack([y_clss_p, y_clss_n])
        y_clss_proba_norm = aggr_proba[conf_examples,
1]/(aggr_proba[conf_examples, 1]+aggr_proba[conf_examples, 0])
        y_clss_proba = np.vstack((y_clss_proba,
y_clss_proba_norm.reshape(len(y_clss_proba_norm),1)))

        #Приведение предыдущих массивов к единому виду
```

```
        y_tmp = np.asarray(y_tmp)
        y_tmp = y_tmp.reshape(len(y_tmp), 1)

        y_tmp_prob = np.asarray(y_tmp_prob)
        y_tmp_prob = y_tmp_prob.reshape(len(y_tmp_prob), 1)

        # добавление меток класса
        y_tmp_ch = copy.copy(np.vstack([y_tmp, y_clss]))
        y_tmp_prob_ch = copy.copy(np.vstack([y_tmp_prob, y_clss_proba]))

        # корректировка неразмеченных массивов
        # backup
        df_X1_U_tmp_backup = df_X1_U_tmp.copy()
        df_X2_U_tmp_backup = df_X2_U_tmp.copy()

        df_X1_U_tmp_ch = np.delete(df_X1_U_tmp, conf_examples, axis=0)
        df_X2_U_tmp_ch = np.delete(df_X2_U_tmp, conf_examples, axis=0)

        #Автокоррекция подгонки модели
        clf1.fit(df_X1_L_tmp_ch, y_tmp_ch.ravel())
        clf2.fit(df_X2_L_tmp_ch, y_tmp_ch.ravel())

        model_qual_t1 = CoTrain_Classifier_Test(dataframe_test, X1, X2, clf1, clf2,
y_data_test)
        print(Качество до текущей разметки:, {0:.5f}.format(model_qual_t0))
        print(Качество после текущей разметки:, {0:.5f}.format(model_qual_t1))
        #print(len(df_X2_U_tmp_ch))
        #print(len(df_X1_U_tmp_backup))

        print(Согласованных объектов для разметки: ,len(conf_examples))
        print(Positive: , pos_aggr_prob_ind)
        print(Negative: , neg_aggr_prob_ind)
        print(conf_examples)
        if model_qual_t1 > model_qual_t0:
            GREEN_LIGHT = 1

            #k+=1
            print(Качество повышено на ,model_qual_t1 - model_qual_t0, , объекты
размечаются)
            #Можно объявить массивы основными данными для подгонки
            y_tmp = y_tmp_ch
            y_tmp_prob = y_tmp_prob_ch

            df_X1_L_tmp = df_X1_L_tmp_ch
            df_X2_L_tmp = df_X2_L_tmp_ch
```

```
            df_X1_U_tmp = df_X1_U_tmp_ch
            df_X2_U_tmp = df_X2_U_tmp_ch

        else:
            #k+=1 # убрать
            GREEN_LIGHT = 0

            df_X1_U_tmp = df_X1_U_tmp_backup
            df_X2_U_tmp = df_X2_U_tmp_backup
            objs_keys_U_tmp

            randomize = np.arange(len(df_X1_U_tmp))
            np.random.shuffle(randomize)

            df_X1_U_tmp = df_X1_U_tmp[randomize]
            df_X2_U_tmp = df_X2_U_tmp[randomize]
            objs_keys_U_tmp = objs_keys_U_tmp[randomize]

            print(Качество снижено, объекты не размечаются)


        unlabeled_rows = len(df_X1_U_tmp)


        if len(objs_keys) == 0:
            print(objs_keys is None, Ключи для БД не заданы)
        else:
            print(objs_keys is not None, Добавление ключей БД)
            if GREEN_LIGHT == 1:
                objs_keys_L_tmp = np.vstack([objs_keys_L_tmp,
objs_keys_U_tmp[conf_examples]])
                objs_keys_U_tmp = np.delete(objs_keys_U_tmp, conf_examples, axis=0)
            else:
                print(Добавление ключей БД отменено, качество модели не
повышено)

        print(Неразмечено объектов = , unlabeled_rows)
        print(Размечено объектов = , len(df_X1_L_tmp))
        #print(Меток поставлено = , len(y_tmp))


    elif conv_indicator ==0:

        if conv_mark == aggr_proba_max:
            print(Корректировка сходимости, метод:,conv_mark )
            #1 – единоразовая разметка
```

```
        proba_clf1 = clf1.predict_proba(df_X1_U_tmp)
        proba_clf2 = clf2.predict_proba(df_X2_U_tmp)
        aggr_proba = (proba_clf1 + proba_clf2)/2

        y_conv_1_proba = np.asarray([])
        y_conv_1_proba = y_conv_1_proba.reshape(len(y_conv_1_proba), 1)

        logic = aggr_proba[:,1]> aggr_proba[:,0] #True – класс 1, False – класс 0

        pos = np.where(logic==True)
        neg = np.where(logic==False)
        all_ind = np.concatenate((pos[0],  neg[0]))

        y_clss_p = np.ones( shape =(len(pos[0]),1), dtype = int)
        y_clss_n = np.zeros( shape =(len(neg[0]),1), dtype = int)

        y_conv_1 = np.vstack([y_clss_p, y_clss_n])
        y_conv_proba_norm = aggr_proba[all_ind, 1]/(aggr_proba[all_ind, 1]+
aggr_proba[all_ind, 0])
        y_conv_1_proba = np.vstack((y_conv_1_proba,
y_conv_proba_norm.reshape(len(y_conv_proba_norm),1)))


        # Добавление переменных по меткам классов
        df_X1_L_tmp = np.vstack([df_X1_L_tmp, df_X1_U_tmp[all_ind]])
        df_X2_L_tmp = np.vstack([df_X2_L_tmp, df_X2_U_tmp[all_ind]])


        y_clss = np.asarray(y_conv_1)
        y_clss = y_clss.reshape(len(y_clss), 1)

        y_clss_proba = np.asarray(y_conv_1_proba)
        y_clss_proba = y_clss_proba.reshape(len(y_clss_proba), 1)

        y_tmp = np.asarray(y_tmp)
        y_tmp = y_tmp.reshape(len(y_tmp), 1)

        y_tmp_prob = np.asarray(y_tmp_prob)
        y_tmp_prob = y_tmp_prob.reshape(len(y_tmp_prob), 1)

        y_tmp = np.vstack([y_tmp, y_clss])
        y_tmp_prob = np.vstack([y_tmp_prob, y_clss_proba])

        df_X1_U_tmp = np.delete(df_X1_U_tmp, all_ind, axis=0)
        df_X2_U_tmp = np.delete(df_X2_U_tmp, all_ind, axis=0)
```

```
            unlabeled_rows = len(df_X1_U_tmp)

            if len(objs_keys) == 0:
                print(objs_keys is None, Ключи для БД не заданы)
            else:
                print(objs_keys is not None, Добавление ключей БД)
                objs_keys_L_tmp = np.vstack([objs_keys_L_tmp, objs_keys_U_tmp])



            print(Неразмечено объектов = , unlabeled_rows)
            #print(Размечено объектов = , len(df_X1_L_tmp))
            print(Меток поставлено = , len(y_tmp))

        #2 - снижения границы aggr_proba_th

        elif conv_mark ==proba_th_min:
            print(Корректировка сходимости, заданная вероятность снижена на:
,th_step )
            aggr_proba_th = aggr_proba_th - th_step #смягчаем условие

            print(40 * /, ₩n, 40 * /)
            print(Co-train iters = , cotrain_iters)
            print(40 * /, ₩n, 40 * /)

            # Обучение
            clf1.fit(df_X1_L_tmp, y_tmp.ravel())
            clf2.fit(df_X2_L_tmp, y_tmp.ravel())

            model_qual_t0 = CoTrain_Classifier_Test(dataframe_test, X1, X2, clf1, clf2,
y_data_test)

            # Оценка вероятностей классов
            proba_clf1 = clf1.predict_proba(df_X1_U_tmp)
            proba_clf2 = clf2.predict_proba(df_X2_U_tmp)
            unmkd_tmp = len(df_X1_U_tmp)

            aggr_proba = (proba_clf1 + proba_clf2)/2

            pos_aggr_prob_ind = np.asarray(np.where(aggr_proba[ :, 1] >
aggr_proba_th)).ravel()
            neg_aggr_prob_ind = np.asarray(np.where(aggr_proba[ :, 0] >
aggr_proba_th)).ravel()

            randomize_p = np.arange(len(pos_aggr_prob_ind))
            randomize_n = np.arange(len(neg_aggr_prob_ind))
```

```
        np.random.shuffle(randomize_p)
        np.random.shuffle(randomize_n)
        pos_aggr_prob_ind = pos_aggr_prob_ind[randomize_p]
        neg_aggr_prob_ind = neg_aggr_prob_ind[randomize_n]

        neg_aggr_prob_ind= neg_aggr_prob_ind[:len(pos_aggr_prob_ind)]


        conf_examples = np.concatenate((pos_aggr_prob_ind,  neg_aggr_prob_ind))

        y_clss = [ ]        # Размеченный класс
        y_clss_proba = [ ]  # Вероятность принадлежности к 1 классу


        unmkd = unmkd_tmp – len(conf_examples)
        print(Разметка данных)
        print(Неразмеченных объектов= , unmkd)

        # добавление переменных по меткам классов
        df_X1_L_tmp_ch = copy.copy(np.vstack([df_X1_L_tmp,
df_X1_U_tmp[conf_examples]]))
        df_X2_L_tmp_ch = copy.copy(np.vstack([df_X2_L_tmp,
df_X2_U_tmp[conf_examples]]))

        y_clss = np.asarray(y_clss)
        y_clss = y_clss.reshape(len(y_clss), 1)
        y_clss_proba = np.asarray(y_clss_proba)
        y_clss_proba = y_clss_proba.reshape(len(y_clss_proba), 1)

        y_clss_p = np.ones( shape =(len(pos_aggr_prob_ind),1), dtype = int)
        y_clss_n = np.zeros( shape =(len(neg_aggr_prob_ind),1), dtype = int)

        #Итоговые массивы для добавления
        y_clss = np.vstack([y_clss_p, y_clss_n])
        y_clss_proba_norm = aggr_proba[conf_examples,
1]/(aggr_proba[conf_examples, 1]+aggr_proba[conf_examples, 0])
        y_clss_proba = np.vstack((y_clss_proba,
y_clss_proba_norm.reshape(len(y_clss_proba_norm),1)))

        #Приведение предыдущих массивов к единому виду
        y_tmp = np.asarray(y_tmp)
        y_tmp = y_tmp.reshape(len(y_tmp), 1)

        y_tmp_prob = np.asarray(y_tmp_prob)
        y_tmp_prob = y_tmp_prob.reshape(len(y_tmp_prob), 1)
```

```
        # Добавление меток класса
        y_tmp_ch = copy.copy(np.vstack([y_tmp, y_clss]))
        y_tmp_prob_ch = copy.copy(np.vstack([y_tmp_prob, y_clss_proba]))

        # Корректировка неразмеченных массивов
        # backup
        df_X1_U_tmp_backup = df_X1_U_tmp.copy()
        df_X2_U_tmp_backup = df_X2_U_tmp.copy()

        df_X1_U_tmp_ch = np.delete(df_X1_U_tmp, conf_examples, axis=0)
        df_X2_U_tmp_ch = np.delete(df_X2_U_tmp, conf_examples, axis=0)

        #Автокоррекция подгонки модели
        clf1.fit(df_X1_L_tmp_ch, y_tmp_ch.ravel())
        clf2.fit(df_X2_L_tmp_ch, y_tmp_ch.ravel())

        model_qual_t1 = CoTrain_Classifier_Test(dataframe_test, X1, X2, clf1, clf2,
y_data_test)
        print(Качество до текущей разметки:, {0:.5f}.format(model_qual_t0))
        print(Качество после текущей разметки:, {0:.5f}.format(model_qual_t1))
        #print(len(df_X2_U_tmp_ch))
        #print(len(df_X1_U_tmp_backup))

        print(Согласованных объектов для разметки: ,len(conf_examples))
        print(Positive: , pos_aggr_prob_ind)
        print(Negative: , neg_aggr_prob_ind)
        print(conf_examples)
        if model_qual_t1 > model_qual_t0:
            GREEN_LIGHT = 1

            #k+=1
            print(Качество повышено на ,model_qual_t1 - model_qual_t0, ,
объекты размечаются)
            #Можно объявить массивы основными данными для подгонки
            y_tmp = y_tmp_ch
            y_tmp_prob = y_tmp_prob_ch

            df_X1_L_tmp = df_X1_L_tmp_ch
            df_X2_L_tmp = df_X2_L_tmp_ch

            df_X1_U_tmp = df_X1_U_tmp_ch
            df_X2_U_tmp = df_X2_U_tmp_ch

        else:
            #k+=1 # убрать
```

```
                GREEN_LIGHT = 0

                df_X1_U_tmp = df_X1_U_tmp_backup
                df_X2_U_tmp = df_X2_U_tmp_backup
                objs_keys_U_tmp

                randomize = np.arange(len(df_X1_U_tmp))
                np.random.shuffle(randomize)

                df_X1_U_tmp = df_X1_U_tmp[randomize]
                df_X2_U_tmp = df_X2_U_tmp[randomize]
                objs_keys_U_tmp = objs_keys_U_tmp[randomize]

                print(Качество снижено, объекты не размечаются)


            unlabeled_rows = len(df_X1_U_tmp)


            if len(objs_keys) == 0:
                print(objs_keys is None, Ключи для БД не заданы)
            else:
                print(objs_keys is not None, Добавление ключей БД)
                if GREEN_LIGHT == 1:
                    objs_keys_L_tmp = np.vstack([objs_keys_L_tmp,
    objs_keys_U_tmp[conf_examples]])
                    objs_keys_U_tmp = np.delete(objs_keys_U_tmp, conf_examples,
    axis=0)
                else:
                    print(Добавление ключей БД отменено, качество модели не
    повышено)

            print(Неразмечено объектов = , unlabeled_rows)
            print(Размечено объектов = , len(df_X1_L_tmp))
            #print(Меток поставлено = , len(y_tmp))


elif method == k_iters:
    print(Метод  = , method)
    print(Предельная вероятность = , aggr_proba_th)

    #k=0
    for i in range(1,co_train_k+1):

        print(40 * /, ₩n, 40 * /)
        print(Итерация CoTrain = , i)
```

```
    print(40 * /, ₩n, 40 * /)

    if len(df_X1_U_tmp)>0:
        # Обучение
        clf1.fit(df_X1_L_tmp, y_tmp.ravel())
        clf2.fit(df_X2_L_tmp, y_tmp.ravel())

        model_qual_t0 = CoTrain_Classifier_Test(dataframe_test, X1, X2, clf1, clf2,
y_data_test)

        # Оценка вероятностей классов
        proba_clf1 = clf1.predict_proba(df_X1_U_tmp[0:p_const])
        proba_clf2 = clf2.predict_proba(df_X2_U_tmp[0:p_const])
        unmkd_tmp = len(df_X1_U_tmp)

        aggr_proba = (proba_clf1 + proba_clf2)/2

        pos_aggr_prob_ind = np.asarray(np.where(aggr_proba[ :, 1] >
aggr_proba_th)).ravel()
        neg_aggr_prob_ind = np.asarray(np.where(aggr_proba[ :, 0] >
aggr_proba_th)).ravel()

        randomize_p = np.arange(len(pos_aggr_prob_ind))
        randomize_n = np.arange(len(neg_aggr_prob_ind))

        np.random.shuffle(randomize_p)
        np.random.shuffle(randomize_n)
        pos_aggr_prob_ind = pos_aggr_prob_ind[randomize_p]
        neg_aggr_prob_ind = neg_aggr_prob_ind[randomize_n]

        pos_aggr_prob_ind = pos_aggr_prob_ind[ :p]
        neg_aggr_prob_ind= neg_aggr_prob_ind[:n]


        conf_examples = np.concatenate((pos_aggr_prob_ind,  neg_aggr_prob_ind))

        y_clss = [ ]        # Размеченный класс
        y_clss_proba = [ ]  # Вероятность принадлежности к 1 классу


        unmkd = unmkd_tmp - len(conf_examples)
        print(Разметка данных)
        print(Неразмеченных объектов= , unmkd)

        # Добавление переменных по меткам классов
```

```
        df_X1_L_tmp_ch = copy.copy(np.vstack([df_X1_L_tmp,
df_X1_U_tmp[conf_examples]]))
        df_X2_L_tmp_ch = copy.copy(np.vstack([df_X2_L_tmp,
df_X2_U_tmp[conf_examples]]))

        y_clss = np.asarray(y_clss)
        y_clss = y_clss.reshape(len(y_clss), 1)
        y_clss_proba = np.asarray(y_clss_proba)
        y_clss_proba = y_clss_proba.reshape(len(y_clss_proba), 1)

        y_clss_p = np.ones( shape =(len(pos_aggr_prob_ind),1), dtype = int)
        y_clss_n = np.zeros( shape =(len(neg_aggr_prob_ind),1), dtype = int)

        #Итоговые массивы для добавления
        y_clss = np.vstack([y_clss_p, y_clss_n])
        y_clss_proba_norm = aggr_proba[conf_examples,
1]/(aggr_proba[conf_examples, 1]+aggr_proba[conf_examples, 0])
        y_clss_proba = np.vstack((y_clss_proba,
y_clss_proba_norm.reshape(len(y_clss_proba_norm),1)))

        #Приведение предыдущих массивов к единому виду
        y_tmp = np.asarray(y_tmp)
        y_tmp = y_tmp.reshape(len(y_tmp), 1)

        y_tmp_prob = np.asarray(y_tmp_prob)
        y_tmp_prob = y_tmp_prob.reshape(len(y_tmp_prob), 1)

        # Добавление меток класса
        y_tmp_ch = copy.copy(np.vstack([y_tmp, y_clss]))
        y_tmp_prob_ch = copy.copy(np.vstack([y_tmp_prob, y_clss_proba]))

        # Корректировка неразмеченных массивов
        # backup
        df_X1_U_tmp_backup = df_X1_U_tmp.copy()
        df_X2_U_tmp_backup = df_X2_U_tmp.copy()

        df_X1_U_tmp_ch = np.delete(df_X1_U_tmp, conf_examples, axis=0)
        df_X2_U_tmp_ch = np.delete(df_X2_U_tmp, conf_examples, axis=0)

        #Автокоррекция подгонки модели
        clf1.fit(df_X1_L_tmp_ch, y_tmp_ch.ravel())
        clf2.fit(df_X2_L_tmp_ch, y_tmp_ch.ravel())

        model_qual_t1 = CoTrain_Classifier_Test(dataframe_test, X1, X2, clf1, clf2,
y_data_test)
        print(Качество до текущей разметки:, {0:.5f}.format(model_qual_t0))
```

```
        print(Качество после текущей разметки:, {0:.5f}.format(model_qual_t1))
        #print(len(df_X2_U_tmp_ch))
        #print(len(df_X1_U_tmp_backup))

        print(Согласованных объектов для разметки: ,len(conf_examples))
        print(Positive: , pos_aggr_prob_ind)
        print(Negative: , neg_aggr_prob_ind)
        print(conf_examples)
        if model_qual_t1 > model_qual_t0:
            GREEN_LIGHT = 1

            #k+=1
            print(Качество повышено на ,model_qual_t1 - model_qual_t0, , объекты
размечаются)
            #Можно объявить массивы основными данными для подгонки
            y_tmp = y_tmp_ch
            y_tmp_prob = y_tmp_prob_ch

            df_X1_L_tmp = df_X1_L_tmp_ch
            df_X2_L_tmp = df_X2_L_tmp_ch

            df_X1_U_tmp = df_X1_U_tmp_ch
            df_X2_U_tmp = df_X2_U_tmp_ch

        else:
            #k+=1 # убрать
            GREEN_LIGHT = 0

            df_X1_U_tmp = df_X1_U_tmp_backup
            df_X2_U_tmp = df_X2_U_tmp_backup
            objs_keys_U_tmp

            randomize = np.arange(len(df_X1_U_tmp))
            np.random.shuffle(randomize)

            df_X1_U_tmp = df_X1_U_tmp[randomize]
            df_X2_U_tmp = df_X2_U_tmp[randomize]
            objs_keys_U_tmp = objs_keys_U_tmp[randomize]

            print(Качество снижено, объекты не размечаются)


        unlabeled_rows = len(df_X1_U_tmp)


        if len(objs_keys) == 0:
```

```
            print(objs_keys is None, Ключи для БД не заданы)
        else:
            print(objs_keys is not None, Добавление ключей БД)
            if GREEN_LIGHT == 1:
                objs_keys_L_tmp = np.vstack([objs_keys_L_tmp,
objs_keys_U_tmp[conf_examples]])
                objs_keys_U_tmp = np.delete(objs_keys_U_tmp, conf_examples, axis=0)
            else:
                print(Добавление ключей БД отменено, качество модели не
повышено)

        print(Неразмечено объектов = , unlabeled_rows)
        print(Размечено объектов = , len(df_X1_L_tmp))
        #print(Меток поставлено = , len(y_tmp))

    else:
        print(Разметка завершена раньше заданного чсила итераций CoTrain)


if len(df_X1_U_tmp) > 0:

    #Единая разметка оставшихся объектов
    proba_clf1 = clf1.predict_proba(df_X1_U_tmp)
    proba_clf2 = clf2.predict_proba(df_X2_U_tmp)
    aggr_proba = (proba_clf1 + proba_clf2)/2

    y_conv_1_proba = np.asarray([])
    y_conv_1_proba = y_conv_1_proba.reshape(len(y_conv_1_proba), 1)

    logic = aggr_proba[:,1]> aggr_proba[:,0] #True – класс 1, False – класс 0

    pos = np.where(logic==True)
    neg = np.where(logic==False)
    all_ind = np.concatenate((pos[0],  neg[0]))

    y_clss_p = np.ones( shape =(len(pos[0]),1), dtype = int)
    y_clss_n = np.zeros( shape =(len(neg[0]),1), dtype = int)

    y_conv_1 = np.vstack([y_clss_p, y_clss_n])
    y_conv_proba_norm = aggr_proba[all_ind, 1]/(aggr_proba[all_ind, 1]+
aggr_proba[all_ind, 0])
    y_conv_1_proba = np.vstack((y_conv_1_proba,
y_conv_proba_norm.reshape(len(y_conv_proba_norm),1)))


    # Добавление переменных по меткам классов
```

```
        df_X1_L_tmp = np.vstack([df_X1_L_tmp, df_X1_U_tmp[all_ind]])
        df_X2_L_tmp = np.vstack([df_X2_L_tmp, df_X2_U_tmp[all_ind]])

        y_clss = np.asarray(y_conv_1)
        y_clss = y_clss.reshape(len(y_clss), 1)

        y_clss_proba = np.asarray(y_conv_1_proba)
        y_clss_proba = y_clss_proba.reshape(len(y_clss_proba), 1)

        y_tmp = np.asarray(y_tmp)
        y_tmp = y_tmp.reshape(len(y_tmp), 1)

        y_tmp_prob = np.asarray(y_tmp_prob)
        y_tmp_prob = y_tmp_prob.reshape(len(y_tmp_prob), 1)

        y_tmp = np.vstack([y_tmp, y_clss])
        y_tmp_prob = np.vstack([y_tmp_prob, y_clss_proba])

        df_X1_U_tmp = np.delete(df_X1_U_tmp, all_ind, axis=0)
        df_X2_U_tmp = np.delete(df_X2_U_tmp, all_ind, axis=0)

        unlabeled_rows = len(df_X1_U_tmp)

        if len(objs_keys) == 0:
            print(objs_keys is None, Ключи для БД не заданы)
        else:
            print(objs_keys is not None, Добавление ключей БД)
            objs_keys_L_tmp = np.vstack([objs_keys_L_tmp, objs_keys_U_tmp[all_ind]])
            objs_keys_U_tmp = np.delete(objs_keys_U_tmp, all_ind, axis=0)

        print(Неразмечено объектов = , unlabeled_rows)
        print(Размечено объектов = , len(df_X1_L_tmp))
        #print(Меток поставлено = , len(y_tmp))


# Если тестируется качество
if dataframe_test is not None:

    y_test_pred =  CoTrain_Classifier_Test(dataframe_test, X1, X2, clf1, clf2)


# Если привязывается к ключам БД
if len(objs_keys) != 0:
    result = pd.DataFrame({Client class: y_tmp.ravel(),
                           Class prob: y_tmp_prob.ravel(),
                           Client ID: objs_keys_L_tmp.ravel()})
```

```
    #result = [y_tmp,y_tmp_prob]
    if dataframe_test is None:
        #вернуть предсказанное
        return result
    # Если тестируется качество и выводится весь предсказанный массив
     объектов
    if dataframe_test is not None:
        return result, y_test_pred  # Все размеченные данные #Внешние данные для
     тестирования

else:
    result = y_tmp.ravel()
    if dataframe_test is None:
        return result, y_test_pred
    if dataframe_test is None:
        return result
```

3. Пример работы программы Co-Training Classifier:

Разработанная программа, применяющая исходный код метода частичного обучения проста в своём применении и имеет понятный интерфейс для работы.

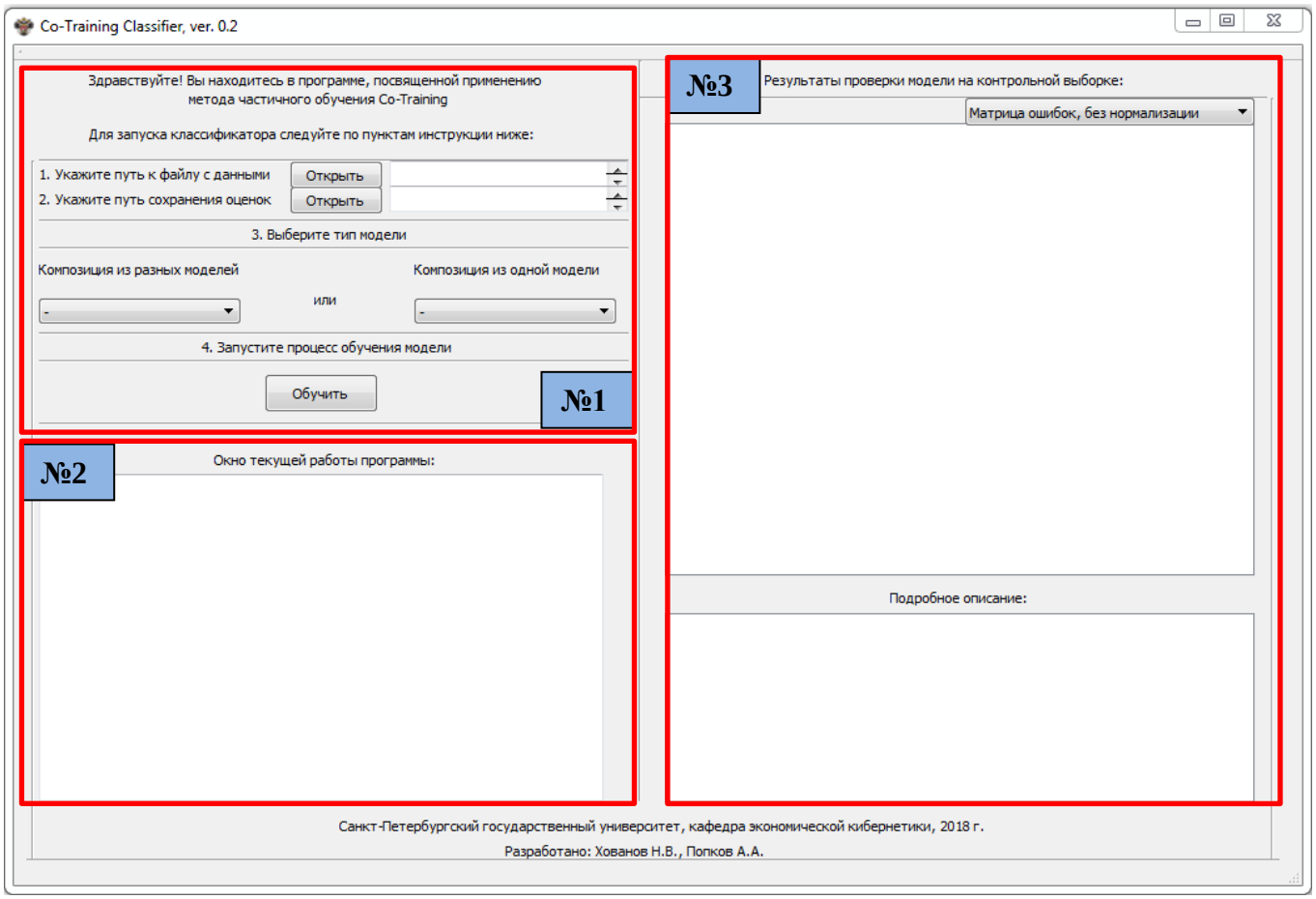

Рисунок 10: Интерфейс программы Co-Training Classifier

Условно окно приложения можно разделить на 3 отмеченные зоны. В первой области указывается путь к файлу с базой объектов и их признаков, путь для сохранения базы, оцененной с точки зрения лояльности, два выпадающих списка, которые предлагают выбрать ту или иную внутреннюю модель для обучения и стартовую кнопку для обучения. Во второй области содержится окно вывода основной информации из журнала событий программы. Третья область состоит из двух зон, первая из которых прорисовывает графики матрицы ошибок и оценки качества модели на итерациях, а вторая формирует таблицу подробной оценки качества одели на контрольной выборке.

Обучение модели осуществляется в 3 простых шага установки базовых параметров.

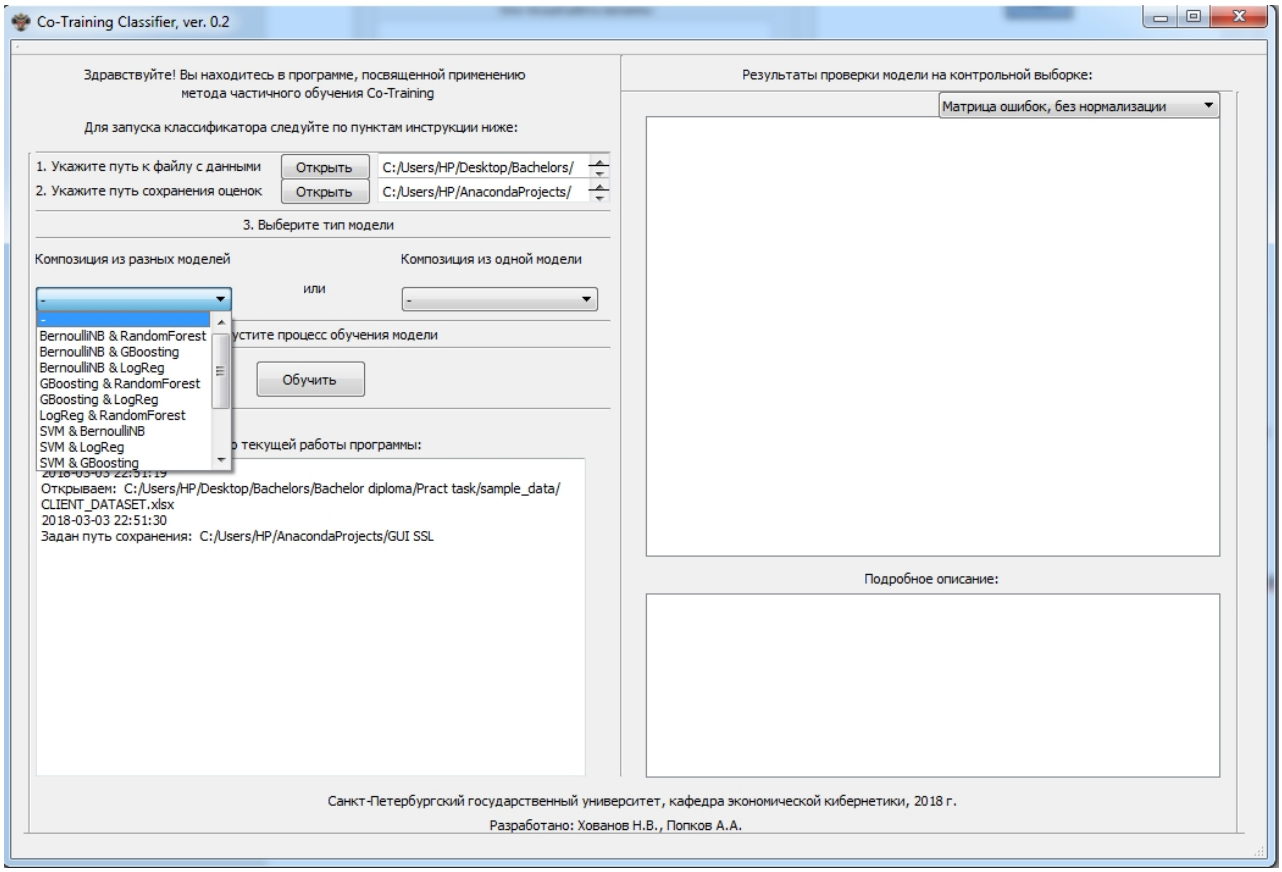

Рисунок 11: Список композиций из разных типов моделей в рамках метода Co-Training

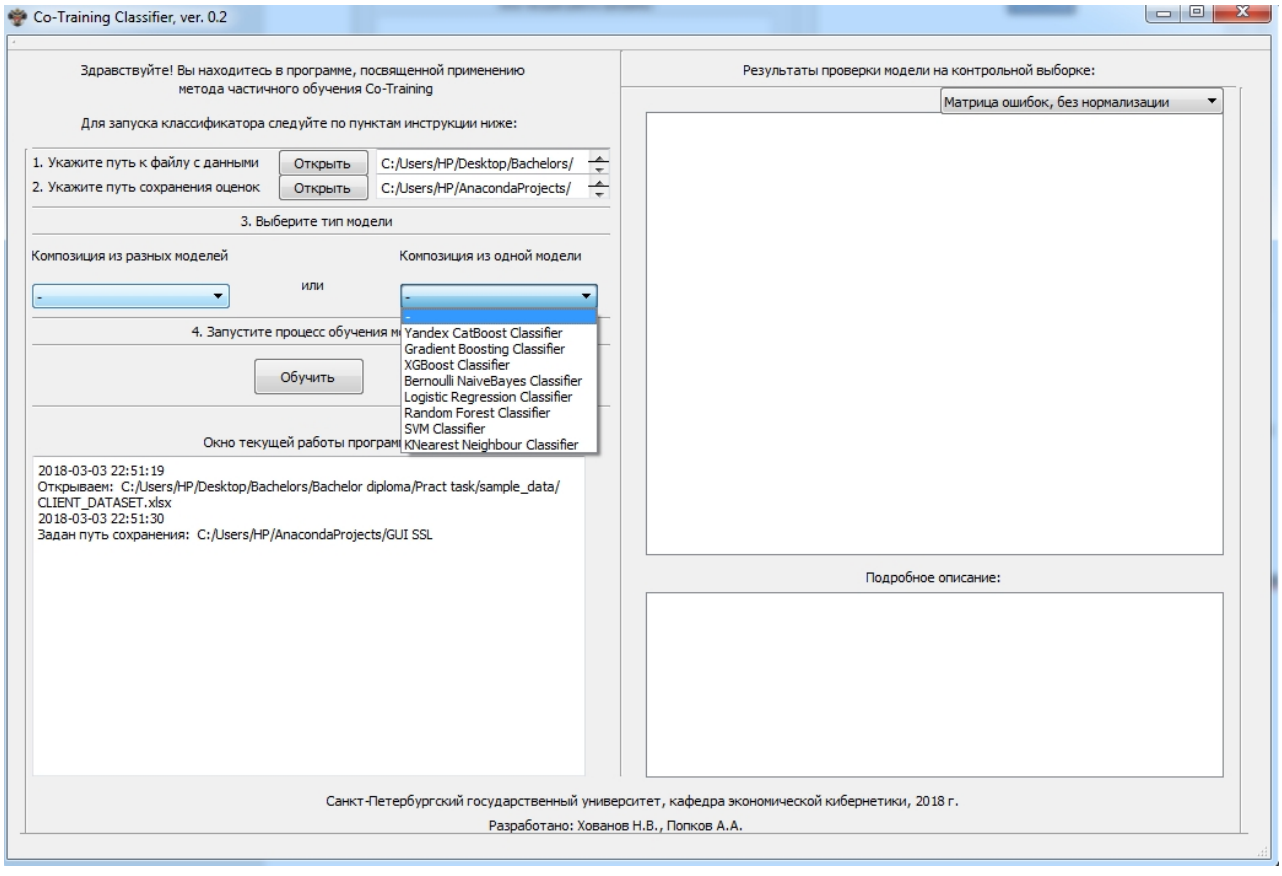

Рисунок 12: Список композиций из одного типа моделей в рамках метода Co-Training

На Рис. 11,12 окно текущей работы программы показывает, что пути к файлу с базой данных, а также путь сохранения прогнозов определены. Для примера выбрана композиция из двух наивных байесовский моделей:

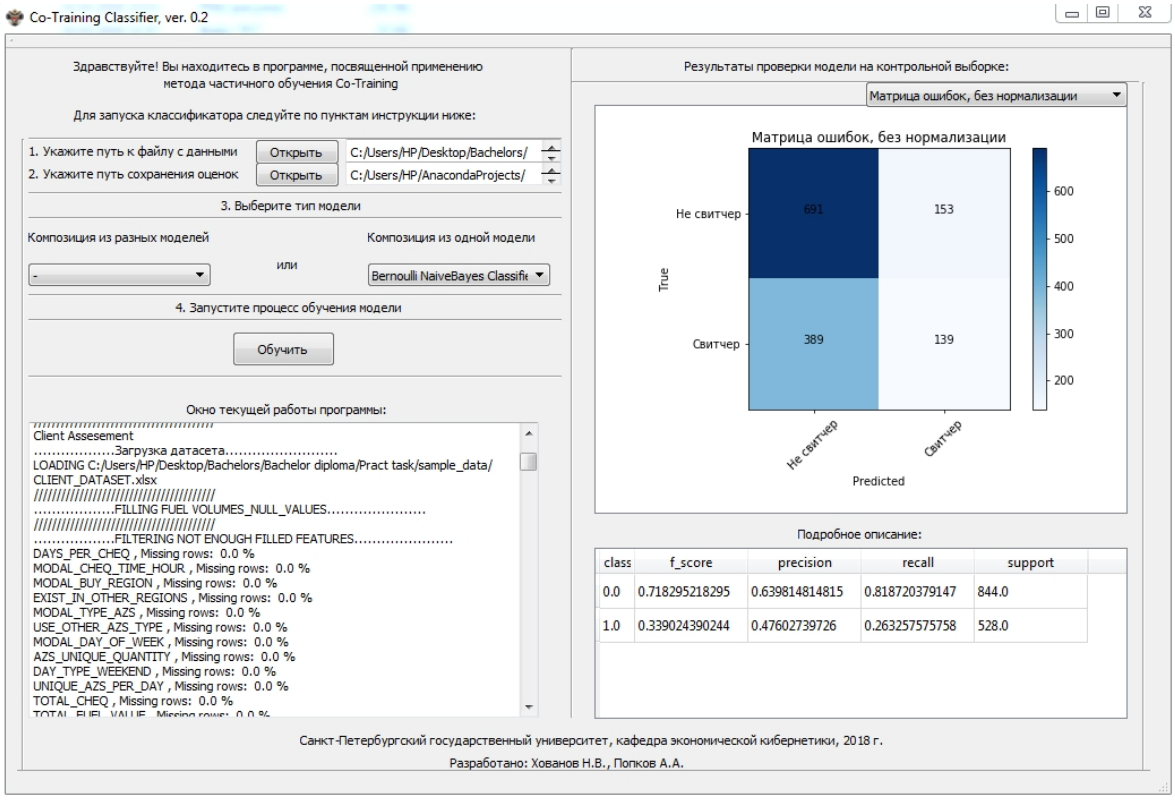

Рисунок 13: Пример вывода результатов работы программы (1)

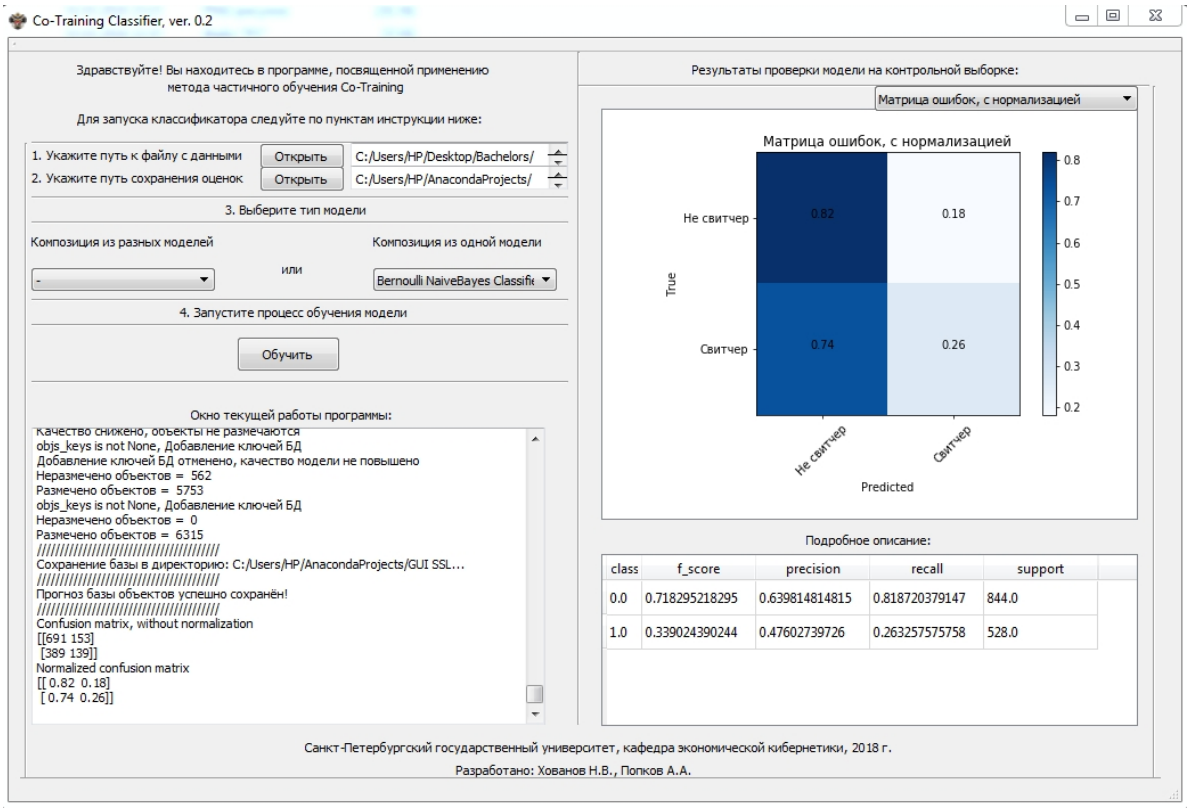

Рисунок 14: Пример вывода результатов работы программы (2)

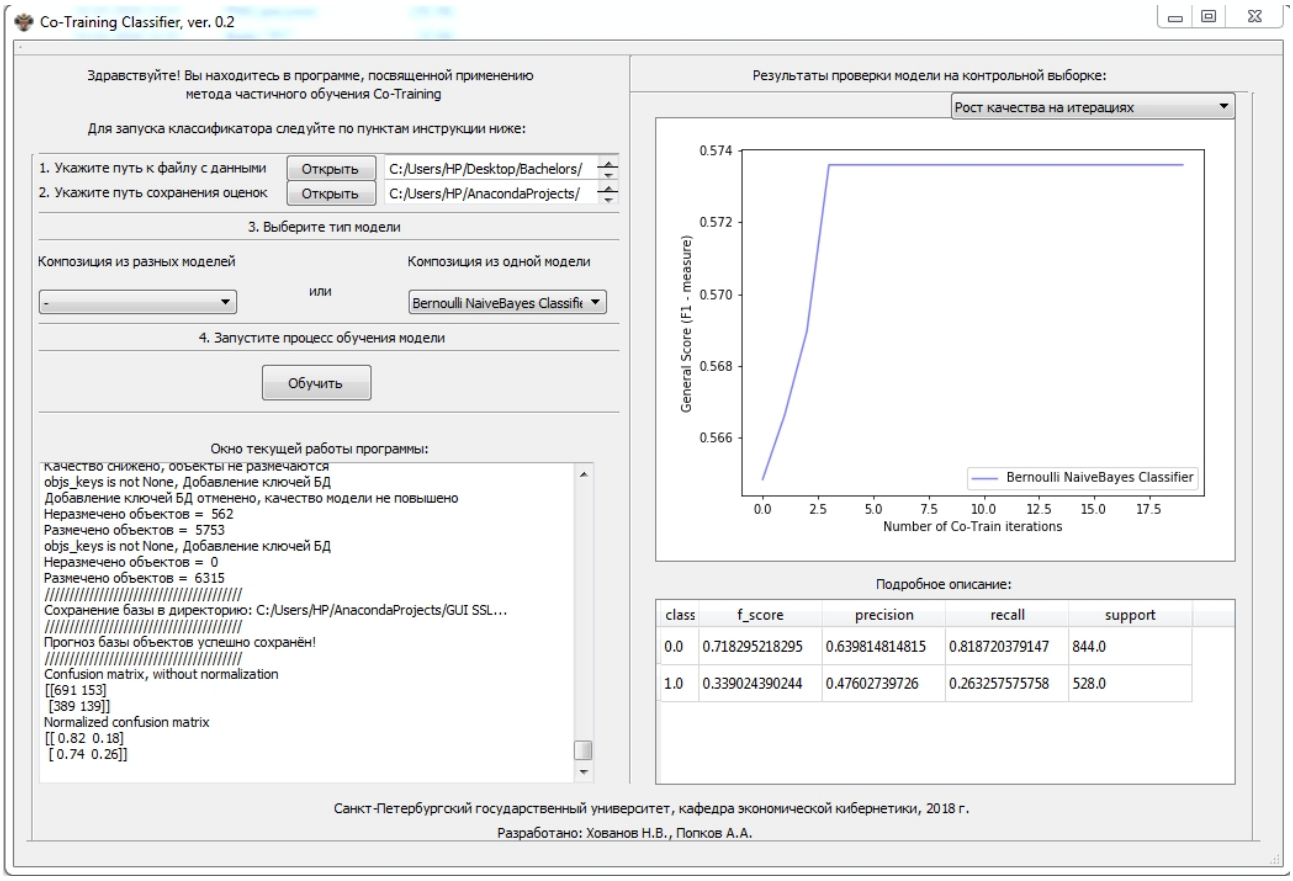

Рисунок 15: Пример вывода результатов работы программы (3)