Санкт-Петербургский государственный университет
Кафедра технологии программирования

Прокопьева Анна Анатольевна

Выпускная квалификационная работа бакалавра

Определение скрытых атрибутов пользователей социальных сетей с помощью анализа социального графа

Направление 010300
Фундаментальная информатика и информационные технологии

Научный руководитель,
ст. преподаватель
Мишенин А.Н.

Санкт-Петербург
2016

Содержание

Введение

В настоящее время распространенность и значимость социальных сетей весьма высока. Существует достаточно большое число различных площадок, которые охватывают разные сферы интересов человека. Некоторые из таких сетей небольшие, но узко специализированные, например, сообщество Executive.ru [1] объединяет менеджеров, которые заинтересованы в профессиональном росте и в получении новых знаний для повышении квалификации. Сообщество, по его данным, насчитывает около 300 тысяч пользователей. Среди широкого спектра различных социальных сетей находятся и известные крупные сети глобального характера. Среди русских сетей это, например, «Одноклассники» и «ВКонтакте», а среди мировых — «Facebook» и «Google+». В сети «ВКонтакте» по данным компании зарегистрировано более 350 миллионов пользователей [2], а «Facebook» ежемесячно посещают 1,59 млрд. активных пользователей по данным от 31 декабря 2015 года [3].

Крупнейших сетей по всему миру насчитывают в количестве около 22 штук [4]. В них люди взаимодействуют с другими пользователями, создают свой круг общения по интересам и обмениваются информацией. В социальной сети у каждого пользователя есть профиль — некоторая информация, доступная другим пользователям. Атрибутами профиля, например, могут быть: имя, фамилия, возраст, город, университет и др. К сожалению, многие пользователи не стремятся указывать полную информацию о себе, например, возраст в профиле может быть неверным или вовсе опущен. Данное обстоятельство затрудняет поиск новых друзей, а также ограничивает возможности в разных областях человеческой деятельности.

Многие владельцы бизнеса продвигают свои услуги и продукты в социальных сетях, создавая для этого отдельные страницы или предлагая их в переписке другим пользователям. Любая компания имеет свою целевую аудиторию, и для нее очень важно продвигать свой продукт заинтересованным людям.

Кроме того, социальные сети — это удобная площадка для проведения социологических исследований. Важнейшими параметрами в социологическом опросе являются пол и возраст человека. Определение возраста пользователей социальной сети позволит сформировать репрезентативную

выборку, то есть каждый пользователь, вне зависимости от того, указал он свой возраст или нет, имеет шанс попасть в данную выборку. Таким образом в итоговой выборке будут присутствовать представители разных подгрупп, что обеспечивает правильность дальнейших расчетов и исследований.

Таким образом, наличие полной информации из профиля пользователя является важным критерием для успешной работы в самых разных сферах человеческой деятельности.

Постановка задачи

Целью данной работы является разработка и реализация общего метода определения скрытых атрибутов пользователей социальных сетей на примере сети «ВКонтакте», а также его тестирование на атрибуте «Возраст» и сравнение с тривиальными (эвристическими) алгоритмами. Данный атрибут является сложной некатегориальной величиной, которая указывается большим числом пользователей и позволяет получать достаточно сложные метрики для оценки качества. Эксперименты проводятся на двух задачах:

1. Первая задача: по данной эго-сети пользователя [5] определить неизвестные атрибуты только одного центра. Решив эту задачу, можно решить следующую, но не обязательно, что один и тот же метод будет эффективен для обеих задач;
2. Вторая задача: в данном социальном графе [5] найти значения атрибутов всех пользователей, для которых этот атрибут неизвестен.

Для достижения поставленной цели предполагается решить следующие задачи:

1. Сбор данных и их хранение;
2. Выявление взаимосвязи между атрибутами и дружбой пользователей;
3. Разработка эвристических алгоритмов, основанных на наличии дружественной связи;
4. Реализация обучения с частичным привлечением учителя для задачи классификации на социальном графе;
5. Реализация методов машинного обучения для определения схожести отдельных атрибутов;
6. Сведение к задаче линейного программирования;
7. Вывод об эффективности разработанного метода.

Обзор литературы

Задача определения скрытых атрибутов пользователей социальных сетей очень актуальна в виду широкого распространения последних и достаточно активно изучается. В последние годы выходит множество работ, так или иначе посвященных данной теме. Отметим некоторые из них.

Одной из недавних статей по теме определения скрытых атрибутов пользователей социальных сетей является [6]. В ней отражено такое базовое и важное понятие, как принцип социального влияния. Предсказания скрытых атрибутов в этой статье проводились схожим с применяемым в рамках данной работы образом — с помощью метода обучения с частичным привлечением учителя на графе. Авторы используют различные метрики для определения схожести двух пользователей, например, длину пути от одного узла графа до другого.

Следующей статьей, внесшей большой вклад в данную работу, является статья [7]. В ней рассматривается адаптация метода обучения с частичным привлечением учителя для задачи классификации с последующей постановкой оптимизационной задачи и ее решения. В этой статье приводится доказательство справедливости предложенного решения.

Кроме того, статья [8] демонстрирует подход к задаче классификации на графе путем сведения к задаче линейного программирования. В этой статье используются понятия, которые также нашли применение в задаче определения скрытых атрибутов пользователей социальных сетей.

Основные определения

Введем основные понятия и термины для обеспечения ясности изложения.

Социальная сеть — это граф $G = (V, E)$. Каждый пользователь — это узел в графе G, а каждое ребро показывает отношения между двумя пользователями. Множество узлов будем обозначать V, а множество ребер — E. Для каждого узла $v_i \in V, i = 1, \ldots, n$, существует вектор A_i, который описывает атрибуты пользователя i:

$$A_i = (a_i^1, \ldots, a_i^m). \tag{1}$$

Переменная m в выражении (1) — это число атрибутов пользователя i. Параметр $a_i^j, j \in (1, \ldots, m)$ показывает j-ый атрибут пользователя i. Вес ребра между узлами i и j будем обозначать ω_{ij}. Он определяет меру схожести между пользователями i и j, т. е. близость значений их конкретных атрибутов.

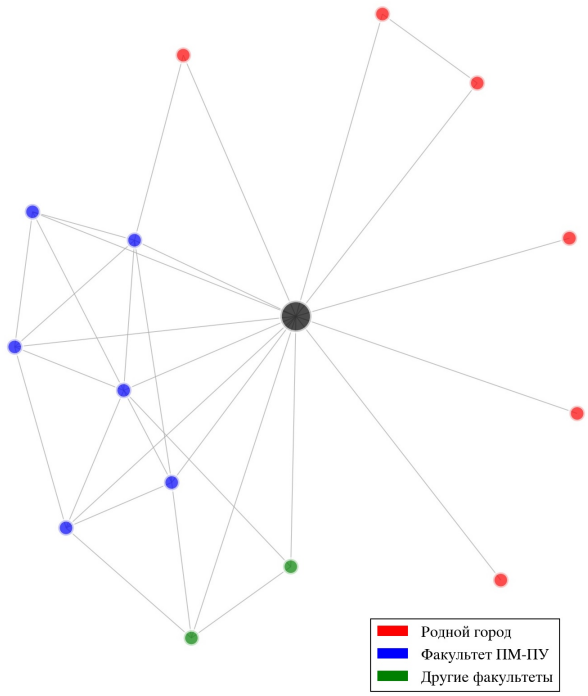


Рис. 1: Эго-сеть

Каждый пользователь имеет собственную *эго-сеть* — это социальная сеть, центральным узлом которой является выбранный пользователь, а остальными узлами обозначаются его друзья. Пример эго-сети представлен на Рис. 1

Глава 1. Данные и работа с ними

В текущей главе будет рассмотрен процесс формирования данных, способ их хранения, основные предположения и их доказательства, основанные на данных, которые необходимы для дальнейшей работы.

1.1. Коллекция данных

Для начала необходимо определить, какие данные понадобятся для работы и как их собрать.

Сбор данных

Среди всех социальных сетей выбор остановился на «ВКонтакте», так как эта сеть пользуется наибольшей известностью в России. По данным SimilarWeb, «ВКонтакте» является первым по популярности сайтом в России и на Украине [9].

«ВКонтакте» предоставляет API (Application Programming Interface) — набор готовых функций и процедур, которые разработчики могут использовать в своих внешних программных продуктах [10]. С помощью этого интерфейса можно загрузить из сети данные о пользователе, работать с его фотографиями и записями, просматривать друзей и т. д. Необходимо заметить, что к методам API, предоставляемым «ВКонтакте», можно обращаться не чаще 3 раз в секунду от имени одного пользователя. Данное ограничение существенно влияет на количество затрачиваемого времени на сбор данных.

Для дальнейших исследований была скачана информация из профилей порядка 1000 пользователей. Сюда входят значения атрибутов профиля пользователя: имя, фамилия, id пользователя, пол, возраст, город, университет (название и год его окончания) и школа, — а также социальный граф пользователя, содержащий всех друзей пользователя и связи между ними. Кроме того, необходимо сохранить значения атрибутов профилей друзей пользователя. Данная выборка из порядка 1000 пользователей формировалась следующим образом:

1. Путем случайной генерации чисел выбирался id пользователя в сети «ВКонтакте»;

2. Проверялось, указал ли пользователь с найденным на предыдущем шаге id в своем профиле дату рождения. Если возраст указан, то проверяется, чтобы у него было не менее 30 друзей для отсеивания фейковых (с ложными данными) страниц;

3. Если условия шага 2 выполнены, загружается информация о пользователе и о его друзьях;

4. Повторяется до тех пор, пока не будет сформирована вся выборка.

Условие известности возраста необходимо для оценки точности работы алгоритмов.

Всего было загружено 235263 профиля и 1000 социальных графов. В среднем в одном социальном графе с единичной глубиной (т. е. только пользователь и его друзья) содержится примерно 230 узлов.

Хранение данных

Следует сразу уточнить, что организация базы данных для хранения собранной информации выходит за рамки данной работы и не рассматривалась в ее процессе. Все данные занимают достаточно небольшое дисковое пространство, например, информация о выборке из 1000 пользователей занимает 194 Мб.

Все данные хранятся в файловой системе. Структура файлов с данными из профилей основана на формате JSON [11] в виде коллекции пар «ключ/значение», где ключами/значениями являются, соответственно, наименования/значения атрибутов. Структура файлов, в которых хранятся эго-сети пользователей (графы), также организована как коллекция пар «ключ/значение», где ключами являются id друзей пользователя, а значениями — списки, состоящие из id таких друзей пользователя, которые дружат с пользователем, чей id указан в ключе. Иными словами, в значении (списке) указаны общие друзья пользователя и его друга с id, хранящемся в ключе.

1.2. Взаимосвязь между дружбой пользователей и их атрибутами

Основным предположением для определения скрытых атрибутов пользователя является принцип социального влияния [6]. Необходимо первым

Атрибут	$\boldsymbol{C_a}$
Пол	0,97
Возраст	1,41
Город	1,17
Университет	2,17
Школа	8,27
Страна	1,01

Таблица 1: Значение параметра C_a в зависимости от атрибута

делом показать, что собранные данные удовлетворяют этому принципу. Для этого нужно ввести следующие определения.

$$S_a = \frac{|(i,j) \in E : a_i = a_j|}{|E|}, \quad a \in A, \quad i,j \in \{1,\ldots,n\}, \tag{2}$$

$$P_a = \frac{\sum_{i=1}^{k} T_i \left(T_i - 1\right)}{\left|\sum_{i=1}^{k} T_i\right| \left(\left|\sum_{i=1}^{k} T_i\right| - 1\right)}. \tag{3}$$

S_a показывает вероятность того, что пользователь i и пользователь j станут друзьями, потому что они имеют одинаковое значение атрибута a. В выражении (2) a_i обозначает атрибут a пользователя i. В выражении (3) k — это число возможных значений параметра a, T_i показывает число пользователей, атрибут a которых принимает i-ое значение, P_a обозначает эмпирическую вероятность того, что пользователи, указавшие одинаковые значение атрибута a, станут друзьями.

Параметр C_a определяется выражением

$$C_a = \frac{S_a}{P_a}.$$

Он показывает отношение вероятности S_a к эмпирической вероятности P_a. Если $C_a > 1$, то существует взаимосвязь между дружбой и атрибутом a. Чем больше C_a, тем более вероятно, что пользователи станут друзями, потому что они имеют одинаковое значение атрибута a. Таблица 1 показывает, какие значения принимает параметр C_a на собранных данных.

С другой стороны, можно предположить, что если пользователи «дружат», то очень вероятно, что большинство из них имеют одинаковые значения атрибутов. Основываясь на этом предположении, можно использовать отношения между пользователями для предсказания скрытых атрибутов.

Кроме того, чем больше значение коэффициента C_a для атрибута a, тем выше точность предсказания.

Глава 2. Работа с социальным графом

В данной главе рассмотрены некоторые методы определения атрибутов пользователя, которые экспериментально подтверждают принцип социального влияния [6], изложенный выше, на примере атрибута «Возраст». Представленные алгоритмы могут решать только первую задачу — определение атрибута центра эго-сети. В первом параграфе приводится описание методов, а во втором результаты их работы.

2.1. Эвристические алгоритмы

2.1.1. Определение возраста по среднему возрасту всех друзей

В социальном графе пользователя необходимо выбрать только те узлы, значение атрибута «Возраст» которых известен. Обозначим a_i данное значение для i-го узла. Тогда возраст пользователя определяется по формуле

$$A_1 = \frac{\sum_{i=1}^{k} a_i}{k},$$

где k — количество узлов, значение атрибута a которых известно.

2.1.2. Определение возраста по среднему значению мод возрастов в группах

Данный метод основан на выделении групп пользователей. В результате кластеризации социального графа [12] имеются некоторые группы друзей пользователя:

$$N = N_1, \ldots, N_p, \quad \sum_{i=1}^{p} N_i = n,$$

где n — количество всех пользователей. Необходимо выбрать такие группы, которые содержат больше трех узлов, чтобы дальнейшие действия имели смысл:

$$N^* = \left\{ N_i \middle| |N_i| > 3, \quad i = \overline{1,p} \right\},$$
$$|N^*| = t, \quad t \leq p.$$

В каждом выбранном кластере определим моду среди возрастов:

$$m_i = mode\left(N_i\right), \quad N_i \in N^*, \quad i = \overline{1,t}.$$

Тогда возраст пользователя определяется по формуле

$$A_2 = \frac{\sum_{i=1}^{t} m_i}{t}.$$

Метод основан на связях между друзьями пользователя и наиболее часто встречающемся возрасте в отдельных группах (в англ. литературе community).

2.1.3. Определение возраста на основе наибольшего кластера

Данный метод также подразумевает выделение групп среди пользователей. Среди всех найденных кластеров выбирается тот, в котором наибольшее количество узлов:

$$N_{max} = max\left|N_i\right|, \quad N_i \in N, \quad i = \overline{1,p}.$$

Заметим, что таких кластеров может быть несколько. Возраст пользователя определяется по формуле

$$A_3 = mode\left(N_{max}\right).$$

В выбранном кластере содержится наибольшее количество «знакомых» между собой друзей пользователя. Применительно к жизни, это скорее всего означает тесное знакомство большого количества людей, их общие интересы и совместную работу, т. е. круг общения пользователя. Как правило, возраст людей в таких группах близок к возрасту пользователя.

2.1.4. Определение возраста на основе кластера с минимальным разбросом

Данный метод использует понятие дисперсии в кластерах. Для всех кластеров, содержащих большего одного узла (иначе дисперсия равна нулю), необходимо посчитать дисперсию ($variance$) возрастов:

$$N^* = \left\{ N_i \middle| \left|N_i\right| > 1, \quad i = \overline{1,p} \right\},$$

$$\left|N^*\right| = t, \quad t \leq p,$$

$$\sigma_i = variance\left(N_i\right), \quad N_i \in N^*, \quad i = \overline{1,t}.$$

Среди полученных значений дисперсий находится минимальное, т. е. выбирается такой кластер, в котором возраста друзей пользователя отклоняются от среднего возраста с наименьшим значением:

$$\sigma_{min} = min\left(\sigma_i\right), \quad i = \overline{1,t}.$$

Тогда возраст пользователя определяется в выбранном кластере по формуле

$$A_4 = \frac{\sum_{i=1}^{k} a_i}{k},$$

где k — число узлов в выбранном кластере, значение атрибута a («Возраст») которых известно.

2.1.5. Определение возраста с помощью коэффициентов влияния

Данный метод основан на вычислении «веса» каждого кластера. Для каждой группы, имеющей больше одного узла (иначе дисперсия равна нулю), определяется ее дисперсия ($variance$) и среднее значение ($average$):

$$N^* = \left\{ N_i \middle| |N_i| > 1, \quad i = \overline{1,p} \right\},$$

$$|N^*| = t, \quad t \leq p,$$

$$\sigma_i = variance\left(N_i\right), \quad n_i = average\left(N_i\right),$$

$$N_i \in N^*, \quad i = \overline{1,t}.$$

Полученные значения дисперсий необходимо преобразовать так, чтобы определить «вклад» каждого кластера в итоговый возраст пользователя. Другими словами, чем меньше разброс в значениях внутри кластера, тем сильнее его «влияние» на результат. Коэффициенты влияния k_i находятся по формулам

$$\sigma = \sum_{i=1}^{t} \sigma_i, \quad \sigma_i' = \sigma - \sigma_i, \quad \sigma' = \sum_{i=1}^{t} \sigma_i',$$

$$k_i = \frac{\sigma_i'}{\sigma'}.$$

Сумма полученных коэффициентов равна единице, а наибольший коэффициент влияния имеет кластер с наименьшей дисперсией. Таким образом, возраст пользователя определяется по формуле

$$\sum_{i=1}^{t} n_i k_i.$$

2.2. Результаты

Все описанные выше эмпирические алгоритмы запускались на выборке из 1000 пользователей, возраст которых был известен.

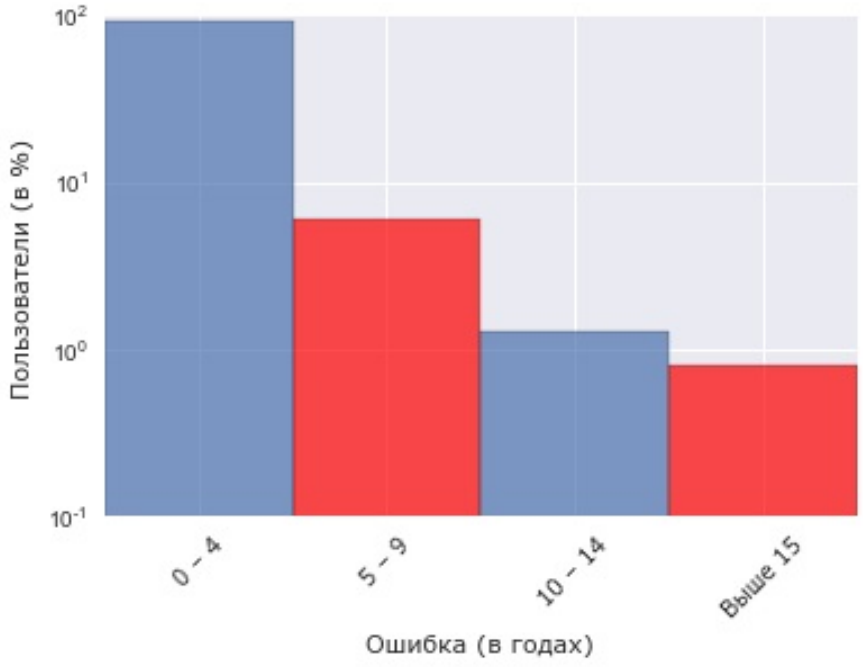


Рис. 2: Среднеквадратичная ошибка метода 2.1.1

На Рис. 2 видно, что возраст некоторых пользователей сильно отличается от найденных значений. Часто оказывается, что такие пользователи не указывают свой реальный возраст в профиле, поэтому отметим их как «выбросы» на графиках и не будем их учитывать.

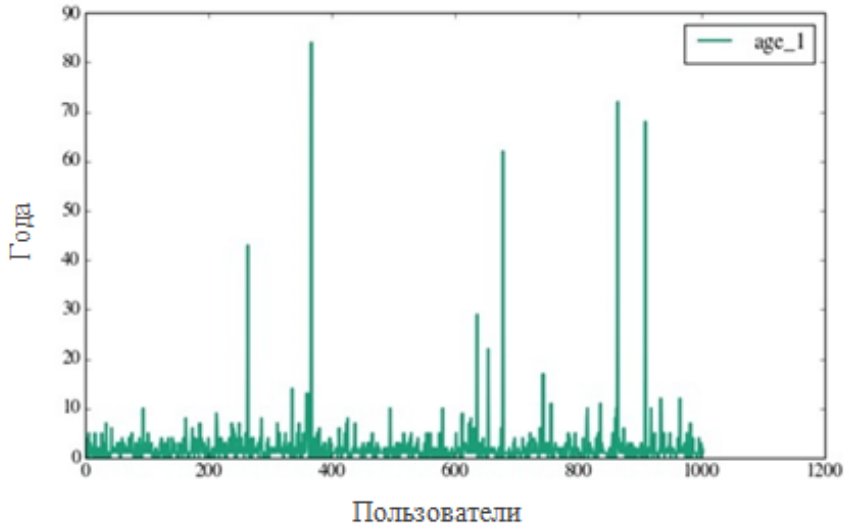
Приводить графики ошибок всех алгоритмов нерационально, поэтому на Рис. 3 изображены только графики методов 2.1.1 и 2.1.3 соответственно. На них изображены ошибки (абсолютная разница между реальным и найденным значением атрибута «Возраст») для каждого пользователя соответственно.

	1	2	3	4	5
RMSE	2,45	3,0	4,24	2,83	2,65

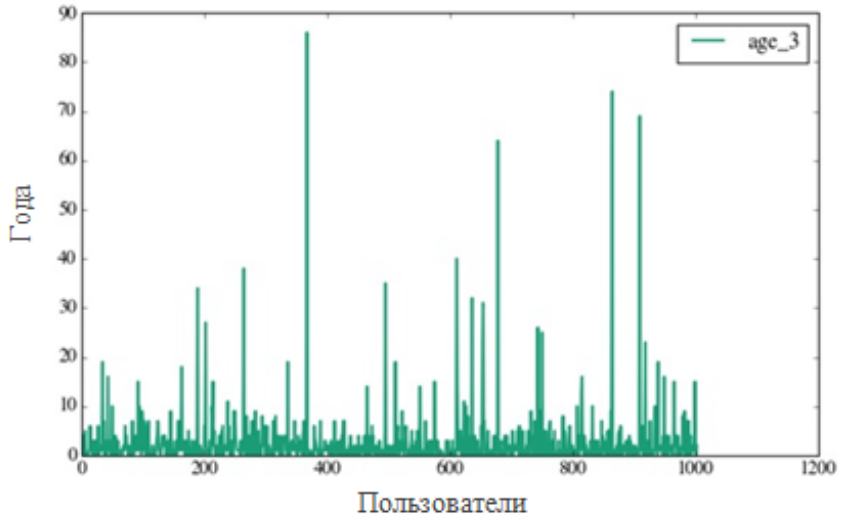
Таблица 2: Среднеквадратичные ошибки методов 2.1.1-2.1.5

В Таблице 2 представлены среднеквадратичные ошибки (RMSE) всех методов. Как видно из данной таблицы, самым эффективным методом оказалось определение возраста по среднему всех возрастов друзей пользователя.

График на Рис. 4 демонстрирует общую статистику по всем методам, а именно, у скольких пользователей удалось верно определить их возраст (столбец True), а у скольких нет (столбец False). Таким образом, примерно



а)



б)

Рис. 3: Ошибки методов а) 2.1.1; б) 2.1.3

у 750 пользователей хотя бы одним методом удалось верно определить их возраст.

2.3. Вывод

Основываясь на приведенных выше результатах, можно сделать вывод, что рассмотренные эмпирические алгоритмы достаточно эффективно определяют скрытый атрибут пользователя, базируясь только на дружеских связях внутри его социального графа.

Однако недостатком разработанных методов является то, что они не учитывают полную структуру внутренних ссылок социального графа, так как основаны только на наличии дружеской связи. Они не рассматривают

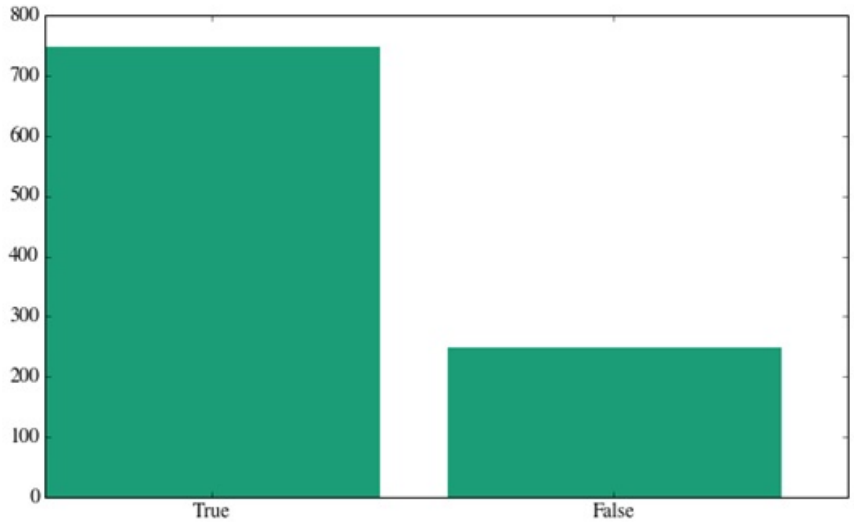


Рис. 4: Количество верно и неверно определенных возрастов

равенство других атрибутов, например, «Университет» или «Город». Кроме того, данные методы могут определить атрибут только одного пользователя — центра рассматриваемой эго-сети. Таким образом, приведенными способами можно решить только одну задачу — определение скрытого атрибута пользователя по его эго-сети. В следующих главах будут приведены методы определения скрытых атрибутов, устраняющие выявленные недостатки.

Глава 3. Метод обучения с частичным привлечением учителя

3.1. Описание метода

Одной из главных задач машинного обучения является задача классификации. В общих словах, задача классификации состоит в присваивании каждому рассматриваемому объекту некоторого класса, основываясь на каких-либо наблюдаемых данных.

Обычно методы классификации используют в качестве обучающей выборки только размеченные данные, но на практике часто встречается ситуация, когда корпус данных достаточно велик, но большая часть данных не размечена. Для решения задач такого типа в качестве эффективного средства может рассматриваться метод обучения с частичным привлечением учителя. Данный метод использует как данные с метками, так и без, что позволяет уменьшить затрачиваемые человеком усилия на разметку данных и дает достаточно высокую точность.

Для решения поставленной задачи попытаемся адаптировать метод с частичным привлечением учителя на графе [7]. Метод работает с графом, в котором одна часть узлов имеют некоторую метку, а узлы другой не размечены, ребра (могут быть взвешенными) показывают степень сходства узлов. В рамках поставленной задачи размеченные узлы соответствуют тем друзьям пользователя (эго-сеть которого исследуется), которые указали определяемый атрибут в своих профилях, неразмеченные наоборот. В большинстве случаев размеченная часть друзей оказывается существенно меньше той, атрибут пользователей в которой неизвестен.

Пусть входной вектор данных $X = \{X_1, \ldots, X_P, X_{P+1}, \ldots, X_N\}$. Элементы вектора X могут принимать действительные значения определяемого атрибута. Пусть имеется K классов и первые P элементов вектора имеют метки $k(i) \in 1, \ldots, K$, $i = \overline{1, P}$. Определим симметричную $(N \times N)$-матрицу W, элементы которой обозначают степень схожести объектов из X друг с другом. Построить матрицу W можно разными способами [7], например:

$$\omega_{ij} = \exp\left(-\|X_i - X_j\|^2/\gamma\right).$$

где γ — некоторый нормирующий множитель.

Определим диагональную матрицу D следующим образом:

$$d_{ii} = \sum_{j=1}^{N} \omega_{ij}, \quad i = \overline{1, N}.$$

Определим $(N \times K)$-матрицу Y с элементами

$$y_{ik} = \begin{cases} 1, & \text{если } X_i \text{ помечен как } k(i) = k, \\ 0, & \text{иначе.} \end{cases}$$

Каждый столбец y_{*k} матрицы Y — это функция разметки. Также определим $(N \times K)$-матрицу F и назовем ее столбцы f_{*k} функциями классификации. Строки матриц F и Y обозначим, соответственно, f_{k*} и y_{k*}. Основная идея метода заключается в поиске таких функций классификации, чтобы, с одной стороны, они были близки к соответствующей функции разметки, а с другой стороны, плавно изменялись над графом, другими словами, чем «ближе» две вершины графа друг к другу, тем более вероятно, что они будут иметь одинаковые метки. Эта идея может быть выражена так:

$$\min_{F} \left\{ \sum_{i=1}^{N} \sum_{j=1}^{N} \omega_{ij} \| f_{i*} - f_{j*} \|^2 + \mu \sum_{i=1}^{N} d_{ii} \| f_{i*} - y_{i*} \|^2 \right\}. \tag{4}$$

В выражении (4) параметр μ показывает компромисс между требованиями к функции классификации.

3.2. Решение задачи оптимизации

В этом параграфе введены некоторые уточняющие условия задачи (4) и приведено ее решение.

В зависимости от возможных значений искомого атрибута определяется количество классов. В экспериментах количество классов K будем считать равным ста — номер класса соответствует определенному значению возраста. В действительности можно было бы разбить диапазон возрастов на группы по пять лет, провести классификацию на двадцати классах, а затем повторить ее внутри выделенной группы среди пяти классов. Такой подход экономит занимаемую память при работе с большими данными, но было решено остановиться на первом подходе, так как он достаточно быстро работает, данные не занимают много места, и, кроме того, экономия памяти не входит в цели работы.

В терминах первого параграфа коэффициенты (степени схожести, или веса ребер) определены весьма абстрактно, поэтому, как уже было замечено, матрицу весов W можно строить по-разному. В рамках данной работы реализовано несколько подходов к построению матрицы W.

Первый заключается в использовании матрицы смежности графа, то есть в качестве меры близости узлов (друзей пользователя) используется единица, если пользователи дружат, и ноль в противном случае.

Второй подход основан на методах машинного обучения, а именно:

1. Логистическая регрессия;
2. Метод ближайших соседей (KNN);
3. Случайный лес.

В качестве входных данных этих методов используется обучающая выборка, построение которой будет подробно рассмотрено в следующем параграфе. Выходным значением первого метода является некоторый вектор коэффициентов, задающий разделяющую гиперплоскость, второго метода — размеченная область, третьего — усредненное дерево решений. На основе этих данных для каждой пары друзей пользователя вычисляется мера их сходства — число в диапазоне от нуля до единицы. Эти числа и есть элементы матрицы W.

Результаты работы метода обучения с частичным привлечением учителя с применением каждого из описанных подходов представлены в параграфе 3.4.

Рассмотрим решение задачи (4). Функции классификации (столбцы матрицы F) определяются по формуле

$$f_{*k} = \frac{\mu}{2+\mu}\left(I - \frac{2}{2+\mu}D^{-\sigma}WD^{\sigma-1}\right)^{-1} y_{*k},$$

где $k = 1, \ldots, K$. Доказательство данного утверждения приведено в статье [7]. Значение параметра μ определяется экспериментально, в рамках данной работы примем $2/(2+\mu) = 0{,}5$. Параметр σ определяет некоторые частные случаи: при $\sigma = 1$ получается стандартный метод, основанный на лапласиане графа [13], при $\sigma = 1/2$ — нормализованный метод, основанный на лапласиане графа [14], и метод, основанный на PageRank [15], при $\sigma = 0$. Для решения задачи применим третий случай, так как главным

преимуществом этого метода является его квазилинейная сложность, и он достаточно хорошо работает с собранными данными.

3.3. Машинное обучение

В этом параграфе рассмотрены особенности построения обучающей выборки в рамках данной работы для некоторых методов машинного обучения. В качестве языка программирования, на котором реализовывались исследуемые методы, выбран Python, так как для данного языка существует множество необходимых библиотек и расширений, находящихся в свободном доступе. Кроме того, с помощью языка Python процесс визуализации результатов становится достаточно простым и удобным.

3.3.1. Построение обучающей выборки

Для построения обучающей выборки среди всех загруженных профилей пользователей выберем случайным образом три тысячи пар друзей, которые указали определяемый атрибут на своей странице. Половина таких пар имеют разное значение атрибута, другая половина одинаковый. Далее для каждой пары строим вектор, бинарные (кроме первого) элементы которого показывают совпадение/несовпадение значений некоторых других атрибутов их профилей.

Для проведения экспериментов в качестве атрибутов, по которым строится вектор признаков для каждой пары, возьмем следующие: «Количество общих друзей», «Пол», «Город», «Страна», «Университет», «Год окончания университета», «Школа», «Школа $\wedge$ Университет». Таким образом, для каждой пары друзей получим вектор, в котором единица указывает на совпадение значений соответствующего атрибута, а ноль на различие. В Таблице 3 продемонстрирован пример такого построения. Последний столбец показывает совпадение или несовпадение значения определяемого атрибута (в экспериментах это «Возраст») у пары друзей и является значением целевой функции в обучении.

Выборку в таком виде назовем «Первой». «Первая» выборка имеет существенный недостаток: некоторые векторы могут совпадать, но при этом значения целевой функции (цифра в последнем столбце) могут быть разными. Данное обстоятельство негативно влияет на точность обучения. Поэтому попробуем ее видоизменить путем нормализации или с помощью

Количество общих друзей	Пол	Город	Страна	Университет	Год окончания университета	Школа	Школа ∧ Университет	Значение
9	1	0	1	0	0	0	0	1
6	0	0	0	0	0	0	0	0
10	0	1	1	1	1	0	0	1
5	1	1	1	0	0	0	0	0
22	1	1	1	0	0	0	0	1
3	0	0	1	0	0	0	0	0

Таблица 3: Построение «Первой» обучающей выборки

других методов для повышения точности обучения.

«Вторая» выборка строится на основе «Первой» по следующим правилам: значение целевой функции z определяется по формуле

$$z = \begin{cases} 1, & \text{если } |a_i - a_j| < 5, \\ 0, & \text{иначе,} \end{cases}$$

где a_i и a_j значения атрибута «Возраст» пользователей i и j соответственно; число общих друзей m становится бинарным:

$$m = \begin{cases} 1, & \text{если } friends_{ij} \geq 5, \\ 0, & \text{иначе,} \end{cases}$$

где $friends_{ij}$ — начальное число общих друзей (i, j) пары пользователей. В таком виде повторы тоже остаются, но для сравнительного анализа примем этот вариант.

В «Третьей» выборке остается правило для подсчета значения целевой функции, но число общих друзей нормализуется по формуле

$$m = \frac{friends_{ij}}{\max(friend_i, friend_j)},$$

где $friend_i$ и $friend_j$ — количество друзей пользователей i и j соответственно.

«Четвертая» выборка строится на основе «Третьей» с помощью ее стандартизации по формуле

$$x' = \frac{x - \bar{x}}{\sigma}$$

где σ — среднеквадратичное отклонение и $\bar{x}$ — среднее значение для рассматриваемого атрибута (столбца), а x — значение текущего элемента вектора (строки).

Элементы обучающей и тестовой выборок были взяты случайным образом из сформированной выборки.

Результаты вычислений по всем выборкам будут рассмотрены в пунктах 2-4 текущего параграфа для каждого метода обучения отдельно.

3.3.2. Логистическая регрессия

Логистическая регрессия используется для определения вероятности того, что событие произойдет/не произойдет (1 или 0), или, другими словами, для вывода о принадлежности элемента к определенному классу. Основная идея модели состоит в разделении пространства некоторой линейной границей (гиперплоскостью) на две части, соответствующих двум классам элементов. В данной работе логистическая регрессия применяется для предсказания степени схожести двух пользователей по некоторому набору признаков, т. е. определение класса элемента (связи между людьми) не требуется.

Модель логистической регрессии, как и многие другие модели, уже реализована в ряде открытых библиотек. Для решения задачи в рамках данной работы используется библиотека *scikit learn* [16] и ее модуль *linear model*. В документации подробно описано, по каким формулам производится расчет. Кроме того, для автоматизации процесса подбора коэффициентов регуляризации используется модуль *grid_search* [16]. Из входного набора коэффициентов он позволяет автоматически выбрать самый подходящий. Оценка качества обучения осуществляется с помощью ROC-кривой [17]. Для этого в метод *GridSearchCV* из модуля *grid_search* необходимо передать в качестве входного параметра *scoring='roc_auc'*.

После задания основных параметров происходит обучение и тести-

	«Первая»	«Вторая»	«Третья»	«Четвертая»
Качество	0,54	0,6	0,58	0,61
Коэффициент регуляризации	0,89	1,0	1,0	1,0
Вид регуляризации	L2	L2	L2	L2
Алгоритм решения оптимизационной задачи	liblinear	liblinear	liblinear	liblinear
Результирующие коэффициенты	[-0,002; 0,302; 0,206; 0,103; 0,164; -0,262; 1,018; -0,270]	[0,490; 0,123; 0,127; 0,500; -0,235; -0,398; 0,693; -0,094]	[2,377; -0,030; 0,248; 0,442; -0,412; -0,545; 0,393; 0,298]	[0,351; 0,068; 0,028; 0,102; -0,230; -0,056; 0,199; 0,070]
Время работы (сек)	2,02	1,73	1,8	1,79

Таблица 4: Результат обучения модели логистической регрессии

рование результатов на соответствующей выборке. Основные значения параметров, качество обучения, а также итоговые коэффициенты, задающие гиперплоскость, для каждой из выборок представлены в Таблице 4.

Качество обучения по всем выборкам достаточно низкое, что в принципе объясняется несбалансированностью указанных данных в профилях пользователей. Многие частично закрывают доступ к своей странице или просто не указывают такую информацию, как, например, «Год окончания университета». Тем не менее, показатель AUC (площадь под ROC-кривой) больше 0,5.

Для дальнейшего исследования выберем модель, обученную на «Четвертой» выборке, т. к. она показала наилучший результат.

Фрагмент программы, реализовывающий обучение модели логистической регрессии представлен в Приложении А.1.

3.3.3. Метод K ближайших соседей (KNN)

Метод K ближайших соседей — алгоритм классификации объектов, основанный на оценивании сходства объектов [18]. Идея метода заключается в том, что объекту присваивается тот класс, который наиболее распространен среди его ближайших соседей. Главное преимущество метода — его простота.

Выбор параметра K (числа ближайших соседей) основан на методе кросс-валидации или, по-другому, скользящего контроля [19]. Метод реализовывается путем рассмотрения различных параметров настройки алгоритма для некоторых разбиений выборки на обучающую и контрольную подвыборки. Для каждого такого разбиения алгоритм запускается на обучающей подвыборке, и его средняя ошибка оценивается по контрольной подвыборке. Таким образом можно выбрать наилучшие параметры, предписанные конкретному разбиению.

Метод K ближайших соседей, как и модель логистической регрессии уже представлен в некоторых программных модулях. Для реализации данного метода взята библиотека *scikit learn* и ее модуль *Nearest Neighbors*. В этом модуле имеется класс *KNeighborsClassifier* [16], который и используется в конечном итоге для поиска меры схожести между двумя узлами графа.

	«Первая»	«Вторая»	«Третья»	«Четвертая»
Качество	0,57	0,5	0,57	0,58
Метрика	Минковского с $p = 2$, т. е. Евклидова	Минковского с $p = 2$, т. е. Евклидова	Минковского с $p = 2$, т. е. Евклидова	Минковского с $p = 2$, т. е. Евклидова
Число соседей	17	15	19	19
Время работы (сек)	1,02	1,17	1,18	1,18

Таблица 5: Результат обучения методом KNN

Как и в предыдущем пункте, для автоматизации подбора параметра K используется метод *GridSearchCV*, в качестве метрики выбрана метрика по умолчанию — расстояние Минковского, а также ближайшие соседи

имеют большее влияние на исследуемый объект, т. е. *weights = 'distance'*. Оценка качества обучения определяется также с помощью ROC-кривой.

После задания параметров проводится обучение, а также автоматический выбор наилучших значений настроек алгоритма после тестирования. Основные значения параметров метода K ближайших соседей, или метода KNN, и качество обучения для каждой из выборок, представлены в Таблице 5.

Для дальнейшего исследования выберем модель, обученную на «Четвертой» выборке, т. к. она показала наилучший результат.

Фрагмент программы, реализовывающий обучение методом KNN представлен в Приложении А.2.

3.3.4. Случайный лес (Random forest)

Случайный лес — один из алгоритмов машинного обучения. Основан на использовании ансамбля деревьев принятия решений. Решение о присваивании объекту определенного класса принимается на основе голосования — берется наиболее распространенный класс среди всех деревьев. Алгоритм хорошо работает с данными при большом числе признаков и классов. Еще одним из преимуществ метода является его способность к распараллеливанию и масштабированию. Как и предыдущие методы, данный алгоритм в рамках этой работы применяется для определения степени схожести двух пользователей (узлов) в социальном графе.

Для реализации этого метода также используется библиотека *scikit learn*. В ее модуле *ensamble* имеется класс *RandomForestClassifier* [16], который и описывает метод случайного леса.

Глубина решающих деревьев определяется с помощью метода скользящего контроля, как определяется и количество деревьев. Параметр *max features* определяет максимальное число рассматриваемых признаков в узле при построении решающего дерева, и также находится методом скользящего контроля.

Кросс-валидация представлена методом *GridSearchCV*, который упоминался выше. Оценка качества обучения определяется также с помощью ROC-кривой. Значения основных параметров, качество обучающей выборки, а также итог обучения представлены в Таблице 6.

Для дальнейшего исследования выберем модель, обученную на «Чет-

	«Первая»	«Вторая»	«Третья»	«Четвертая»
Качество	0,6	0,6	0,6	0,61
Число деревьев	4000	4000	4000	4000
Глубина деревьев	4	4	4	4
Число признаков в узле	3	2	2	2
Результирующие коэффициенты	[0,591; 0,128; 0,068; 0,037; 0,018; 0,015; 0,131; 0,011]	[0,252; 0,047; 0,093; 0,215; 0,126; 0,135; 0,064; 0,067]	[0,492; 0,020; 0,103; 0,153; 0,068; 0,104; 0,024; 0,036]	[0,489; 0,039; 0,032; 0,096; 0,131; 0,084; 0,063; 0,065]
Время работы	3мин 39сек	3мин 46сек	4мин 8сек	4мин 7сек

Таблица 6: Результат обучения методом RandomForest

вертой» выборке, т. к. она показала наилучший результат.

Фрагмент программы, реализовывающий обучение методом случайного леса представлен в Приложении А.3.

3.4. Результаты

В данном параграфе приведены результаты работы метода с частичным привлечением учителя для определения возраста пользователей при использовании различных способов нахождения коэффициентов схожести между узлами социального графа.

Для подсчета точности определения возраста друзей пользователей (вторая задача) из тех их друзей, которые указали возраст в своем профиле, было выбрано в среднем около 30 человек в каждом социальном графе. Этих друзей при предобработке данных не относили ни к одному известному классу. Кроме того, точно как и в оценке эвристических методов не учитываются выбросы (шум) — пользователи, которые указали неверный возраст, т. е. разница в найденном и указанном значениях достаточно велика (больше 40 лет).

Матрица смежности	Машинное обучение		
	Лог. регрессия	KNN	Случайный лес
4,83	4,50	4,51	4,39

Таблица 7: Среднеквадратичные ошибки при определении возраста друзей пользователя

Матрица смежности	Машинное обучение		
	Лог. регрессия	KNN	Случайный лес
1,43	0,93	1,36	0,76

Таблица 8: Среднеквадратичные ошибки при определении возраста пользователя

В Таблице 7 приведены среднеквадратичные ошибки определения возраста всех пользователей в одном социальном графе при различных подходах (первая задача), а в Таблице 8 — среднеквадратичные ошибки в нахождении возраста самих центров рассматриваемых эго-сетей (вторая задача). Как было указано в первой главе, всего таких графов 1000 штук.

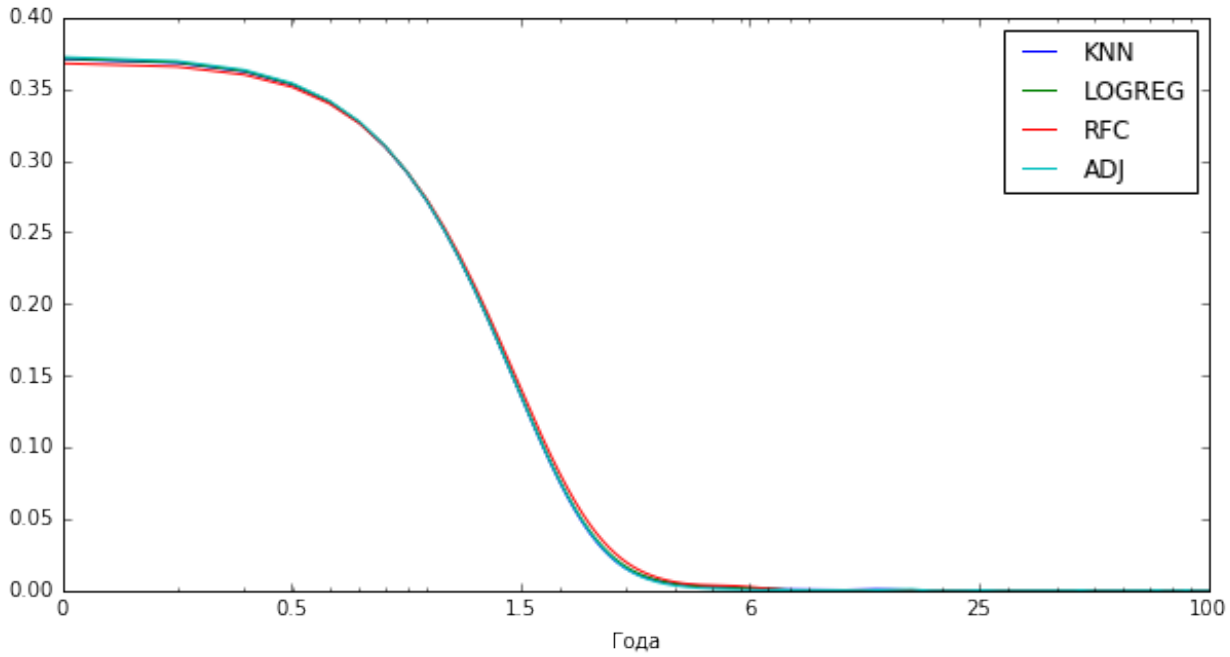


Рис. 5: Плотность распределения ошибок для центра эго-сети (логарифмическая шкала)

На Рис. 5 изображена диаграмма, на которой представлена плотность распределения ошибок (абсолютная разница между найденным и реальным возрастом) рассматриваемых подходов и метода среднего значения, изученного во второй главе. Данный график демонстрирует результаты

эксперимента на задаче определения атрибута для пользователя по его эго-сети.

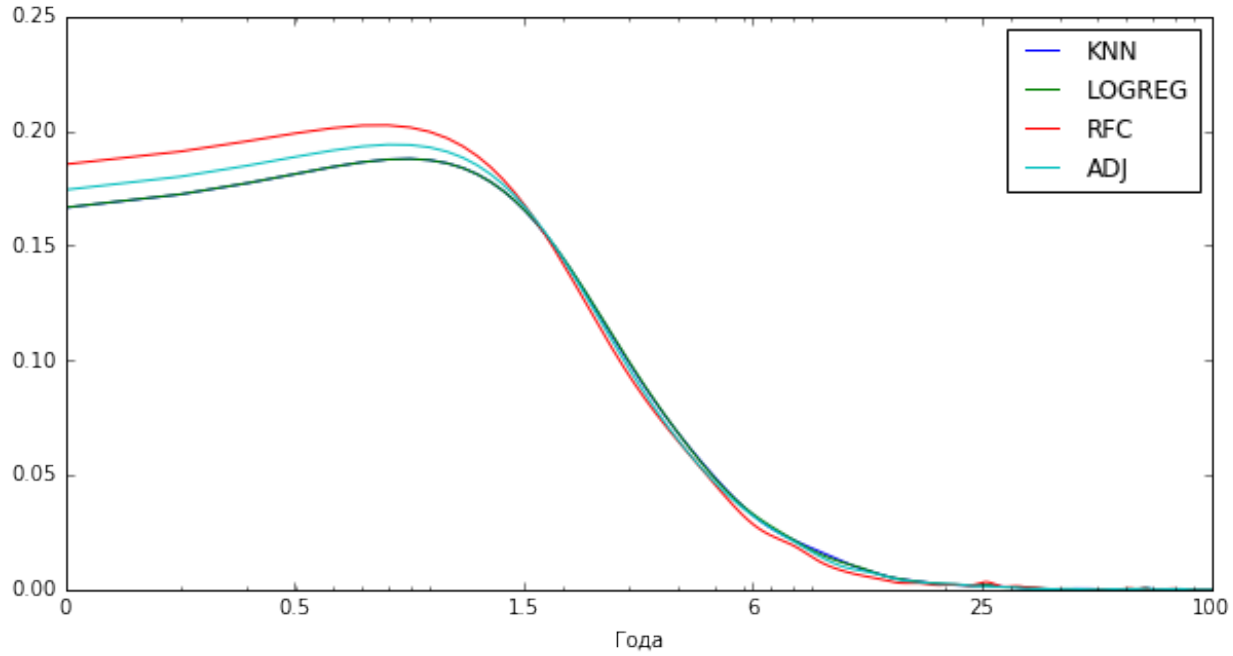


Рис. 6: Плотность распределения ошибок для всех пользователей (логарифмическая шкала)

На Рис. 6 представлены аналогичные результаты эксперимента на задаче определения скрытого атрибута для всех пользователей рассматриваемой социальной сети.

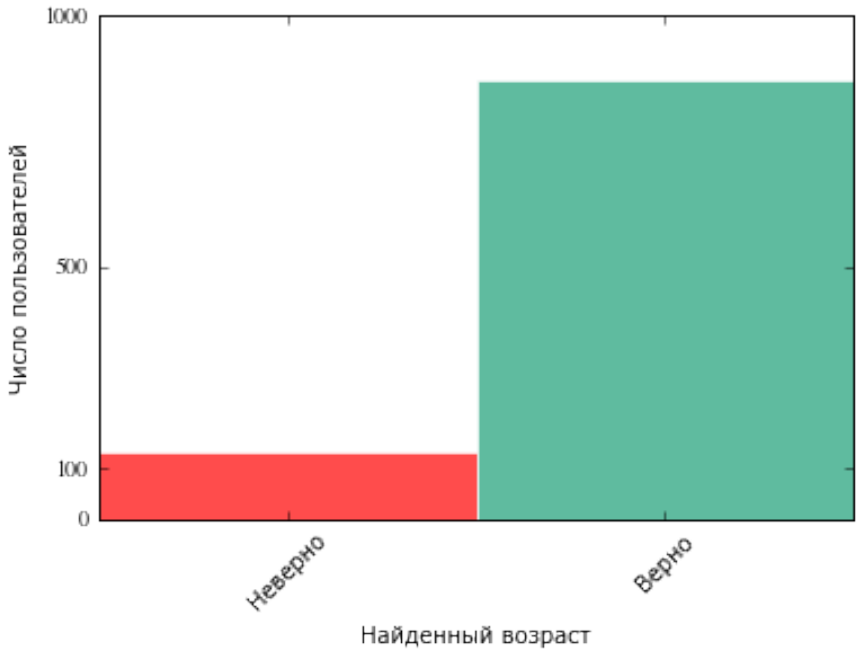


Рис. 7: Количество верно и неверно найденных возрастов с точностью до года

Рис. 7 демонстрирует, у скольких пользователей удалось верно определить возраст хотя бы одним методом с точностью до года, а у скольких нет. Рис. 8 показывает тоже самое, но с точностью до 3 лет.

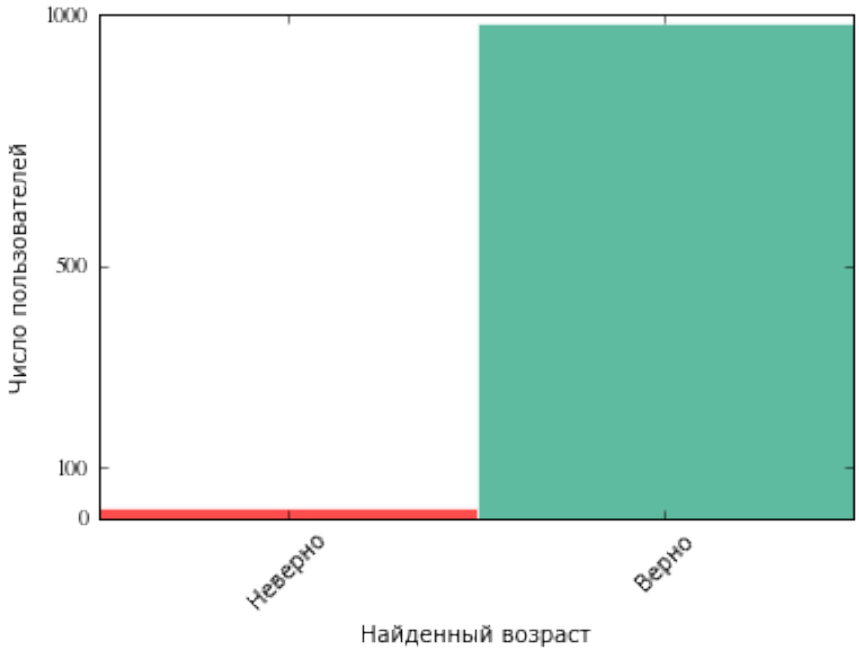


Рис. 8: Количество верно и неверно найденных возрастов с точностью до трех лет

3.5. Вывод

Основываясь на приведенных выше результатах, можно сделать вывод, что рассмотренный метод обучения с частичным привлечением учителя достаточно эффективно определяет скрытые атрибуты пользователя (центра эго-сети), базируясь на известной информации из профилей его друзей и дружественных связях. Кроме того, метод способен определять скрытые атрибуты всех пользователей в заданном социальном графе с некоторой погрешностью. Таким образом, предложенный метод способен решать обе задачи.

Для данного метода можно задавать различные метрики и добиваться таким образом наилучшего результата. Среди предложенных подходов к нахождению меры схожести узлов в социальном графе самым эффективным оказался метод *Случайного леса*, но его существенный недостаток — долгое время выполнения. Поэтому, если необходимо добиться быстрого отклика, лучше использовать метод *Логистической регрессии*.

Глава 4. Задача линейного программирования

4.1. Постановка задачи

В работе [8] приведена идея сведения задачи классификации на графах к задаче линейного программирования. Попробуем применить эту идею для определения скрытых атрибутов пользователей социальной сети на примере атрибута «Возраст». В целом исследуемый способ способен решить обе поставленные задачи.

Введем обозначение графа $G = (P, E)$, где E — множество ребер. Пусть P — множество n объектов (узлов в графе G), которые необходимо классифицировать, а L — множество k меток (классов). Число меток $a \in L$ для тестовой задачи определения возраста задается как было оговорено ранее — примем k равным 100. Разметка (или маркировка) множества P над множеством L — это функция $f : P \to L$, ставящая в соответствие метку каждому объекту. Качество маркировки зависит от следующих понятий:

1. Для каждого объекта $p \in P$ задана некоторая числовая характеристика — неотрицательная функция стоимости $c(p, a)$, где a — метка, присваиваемая объекту p. В рамках данной работы определим функцию стоимости по формуле

$$c(p, a) = \begin{cases} |t - a|, & \text{если значение } t \text{ известно,} \\ 1, & \text{иначе.} \end{cases}$$

   где t — значение атрибута «Возраст» объекта p. Кроме того, можно рассмотреть сглаживание этой функции, например, $\log(1 + |t - a|)$, но это вопрос дальнейших исследований.

2. Связи между объектами. Если элементы p и q связаны друг с другом, то более вероятно, что они будут иметь одинаковые (или близкие) метки. Применительно к графу это означает следующее: ребро $e = (p, q), e \in E$ между двумя вершинами p и q показывает, что данные объекты связаны между собой, и вес ребра w_e определяет силу этой связи. Вес w_e может задаваться разными способами. В рамках данной работы он находится с помощью машинного обучения, как и в предыдущем методе.

В статье [8] рассматривалась задача целочисленного линейного про-

граммирования, в рамках же данной работы исследуется релаксация этой задачи, т. е. условие целочисленности решения опускается.

Основываясь на этом, введем следующие определения. Пусть x_{pa} — неотрицательная величина, определенная для каждого объекта p и метки a. Эта величина должна удовлетворять условию $\sum_{a \in L} x_{pa} = 1$; ее можно интерпретировать как вероятность того, что объект p имеет метку a. Стоимость присваивания метки объекту p выражается как $\sum_{a \in L} c(p, a) x_{pa}$. Определим расстояние между метками объектов p и q как $d(f(p), f(q)) = \frac{1}{2} \sum_{a \in L} |x_{pa} - x_{qa}|$. Далее введем понятия разделяющей стоимости для ребра $e = (p, q)$, которая определяется выражением $\frac{1}{2} w_e \sum_{a \in L} |x_{pa} - x_{qa}|$. Для краткости введем переменную z_e для ребра $e = (p, q)$, чтобы выразить дистанцию между метками, присвоенных объектам p и q, а также переменную z_{ea}, чтобы выразить абсолютное значение $|x_{pa} - x_{qa}|$ для меток $a \in L$. Теперь можно формально описать задачу линейного программирования:

$$
\begin{aligned}
&\sum_{e \in E} w_e z_e + \sum_{p \in P, a \in L} c(p, a) x_{pa} \to \min, \\
&\sum_{a \in L} x_{pa} = 1, && p \in P, \\
&z_e = \tfrac{1}{2} \sum_{a \in L} z_{ea}, && e \in E, \\
&z_{ea} \geq x_{pa} - x_{qa}, && e = (p, q), a \in L, \\
&z_{ea} \geq x_{qa} - x_{pa}, && e = (p, q), a \in L, \\
&x_{pa} \geq 0, && a \in L, p \in P.
\end{aligned}
\tag{5}
$$

Обзор возможного решения данной задачи рассмотрено в следующем параграфе. Воспользуемся уже изученными в предыдущей главе подходами к определению весов ребер w_e графа G.

4.2. Решение задачи

Рассмотрим один из возможных способов решения задачи (5).

Исследуемая задача имеет достаточно большую размерность, так как пользователей в одном социальном графе в среднем двести, а число меток принимается равным ста, т.е. всего в среднем около двадцати тысяч переменных. Это говорит о том, что прямое применение симплекс-метода становится весьма трудоемким, а зачастую и вовсе невозможным.

Однако, существуют программные пакеты с реализацией методов, пригодных для решения задач такой размерности. К сожалению, далеко не

все из них открыты и бесплатны. В рамках данной работы рассмотрим свободно распространяемый солвер COIN-OR [20], для которого в Python есть «обертка» — PuLP [21]. Библиотека PuLP позволяет пользователям описывать математические задачи на декларативном языке. Работает в пределах синтаксиса языка Python и использует его природные конструкции, что делает ее достаточно удобной и понятной. Как уже было отмечено выше, PuLP имеет лицензию с открытым исходным кодом и распространяется бесплатно для широкого диапазона платформ.

Для того, чтобы воспользоваться методами решения библиотеки PuLP, необходимо описать поставленную задачу в нотациях этой библиотеки, подробнее в документации [22]. После того, как описаны все переменные, ограничения и целевая функция, необходимо записать файл в формате LP [23], в котором сохраняется созданная модель. Затем в выбранный метод решения подается записанная модель с помощью функции *LpProblem.solve* и на выходе получается решение рассматриваемой задачи.

Библиотека PuLP — отличное и бесплатное решение для подобных задач линейного программирования. Таким образом, в качестве дальнейшей работы можно рассматривать реализацию решения поставленной задачи линейного программирования с помощью рассмотренной библиотеки, а также анализ полученного результата. Стоит заметить, что в рамках библиотеки PuLP можно переходить от одного солвера к другому (даже коммерческому) путем манипуляции параметрами в функции *LpProblem.solve*.

4.3. Вывод

Метод сведения задачи классификации на графе к задаче линейного программирования способен определять скрытые атрибуты в обоих случаях: как для центра эго-сети, так и для всех пользователей сразу. Такая постановка задачи линейного программирования может оказаться эффективнее метода обучения с частичным привлечением учителя в связи с тем, что в ней вводится понятие стоимости присваивания метки $c(p, a)$, или, штрафа. То есть, варьируя этот параметр (помимо весов ребер графа), можно добиваться наилучших результатов. Однако, время работы данного метода будет существенно выше метода машинного обучения. Кроме того, представление качества работы метода на текущих тестовых данных — предмет будущей работы.

Анализ методов

В текущем разделе представлена сравнительная характеристика методов, предложенных во 2 и 3 главах, на основе проведенных экспериментов и полученных результатах.

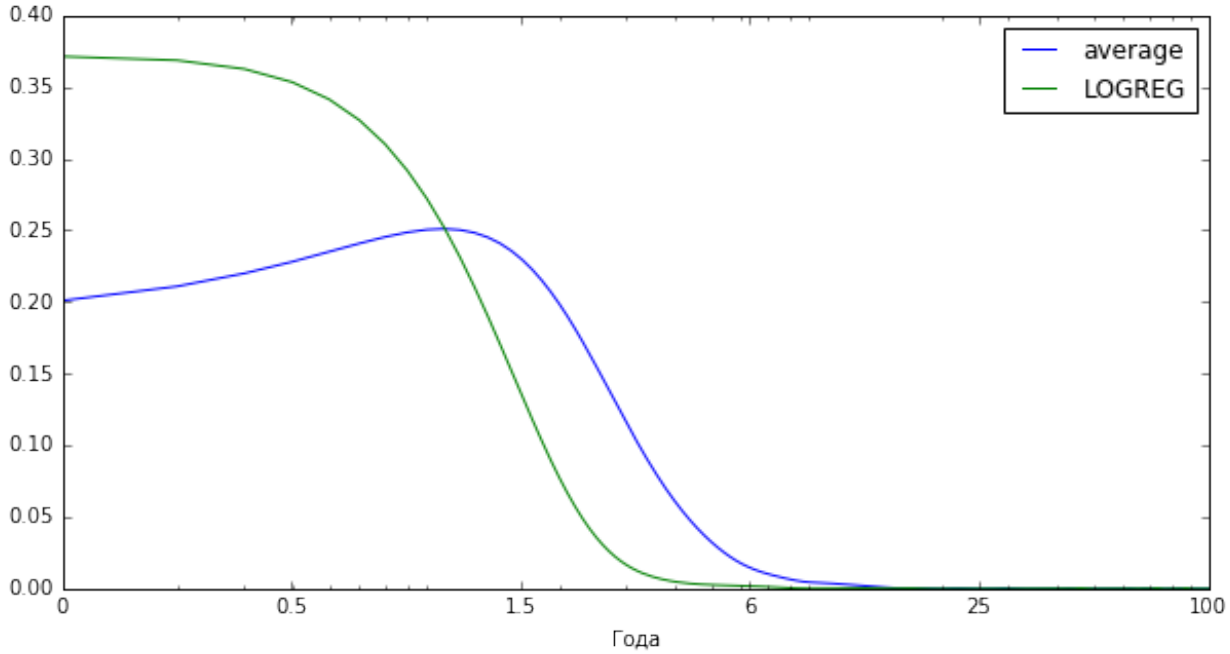


Рис. 9: Плотность распределения ошибки для центра эго-сети (логарифмическая шкала)

С одной стороны, метод обучения с частичным привлечением учителя показал более качественное определение скрытого атрибута пользователя (центра эго-сети), нежели эвристические алгоритмы (решение первой задачи). Это демонстрирует Рис. 9, на котором представлены результаты работы метода машинного обучения на основе метрики, полученной с помощью *Логистической регрессии*, и метода среднего значения атрибутов в социальном графе. Графики дают наглядное представление о плотности распределения ошибок (абсолютной разницы между реальным и найденным значениями атрибута) указанных методов.

Рис. 10 демонстрирует то же самое для всех рассмотренных в рамках данной работы алгоритмов. Сюда входят результаты пяти эвристических методов из главы 2 и обучения с частичным привлечением учителя на различных метриках из главы 3.

С другой стороны, эвристические алгоритмы работают гораздо быстрее и не требуют подготовительных расчетов (некоторые работают на основе кластеризации), в то время как для метода обучения необходимо сначала

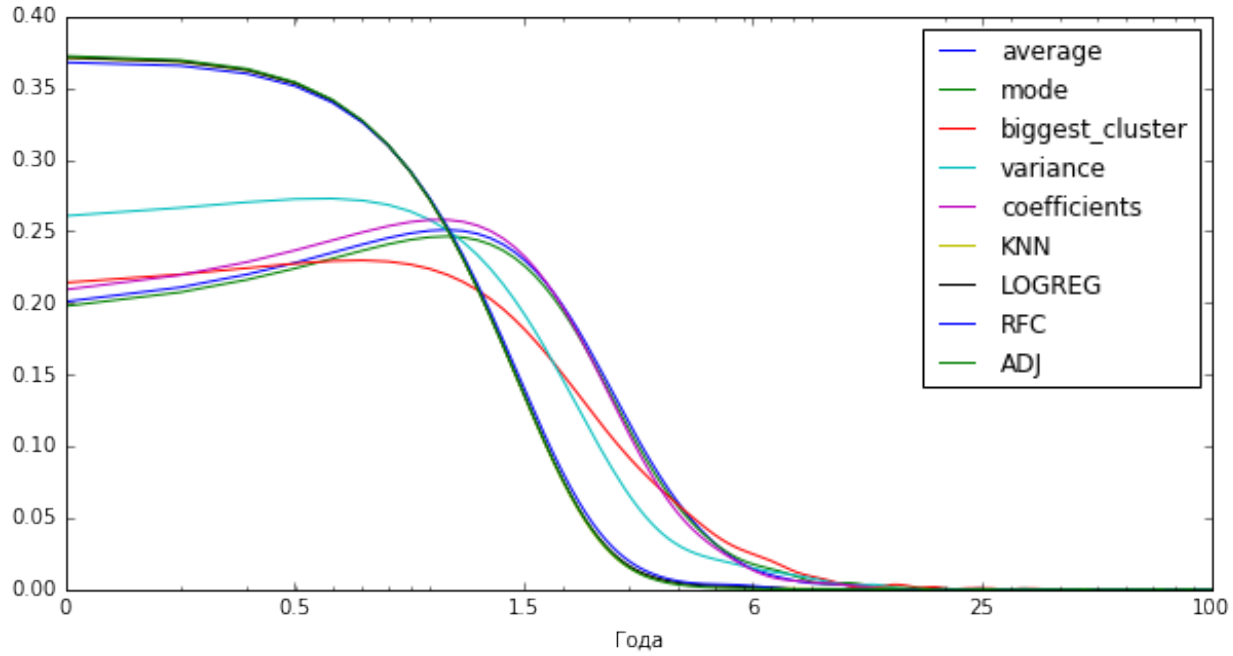


Рис. 10: Плотность распределения ошибки для центра эго-сети (логарифмическая шкала)

произвести подсчет метрик.

Таким образом, ответ на вопрос о том, какой метод применить в задаче определения скрытого атрибута пользователя на основе его эго-сети, зависит от конечной цели: скорость или точность.

Как было замечено в выводах второй главы, эвристические алгоритмы не способны определять скрытый атрибут всех пользователей рассматриваемой социальной сети одновременно, наравне с центром этой сети. Поэтому сравнение эффективности работы методов невозможно для второй задачи. Из предложенных метрик для метода из главы 3 самой точной оказалась метрика, найденная с помощью модели *Случайного леса* (см. Таблицу 7 из третьей главы).

Рассмотренная в четвертой главе задача линейного программирования имеет достаточно большую размерность, поэтому ее решение и исследование эффективности — это возможности для дальнейшей работы.

Также результаты работы предложенных алгоритмов должны быть не хуже для других атрибутов в обеих задачах, так как значения коэффициентов социального влияния (см. Таблицу 1 из первой главы) не сильно меньше, а у некоторых даже и больше, но это тоже тема для дальнейших исследований.

Заключение

В рамках данной работы реализован и исследован общий метод определения скрытых атрибутов пользователей социальных сетей. Более того, проведены эксперименты на двух задачах: определение атрибутов пользователя по его эго-сети; определение атрибутов всех пользователей в рассматриваемом социальном графе.

В начале исследования проведен анализ социальных сетей, в результате которого выбрана наиболее подходящая сеть, а также отражен процесс загрузки данных и их хранение.

В ходе работы установлена зависимость между равенством атрибутов и дружбой пользователей, которая позволила разработать эвристические методы для определения атрибута пользователя на основе информации из его эго-сети. Рассмотрен ряд недостатков этих методов и способ их устранения — метод обучения с частичным привлечением учителя. Для этого сформулирована оптимизационная задача, предложен способ ее решения и проведено исследование с различными метриками на двух поставленных задачах. Кроме того, в качестве возможного решения поставленных задач адаптирована задача линейного программирования и предложен способ ее решения.

В результатах работы отражен анализ предложенных методов для обеих поставленных задач. Метод обучения с частичным привлечением учителя показал наиболее точные результаты, достигая тем самым цели работы.

Список литературы

[1] Executive.ru URL: http://www.e-xecutive.ru/pages/about (дата обращения: 10.04.2016).

[2] ВКонтакте URL: http://vk.com/about?w=page-47200925_44240810 (дата обращение: 10.04.2016).

[3] Facebook newsroom URL: http://newsroom.fb.com/company-info/ (дата обращения: 10.04.2016).

[4] Список социальных сетей // Википедия URL: https://ru.wikipedia.org/wiki/Список_социальных_сетей (дата обращения: 10.04.2016).

[5] Jure Leskovec, Julian J. Mcauley Learning to Discover Social Circles in Ego Networks // Advances in Neural Information Processing Systems 25. 2012. P. 548 -556.

[6] Yuxin Ding, Shengli Yan, YiBin Zhang, Wei Dai, Li Dong Predicting the attributes of social network users using a graph-based machine learning method // Computer Communications. 2016. No 73. P. 3 –11.

[7] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, Bernhard Schölkopf Learning with local and global consistency // Advances in neural information processing systems. 2004. No 16. P. 321 –328.

[8] Jon Kleinberg, 'Eva Tardos Approximation Algorithms for Classification Problems with Pairwise Relationships: Metric Labeling and Markov Random Fields // Journal of the ACM, Vol. 49. 2002. No 5. P. 616 -639.

[9] SimilarWeb // Википедия URL: https://ru.wikipedia.org/wiki/SimilarWeb (дата обращения: 10.04.2016).

[10] ВКонтакте URL: https://vk.com/dev (дата обращения: 10.04.2016).

[11] JSON URL: http://www.json.org/json-ru.html (дата обращения: 10.04.2016).

[12] Community detection for NetworkX's documentation // [Электронный ресурс]: URL: http://perso.crans.org/aynaud/communities/ (дата обращения: 08.03.2016).

[13] Dengyong Zhou and Christopher J. C. Burges Spectral clustering and transductive learning with multiple views // In Proceedings of the 24th iternational conference on Machine learning. ACM, 2007. ICML '07. P. 1159–1166.

[14] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, Bernhard Scholkopf Learning with local and global consistency // In Advances in Neural Information Processing Systems 16. MIT Press, 2004. P. 321–328.

[15] Konstantin Avrachenkov, Vladimir Dobrynin, Danil Nemirovsky, Son Kim Pham, Elena Smirnova Pagerank based clustering of hypertext document collections // In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2008. SIGIR '08. P. 873–874.

[16] Scikit learn URL: http://scikit-learn.org/stable/documentation.html (дата обращения: 15.04.2016).

[17] Кривая ошибок // MachineLearning.ru URL: http://www.machinelearning.ru/wiki/index.php?title=ROC-кривая (дата обращения: 16.04.2016).

[18] Метод ближайших соседей // MachineLearning.ru URL: http://www.machinelearning.ru/wiki/index.php?title=Метод_ближайшего_соседа (дата обращения: 16.04.2016).

[19] Скользящий контроль // MachineLearning.ru URL: http://www.machinelearning.ru/wiki/index.php?title=Скользящий_контроль (дата обращения: 16.04.2016).

[20] COIN-OR URL: http://www.coin-or.org (дата обращения 04.05.2016).

[21] PuLP 1.1 // Python URL: https://pypi.python.org/pypi/PuLP/1.1 (дата обращения: 04.05.2016).

[22] Optimization with PuLP // PuLP 1.6.0 documentation URL: http://pythonhosted.org/PuLP/ (дата обращения: 04.05.2016).

[23] LP file format URL: http://lpsolve.sourceforge.net/5.1/lp-format.htm (дата обращения: 04.05.2016).

Приложение А

А.1. Фрагмент программы, реализующий обучение модели логистической регрессии

```
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
import numpy as np
import pandas as pd
from sklearn.metrics import roc_auc_score
from sklearn.grid_search import GridSearchCV

p = "training_features.csv"
df = np.array(pd.read_csv(p, sep = "\t"))
X = df[:2:-1]
y = df[:-1]

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                        test_size = 0.3)

t = np.logspace(-5, 0, 100)
params = {"C": t}
grid_searcher = GridSearchCV(linear_model.
                LogisticRegreson(), params, cv = 5,
                scoring = "roc_auc")
grid_searcher.fit(X_train, y_train)

clf = grid_searcher.best_estimator_.fit(X_train, y_train)
results = clf.predict_proba(X_test)[:1]

print roc_auc_score(y_test, results)
```

A.2. Фрагмент программы, реализующий обучение методом KNN

```
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.cross_validation import train_test_split
import numpy as np
import pandas as pd
from sklearn.metrics import roc_auc_score
from sklearn.grid_search import GridSearchCV

p = "training_features.csv"
df = np.array(pd.read_csv(p, sep = "\t"))
X = df[:2:-1]
y = df[:-1]

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                        test_size = 0.3)

params = {"n_neighors": range(1, 20, 2), "weights" :
          ["distance"]}
grid_searcher = GridSearchCV(KNN(), params, cv = 5,
                 scoring = "roc_auc")
grid_searcher.fit(X_train, y_train)

clf = grid_searcher.best_estimator_.fit(X_train, y_train)
results = clf.predict_proba(X_test)[:1]

print roc_auc_score(y_test, results)
```

А.3. Фрагмент программы, реализующий обучение методом случайного леса

```
from sklearn.ensemble import RandomForestClassifier as RFC
from sklearn.cross_validation import train_test_split
import numpy as np
import pandas as pd
from sklearn.metrics import roc_auc_score
from sklearn.grid_search import GridSearchCV

p = "training_features.csv"
df = np.array(pd.read_csv(p, sep = "\t"))
X = df[:2:-1]
y = df[:-1]

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                    test_size = 0.3)
params = {"n_estimators" : [4000], "bootstrap" : [True],
          "max_depth" : [4, 5], "random_state" :
          [np.random.randint(1, 10000000)],
          "max_features" : [1, 2, 3],  "max_leaf_nodes" :
          [4, 5], "min_samples_split" : [2],
          "min_samples_leaf" : [2]}
grid_searcher = GridSearchCV(RFC(), params, cv = 5,
                  scoring = "roc_auc")
grid_searcher.fit(X_train, y_train)

clf = grid_searcher.best_estimator_.fit(X_train, y_train)
results = clf.predict_proba(X_test)[:1]

print roc_auc_score(y_test, results)
```