МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»

Факультет Информационные технологии и прикладная математика Кафедра 813
Направление подготовки 02.04.02 ФИИТ Группа М8О-213М-18
Квалификация (степень) магистр

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

На тему: Прогнозирование доходов пользователей социальных сетей по сетевому профилю

Автор диссертации Меркулов Михаил Владимирович (_____)
(Фамилия, имя, отчество)
Научный руководитель Филимонов Александр Борисович (_____)
(Фамилия, имя, отчество)
Рецензент Ивченко Валерий Дмитриевич (_____)
(Фамилия, имя, отчество)

К з а щ и т е д о п у с т и т ь

Зав. кафедрой Денисова И. П. (_____)
(Фамилия, инициалы)
“ 24 ” мая 2020 г.

Москва 2020 г.

РЕФЕРАТ

Общий объем работы составляет 62 страницы, среди которых 17 рисунков, 4 таблицы и 27 использованных источников.

СОЦИАЛЬНЫЕ СЕТИ, МАШИННОЕ ОБУЧЕНИЕ, RANDOM FOREST, EMBEDDINGS, NODE2VEC, DATA MINING

Магистерская диссертация посвящена проблематике применения DataMining для решения задачи прогнозирования доходов пользователей социальных сетей, на основе данных сетевого профиля и социального окружения в социальных сетях.

Цель диссертации: исследование перспектив использования данных о социальном окружении пользователей в социальных сетях для прогнозирования заработной платы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время повсеместное применение цифровых технологий в различных областях деятельности человека привело к значительному росту объема накапливаемой информации, это открывает возможность продуктивного анализа и применения данных, извлекаемых для принятия решений, с помощью методов интеллектуального анализа и технологий автоматизированного сбора данных из сети Интернет.

Магистерская диссертация посвящена проблематике применения Data-Minig для решения задачи прогнозирования доходов пользователей социальных сетей, на основе данных сетевого профиля и социального окружения в социальных сетях.

Стоит заметить, что политологические, социологические, маркетинговые и банковские информационно-аналитические системы [4] используют доход как один из ключевых показателей для принятия решений.

Особенность существующих решений рассматриваемой задачи [10, 14, 18, 19, 20, 25] заключается в том, что для прогнозирования дохода они используются только характеристики сетевого профиля пользователя. Например: возраст, пол, место жительства, сведения об образовании.

За последнее время в области машинного обучения появился ряд подходов позволяющих представлять части графа как точки в пространстве векторов с низкой размерностью – embeddings. Эти подходы открывают новые возможности для эффективного решения задач машинного обучения.

Графы социальных сетей содержат различную полезную информацию о своих пользователях и связях между ними [5]. С помощью графов социальных сетей решаются такие задачи, как генерация рекомендаций медиаконтента и новостей, а также таргетированная реклама.

Основную часть доходов большинства людей составляет заработная плата. Прогнозирование заработной платы позволяет с определенной точностью прогнозировать и доходы.

Цель диссертации: исследование перспектив использования данных о

социальном окружении пользователей в социальных сетях для прогнозирования заработной платы.

Предмет исследования – методы машинного обучения для получения embeddings на основе данных социальных сетей.

Объект исследования – проблематика прогнозирования заработной платы на основе данных о социальном окружении пользователей в социальных сетях.

В соответствии с целью исследования были выдвинуты следующие гипотезы, которые необходимо либо подтвердить, либо опровергнуть:

1. Существует определенная взаимосвязь между социальным окружением пользователя социальных сетей и размером его заработной платы.
2. Возможно эффективно прогнозировать размер заработной платы пользователей на основе информации о социальном окружении в социальных сетях.

В соответствии с целью исследования и выдвинутыми гипотезами, были поставлены следующие задачи:

1. Анализ современного состояния аспектов проблемы исследования.
2. Разработка программного обеспечения для автоматического сбора данных о сетевом профиле и заработной плате пользователей, а также о социальном окружении пользователей в социальных сетях.
3. Анализ перспектив применения методов с использованием данных о социальном окружении пользователей в социальных сетях для прогнозирования заработной платы.
4. Разработка методов решения для задачи прогнозирования зарплаты пользователей.
5. Разработка программных средств для решения задачи прогнозирования зарплаты пользователей.

ОСНОВНАЯ ЧАСТЬ

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 СОЦИАЛЬНЫЕ ГРАФЫ

1.1.1 СОЦИАЛЬНАЯ СЕТЬ

Социальная сеть – это онлайн-сервис, предназначенный для построения, отображения, и организации социальных взаимоотношений [2]. Чтобы считаться социальной сетью, сервис должен предоставлять следующие возможности:

1. Создание для каждого пользователя персональной публичной или частично публичной страницы профиля, содержащей его данные.
2. Создание и редактирование списка пользователей, с которыми данный пользователь состоит в определенных отношениях.
3. Возможность объединения пользователей в сообщества в соответствии с их интересами.

Социальные сети являются универсальным источником данных о личной жизни и интересах реальных людей. Они открывают широкие возможности для решения многих исследовательских и бизнес задач. Этим обуславливается повышенный интерес к сбору и анализу социальных данных со стороны компаний и исследовательских центров.

Специалисты из исследовательских центров и компаний по всему миру используют данные социальных сетей для моделирования социальных, экономических, политических и других процессов, с целью разработки механизмов воздействия на эти процессы, а также создания аналитических и бизнес-приложений и сервисов.

Поскольку сценарии использования интерфейсов социальных сетей не предполагают автоматического сбора данных из профилей, во время сбора возникает ряд проблем:

1. Приватность данных. Зачастую доступ к данным пользователей разрешён только для зарегистрированных и авторизованных участников сети, что требует поддержки эмуляции пользовательской сессии с по-

мощью специальных учётных записей.

2. Ограничения доступа и блокировки. С целью предотвращения несанкционированного автоматического сбора данных и ограничения нагрузки на инфраструктуру сервиса социальной сети, владельцы сервиса вводят явные или скрытые ограничения на допустимое количество запросов от одного пользовательского аккаунта и/или IP-адреса в единицу времени.
3. Размерность данных обуславливает необходимость в параллельном методе сбора данных, а также в методах получения репрезентативной выборки пользователей социальной сети.
4. Истинность данных. При заполнении своего профиля в социальной сети пользователи зачастую по ошибке или преднамеренно не заполняют некоторые поля либо дают ложную информацию о фактах своей биографии, интересах и предпочтениях. Кроме того, в контентных сетях (Twitter, YouTube) пользовательский профиль часто ограничен набором базовых атрибутов, недостаточным для решения многих задач, предполагающих персонализацию результатов.
5. Обновление и изменение функционала и модели социальной сети, вследствие ее дальнейшего развития.

1.1.2 СОЦИАЛЬНЫЙ ГРАФ

Социальный граф – это граф, узлами которого являются социальные объекты: пользовательские профили с различными атрибутами, сообщества, медиа-контент и т.д., а ребра – это социальные связи между ними (см. рис. 1.1).

Социальные графы обладают двумя полезными свойствами:

1. В большинстве социальных графов присутствует одна большая компонента связности, которая захватывает большинство вершин. Например, в социальной сети Facebook 99.91% вершин находятся в одной компоненте связности.
2. Социальные графы в среднем имеют малое расстояние между двумя

случайными вершинами. Это объясняется теорией шести рукопожатий: любые два человека на Земле разделены в среднем пятью уровнями общих знакомых.

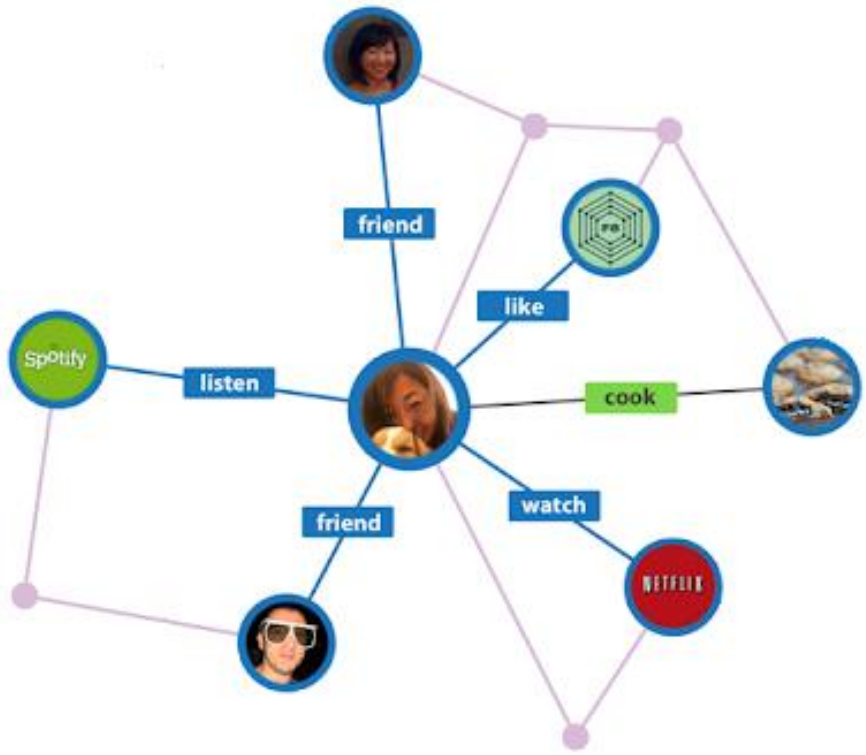

Рис. 1.1 Пример социального графа

В задачах с использованием социальных графов используют понятие метрик - показателей, которые в числовой форме отображают характеристики социальных объектов, групп объектов и их связей.

Характер взаимоотношений одного социального объекта с другими отображают метрики взаимодействий:

1. Гомофилия – характеризует степень, в которой пользователь образует связи с себе подобными. Сходство может быть определено по полу, возрасту, социальному статусу, уровню образования и т.д.
2. Множественность – число связей, соединяющих одну вершину с другой.
3. Взаимность – характеризует степень, в которой два пользователя отвечают взаимностью на действия друг друга.

Особенности связей отдельных объектов и всего графа в целом отображают метрики связей:

1. Мост – пользовательский профиль, чьи слабые связи заполняют пропуски, обеспечивающие единственную связь между другими профилями или кластерами профилей.
2. Центральность – степень, характеризующая важность или меру влияния определенного пользователя в графе.
3. Плотность – доля прямых связей в сети по отношению к общему числу возможных.

Кластеры социального графа, имеющие отличительные друг от друга особенности, отображают метрики сегментаций:

1. Коэффициент кластеризации – вероятность того, что два сетевых профиля, связанные с конкретным сетевым профилем, связаны между собой.
2. Сплоченность – характеризует степень, в которой профили связаны между собой одной общей связью.

1.1.3 ГРАФ ИНТЕРЕСОВ

Неявный социальный граф – это граф, формируемый на основе социальных связей в исходном социальном графе.

Граф интересов – это неявный социальный граф, состоящий из вершин пользовательских профилей и вершин интересов, соединенных связями.

Связи в графе интересов бывают трех типов:

1. Пользовательский профиль – пользовательский профиль. Пользователи в социальной сети взаимодействуют напрямую.
2. Пользовательский профиль – интерес. То чем интересуется пользователь в социальной сети.
3. Интерес – интерес. В случае схожих или взаимосвязанных интересов.

С помощью графа интересов можно получить много полезной информации. Например, что человек хочет сделать или купить, куда хочет пойти, с кем может встретиться, за чьими сообщениями ему интересно следить или за кого он готов проголосовать.

Граф интересов также может быть представлен в виде взвешенного гра-

фа, в этом случае вес ребра означает меру взаимосвязи между вершинами. При построении такого графа изначально вводится предположение о том, что изначально никаких взаимосвязей не существует.

Например, взаимосвязь интересов к спорту и здоровому питанию изначально неизвестна, поэтому вес ребра между этими двумя интересами устанавливается в ноль. Затем, если будет обнаружено, что люди, интересующиеся спортом, также увлекаются и здоровым питанием, то значение веса ребра между вершинами, обозначающими данные увлечения, будет увеличено.

Существует несколько способов использования графа интересов. В сочетании с социальным графом, граф интересов может быть применён для установления связей между пользователями в социальных сетях или в реальном мире. В таких сетях пользователи могут указывать и делиться своими увлечениями, но при этом им не обязательно знать друг друга.

Граф интересов так же может быть применён в маркетинге, в целях анализа аудитории проекта и дальнейших продаж на основе этой информации, для анализа тональности текста и для таргетированной рекламы.

Также его можно использовать при создании продукции с учётом пожеланий потребителя, он помогает определить какие особенности и возможности следует предоставить в следующих выпусках или версиях.

1.1.4 ИДЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ

Несмотря на то, что существуют попытки по обеспечению единого способа взаимодействия между различными социальными платформами (например, OpenSocial), они не получили широкого применения, а новые социальные сервисы продолжают появляться.

Одной из фундаментальных проблем при использовании социальной информации о пользователе является её фрагментированность среди множества различных онлайновых социальных сетей и сайтов [3].

Существует множество как универсальных, так и узкоспециализированных сервисов. Для активных пользователей сети Интернет свойственно иметь несколько профилей в различных социальных сетях и на различных сайтах.

Идентификация пользователя в различных социальных сетях или на сайтах позволяет получить более полную картину о социальном поведении данного пользователя в сети Интернет. Обнаружение аккаунтов, принадлежащих одному человеку, в нескольких социальных сетях, позволяет получить более полный социальный граф, что может быть полезно во многих задачах, таких как информационный поиск, интернет-реклама, рекомендательные системы и т.д.

Большинство пользователей сами выкладывают информацию о себе в публичный доступ, указывая: фамилию, имя, отчество, место жительства, сведения об образовании, номера телефонов, электронную почту, а также фотографии. Получая эти данные в одной социальной сети или на одном сайте, можно проводить однозначную идентификацию пользователя на других сайтах или в других социальных сетях. Однако при этом стоить учитывать, что некоторые поля по ошибке или преднамеренно могут быть либо пропущены, либо заполнены ложной информацией.

1.2 МАШИННОЕ ОБУЧЕНИЕ

1.2.1 ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Искусственный интеллект – это наука и технология создания интеллектуальных машин, алгоритмов и компьютерных программ. Машинное обучение – это подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться.

Обучение с учителем – раздел машинного обучения, посвященный решению следующей задачи. Дано множество объектов и соответствующее им множество ответов. Требуется построить алгоритм (отображение), способный для любого объекта выдать соответствующий ему ответ. Под учителем понимается множество объектов и соответствующее им множество ответов.

Не все ответы должны обязательно быть точными. Для определения точности ответов алгоритма вводится определенная мера качества.

С помощью обучения с учителем решаются следующие задачи:

1. Классификация. В задаче классификации множество ответов, меток класса, конечно. Класс – это множество всех объектов с заданным значением метки.
2. Регрессия. От задачи классификации отличается тем, что множество ответов состоит из действительных чисел или числовых векторов.

Формально задачу классификации можно описать следующим образом. Входные данные: матрица X_{nm}, состоящая из m-мерных строк с характеристиками n объектов x_n, и вектор $Y = (y_1, \dots, y_n)$, состоящий из меток классов для объектов. Каждый объект x_n является точкой в m-мерном пространстве характеристик. Целью является нахождение отображения $f: x_n \twoheadrightarrow y_n$ сопоставляющего каждому объекту его метку класса.

Обучение без учителя – раздел машинного обучения, посвященный решению задач, в которых дано только множество объектов, для которого требуется обнаружить внутренние взаимосвязи, зависимости или закономерности.

Кластеризация – это одна из задач, решаемых в обучении без учителя. Ее суть заключается в разделении объектов на непересекающиеся подмножества, кластеры, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались друг от друга.

Обучающее множество – часть объектов, используемых для обучения алгоритма. Тестовое множество – часть объектов, используемых для определения качества работы алгоритма. Обучающее и тестовое множества не должны пересекаться.

Каждый объект описывается набором характеристик. Характеристики могут быть количественными или категориальными.

1.2.2 ДЕРЕВЬЯ РЕШЕНИЙ

Дерево решений – это метод, задающийся связным ациклическим графом [15], разделяющий пространство характеристик объектов в набор гиперпрямоугольников.

Дерево состоит из узлов, “ветвей” и “листьев”, как на рисунке 1.2. Узел

представляет собой функцию, по значению которой происходит переход в один дочерних узлов. Ветвь – это ребро, содержащее одно из возможных значений функции родительского узла. Лист – это узел, атрибутом которого является, в случае классификации, значение класса. Работа алгоритма начинается из корневого узла.

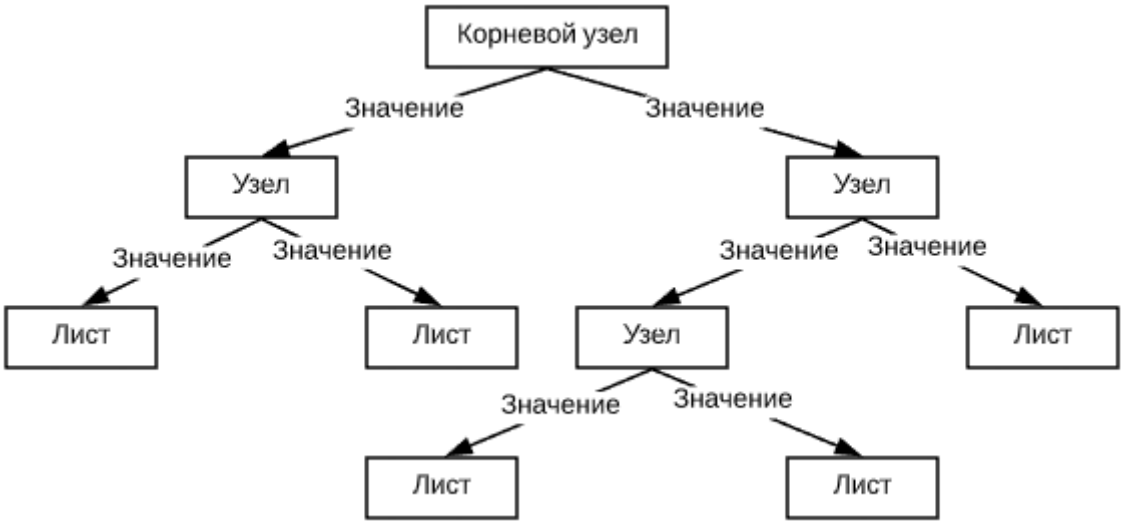

Рис 1.2 Пример дерева решений

Формально, дерево решений – это функция $f(x) = (v_0, V, L, S_v, B_v)$, где:

1. v_0 – корневой узел.
2. V – множество узлов.
3. L – множество листьев.
4. $B_v: X \rightarrow D_v$ - функция ветвления.
5. $S_v: D_v \rightarrow V_v$ – функция перехода по значению функции ветвления.

Бинарное дерево решений – это дерево решений, в котором $D_v = \{0,1\}$.

Пусть дана задача классификации объектов по двум признакам X_1 и X_2. На рисунке 1.3 показано разделение пространства объектов линиями, параллельными координатным осям. Каждой области пространства соответствует определенная метка класса: Y_1, Y_2, Y_3, Y_4, Y_5. Некоторые из областей, например Y_2, можно довольно просто описать двумя условиями: $X_1 < c_1$ и $c_2 < X_2 < c_3$. Тогда как другие, например Y_3, намного сложнее.

Пусть дана задача классификации объектов по двум признакам X_1 и X_2.

На рисунке 1.4 показано разделение пространства объектов линиями, параллельными координатным осям. Каждой области пространства соответствует определенная метка класса: R_1, R_2, R_3, R_4, R_5.

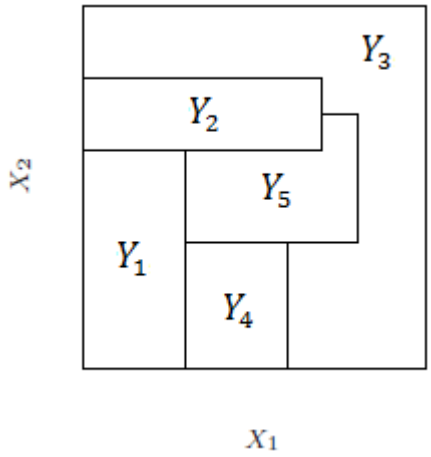

Рис 1.3 Сложное разделение объектов в двумерном пространстве

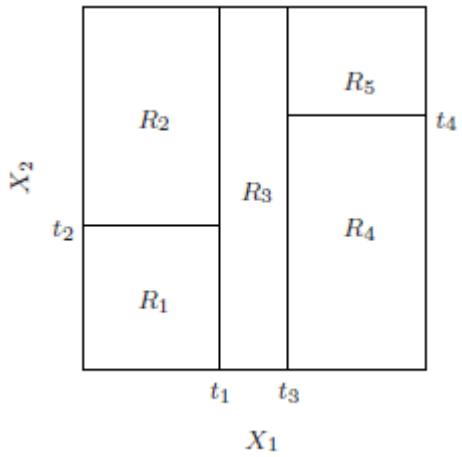

Рис 1.4 Простое разделение объектов в двумерном пространстве

В рамках алгоритма построения бинарного дерева решений, для рисунка 1.4, пространство сначала будет разбито на две области. Выбирается характеристика и ее значение, соответствующие лучшему разделению, в данном случае это - значение t_1 для характеристики X_1. В обучаемом дереве будет добавлен корневой узел, с переходами в два дочерних узла в зависимости от значения характеристики t_1. Затем одна или обе полученные области будут разбиты еще на подобласти. Процесс продолжается до тех пор, пока не будет применено определенное правило остановки. Полученное в результате построения дерево решений может быть представлено графом на рисунке 1.5.

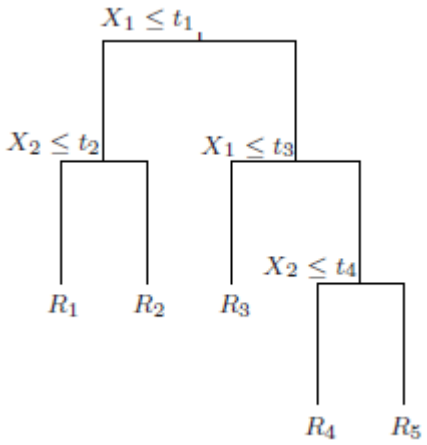

Рис 1.5 Дерево решений, полученное в результате простого разделения

В не зависимости от решаемой задачи предпочтительнее использовать бинарное разделение, нежели многолинейное [1]. Поскольку при многолинейном разделении объектов на каждом этапе, обучающее множество может закончиться слишком быстро, не оставив достаточно информации для разделения на последующих этапах. Многолинейное разделение может быть заменено серией бинарных разделений, при этом произойдет меньшее искажение информации на каждом последующем этапе и, следовательно, будет достигнута более качественная классификация.

Основной проблемой деревьев решений является высокая дисперсия для результирующего значения. Небольшое изменение в обучающем множестве может привести к сильно отличающимся друг от друга сериям разделений. Причиной этого является иерархическая структура дерева: ошибка, совершенная на верхнем этапе разделения, распространяется на все нижние этапы.

Также проблемой деревьев решений является необходимость наиболее равномерного распределения объектов по классам. Построение деревьев решений требует определение оптимального разделения в каждом узле. Однако выбор наилучшего результата на текущем шаге не гарантирует, что на последующих будет достигнуто оптимальное решение.

Например, пусть в рамках задачи классификации дано обучающее множество, состоящее из 100 объектов с большим количеством характеристик m, разделенных на 2 класса. 95 объектов относятся к первому классу, 5 ко вто-

рому.

Оптимальным решением разбиения на первом шаге будет присвоение любому объекту метки со значением 1. В результате получится дерево, состоящее из одного корневого листа со значением класса 1. На обучающем множестве данное дерево будет работать с точностью 95%.

Пусть дано тестовое множество, состоящее из 100 объектов. Первые 40 объектов относятся к классу 1, а следующие 60 объектов к классу 2. В этом случае точность на тестовом множестве составит всего 40%.

1.2.3 БЭГГИНГ

В контексте задач классификации, бэггинг – это метаалгоритм, предназначенный для уменьшения дисперсии метки класса путем усреднения определенного количества классификаторов [15]. Схема работы бэггинга приведена на рисунке 1.6.

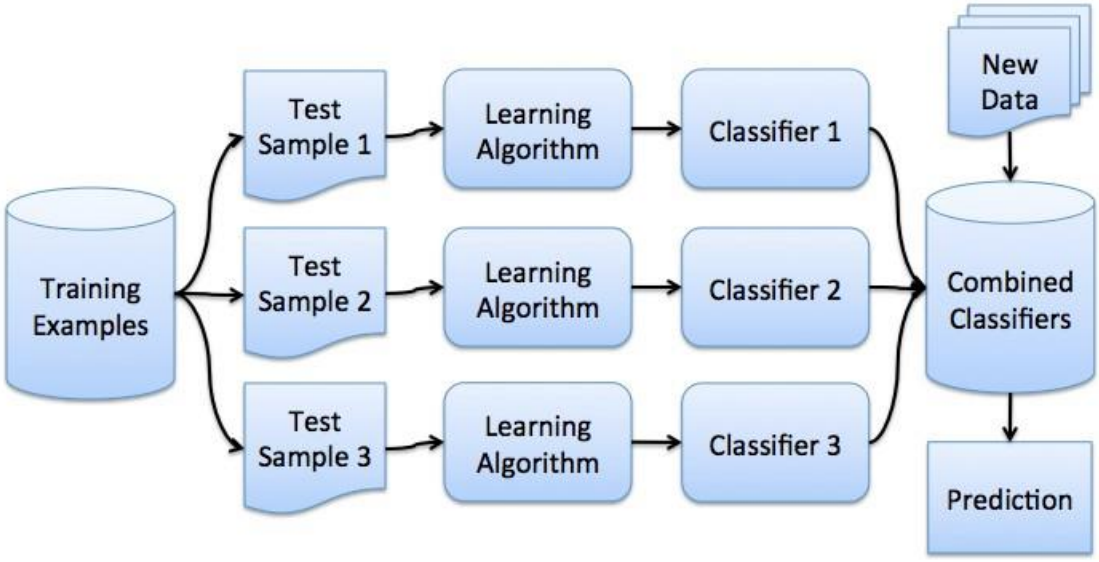

Рис 1.6 Схема работы бэггинга

Усреднение представляет собой голосование. Каждый классификатор независимо от других определяет значение метки класса для объекта. Окончательное решение о значении метки принимается во время голосования любым способом, например:

1. Выбор большинства - выбирается значение метки, за которое проголосовало большинство классификаторов.

2. Взвешивание голосов классификаторов – каждому классификатору сопоставляется в соответствие вес, на который умножается его голос во время голосования. Выбирается значение метки, набравшей больший суммарный вес.

Для B случайных величин, дисперсия каждой из которых q^2, дисперсия усреднения составит $\frac{q^2}{B}$. Если величины простые (например, одинаково распределены, но необязательно независимы) с попарной корреляцией r, то дисперсия усреднения составит:

$$q^2(r + \frac{1-r}{B}) \quad (1.1)$$

По мере увеличения B значение второго члена в скобках уменьшается, первый остается неизменным. Следовательно, преимущество усреднения ограничивает только значение попарной корреляции.

Классификаторы, используемые в беггинге, должны быть независимы друг от друга и обучены на разных наборах данных. Иначе в результате голосования они будут выдавать одинаковые значения метки класса.

В качестве модели для бэггинга хорошо подходят деревья решений [1]. Они обладают высокой дисперсией, но при этом в них можно фиксировать сложные структурные взаимодействия данных.

Бэггинг хорошо подходит для классификации многомерных объектов. Он позволяет добиться качественной классификации в условиях, когда разделить объекты на группы на всем пространстве параметров не представляется возможным.

В рамках беггинга происходит разделение всего пространства характеристик на подмножества, объединенные по смыслу. Классификация на каждом подпространстве происходит отдельно, затем ее результаты учитываются в голосовании. Такой подход позволяет повысить вероятность более качественной классификации, нежели без деления на подпространства, так как параметры, по которым объекты разных классов неотличимы, с более высокой вероятностью попадут не во все подмножества.

1.2.4 RANDOM FOREST

Random forest – алгоритм машинного обучения, основанный на беггинге с применением деревьев решений.

Идея алгоритма состоит в улучшении уменьшения дисперсии, достигаемого с помощь бэггинга, за счет уменьшения значения попарной корреляции между деревьями [15]. Попарная корреляция уменьшается путем случайного выбора характеристик объектов для классификации.

Псевдокод обучения Random forest для задачи классификации, описанной в разделе 1.5.1, с обучением B деревьев, выглядит следующим образом:

1. for i $=$ 1 to B:
    1) Взять случайным образом N элементов Z^* из обучающего множества.
    2) Построить дерево решений T_i, используя Z^*, путем повторения следующих действий для каждого узла дерева, до тех пор, пока не будет достигнуто минимальное количество узлов в дереве n_{min}:
        I. Выбрать случайным образом p характеристик из m.
        II. Выбрать лучшую характеристику из p, разделяющую оставшиеся на текущем шаге объекты из Z^*.
        III. Добавить дочерние узлы для текущего узла.
2. Вернуть обученные деревья $\{T_B\}_1^B$.

По умолчанию $p = \sqrt{m}$, а $n_{min} = 1$. На практике значения этих параметров зависят от решаемой задачи. При увеличении p увеличивается время обучения алгоритма, а деревья решений становятся более однообразными [6].

При увеличении B, качество на обучающей выборке повышается, а на тестовой выборке выходит на асимптоту [6]. Вместе с этим пропорционально увеличиваются время обучения и работы алгоритма.

1.3 EMBEDDINGS

1.3.1 ГРАФЫ В МАШИННОМ ОБУЧЕНИИ

Основной проблемой в машинном обучении с использованием графов

является поиск способа включения информации о структуре графа в модель [27]. Например, в случае классификации, может потребоваться информация о глобальной позиции вершины или структуре её локальной окрестности.

Большинство традиционных подходов решения этой проблемы основываются на: вычислении различных статистик (среднее количество ребер у вершины, коэффициенты кластеризации), функциях ядер, определении функций для описания локальных окрестностей вершины [12]. Традиционные подходы имеют ряд ограничений в применении, поскольку они не могут адаптироваться в процессе обучения, а их внедрение является трудоемким и затратным процессом.

За последнее время появилось несколько новых подходов, основанных на прямом кодировании. Идея этих подходов заключается в получении отображения, представляющего вершины или части графа как точки в пространстве векторов с низкой размерностью R^d – embeddings [17]. Целью является оптимизация отображения с сохранением информации о структуре исходного графа. После оптимизации, полученные представления могут быть использованы в качестве входных данных для последующих задач машинного обучения.

Новые подходы, в отличие от традиционных, решают проблему, как задачу машинного обучения. Оптимизация может выполняться без учителя, для тех случаев, когда последующая задача машинного обучения не известна. Также можно использовать классификационные и регрессионные метки, связывая их с отдельными вершинами или частями графа.

Формально задачу можно описать следующим образом. Входные данные: неориентированный граф $G = (V, E)$, с ассоциативной двоичной матрицей смежности A и вещественной матрицей атрибутов вершин $X \in R^{m|V|}$. Целью является отображение информации о каждой вершине из A и X в вектор $z \in R^d$, где $d \ll V$.

1.3.2 КОДЕР-ДЕКОДЕР

Огромное количество исследований по внедрению информации об узлах в модель привело к сложному разнообразию концептуальных идей и подходов. Обобщением для всех них является модель кодер-декодер (см. рис 1.7). Она состоит из двух ключевых функций: кодер – отображающий информацию о вершине или части графа в низкоразмерный вектор, декодер - извлекающий информацию из набора низкоразмерных векторов [13].

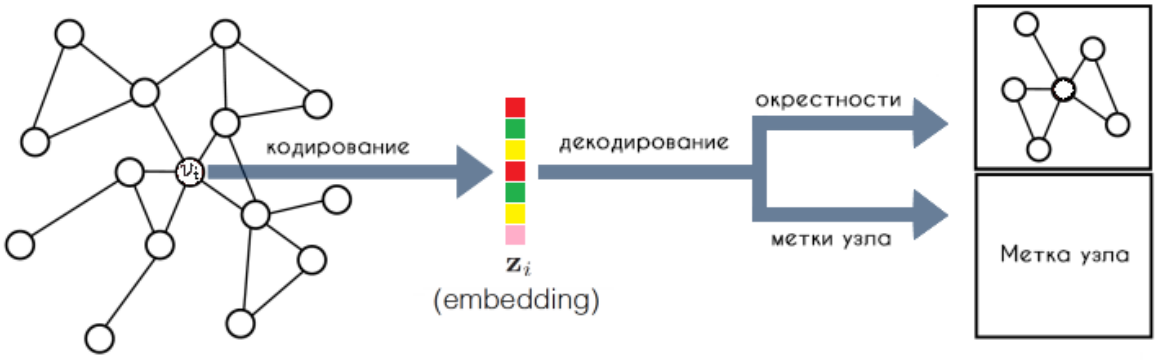

Рис. 1.7 Модель кодер-декодер

Формально кодер и декодер это функции:

$$\mathrm{ENC}: V \to R^d \tag{1.2}$$

$$\mathrm{DEC}: R^d \times R^d \to R^+ \tag{1.3}$$

Различия между подходами представления вершин заключаются в том, как они определяют четыре компонента:

1. Заранее определяемая функция близости двух узлов в графе $S(v_i, v_j)$.
2. Функция кодера ENC.
3. Функция декодера DEC.
4. Заранее определяемая функция погрешности, связанной с кодированием, $l = R * R \to R$. Позволяет измерить расхождение между декодированными значениями близости $\mathrm{DEC}(z_i, z_j)$ и истинными значениями $S(v_i, v_j)$.

На практике чаще всего используется попарный декодер, предназначенный для отображения пары представлений вершин в меру близости этих вершин в исходном графе. При использовании попарного декодера целью яв-

ляется оптимизация отображений для минимизации ошибки таким образом, чтобы:

$$\mathrm{DEC}\left(\mathrm{ENC}(v_i), \mathrm{ENC}\left(v_j\right)\right) = \mathrm{DEC}(z_i, z_j) \approx S(v_i, v_j) \quad (1.4)$$

Большинство подходов реализующих восстановление, описанное уравнением (1.3), опираются на минимизацию эмпирических потерь L по набору пар из обучающих узлов D:

$$L = \sum_{(v_i, v_j) \in D} l(\mathrm{DEC}(z_i, z_j), S(v_i, v_j)) \quad (1.5)$$

1.3.3 ЦЕПИ МАРКОВА

Пусть дан ориентированный граф $G = (V, E)$, $|V| = n, |E| = m$. Каждая дуга графа $(u, v) \in E$ имеет вес $M_{uv} \in [0; 1]$. M_{uv} определяет вероятность перехода из вершины u в v на одном шаге. Сумма вероятностей переходов из вершины u в каждую из вершин, с которыми она связана дугами $N(u)$, равна 1:

$$\forall u: \sum_{v \in N(u)} M_{uv} = 1 \quad (1.6)$$

Цепь Маркова – это последовательность случайных событий с конечным или счетным числом исходов (см. рис. 1.8), в которой вероятность перехода в следующее состояние Pr зависит только от текущего состояния [7]. Формально:

$$\Pr[X_{t+1} = v | X_t = u, X_{t-1} = u_1, \dots, X_0 = u_t] = \Pr[X_{t+1} = v | X_t = u] = M_{uv} \quad (1.7)$$

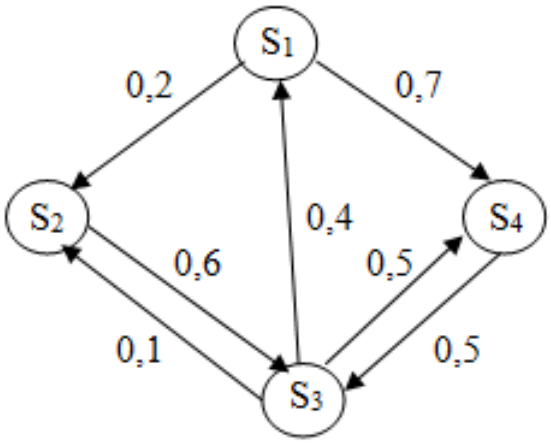

Рис. 1.8 Пример цепи Маркова

Пусть P_i определяет вероятность оказаться в вершине i, а P^t обозначает

распределение вероятностей после t шагов:

$$P^t = (P_1, \dots, P_n) \tag{1.8}$$

Тогда:

$$P^{t+1} = P^t M \Longrightarrow P^t = P^0 M^t \tag{1.9}$$

Определение 1. Случайное распределение П является стационарным, если $\Pi M = P$.

В некоторых случаях невозможно достичь стационарного распределения для заданной цепи Маркова. Даже если оно существует, как на рисунке 1.9. Здесь стационарным распределением является вектор $\left(\frac{1}{2}, \frac{1}{2}\right)$. Не является достижимым с начальным распределением $(1,1)$ или $(1,1)$.

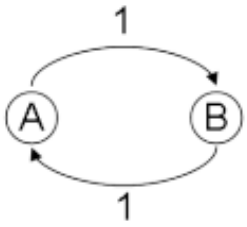

Рис. 1.9 Пример цепи Маркова с недостижимым стационарным распределением

Определение 2. Вершины u и v являются смежными, если существует ненулевая вероятность попасть из u в v, и наоборот, за n шагов:

$$M_{uv}^n > 0, M_{vu}^n > 0 \tag{1.10}$$

Определение 3. Цепь Маркова называется неприводимой, если в ней из любого состояния можно попасть в любое другое.

Замечание 1. Цепь Маркова над графом G является неприводимой, если G – сильно связанный граф.

Определение 4. Вершина v называется периодической, если:

$$\exists \Delta > 1 \in N: \Pr[X_{t+s} = v | X_t = v] = 0$$

При том, что t делит Δ нацело. Другими словами, если начать серию переходов из вершины v, то только каждые Δ шагов система может вернуться в состояние с вершиной v.

Определение 5. Цепь Маркова называется непериодической, если все её состояния непериодические.

Теорема 1. Фундаментальная теорема цепей Маркова. Каждая конечная, неприводимая, непериодическая цепь Маркова имеет стационарное случайное распределение П, которое может быть достигнуто из любого начального распределения.

Обозначим T_i как ожидаемое количество шагов, затрачиваемое на возвращение в i-ю вершину после начала изменения состоя из неё. Тогда стационарное распределение будет выглядеть следующим образом:

$$П = (П_1, \dots, П_n), \text{ где } П_i = \frac{1}{T_i} \quad (1.11)$$

Пусть u и v – две вершины графа G, а r_{uv}^t определяет вероятность достижения v из u ровно за t шагов, тогда ожидаемое количество шагов:

$$h_{uv} = \sum_{t \geq 1} tr_{uv}^t \quad (1.12)$$

А элемент стационарного распределения:

$$П_i = \frac{1}{h_{ii}} \quad (1.13)$$

1.3.4 СЛУЧАЙНОЕ БЛУЖДАНИЕ

Пусть $G = (V, E),\ |V| = n, |E| = m$ – это не двудольный, неориентированный граф. А матрица M является его матрицей смежности и содержит в себе вероятности перехода из одной из вершины u в одну из ее соседних вершин v или 0, если вершины не связаны:

$$M_{uv} = \frac{1}{deg(u)} \quad (1.14)$$

Замечание 2. Цепь Маркова над не двудольным, связным, неориентированным графом является непериодической.

Замечание 3. Стационарное распределение цепи Маркова над G:

$$П_v = \frac{deg(v)}{2m} \quad (1.15)$$

Замечание 4. Пусть u и v – две вершины графа G, тогда ожидаемое количество шагов для перехода из u в v:

$$h_{uv} < 2m \quad (1.16)$$

Пусть v – вершина графа G. Обозначим C_v, как ожидаемое количество шагов, необходимое, чтобы пройти все вершины графа, начиная с v. C_v также

называется временем обхвата графа, начиная с вершины v.

Замечание 5. $\forall v \Longrightarrow C_v \leq 4nm$.

Случайное блуждание подразумевает серию переходов из одной вершины графа в другую, с началом в некоторой вершине, по ребрам, в зависимости от вероятности перехода [9]. Количество переходов может быть фиксированным, также можно указать вероятность остановки процесса случайного блуждания.

Алгоритм случайного блуждания предназначен для определения достижимости вершины u из вершины v, за определенное количество шагов или с учетом вероятности остановки процесса. В рамках алгоритма начинается процесс случайного блуждания из вершины v, количество шагов равно $8n^3$.

Если между v и u не существует пути в графе, то алгоритм всегда будет возвращать результат, обозначающий недостижимость u. Если существует, то алгоритм вернет результат, обозначающий достижимость, с вероятностью $\geq \frac{1}{2}$, это следует из неравенства Маркова. Для достижения вершины u ожидаемое количество шагов равно $4n^3$. Отсюда, вероятность недостижимости u за $8n^3$ шагов:

$$Pr \leq \frac{4n^3}{8n^3} \leq \frac{1}{2} \tag{1.17}$$

1.3.5 NODE2VEC

Пусть $G = (V, E)$, $|V| = n, |E| = m$ – это не двудольный, неориентированный граф.

Случайное блуждание первого порядка – это процесс с фиксированным количеством переходов l по графу G, с началом в определенной вершине u. Вероятность перехода в следующую вершину c_i из текущей c_{i-1} определяется следующим образом:

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z}, \text{если } (v, x) \in E \\ 0, \text{в противном случае} \end{cases} \tag{1.18}$$

Где π_{vx} – ненормализованная вероятность перехода из v в x, а Z - константа нормализации. В простейшем случае π_{vx} зависит от веса связи w_{vx},

например $\pi_{vx} = w_{vx}$. В случае невзвешенного графа $w_{vx} = 1$.

Рассмотрим неориентированный граф на рисунке 1.10, в котором был совершен переход из вершины t в v. Каждому ребру вершины v было поставлено в соответствие число $d_{tx} \in \{0,1,2\}$. 0 – если ребро связано с вершиной t. 1- если ребро связано с вершиной, которая входит в кратчайший путь от t в v, не содержащий ребра (t, v). 2 – в остальных случаях.

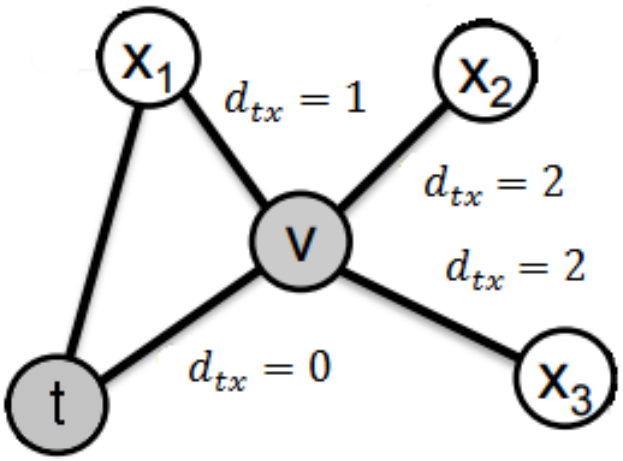

Рис 1.10 Пример случайного блуждания второго порядка

Случайное блуждание второго порядка – это случайное блуждание первого порядка с параметрами p и q, в котором $\pi_{vx} = w_{vx}\alpha_{pq}(t, x)$, где:

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p}, \text{если } d_{tx} = 0 \\ 1, \text{если } d_{tx} = 1 \\ \frac{1}{q}, \text{если } d_{tx} = 2 \end{cases} \quad (1.19)$$

Параметры p и q позволяют ограничить зону случайного блуждания, как на рисунке 1.11. В рамках выполняемой задачи может потребоваться проход только локальной окрестности начального узла u - проход в ширину. В другой задаче наоборот, необходимо как можно дальше удаляться от начальной вершины – проход в глубину.

Параметр p позволяет контролировать вероятность немедленного повторного посещения узла в блуждании. Выбор высокого значения для этого параметра (больше единицы) гарантирует, что уже посещенные узлы будут выбраны с меньшей вероятностью на следующих 2 шагах. Низкое значение (меньше единицы) позволяет ограничить пределы случайного блуждания в

локальной окрестности начального узла u.

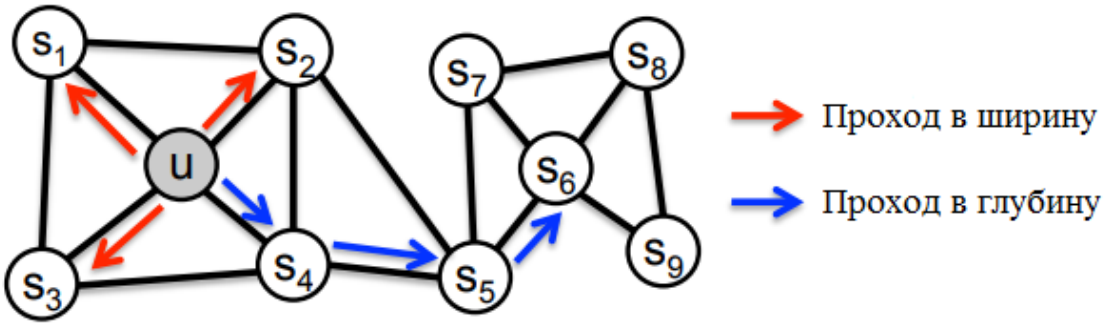

Рис 1.11 Разницы между проходом в глубину и ширину

Параметр q позволяет проводить разделение узлов как внутренних (относящихся к локальной окрестности узла u) и внешних (находящихся дальше от узла u). Низкое значение для параметра q (меньше единицы) обеспечивает выбор тех узлов во время случайного блуждания, которые находятся дальше от начального узла u. Высокое значение (больше единицы) позволяет ограничить пределы случайного блуждания в локальной окрестности начального узла u.

Параметр q позволяет проводить разделение узлов как внутренних (относящихся к локальной окрестности узла u) и внешних (находящихся дальше от узла u). Низкое значение для параметра q (меньше единицы) обеспечивает выбор тех узлов во время случайного блуждания, которые находятся дальше от начального узла u. Высокое значение (больше единицы) позволяет ограничить пределы случайного блуждания в локальной окрестности начального узла u.

Ключевым нововведением методов прямого кодирования, основывающихся на случайном блуждании, является оптимизация, при которой две вершины имеют наиболее похожие представления, если случайные блуждания, начинающиеся в них, имеют похожую форму. То есть, вместо детерминированной меры близости, в этих в этих методах используется гибкая и стохастическая.

Nod2Vec – алгоритмический каркас, основанный на случайном блужда-

нии второго порядка и стохастическом градиентном спуске, предназначенный для репрезентативного представления узлов графа [11].

Алгоритм состоит из трех этапов: предварительная обработка для вычисления вероятностей перехода (PreprocessModifiedWeights), моделирование случайного блуждания (node2vecWalk) и оптимизация с использованием стохастического градиентного спуска (StochasticGradientDescent). Все три этапа выполняются последовательно. Вычисления на каждом этапе можно выполнять в асинхронном режиме.

Вероятности перехода π_{vx} могут быть вычислены предварительно, следовательно, выборка узлов при моделировании случайного блуждания может быть выполнена за время $O(1)$.

Псевдокод алгоритма выглядит следующим образом:

```
LearnFeatures(G = (V, E, W), r, l, p, q)
   pi = PreprocessModifiedWeights(G, p ,q)
   G' = (V, E, pi)
   init walks to empty

   for all u in V do
      for i = 1 to r do
         walk = node2vecWalk(G', u, l)
         walks add walk

   f = StochasticGradientDescent(walks)

   return f
```

```
node2vecWalk(G' = (V, E, pi), u, l)
   init walk to [u]

   for i = 1 to l do
      curr = walk.last
      v_cur = GetNeighbors(curr, G')
      s = aliasSample(v_curr, pi)
      walk add s

   return walk
```

Где:

1. G – исходный граф.
2. r – количество случайных блужданий, совершаемых из каждого узла.
3. l – длина одного случайного блуждания.
4. walks – совокупность всех выполненных случайных блужданий.
5. f – результат стохастического градиентного спуска, embedding.

1.4 DATA MINING

1.4.1 ОПРЕДЕЛЕНИЯ И ЗАДАЧИ

Знания - совокупность фактов, закономерностей и эвристических правил, с помощью которых решается определенная задача. Data Mining – это мультидисциплинарная область, возникшая и развивающаяся на базе прикладной статистики, распознавания образов, искусственного интеллекта и др. Она представляет собой анализ данных для обнаружения в них скрытых знаний.

В основу технологии положена концепция шаблонов, представляющих собой закономерности. В результате обнаружения закономерностей решаются поставленные задачи. Различным типам закономерностей, которые могут быть выражены в форме, понятной человеку, соответствуют определенные задачи.

Единого мнения относительно того, какие задачи следует относить к Data Mining, нет. В основном перечисляются следующие: классификация, кластеризация, прогнозирование. Для таких задач широко применяются методы математической статистики, машинное обучение, нейронные сети и др.

Задачи подразделяются по типам производимой информации, это наиболее общая классификация задач Data Mining.

1.4.2 ОСНОВНЫЕ ЭТАПЫ

Процесс Data Mining является своего рода исследованием. Он состоит из определенных этапов (см. рис. 1.12), включающих в себя элементы: сравнения, типизации, классификации, обобщения, абстрагирования, повторения.

В результате исследования выстраивается модель, используемую в про-

цессе принятия решений.

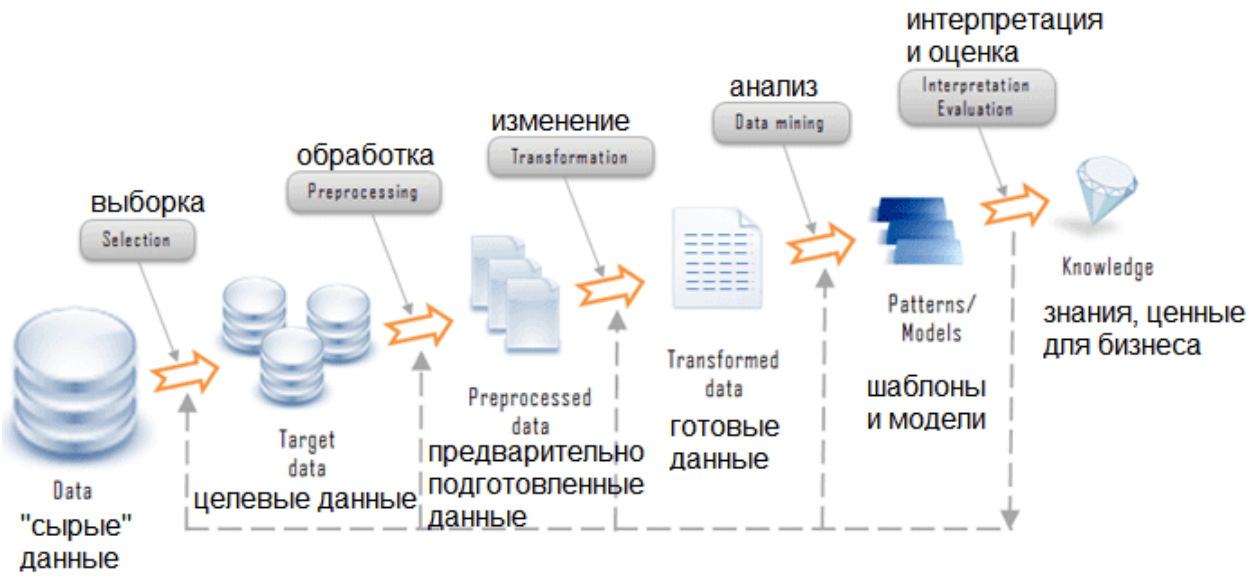

Рис 1.12 Процесс Data Mining

Процесс Data Mining включает следующие этапы:

1. Анализ предметной области.
2. Постановка задачи.
3. Подготовка данных.
4. Построение моделей.
5. Проверка и оценка моделей.
6. Выбор модели.
7. Применение модели.
8. Коррекция и обновление модели.

1.4.3 ПРОВЕРКА И ОЦЕНКА МОДЕЛЕЙ

Адекватность модели – это соответствие модели моделируемому объекту или процессу. Проверка модели подразумевает проверку ее достоверности или адекватности. Она заключается в определении степени соответствия модели реальности. Адекватность модели проверяется путем тестирования.

Понятия достоверности и адекватности являются условными, поскольку невозможно рассчитывать на полное соответствие модели реальному объекту. В процессе моделирования следует учитывать адекватность не модели вообще, а именно тех ее свойств, которые являются существенными с точки

зрения проводимого исследования.

В процессе проверки модели необходимо установить включение в модель всех существенных факторов. Сложность решения этой проблемы зависит от сложности решаемой задачи.

Тестирование модели заключается в использовании построенной модели, заполненной данными, с целью определения ее характеристик, а также в проверке ее работоспособности. Тестирование модели включает в себя проведение множества экспериментов. На вход модели могут подаваться наборы данных различного объема.

С точки зрения статистики, точность модели увеличивается с увеличением количества исследуемых данных. Алгоритмы, являющиеся основой для построения моделей на сверхбольших объемах данных, должны обладать свойством масштабирования.

Если модель достаточно сложна, а значит, требуется много времени на ее обучение и последующую оценку, то иногда бывает полезно построить и протестировать модель на небольшой части данных.

Построенные модели рекомендуется тестировать на различных наборах для определения их обобщающих способностей. В ходе экспериментов можно варьировать количество записей и использовать наборы различной сложности.

Для оценки результатов полученных моделей следует использовать знания специалистов предметной области. Если результаты полученной модели эксперт считает неудовлетворительными, следует вернуться на один из предыдущих шагов процесса Data Mining, а именно: подготовка данных, построение модели, выбор модели. Если же результаты моделирования эксперт считает приемлемыми, ее можно применять для решения реальных задач.

1.4.4 ВЫБОР МОДЕЛИ

Если в результате моделирования было построено несколько различных моделей, то на основании их оценки можно осуществить выбор лучшей. В ходе проверки и оценки различных моделей на основании их характеристик,

а также с учетом мнения экспертов, следует выбор наилучшей. Часто это оказывается непростой задачей.

Основные характеристики модели, которые определяют ее выбор, – это точность модели и эффективность работы алгоритма.

1.4.5 ПРИМЕНЕНИЕ, КОРРЕКЦИЯ И ОБНОВЛЕНИЕ МОДЕЛИ

После тестирования, оценки и выбора модели следует этап применения модели. На этом этапе выбранная модель используется применительно к новым данным с целью решения задач, поставленных в начале процесса Data Mining. Для классификационных и прогнозирующих моделей на этом этапе прогнозируется целевое значение.

По прошествии определенного установленного промежутка времени с момента начала использования модели Data Mining следует проанализировать полученные результаты. Определить, действительно ли она решает задачу или же возникли проблемы и сложности в ее использовании.

Однако даже если модель решает задачу, ее не следует считать абсолютно верной. Необходимо периодически оценивать адекватность модели набору данных, а также текущей ситуации (следует учитывать возможность изменения внешних факторов). Даже самая точная модель со временем может перестать быть таковой. Для того чтобы построенная модель выполняла свою функцию, следует работать над ее коррекцией (улучшением). При появлении новых данных требуется повторное обучение модели. Этот процесс называется обновлением модели. Работы, проводимые с моделью на этом этапе, также называют контролем и сопровождением модели.

Существует много причин, требующих обучить модель заново, т.е. обновить ее, чтобы отразить определенные изменения. Основными являются следующие:

- Изменились входящие данные или их поведение.
- Появились дополнительные данные для обучения.
- Изменились требования к форме и количеству выходных данных.

- Изменились цели задачи, которые повлияли на критерии принятия решений.
- Изменилось внешнее окружение или среда (макроэкономика, политическая ситуация, научно-технический прогресс, появление новых конкурентов и товаров и т.д.).

Причины, перечисленные выше, могут обесценить допущения и исходную информацию, на которых основывалась модель при построении.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 СБОР СЕТЕВЫХ ПРОФИЛЕЙ

2.1.1 СБОР РЕЗЮМЕ

Рекрутмент – это процесс привлечения, отбора и подбора квалифицированных специалистов для работы. HeadHunter — крупнейшая российская компания интернет-рекрутмента. Клиентами HeadHunter являются порядка одного миллиона компаний. Обширная база кандидатов HeadHunter содержит более чем 30 миллионов резюме, а среднее дневное количество вакансий превышает 450 тыс. HeadHunter занимает третье место в мире по популярности среди порталов по поиску работы и сотрудников. Официальный сайт компании - hh.ru.

У сайта hh.ru существует публичный API [16], с помощью которого можно получать информацию о соискателях и компаниях, а также использовать функциональность HeadHunter для сторонних сайтов или приложений.

Для поискового запроса по названию профессии API возвращает массив json-объектов, каждый json-объект содержит данные из резюме соискателя. Пример json-объекта с полями резюме:


```
{
   "photo":"URL",
   "salary":{
      "currency":"RUR",
      "amount":30000
   },
   "title":"Секретарь",
   "area":{
      "url":"https://api.hh.ru/areas/1",
      "id":"1",
      "name":"Москва"
   },
   "published_at":"2013-07-08T16:17:21+0400",
   "employer":{
      "logo_urls":{
         "90":"https://hh.ru/employer-logo/289027.png",
         "240":"https://hh.ru/employer-logo/289169.png",
         "original":"https://hh.ru/file/2352807.png"
      },
      "name":"HeadHunter",
      "url":"https://api.hh.ru/employers/1455",
      "id":"1455",
      "trusted":true
   },
   "address":{
      "city":"Москва",
```

```
      "street":"улица Годовикова",
      "building":"9с10",
      "description":"на проходной потребуется паспорт",
      "lat":55.807794,
      "lng":37.638699,
      "metro_stations":[
         {
            "station_id":"6.8",
            "station_name":"Алексеевская",
            "line_id":"6",
            "line_name":"Калужско-Рижская",
            "lat":55.807794,
            "lng":37.638699
         }
      ]
   },
   "department":{
      "id":"HH-1455-TECH",
      "name":"HeadHunter::Технический департамент"
   },
   "type":{
      "id":"open",
      "name":"Открытая"
   },
   "id":"8331228",
   "has_test":true,
   "response_url":null,
   "snippet":{
      "requirement":"Высшее образование. Опыт работы в качестве <high-
lighttext>секретаря</highlighttext>, офис-менеджера. Знание делопроизводства,
документооборота. Коммуникативные навыки.",
      "responsibility":"Документооборот (регистрация, отправка, контроль ис-
полнения писем, ведение протоколов, отчетность). Распределение корреспонден-
ции. Прием и распределение телефонных звонков."
   }
}
```


Для построения сетевых профилей, в ходе сбора, из резюме извлекались следующие данные:

- Фотография.
- Дата рождения.
- Пол.
- Место жительства.
- Сведения о высшем образовании.
- Сфера деятельности.
- Название профессии.
- Желаемая зарплата.
- Опыт работы.
- Сведения о предыдущих местах работы.
- График работы.

- Тип занятости.

Поле сфера деятельности может принимать одно из следующих значений:

- IT телеком.
- Бухгалтерия.
- Маркетинг.
- Админ. персонал.
- Банки.
- Управление персоналом.
- Авто.
- Безопасность.
- Топ-менеджер.
- Добыча сырья.
- Искусство медиа.
- Консультирование.
- Медицина.
- Наука и образование.
- Начало карьеры.
- Госслужба.
- Продажи.
- Производство.
- Страхование.
- Строительство.
- Транспорт.
- Туризм, рестораны.
- Юриспруденция.
- Спорт фитнес.
- Инсталляция.
- Закупи.

- Домашний персонал.
- Рабочий персонал.

Поле занятость может принимать одно или несколько из следующих значений:

- Полная занятость.
- Частичная занятость.
- Проектная работа.
- Волонтерство.
- Стажировка.

Поле график работы может принимать одно или несколько из следующих значений:

- Полный день.
- Сменный график.
- Гибкий график.
- Удаленная работа.
- Вахтовый метод.

Сбор резюме проходил в 2018 году. Были использованы запросы, по заранее составленному словарю профессий: менеджер, программист, бухгалтер, инженер, слесарь и т.д.

Всего было собрано 10566 различных резюме людей различного возраста, с различным местом жительства, опытом работы и т.д.

Публично доступный профиль соискателя на hh.ru не предоставляет информацию о ФИО. Но содержит фото и точные данные о месте жительства, образовании, поле и дате рождения, чего достаточно для поиска страницы соискателя в социальных сетях.

2.1.2 OPENFACE

Torch - это библиотека машинного обучения с открытым исходным кодом, научная вычислительная среда и язык сценариев, основанный на языке Lua. OpenFace - это библиотека для распознавания лиц, реализованная на

языке Python [24], с использованием Torch [21].

OpenFace преобразует изображение с лицом человека в вектор, embedding, на гиперсфере в евклидовом пространстве с размерностью 128 [23]. Такое представление обладает полезным свойством, заключающимся в том, что расстояние между векторами можно использовать как меру сходства двух лиц. Это свойство позволяет решать задачи кластеризации, обнаружения сходства и классификации намного проще, чем с использованием других методов распознавания, в которых евклидово расстояние между объектами не имеет смысла.

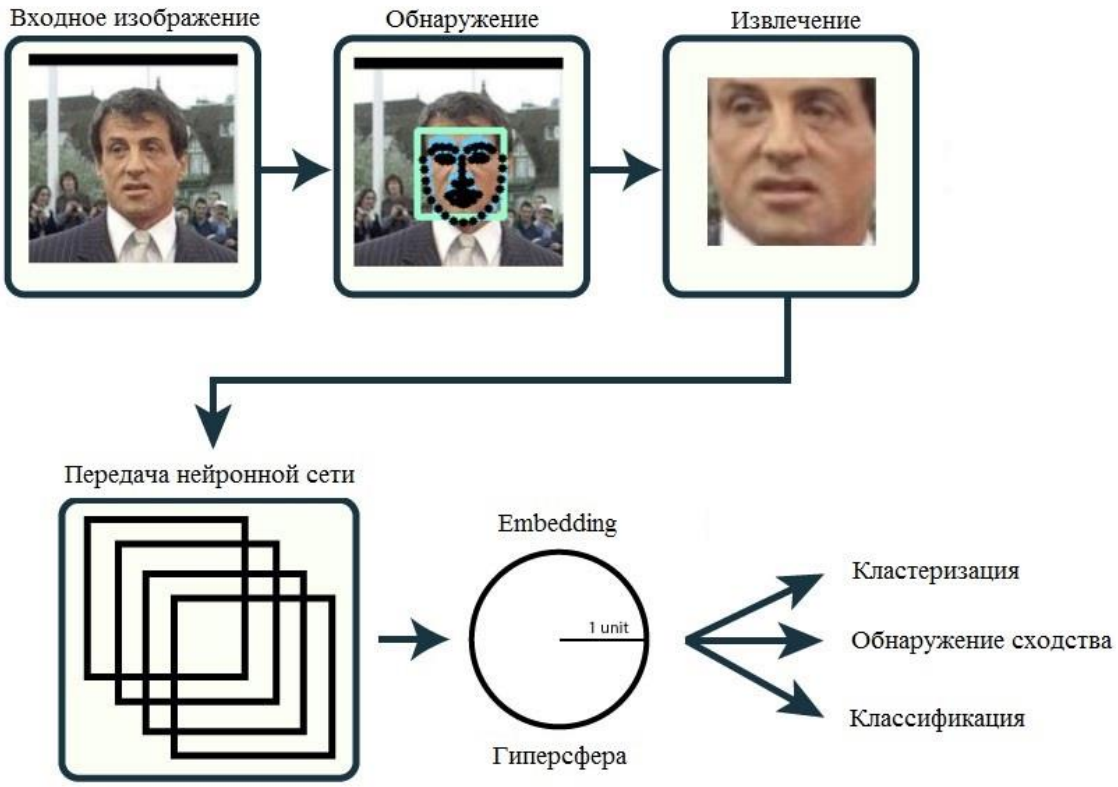

Рис 2.1 Процесс работы с OpenFace

Процесс работы с библиотекой выглядит следующим образом (см. рис. 2.1):

1. Передача входного изображения.
2. Обнаружение лица на входном изображении с помощью предварительно обученной модели, взятой из библиотеки dlib или OpenFace.
3. Извлечение обнаруженного лица из входного изображения.
4. Преобразование извлеченного лица в вектор, embedding.

5. Использование полученного вектора, embedding’a, для выполнения задачи кластеризации, обнаружения сходства или классификации.

Исходный код скрипта на языке программирования Python, предназначенного для сравнения лиц на фотографиях приведен в разделе Приложение 1.

2.1.3 СБОР СТРАНИЦ ВО ВКОНТАКТЕ

ВКонтакте – это российская социальная сеть, среднесуточная аудитория которой составляет более 80 миллионов посетителей, а всего зарегистрировано более 460 миллионов. На сентябрь 2019 года сайт ВКонтакте занимал 12 место по популярности в мире. Официальный сайт vk.com.

У сайта vk.com существует публичный API [26], с помощью которого можно выполнять вход на страницу и поисковые запросы, а также получать данные со страниц пользователей.

Использование информации только о Месте жительства, образовании, поле и дате рождения задает достаточно широкий коридор для поиска во ВКонтакте необходимой страницы, в среднем 10-20 страниц на каждый запрос.

Для решения этой проблемы было реализовано сопоставление фотографий резюме и профиля социальной сети при помощи библиотеки OpenFace. В качестве результирующей страницы для резюме выбиралась та, для которой разница между векторами embeddings для фото страницы и резюме была меньше 0.1 и являлась минимальной.

Метод с сопоставлением лиц имеет высокую точность, но низкую полноту. Всего удалось сопоставить 2692 резюме, из изначальных 10566, со страницами ВКонтакте.

Полученные профили были дополнены следующими данными из ВКонтакте:

- ФИО.
- Дата рождения.
- Пол.

- Место жительства.
- Сведения о высшем образовании.
- Информация о группах, в которых состоит пользователь.
- Информация о друзьях.
- Аналогичная информация для друзей.

2.2 ВЫЧИСЛЕНИЕ EMBEDDINGS

Изначально для построения социального графа, по которому вычислялись embeddings, из ВКонтакте для полученных профилей предполагалось собрать информацию обо всех страницах друзей (друзья первого уровня) и обо всех страницах друзей их друзей (друзья второго уровня).

В виду ограничений по имеющимся аппаратным ресурсам, информация о друзьях второго уровня не была использована для вычислений. Но информация о том, с кем они дружат, помогла дополнить итоговый граф связями между друзьями первого уровня.

Для собранных во ВКонтакте страниц на рисунке 2.2 приведена гистограмма распределения количества друзей и количества страниц с таким количеством друзей.

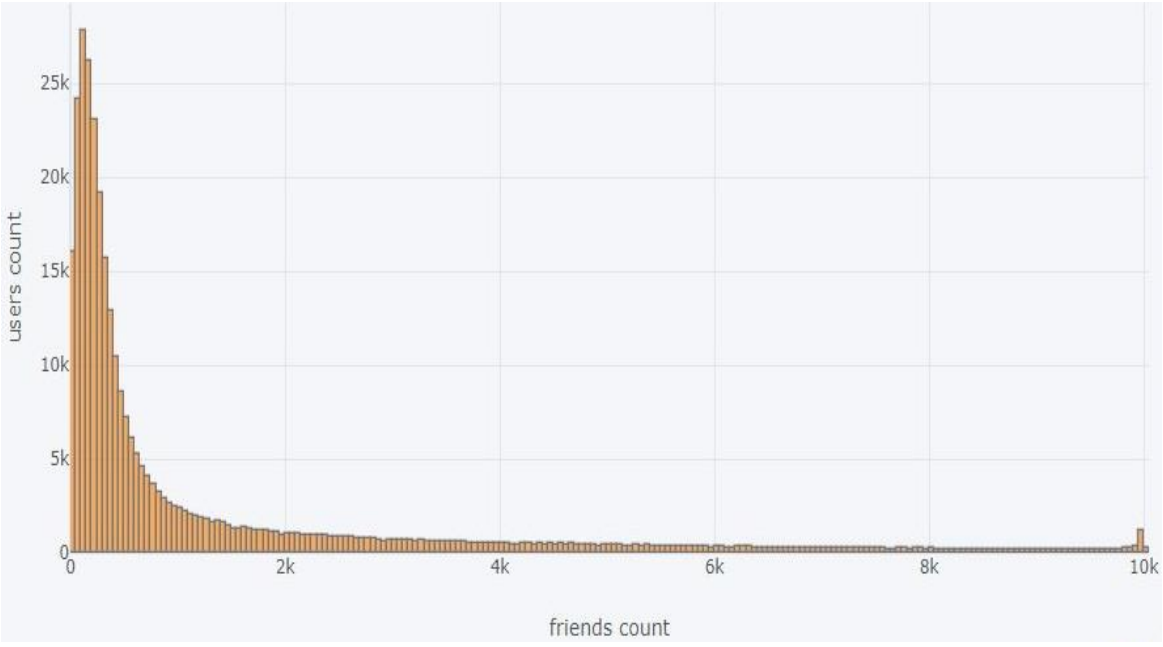

Рис 2.2 Гистограмма распределения друзей

Максимальное количество друзей у страницы не может быть больше

10000, это ограничение социальной сети.

По гистограмме видно, что примерно у 10% страниц количество друзей больше 3000. Вершины этих страниц и их связи увеличивают исходный граф примерно в два раза. Зачастую такие страницы не содержат полезной информации, т.к. являются мостами между остальными частями графа. Это либо фейковые страницы, либо страницы различных компаний, поставляющих определенные услуги. По этим причинам страницы с количеством друзей больше 3000 будут удалены из исходного графа.

До удаления мостов из исходного графа, главная компонента связности составляла 92,20%.

Итоговый граф состоит из 403 624 вершин (полученные профили и друзья первого уровня) и 2 233 946 связей. Размер файла с итоговым графом в формате csv составляет 4,17 гигабайт.

Для вычисления embeddings по итоговому графу использовалась библиотека SNAP [22], реализованная на языке Си.

По графу была выполнена серия случайных блужданий с началом в каждой вершине каждого полученного профиля. Размерность результирующего вектора была подобрана экспериментально, ее значение равно 100. Длина пути равна 20 переходам. В качестве значений для параметров алгоритма Nod2vec, p и q, использовались следующие комбинации:

1. $p = 0, q = 0.$
2. $p = 0, q = 0.3.$
3. $p = 0, q = 0.5.$
4. $p = 0, q = 1.5.$
5. $p = 0, q = 2.$
6. $p = 0.3, q = 0.$
7. $p = 0.3, q = 0.3.$
8. $p = 0.3, q = 0.3.$
9. $p = 0.3, q = 1.5.$
10. $p = 0.3, q = 2.$

11. $p = 0.5, q = 0$.
12. $p = 0.5, q = 0.3$.
13. $p = 0.5, q = 0.5$.
14. $p = 0.5, q = 1.5$.
15. $p = 0.5, q = 2$.
16. $p = 1, q = 1$.
17. $p = 1.5, q = 0$.
18. $p = 1.5, q = 0.3$.
19. $p = 1.5, q = 0.5$.
20. $p = 1.5, q = 1.5$.
21. $p = 1.5, q = 2$.
22. $p = 2, q = 0$.
23. $p = 2, q = 0.3$.
24. $p = 2, q = 0.5$.
25. $p = 2, q = 1.5$.
26. $p = 2, q = 2$.

Собранные на предыдущих этапах профили были дополнены полученными векторами embeddings.

2.3 ПОДГОТОВКА ДАННЫХ

2.3.1 АНОНИМИЗАЦИЯ ПРОФИЛЕЙ

В связи со стремительным развитием и массовым распространением социальных сетей работа с данными, извлекаемыми из них, считается новым, перспективным, но пока мало изученным направлением. Кроме того, эстетические и правовые вопросы использования данных из социальных сетей не проработаны в полной мере [8].

Таким образом, с одной стороны, данные, извлекаемые из социальных сетей, дают новые возможности для развития науки, а с другой – ставят под сомнение правомерность и этичность их использования.

Основная проблема заключается в том, что нарушается главный эстети-

ческий принцип – добровольное участие в исследовании [8]. Также возникает риск вторжения в личную жизнь пользователей, чьи данные были собраны в ходе исследования.

Чтобы не допустить вторжения в личную жизнь пользователей, чьи данные были собраны в ходе текущего исследования, была произведена анонимизация собранных профилей.

Из профилей были удалены фотографии, ФИО, данные о предыдущих местах работы и о друзьях. Даты рождения заменены на возраст.

Из данных о месте жительства были отобраны только названия городов и областей, к которым они относятся, а из сведений о высшем образовании только название высшего учебного заведения. Данные о месте жительства и высшем образовании были прокодированы, подробнее в разделе 2.3.2.

2.3.2 КОДИРОВАНИЕ ДАННЫХ ПРОФИЛЕЙ

К категориальным признакам относятся: пол, город или название области, название учебного заведения, название профессии, сфера деятельности, график работы, тип занятости.

Город и название области были прокодированы в соответствии с рейтингом крупнейших по населению городов России, ссылка на рейтинг http://www.statdata.ru/largest_cities_russia.

Названия следующих городов, а также их области в профилях были заменены на число 2:

- Москва.
- Санкт-Петербург.

Названия следующих городов, а также их области в профилях были заменены на число 1:

- Новосибирск.
- Екатеринбург.
- Нижний Новгород.
- Казань.

- Челябинск.
- Омск.
- Самара.
- Ростов-на-дону.
- Уфа.
- Красноярск.
- Воронеж.
- Пермь.
- Волгоград.

Название всех остальных городов и областей были заменены на число 0.

Названия университетов были прокодированы в соответствии с рейтингом топ-100 российских вузов, по объемам реализации в рублях, ссылка на рейтинг https://expert.ru/ratings/top-100-rossijskih-vuzov-ot-ra-ekspert/.

На число 2 были заменены названия следующих вузов:

- Московский государственный университет имени М.В. Ломоносова.
- Московский государственный технический университет им. Н.Э. Баумана.
- Санкт-Петербургский государственный университет.
- Московский физико-технический институт.
- Высшая школа экономики.
- Московский энергетический институт.
- Национальный исследовательский ядерный университет “МИФИ”.
- Томский политехнический институт.
- Санкт-петербургский государственный политехнический университет.
- Новосибирский государственный университет.

На число 1 были заменены названия следующих вузов:

- Московский государственный институт международных отношений МИД России.
- Сибирский федеральный университет.

- Российский государственный университет нефти и газа им. И.М. Губкина.
- Финансовый университет при правительстве РФ.
- Томский государственный университет.
- Российская академия народного хозяйства и государственной службы.
- Национальный исследовательский технологический университет “МИСиС”.
- Казанский приволжский федеральный университет.
- Уральский федеральный университет им. Первого президента России Б.Н. Ельцина.
- Российский университет дружбы народов.
- Южный федеральный университет.
- Новосибирский государственный технический университет.
- Санкт-Петербургский государственный университет информационных технологий, механики и оптики.
- Российский экономический университет им. Г.В. Плеханова.
- Казанский технологический университет.
- Российский государственный гуманитарный университет.
- Самарский государственный университет.
- Российский медицинский университет им. Н.И. Пирогова.
- Московский государственный университет экономики, статистики и информатики.
- Санкт-Петербургский государственный медицинский университет им. акад. И.П. Павлова.
- Московская международная высшая школа бизнеса “МИРБИС”.
- Национальный минерально-сырьевой университет “Горный”.
- Московский авиационный институт.
- Нижегородский государственный университет им. Н.И. Лобачевского.
- Самарский государственный аэрокосмический университет им. Н.И.

Лобачевского.

- Самарский государственный аэрокосмический университет им. акад. С.П. Королева.
- Российский химико-технологический институт университет им. Д.И. Менделеева.
- Томский государственный университет систем управления радиоэлектроники.
- Северо-Западный государственный медицинский им. И.И. Мечникова.
- 1-й Московский государственный медицинсий университет им. И.М. Сеченова.

Названия остальных вузов были заменены на число 0.

Остальные категориальные признаки также были прокодированы. Каждое уникальное значение для каждого признака было добавлено в отдельную категорию.

К количественным признакам относятся: возраст, опыт работы, количество друзей, медианный возраст друзей, средний возраст друзей, количество групп. Для количественных признаков никакие изменения не проводились.

2.4 АНАЛИЗ ДАННЫХ И ОБУЧЕНИЕ RANDOM FOREST

2.4.1 ОПИСАНИЕ ИТОГОВЫХ ДАННЫХ

Для каждой комбинации параметров алгоритма Nod2vec, p и q, перечисленных в разделе 2.2, было составлено отдельное множество данных. Всего получилось 26 множеств.

Каждое множество данных состоит из 2692 строк, а каждая строка из полей, приведенных в таблице 2.1.

Начало таблицы 2.1 Поля обучающего множества

Название поля	Описание	Тип данных
salary	Желаемая зарплата (в рублях)	float64
gender	Метка класса пола	int64

Окончание таблицы 2.1 Поля обучающего множества

Название поля	Описание	Тип данных
age	Возраст (в годах)	float64
city	Метка класса места жительства	int64
education	Метка класса оконченного ВУЗа	int64
work_exp	Опыт работы (в месяцах)	float64
position	Метка класса профессии	int64
fields_of_activity	Метка класса сферы деятельности	int64
employment	Метка класса типа занятости	int64
work_schedule	Метка класса графика работы	int64
friends_count	Количество друзей во ВКонтакте	float64
friends_median_age	Медианный возраст друзей во ВКонтакте	float64
friends_mid_age	Средний возраст друзей во ВКонтакте	float64
groups_count	Количество групп	float64
0…99	Элементы вектора embedding	floaf64

Поля с названиями от 0 до 99 – это элементы вектора embedding.

На рисунке 2.3 приведет срез одного из множеств итоговых данных.

	salary	gender	age	city	education	position	work_exp	...	99
0	70000.0	0	34.0	2	0	253	132.0	...	0.263777
1	120000.0	1	31.0	2	0	181	103.0	...	-0.461302
2	30000.0	0	24.0	1	0	1311	34.0	...	0.092327
3	35000.0	0	26.0	2	0	1292	104.0	...	-0.026193
4	35000.0	1	33.0	0	0	1796	176.0	...	-0.390551

Рис 2.3 Срез итоговых данных

2.4.2 АНАЛИЗ ДАННЫХ

Для выявления зависимостей между всеми полями множества данных, кроме элементов вектора embedding, была составлена матрица корреляции с использованием коэффициента корреляции Пирсона (см. рис. 2.4).

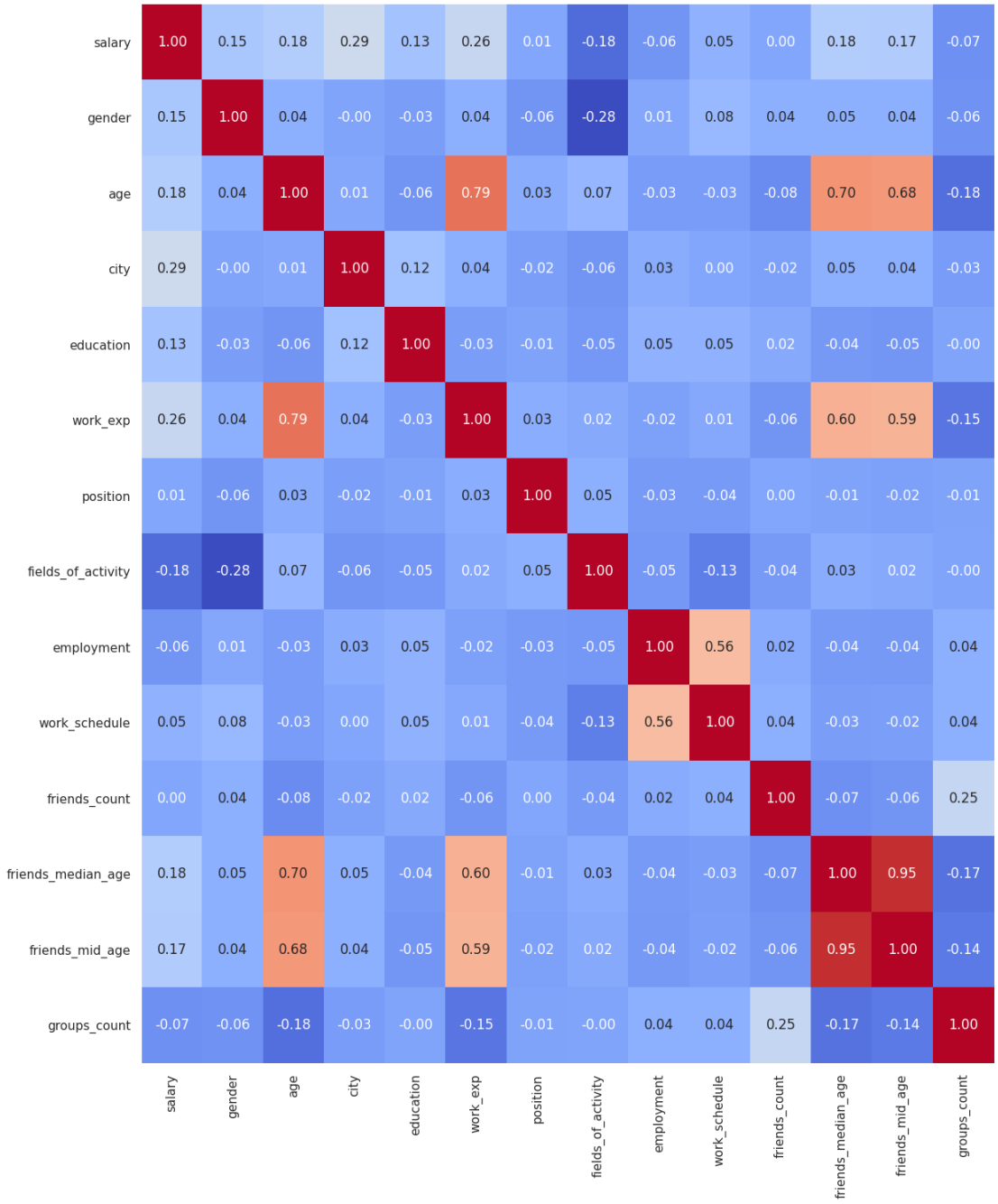

Рис 2.4 Матрица корреляции

Анализируя значения матрицы корреляции можно сделать следующие выводы, принимая во внимание, что в большинстве случаев желаемая заработная плата не сильно отличается от реальной:

Связь желаемой зарплаты и места жительства обусловлена тем, что в

России в больших городах (Москва), как правило, уровень жизни выше, чем в маленьких (Саранск), да и возможностей для заработка больше. Соответственно, в большинстве случаев зарплата и требования к ней в больших городах выше, а в маленьких меньше.

В начале карьеры, пока человек молодой его не особо волнует, сколько он будет зарабатывать. Но чем он старше и чем больше у него опыта работы, тем выше зарплата и требования к ней.

Уровень образования и престиж оконченного также оказывают определенное влияние на желаемую зарплату. Люди, оканчивающие более престижные вузы, обладают более качественным образованием и рассчитывают на более высокую зарплату.

Связь между возрастом пользователя и возрастом его друзей объясняется гомофилией, в социальных сетях в качестве окружения людям свойственно выбирать себе подобных.

2.4.3 ОБУЧЕНИЕ RANDOM FOREST

Между желаемой заработной платой и реальной всегда существует определенная разница. Определить ее значение в рамках данного исследования не представляется возможным. Поэтому будет решаться задача не регрессии, а классификации. Границы классов будут компенсировать разницу между желаемой зарплатой и реальной.

Для проверки гипотез наборы данных были разбиты на поднаборы, состоящие из:

1. Только вектора embedding.
2. Только из всех остальных данных.
3. Из вектора embedding и всех остальных данных.

Каждый поднабор был разбит на обучающее и тестовое множество в пропорции 4 к 1.

С использованием каждого поднабора данных, проводились двухклассовая, трехклассовая и четырехклассовая классификации. При выполнении двухклассовой классификации границей было значение 50000 рублей. При

трехклассовой 40000 и 60000 рублей. При трехклассовой 30000, 50000, 70000 рублей.

В таблицах 2.2, 2.3 и 2.4 приведены результаты качества обучения для выполненных классификаций. Качество измерялось на тестовом множестве.

Начало таблицы 2.2 Двухклассовая классификация

Значения p и q для набора	Только embeddings, %	Остальные данные, %	Вместе, %
p=0 q=0	69.57	74.71	76.19
p=0 q=0.3	59.74	72.24	74.54
p=0 q=0.5	61.78	74.38	74.54
p=0 q=1.5	58.99	75.04	76.35
p=0 q=2	62.15	75.36	74.04
p=0.3 q=0	69.20	73.23	75.70
p=0.3 q=0.3	68.08	73.72	76.35
p=0.3 q=0.5	68.27	74.05	76.35
p=0.3 q=1.5	66.04	73.40	76.02
p=0.3 q=2	67.90	74.38	75.53
p=0.5 q=0	69.01	74.38	77.01
p=0.5 q=0.3	66.60	75.05	76.02
p=0.5 q=0.5	69.38	72.74	76.35
p=0.5 q=1.5	67.16	74.38	77.18
p=0.5 q=2	70.12	72.74	76.51
p=1 q=1	67.71	73.23	76.35
p=1.5 q=0	70.50	74.71	76.84
p=1.5 q=0.3	66.97	75.20	76.35
p=1.5 q=0.5	67.90	75.53	77.50
p=1.5 q=1.5	67.90	73.27	77.17
p=1.5 q=2	64.93	72.57	77.99
p=2 q=0	66.79	72.74	77.01
p=2 q=0.3	66.41	75.53	76.02

Окончание таблицы 2.2 Двухклассовая классификация

Значения p и q для набора	Только embeddings, %	Остальные данные, %	Вместе, %
p=2 q=0.5	68.83	74.38	76.84
p=2 q=1.5	68.64	74.71	77.01
p=2 q=2	68.46	75.53	77.83

Начало таблицы 2.3 Трехклассовая классификация

Значения p и q для набора	Только embeddings, %	Остальные данные, %	Вместе, %
p=0 q=0	53.43	57.14	56.77
p=0 q=0.3	43.97	56.21	57.32
p=0 q=0.5	48.05	55.65	55.65
p=0 q=1.5	45.82	58.99	54.91
p=0 q=2	46.01	53.98	58.25
p=0.3 q=0	51.76	54.73	53.80
p=0.3 q=0.3	51.94	53.80	58.07
p=0.3 q=0.5	48.05	53.80	56.95
p=0.3 q=1.5	51.76	55.84	55.65
p=0.3 q=2	50.83	57.32	56.58
p=0.5 q=0	50.64	55.28	58.44
p=0.5 q=0.3	51.20	56.21	56.95
p=0.5 q=0.5	50.83	54.91	58.07
p=0.5 q=1.5	49.90	56.58	58.25
p=0.5 q=2	51.76	57.51	57.51
p=1 q=1	48.23	56.40	55.84
p=1.5 q=0	51.39	56.40	57.51
p=1.5 q=0.3	51.39	56.58	57.32
p=1.5 q=0.5	52.69	55.84	57.51
p=1.5 q=1.5	50.83	55.65	55.10

Окончание таблицы 2.3 Трехклассовая классификация

Значения p и q для набора	Только embeddings, %	Остальные данные, %	Вместе, %
p=1.5 q=2	49.35	55.10	57.88
p=2 q=0	52.31	56.02	57.88
p=2 q=0.3	52.50	55.84	55.28
p=2 q=0.5	51.02	56.95	57.88
p=2 q=1.5	49.90	57.51	55.65
p=2 q=2	50.83	55.10	54.54

Начало таблицы 2.4 Четырехклассовая классификация

Значения p и q для набора	Только embeddings, %	Остальные данные, %	Вместе, %
p=0 q=0	43.78	46.30	48.11
p=0 q=0.3	35.25	46.46	47.94
p=0 q=0.5	37.47	45.81	47.78
p=0 q=1.5	38.77	46.63	48.11
p=0 q=2	35.99	47.94	49.75
p=0.3 q=0	42.85	45.48	50.08
p=0.3 q=0.3	40.63	46.63	47.61
p=0.3 q=0.5	40.07	45.64	46.46
p=0.3 q=1.5	40.07	45.81	49.75
p=0.3 q=2	39.14	44.99	49.09
p=0.5 q=0	41.00	45.64	49.96
p=0.5 q=0.3	39.88	47.29	48.93
p=0.5 q=0.5	41.18	46.46	49.09
p=0.5 q=1.5	41.18	47.78	48.60
p=0.5 q=2	41.55	46.63	50.08
p=1 q=1	37.66	47.45	46.79
p=1.5 q=0	39.88	45.48	47.78

Окончание таблицы 2.4 Четырехклассовая классификация

Значения p и q для набора	Только embeddings, %	Остальные данные, %	Вместе, %
p=1.5 q=0.3	41.00	44.66	49.75
p=1.5 q=0.5	40.44	45.64	49.42
p=1.5 q=1.5	40.07	45.48	49.58
p=1.5 q=2	39.70	45.64	48.93
p=2 q=0	43.22	44.82	47.61
p=2 q=0.3	43.78	44.82	47.45
p=2 q=0.5	42.30	47.61	49.42
p=2 q=1.5	41.92	47.78	50.08
p=2 q=2	40.81	46.79	49.91

При трехклассовой и четырехклассовой классификациях качество обучения ощутимо падает, это связано с малым размером наборов исходных данных.

Качество обучения, с использованием только embeddings, при двухклассовой классификации подтверждает обе выдвинутые гипотезы. От социального окружения пользователя в социальных сетях зависит размер его доходов. Используя информацию об окружении пользователя в социальных сетях, возможно эффективно прогнозировать размер его доходов.

Дополнив стандартное множество данных, не включающих в себя информацию о структуре социального графа, можно повысить качество обучаемой модели. В данной работе качество повышалось максимально до 5.42%.

Исходный код скрипта на языке программирования Python, предназначенного для обучения модели приведен в разделе Приложение 2.

ЗАКЛЮЧЕНИЕ

Магистерская диссертация посвящена проблематике применения Data-Minig для решения задачи прогнозирования доходов пользователей социальных сетей, на основе данных сетевого профиля и социального окружения в социальных сетях.

Результатами выполненной диссертационной работы являются:

1. Разработано программное обеспечение для автоматического сбора данных о сетевом профиле и заработной плате пользователей, а также о социальном окружении пользователей в социальных сетях.
2. Показана продуктивность использования данных о социальном окружении пользователей в социальных сетях для прогнозирования заработной платы.
3. Разработан новый метод решения для задачи прогнозирования зарплаты пользователей.
4. Разработаны программные средства для решения задачи прогнозирования зарплаты пользователей.

Цель магистерской диссертации выполнена. Обе выдвинутые гипотезы подтвердились. Что свидетельствует о существовании перспектив в применении данных о социальном окружении пользователей в социальных сетях для задачи прогнозирования заработной платы и, следовательно, доходов.

Стоит отметить, что результаты диссертации представляют интерес для ряда областей социально-экономической и политической сфер деятельности.

Анонимизированные и прокодированные множества данных, а также исходный скрипт для анализа данных и обучения на языке программирования Python выложены в репозиторий на GitHub. Ссылка на репозиторий: https://github.com/hungryangry/income_prediction.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Джулли А., Пал С. Библиотека Keras – инструмент глубокого обучения ДМК.: Москва, 2018. 293 с.
2. Коршунов А., Белобородов И., Аванесов В. Анализ социальных сетей: методы и приложения [Электронный ресурс]. URL: https://www.ispras.ru/proceedings/docs/2014/26/1/isp_26_2014_1_439.pdf (дата обращения 11.02.2019).
3. Меркулов М.В., Браун С.А. Выявление скрытых связей между агентами в гетерогенных сетях // Гагаринские чтения – 2018. Сборник тезисов и докладов XLIV Международной молодёжной научной конференции. – Москва, 2019. – С. 237-238.
4. Меркулов М.В., Филимонов А.Б. Применение методов машинного обучения в банковском скорринге // Гагаринские чтения – 2019. Сборник тезисов и докладов XLV Международной молодёжной научной конференции. – Москва, 2019. – С. 379-380.
5. Меркулов М.В., Филимонов А.Б. Прогнозирование доходов пользователей социальных сетей // Ломоносов – 2020. Сборник тезисов и докладов Международной конференции студентов, аспирантов и молодых ученых «Ломоносов-2020». – Москва, 2020. - С. 96-97.
6. Орельен Ж. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем Вильямс.: Москва, 2018. 688 с.
7. Ревюз Д. Цепи Маркова РФФИ.: Москва, 1997. 409 с.
8. Щеглова И.А. Эстетические и правовые аспекты использования данных из социальных медиа. Вестник томского государственного университета.: Томск, 2018. 7 с.
9. Agrawal A. Random walks in graphs [Электронный ресурс]. URL: http://web.cs.iastate.edu/~pavan/633/lec7.pdf (дата обращения 11.02.2019).
10. Chockalingam V., Shah S., Shaw R. Income classification using adult census

data [Электронный ресурс]. URL: https://pdfs.semanticscholar.org/3dd5/e9f335511efbb81d65f1d6d4995019f8b5fd.pdf?_ga=2.199053994.467408706.1584791178121787594 0.1584791178 (дата обращения 11.02.2019).

11. Grover A., Leskovec J. Node2vec: scalable feature learning for networks [Электронный ресурс]. URL: https://cs.stanford.edu/~jure/pubs/node2vec-kdd16.pdf (дата обращения 11.02.2019).
12. Hamilton W., Ying R., Leskovec J. Inductive representation learning on large graphs [Электронный ресурс]. URL: https://arxiv.org/pdf/1706.02216.pdf (дата обращения 11.02.2019).
13. Hamilton W., Ying R., Leskovec J. Representation learning on graphs: methods and applications [Электронный ресурс]. URL: https://www-cs.stanford.edu/people/jure/pubs/graphrepresentation-ieee17.pdf (дата обращения 11.02.2019).
14. Haojun Z. Predicting earning potential using the adult dataset [Электронный ресурс]. URL: https://rstudio-pubs-staic.s3.amazonaws.com/235617_51e06fa6c43b47d1b6daca2523b2f9e4.html (дата обращения 11.02.2019).
15. Hastie T., Tibshirani R., Friedman J. The elements of statistical learning Springer.: Stanford, 2017. 764c.
16. HeadHunter API – описание API и техническая документация [Электронный ресурс]. URL: https://dev.hh.ru/ (дата обращения 01.11.2018).
17. Koehrsen W. Neural network embeddings explained [Электронный ресурс]. URL: https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526 (дата обращения 11.02.2019).
18. Lazar A. Income prediction via support vector machine [Электронный ресурс]. URL: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=73AB890512E62

D87C00B3BD217772746?doi=10.1.1.387.5068&rep=rep1&type=pdf (дата обращения 11.02.2019).

19. Lemon C., Zelazo C., Mulakaluri K. Predicting of income exceeds $50.000 per year based on 1994 US Census data with simple classification techniques [Электронный ресурс]. URL: https://educationdocbox.com/Homework_and_Study_Tips/66686559-Predicting-if-income-exceeds-50-000-per-year-based-on-1994-us-census-data-with-simple-classification-techniques.html (дата обращения 11.02.2019).
20. Menji S. Using decision tree classifier to predict income levels [Электронный ресурс]. URL: https://mpra.ub.uni-muenchen.de/83406/1/MPRA_paper_83406.pdf (дата обращения 11.02.2019).
21. OpenFace – техническая документация [Электронный ресурс]. URL: https://cmusatyalab.github.io/openface/ (дата обращения 01.11.2018).
22. SNAP – техническая документация [Электронный ресурс]. URL: http://snap.stanford.edu/graphwave/ (дата обращения 01.11.2018).
23. Schroff F., Kalenichenko D., Philbin J. FaceNet: A Unified Embedding for Face Recognition and Clustering [Электронный ресурс]. URL: https://www.cvfoundation.org/openaccess/content_cvpr_2015/app/1A_089.pdf (дата обращения 01.11.2018).
24. Shaw Z. Learn Python 3 the hard way Addison-Wesley Professional.: Boston, 2017. 320 с.
25. Topiwalla M. Machine learning on UCI adult data set using various classifier algorithms and scaling up the accuracy using extreme gradient boosting [Электронный ресурс]. URL: https://datascience52.files.wordpress.com/2017/02/machine-learning-on-uci-adult-data-set-using-various-classifier-algorithms-and-scaling-up-the-accuracy-using-extreme-gradient-boosting.pdf (дата обращения 11.02.2019).

26. VK API - описание API и техническая документация [Электронный ресурс]. URL: https://pypi.org/project/vk-api/ (дата обращения 01.11.2018).
27. Zhou J., Cui G., Zhang Z., Yang C. Graph neural networks: a review of methods and applications [Электронный ресурс]. URL: https://arxiv.org/pdf/1812.08434.pdf (дата обращения 11.02.2019).

ПРИЛОЖЕНИЕ 1

Скрипт на языке программирования Python, предназначенный для сравнения лиц людей на фотографиях и использованием библиотеки OpenFace. Входные аргументы:

1. --who – путь к изображению с лицом, которое необходимо опознать.
2. --where – путь к папке, в которой находятся изображения с лицами, которые необходимо сравнить с лицом с изображения --who.
3. --result – путь к файлу, в который будут записаны результаты сравнения. Результаты сравнения представляют собой евклидово расстояние между векторами, embeddings, соответствующими лицам с фотографий.

Код скрипта:

```
import os
import sys
import argparse

import cv2
import openface
import numpy as np

file_dir = os.path.dirname(os.path.realpath(__file__))
model_dir = os.path.join(file_dir, '..', 'models')
dlib_model_dir = os.path.join(model_dir, 'dlib')
openface_model_dir = os.path.join(model_dir, 'openface')

align = openface.AlignDlib(os.path.join(dlib_model_dir,
"shape_predictor_68_face_landmarks.dat"))
net = openface.TorchNeuralNet(os.path.join(openface_model_dir,
'nn4.small2.v1.t7'), 96)

def get_rep(img_path):
    bgr_img = None
    try:
        bgr_img = cv2.imread(img_path)
    except Exception:
        pass
    if bgr_img is None:
        print("Unable to load image: {}".format(img_path))
        return None
    rgb_img = None
    try:
        rgb_img = cv2.cvtColor(bgr_img, cv2.COLOR_BGR2RGB)
    except Exception:
        pass
    bb = None
    try:
        bb = align.getLargestFaceBoundingBox(rgb_img)
    except Exception:
        pass
```

```
    if bb is None:
        print("Unable to find a face: {}".format(img_path))
        return None
    aligned_face = None
    try:
        aligned_face = align.align(96, rgb_img, bb, landmarkIn-
dices=openface.AlignDlib.OUTER_EYES_AND_NOSE)
    except Exception:
        pass
    if aligned_face is None:
        print("Unable to align image: {}".format(img_path))
        return None
    rep = None
    try:
        rep = net.forward(aligned_face)
    except Exception:
        pass
    return rep

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('--who', type=str, help='Path to image with face to
search')
    parser.add_argument('--where', type=str, help='Paths to images where in
search, separated by ; char')
    parser.add_argument('--result', type=str, help='Path to file wehere will
result of search')
    args = parser.parse_args()
    who_img_rep = get_rep(args.who)
    if who_img_rep is None:
        exit(1)
    distances = []
    for where_path in args.where.split(';'):
        img_file_name = where_path[where_path.rfind(os.path.sep) + 1:]
        where_img_rep = get_rep(where_path)
        if where_img_rep is not None:
            d = who_img_rep - where_img_rep
            distances.append(np.dot(d, d))
        else:
            distances.append(100)
    with open(args.result, 'w') as out:
        for distance in distances:
            out.write("%f\n" % distance)

if __name__ == '__main__':
    main()
```

ПРИЛОЖЕНИЕ 2

Скрипт на языке программирования Python, предназначенный для анализа данных и обучения алгоритма RandomForest:

```
import warnings

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score, train_test_split, KFold
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.utils import resample


warnings.simplefilter(action='ignore', category=FutureWarning)
sns.set(style='white', context='notebook', palette='deep')

data_set = pd.read_csv("../dataset/p0_q0.3.csv")
salaries = [50000]

# кореляция численных величин

img = sns.heatmap(data_set[['salary', 'gender', 'age', 'city', 'education',
'work_exp', 'position', 'fields_of_activity', 'employment',
'work_schedule']].corr(), annot=True, fmt=".2f", cmap="coolwarm")
plt.savefig("../dataset_analyze/0.png", figsize=(1000,1000))
plt.clf()
exit(0)
print('Rows count: ', len(data_set))

# удаление эмбедингов
'''
i = 0
while i < 100:
    data_set.drop(labels=[str(i)], axis=1, inplace=True)
    i = i + 1
'''
# удаление остальных данных
'''
data_set.drop(labels=['gender','age','city','education','position','fields_of_activity','employment','work_schedule','friends_count','friends_median_age','friends_mid_age','groups_count'], axis=1, inplace=True)
'''
# замена значений зарплат

def map_salary(salary):
    i = 0

    while i < len(salaries):
        if salary <= salaries[i]:
            break
        i = i + 1

    return i

data_set['salary'] = data_set['salary'].apply(map_salary)
```

```
# разделение данных на обучающее и тестовое множества

validation_size = 0.2
seed = 7
num_folds = 10
scoring = 'accuracy'

train, test = train_test_split(data_set, test_size=validation_size, ran-
dom_state=seed)

# балансировка обучающего множества по классам

frames = list()
vc = train['salary'].value_counts()

for i in data_set['salary'].unique():
    if vc[i] < vc.max():
        frames.append(resample(train[train['salary'] == i], replace = True,
n_samples = vc.max()))
    else:
        frames.append(train[train['salary'] == i])

train = pd.concat(frames)

# обучение

train_array = train.values

x_t = train_array[:, 1:]
y_t = train_array[:, 0]

test_array = test.values

x_v = test_array[:, 1:]
y_v = test_array[:, 0]

print('Start train')

random_forest = RandomForestClassifier(n_estimators=100, max_features=None,
max_depth=None, oob_score=True)
random_forest.fit(x_t, y_t)

# оценка результатов

predictions = random_forest.predict(x_v)

print("Accuracy: %s%%" % (100 * accuracy_score(y_v, predictions)))
print()
print(confusion_matrix(y_v, predictions))
print()
print(classification_report(y_v, predictions))
```