МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра дискретной математики и алгоритмики

СКОКЛЕЕНКО ДАНИИЛ АРТУРОВИЧ

АНАЛИЗ СООБЩЕНИЙ ПОЛЬЗОВАТЕЛЕЙ СОЦИАЛЬНЫХ СЕТЕЙ НА ПРЕДМЕТ ПРИНАДЛЕЖНОСТИ К СПАМОВЫМ

Магистерская диссертация

специальность 1-31 81 09 «Алгоритмы и системы обработки больших объёмов информации»

Научный руководитель
Соболевская Елена Павловна,
доцент,
кандидат физ.-мат. наук

Допущен к защите
«___» _____ 2017 г.
Зав. кафедрой ДМиА
_____ Котов В.М.
доктор физ.-мат. наук,
профессор

Минск, 2017

ОГЛАВЛЕНИЕ

РЕФЕРАТ

Магистерская диссертация, 41 с., 11 табл., 18 рис., 27 источн.

СОЦИАЛЬНЫЕ СЕТИ, СПАМ, ЗАДАЧА КЛАССИФИКАЦИИ, МАШИННОЕ ОБУЧЕНИЕ

Объект исследования — применение алгоритмов машинного обучения для классификации спама в социальных сетях.

Цель работы — исследование методов определения спама в социальных сетях, сравнение подходов для решения задачи классификации спама, построение модели распознавания спама в социальной сети Twitter на основе алгоритмов машинного обучения.

Методы исследования — наивный байесовский классификатор, метод k ближайших соседей, метод опорных векторов (SVM), решающие деревья, случайные леса.

Результатом является предложенный подход для построения классификатора социального спама, не требующий наличия историчных признаков пользователя социальной сети.

Область применения — системы спамообороны.

ABSTRACT

Master thesis, 41 p., 11 tab., 18 fig., 27 ref.

SOCIAL NETWORKS, SPAM DETECTION, CLASSIFICATION PROBLEM, MACHINE LEARNING

Object of research — Machine learning algorithms application in social spam classification problem.

Purpose — studying methods of social spam detection, comparing approaches of solving spam classification problem, building a model for Twitter spam detection based on machine learning algorithms.

Research methods — naive Bayes classifier, *k*-nearest neighbors algorithm, support vector machines (SVM), Decision trees, Random forest.

The result of work is the social spam classifier that does not require historical features of a social network account.

The results can be applied in antispam systems.

ВВЕДЕНИЕ

Проблема спама в социальных сетях, иначе говоря, социального спама, наносит все больший ущерб для таких компаний, как Facebook, Twitter, Pinterest и т.д. Согласно исследованию компании Nexgate, специализирующейся на безопасности пользователей социальных сетей, вышеупомянутые платформы столкнулись с 355% ростом объёма социального спама в первой половине 2013 года [1].

Социальный спам влечет за собой последствия самого различного рода, и как результат, существует несколько определений понятия «социальный спам». Одна из самых популярных социальных платформ Twitter дает собственное определение спаму и предоставляет несколько различных способов сообщать о спамовых активностях. Примером такого сообщения может служить твит «@spam @username», где @username – никнейм потенциального спамера. При этом, будучи коммерческой платформой, Twitter достаточно лояльно относится к сообщениям рекламного характера, при условии, что они не нарушают правила и политику использования социальной сети. В последние годы вышло в свет огромное количество работ и исследований, посвященных выявлению различных скрытых трендов и тенденций в самых разнообразных областях, начиная от финансовых рынков и заканчивая общественно-политической сферой, огромная часть которых строилась на основе открытых данных пользователей Twitter, что подтверждает огромную значимость этой платформы, как для академической, так и для индустриальной части общества. Потребность в выявлении значимой информации в шумовом поле Twitter является одной из ключевых задач подобных исследований, и в этом смысле, огромные обьемы социального спама очень серьезно усложняют эту задачу.

Существующие техники и методы классификации социального спама в основной своей массе используют обширный обьем исторических данных о пользователе. В данной работе была предпринята попытка создания классификатора спама на основе не столь большого набора необходимых данных, а лишь той информации, которую можно извлечь непосредственно из данных об авторе твита в момент его публикации. Значимым плюсом подобного классификатора может служить его способность своевременно определять спамовых пользователей и, как следствие, потенциальное применение в режиме реального времени. Используя классифицированный вручную набор данных твитов, среди которых находились спамовые твиты, была предпринята попытка использования пяти алгоритмов классификации на основе двух различных групп признаков.

Целью работы является исследование и разработка методов автоматического распознавания спам-сообщений в социальной сети Twitter. При написании работы были поставлены следующие задачи:

1. Исследование политики, применяемой социальной сетью Twitter относительно спама, а также техник, используемых спамерами.

2. Проведение анализа существующих методов распознавания спама, а также оцен-

ки их эффективности.

3. Разработка методов автоматического распознавания спама.

4. Реализация разработанных методов и проведение экспериментальной оценки результатов их работы.

Структура данной работы следующая. Глава 1 знакомит читателя с основной характеристикой социальной сети Twitter, а также с существующей на данный момент политикой этой компании относительно спама. В главе 2 приведен обзор и анализ ключевых исследований в сфере классификации социального спама. Глава 3 посвящена описанию техник и приемов, которые я использовал для своего классификатора. В главе 4 приведены результаты попытки построения классификатора спама и оценка его качества.

1. Политика Twitter в отношении спама

1.1. Структура Twitter

Twitter - социальная сеть, позволяющая пользователям писать сообщения, не превышающие 140 символов, которые могут быть увиденными и прокомментированными любым другим пользователем этой социальной сети. Структура Twitter представляет собой направленный граф, где пользователи — узлы, а ребра отражают отношения между ними. Некоторые определения:

Твит (сообщение) - текст длиной до 140 символов, в котором могут содержаться ссылки, хэштеги и упоминания;

Ретвит - перепубликация твита, созданного другим пользователем;

Хэштег - специально зарезервированный символ «#», употребляемый перед словом, обозначающим принадлежность к определенной теме;

Упоминание - специально зарезервированный символ «@», употребляемый перед именем пользователя;

Тренд - набор слов и хэштегов, популярность использования которых на определенный момент времени резко превышает остальные;

В Twitter пользователь может формировать связь с другими пользователями путем подписки на их твиты. Простейший пример взаимоотношений пользователей показаны на Рисунке 1.

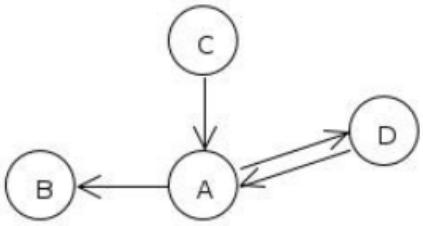

Рисунок 1 — Пример взаимоотношения пользователей Twitter

1.2. Распространенность спама в Twitter

Результаты исследований показывают, что более 3% сообщений в Twitter являются спамом [2], [3], около 8% всех ссылок указывают на вредоносный контент [4]. Также сообщается, что более 90% всех спамовых сообщений содержат ссылки [5].

В силу ограниченности длины статуса 140 символами спамеры распространяют вредоносный контент преимущественно посредством ссылок. Twitter трансформирует все ссылки с помощью собственного сервиса по сокращению URL, что позволяет спамерам не затрачивать дополнительных усилий на маскировку источника. В основной массе исследований приводятся описания следующих выявленных стратегий спамеров:

1. Спам в отношении пользователей, подписанных на спамера (любой твит пользователя автоматически появляется в новостной ленте его подписчиков) [6], [7];

2. Использование упоминаний (твит с упоминанием пользователя появляется в новостной ленте упомянутого, независимо от того, является он подписчиком спамера или нет) [7];

3. Покупка подписчиков с целью создания видимости благонадежной репутации аккаунта [6], покупка ретвитов [4];

4. Спам в отношении пользователей, чьи твиты содержат слова релевантные проводимой спам-кампании [6];

5. Использование в твитах популярных поисковых слов (техника напоминает SEO) [7];

6. Эксплуатация хэштегов популярных тем [7], [8], создание спамерских тем [4];

7. Изменение частей URL, с целью создания видимости различия ссылок [6], [9];

8. Атака фолловеров знаменитости посредством упоминания ее в своих твитах [6];

9. Использование специальных приложений, таких как TweetAttacks и TweetAdder, облегчающих использование техник 4, 8 [6];

10. Взлом аккаунтов пользователей (подбором паролей или фишингом) и рассылка спама от их лица [4], [8];

11. Копирование твитов знаменитостей с добавлением спам-ссылки [4];

12. Использование сервисов для сокращения ссылок (может быть сформирована длинная цепь перенаправлений) [4];

1.3. Политика Twitter относительно спама

Хотя алгоритм борьбы со спамом в Twitter не афишируется, чтобы не облегчить его обход, раздел «Abuse and Spam» Twitter rules [10] содержит перечисление действий, за которые аккаунт может быть заблокирован. Данный список позволяет составить некоторое представление о принципах борьбы Twitter со спамом. Некоторые условия, которые могут служить поводом для блокировки аккаунта, перечисленные в правилах Twitter:

1. Создание множественных аккаунтов;

2. Публикация ссылок на вредоносный контент;

3. Создавние статусов, содержащих преимущественно ссылки;

4. Создавние дублированных статусоа от имени одного или разных аккаунтов;

5. Создавние множественных нерелевантных статусов в темах, в т.ч. популярных;

6. Использование множественных упоминаний в целях привлечения внимания к аккаунту, сервису или ссылке;

7. Многочисленные блокировки и жалобы на спам со стороны других пользователей;

Несмотря на предупреждение о блокировке за вышеперечисленные действия, Twitter по тем или иным причинам блокирует далеко не все аккаунты, удовлетворяющие одному или нескольким из вышеперечисленных условий. Так, существует список спамерских аккаунтов [11], на странице источника которого указано, что автор регулярно отправляет списки обнаруженных спамерских аккаунтов в группу Twitter по борьбе со спамом.

Список был создан в 2009 году, тем не менее, лишь 27% из 632 аккаунтов были заблокированы Twitter к 2016.

2. Методы классификации спама

Методы обнаружения спама в социальных сетях берут начало с техник классификации сообщений электронной почты на спамерские и легитимные. Техники варьируются от надстроек над SMTP, анализа последовательностей его транзакций и составления черных списков отправителей до идентификации спам-сообщений по определенным правилам, и, наконец, применения различных методов машинного обучения: байесовской классификации, нейросетей, марковских моделей [12]. Среди традиционные методов классификации для электронной почты наиболее популярны Наивный байесовский классификатор и SVM, как признанные наилучшими для категоризации текстов [13].

Использование для обнаружение спама в Twitter теми же методами что и в электронной почте неэффективно, в первую очередь из-за ограничения на длину сообщения. Кроме того, спам в Twitter распространяется преимущественно посредством ссылок, поэтому при использовании черных списков ссылок за время до блокировки вредоносной ссылки по ней успевает перейти большое количество пользователей. Поэтому для решения задачи обнаружения спама в Twitter должны применяться методы, учитывающие специфику этой социальной сети.

В этой главе приведено описание существующих подходов классификации спама в социальной сети Twitter.

2.1. Выявление социальных спамеров

В опубликованных ранее работах, задача автоматизации поиска спама рассматривалась с 2-х точек зрения. Первая – подход, основанный на классификации конкретного пользователя, как спамера или не спамера. Формальная постановка задачи для подобного подхода выглядит следующим образом: Для аккаунта $\alpha \in A$, где A - множество аккаунтов, построить функцию $\Theta(\alpha) : A \to \{Spam, Ham\}$, то есть аккаунт классифицируется как спамерский, если функция Θ принимает значение Spam, и как благонадежный иначе.

Подобный подход наиболее популярен в большинстве работ на данный момент (см. [3], [5], [14], [9], [15], [16]) и требует наличия признаков, которые должны быть собраны на основе поведения пользователя до этого, например, прирост или убытие подписок и подписчиков, среднее количество хэштегов, ссылок и упоминаний в предыдущих сообщениях и т.д. Однако это требование не всегда достижимо из-за ограничений в Twitter API.

Ли [9] и Янг [15] использовали различные методы для сбора данных о спамовых аккаунтах (см. пункт 3.1), и проведили всестороннее исследование поведения спамеров. Они оба полагались на твиты, размещенные пользователями в прошлом, а также такие признаки, как частота публикации твитов, частота подписок, процент взаимных подписок и коэффициент локальной кластеризации сетевого графа, и боролись с тактикой уклонения спамеров, поскольку эти признаки сложны для симулирования.

Феррара [16] помимо вышеперечисленных признаков использовал сентимент-оценку текстов сообщений из все того же набора данных [9].

Миллер [17] трактует задачу обнаружения спама, как проблему обнаружения аномалий и предлагает алгоритм кластеризации для ее решения. Такой алгоритм строится на множестве легитимных пользователей, выбросы которых классифицируются как спамовые аккаунты.

Набор признаков, используемых в указанных выше работах, требует сбора исторических данных для каждого пользователя, что не отвечает требованиям обозначенного сценария обнаружения спама в реальном времени.

2.2. Выявление социального спама

Второй, альтернативный вариант, который не так популярен в литературе, это перевести задачу классификации с пользователя на отдельный твит [5]. В этом случае предполагается, что той информации, которая может быть извлечена из сообщения пользователя, достаточно для классификации сообщения как спамового. В данной работе используется именно этот подход.

Сантош [18] исследовал два различных подхода, а именно алгоритмы классификации текста на основе сжатия (т.е. динамическое марковское сжатие и предсказание путем частичного сопоставления) и использование модели «мешок слов» для обнаружения спам-твитов.

Мартинес-Ромо [8] применил расстояние Кульбака-Лейблера и исследовал различие языка, используемого в наборе твитов, содержащих трендовые слова и хэштеги, подозрительных твитов (твиты, содержащие ссылку на веб-страницу) и страниц, на которые ссылаются подозрительные твиты. Эти различия в языковой дивергенции использовались в качестве признаков для классификации. Для разметки спамовых твитов в своей работе авторы использовали несколько черных списков спамовых URL-адресов, поэтому каждый из помеченных ими спам-твитов содержит URL-ссылку и не может идентифицировать другие типы спам-твитов.

2.3. Прочие подходы

Существуют и иные методы обнаружения спама в социальных сетях, в частности, подход, основанный на тенденции спамеров образовывать связи друг с другом (см. Рис. 2), алгоритм поиска спамеров Criminal account Inference Algorithm (CIA) — на основе начального множества вредоносных аккаунтов с помощью «оценки зловредности» калькулируемой из степени связанности аккаунтов и семантической схожести их статусов отслеживаются остальные спамеры. Тем не менее, алгоритм CIA позиционируется не как полноценный алгоритм детектирования, а как «легковесный алгоритм вывода и ранжирования», который может быть встроен в систему обнаружения спамеров в комбинации с другими методами [19].

В своей работе Жань [20] предлагает находить потенциальных спамеров как поль-

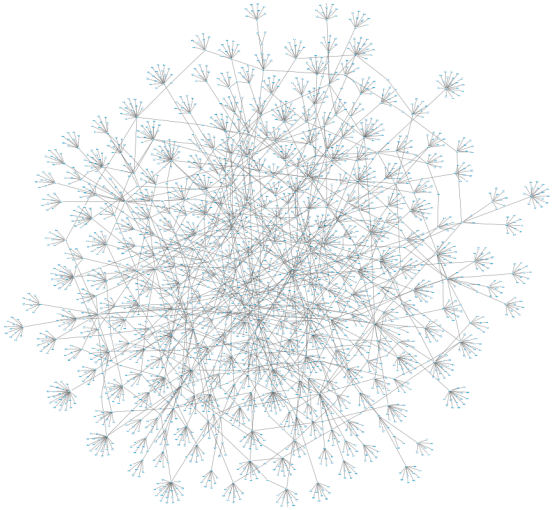

Рисунок 2 — Пример взаимосвязей спамовых аккаунтов Twitter

зователей, имеющих достаточно большое количество твитов, являющимися дубликатами сообщений других пользователей. Для выявления дублированных твитов среди набора твитов всех пользователей в наборе данных используется метод locality-sensitive hashing (LSH), позволяющий оценить коэффициент схожести Жаккара для n-грамм над словами в статусах. Твиты, для которых коэффициент Жаккара превышает 0.8, считаются дубликатами. Использование данного метода в детектировании спамеров предполагает дальнейшую классификацию на основе свойств сообщений и профилей, характерных для спамеров, имеющих дублированные сообщения.

3. Описание методологии

В этой главе будет дано описание начального набора данных, его предобработка, признаки, которые были использованы для построения классификатора, а также описание использовавшихся алгоритмов.

3.1. Набор данных

Наличие размеченной коллекции твитов в задаче классификации спама имеет критически важное значение. Набора данных, который бы полностью удовлетворял всем требованиям найдено не было. В качестве базы был использован набор данных из [9], который распространяется по открытой некоммерческой лицензии Creative Commons. Датасет представляет из себя случайно собранную в период с 30.11.2009 г. по 02.08.2010 г. информацию о 22,223 спамерах и 19,276 легитимных пользователях, а также 2,353,473 и 3,259,693 спамовых и легитимных сообщениях этих аккаунтов соответственно. Данные были получены путем регистрации 60 аккаунтов-приманок для спама, целью которых было имитировать пользовательское поведение. Аккаунт-приманка имел возможность совершать одно из 4-х действий:

1. Пубиковать обычное текстовое сообщение;
2. Упоминать один из других аккаунтов-приманок с помощью символа «@»;
3. Публиковать сообщение, содержащее ссылку;
4. Публиковать сообщение, содержащее одну из ТОП-10 трендовых слов Twitter;

Каждый из аккаунтов-приманок не мог взаимодействовать с пользователями не из круга других аккаунтов-приманок. Как только один из искусственных аккаунтов упоминался в сообщении аккаунта «извне» или появлялся у него в подписках, информация о новом спамовом аккаунте заносились в датасет.

Перед извлечением признаков из набора данных к тексту твитов были применены несколько техник предобработки с целью нормализации и уменьшения «шума» на стадии классификации.

3.2. Описание признаков

Поскольку спамеры и обычные пользователи имеют различные цели при размещении твитов или взаимодействии с другими пользователями в Twitter, справедливо предположение, что характеристики спам-твитов сильно отличаются от обычных твитов. Признаки, присущие твиту, включают, помимо самого твита, набор метаданных, включая информацию о пользователе, разместившем твит, который также легко доступен в потоке твитов, к которому у нас есть доступ. Признаки были разделены по следующим группам:

1. Пользовательские признаки

2. Признаки контента

Пользовательские признаки включают в себя список из атрибутов автора твита (см. Таблицу 1), который генерируется из метаданных каждого сообщения. Например «репутация пользователя» [3] (см. Рис 3), которая определяется как соотношение между числом подписчиков и общим числом подписчиков и подписок и используется для измерения степени влияния пользователя. Такие признаки как число ретвитов не использовались, поскольку они содержат в себе историчность.

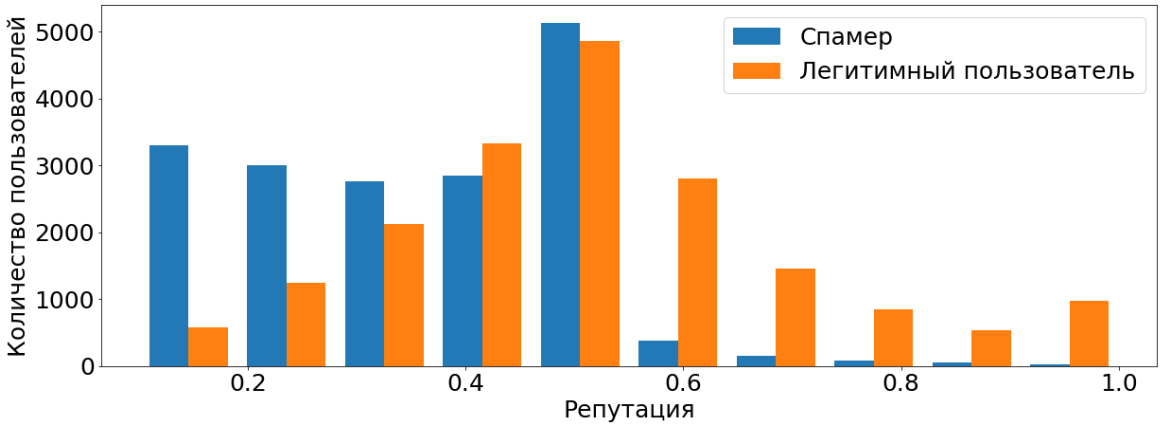

Рисунок 3 — Распределение пользователей по репутации Twitter

Признаки контента фиксируют свойства текста каждого твита (табл. 1). Среди 17 контентных атрибутов присутствует количество спам-слов и количество спам-слов на одно слово, которые генерируются путем сопоставления с известными спамовыми словами[1].

[1] https://github.com/splorp/wordpress-comment-blacklist/blob/master/blacklist.txt

Пользовательские признаки	**Признаки контента**
Длина названия профиля	Количество слов (КС)
Длина описания профиля	Количество символов
Количество подписок (ПС)	Количество пробелов
Количество подписчиков (ПЧ)	Количество слов с заглавной буквы (КЗ)
Количество твитов	КЗ/КС
«Возраст» аккаунта, ч. (ВА)	Максимальная длина слова
Соотношение подписчиков и подписок (ПЧ/ПС)	Средняя длина слова
Репутация пользователь (ПЧ/(ПС + ПЧ)))	Количество символов «!»
Прирост подписок (ПС/ВА)	Количество символов «?»
Количество твитов в день	Количество URL (КЛ)
Количество твитов в неделю	КЛ/КС
	Количество хэштегов (КХ)
	КХ/КС
	Количество упоминаний (КУ)
	КУ/КС
	Количество спамовых слов (КСП)
	КСП / КС

Таблица 1 — Список итоговых признаков

3.3. Используемые классификаторы

Наиболее популярным и эффективным подходом на данный момент является использование машинного обучения с учителем с различными признаками, основанными как на содержании сообщений, так и на свойствах отдельных профилей пользователей.

Машинное обучение (machine learning) – это область научного знания, имеющая дело с алгоритмами, «способными обучаться». Необходимость использования методов машинного обучения объясняется тем, что для многих сложных – «интеллектуальных» – задач (например, распознавание рукописного текста, речи и т. п.) очень сложно (или даже невозможно) разработать «явный» алгоритм их решения, однако часто можно научить компьютер обучиться решению этих задач. Одним из первых, кто использовал термин «машинное обучение», был изобретатель первой самообучающейся компьютерной программы игры в шашки А. Л. Самуэль в 1959 г. [21]. Под обучением он понимал процесс, в результате которого компьютер способен показать поведение, которое в нее не было заложено «явно». Это определение не выдерживает критики, так как не понятно, что означает наречие ”явно”. Более точное определение дал намного позже Т. М. Митчелл [22]: говорят, что компьютерная программа обучается на основе опыта E по отношению к некоторому классу задач T и меры качества P, если качество решения задач из T, измеренное на основе P, улучшается с приобретением опыта E.

В настоящее время машинное обучение имеет многочисленные сферы приложения, такие, как компьютерное зрение, распознавание речи, компьютерная лингвистика и обработка естественных языков, медицинская диагностика, биоинформати-

ка, техническая диагностика, финансовые приложения, поиск и рубрикация текстов, интеллектуальные игры, экспертные системы и др.

На этапе классификации и оценки были протестированы 5 алгоритмов, реализованных с использованием scikit-learn[2]: наивный байесовский классификатор (Naive Bayes classifier), метод *k* ближайших соседей (*k*-nearest neighbors algorithm, *k*-NN)), метод опорных векторов (SVM), дерево принятия решений (Decision tree), случайные леса (Random forest).

3.3.1. Наивный байесовский классификатор

Идея байесовский классификатора для спам-фильтра основана на предположении, что некоторые слова особенно часто встречаются в спаме [23]. Отсюда возникает идея посчитать для каждого слова w из коллекции текстов количество сообщений с ним n_{ws} в спаме (spam) и количесвто сообщений с ним n_{wh} в «не спаме» (ham), а затем оценить вероятность появления каждого слова w в спамном и неспамном тексте:

$$P(w|spam) = n_{ws}/n_s; P(w|ham) = n_{wh}/n_s; \tag{1}$$

Получив текст сообщения, для которого нужно определить, относится оно к спаму или нет, мы можем оценить вероятность появления всего текста в классе «спам» и в классе «не спам» произведением вероятностей слов:

$$P(text|spam) = P(w_1|spam)P(w_2|spam)...P(w_N|spam) \tag{2}$$

$$P(text|ham) = P(w_1|ham)P(w_2|ham)...P(w_N|ham) \tag{3}$$

«Наивность» подхода в этом случае состоит в предположении, что вхождение разных слов в текст – это независимые события.

Получив текст сообщения, для которого нужно определить, относится оно к спаму или нет, мы можем выбрать тот класс, в котором вероятность возникновения этого текста, домноженная на априорную вероятность класса, будет максимальна:

$$a(text) = arg\ max_y\ P(y|text) = P(y) \times P(text|y) \tag{4}$$

При этом важно понимать, что если новый текст содержит хотя бы одно слово, которое до этого не встречалось ни в одном из классов, то мы не сможем классифицировать текст как $spam$ или ham, поскольку $P(text|spam) = P(text|ham) = 0$.

В библиотеке sklearn наивный байесовский классификатор реализуется с помощью модуля sklearn.naive_bayes[3].

[2] http://scikit-learn.org/

[3] http://scikit-learn.org/stable/modules/naive_bayes.html

3.3.2. Метод *k* ближайших соседей

Метод *k* ближайших соседей (*k* nearest-neighbor, *k*-NN) относится к наиболее простым и в то же время универсальным методам, используемым как для решения задач классификации, так и восстановления регрессии [24]. В случае классификации новый объект классифицируется путем отнесения его к классу, являющемуся преобладающим среди *k* ближайших (в пространстве признаков) объектов из обучающей выборки. Если *k* = 1, то новый объект относится к тому же классу, что и ближайший объект из обучающей выборки (см. Рис 4, 5).

Рисунок 4 — Демонстрация метода *k*-NN, *k* = 1

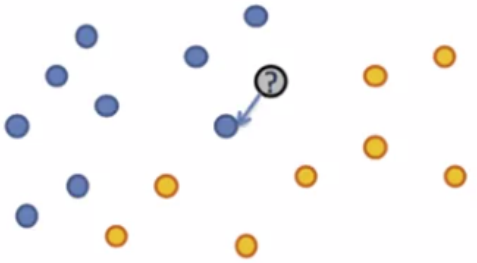

Рисунок 5 — Демонстрация метода *k*-NN, *k* = 1

Можно модифицировать этот метод, поскольку принимать решения по одной точке может быть не очень надежно. Посмотрим на *k* ближайших точек, выберем среди них доминирующий класс и отнесем новую точку к этому классу (см. Рис 6).

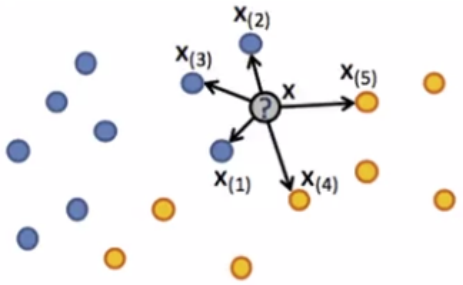

Рисунок 6 — Демонстрация метода *k*-NN, *k* = 5

Также в алгоритм *k*-NN можно добавить веса. Веса могут зависеть от номера соседа $w(x_i = w(i))$ или расстояния до соседа $w(x_i = w(d(x, x_i)))$

Имея функцию весов можно определить класс как:

$$Z_{(1)} = w(x_1) + w(x_2) + ... + w(x_m) \tag{5}$$

$$Z_{(2)} = w(x_{m+1}) + w(x_{m+2}) + ... + w(x_n) \tag{6}$$

$$arg\ max_Z Z_{(i)} \tag{7}$$

Также можно классифицировать обьект посчитав центр обоих классов, как среднее арифметическое точек, которые входят в один и в другой класс (см. Рис 7) и присвоить класс ближайшего центра новой точке:

$$\mu_{-1} = \frac{1}{l_{-1}} \sum_{i:y_i=-1} x_i \tag{8}$$

$$\mu_{+1} = \frac{1}{l_{+1}} \sum_{i:y_i=+1} x_i \tag{9}$$

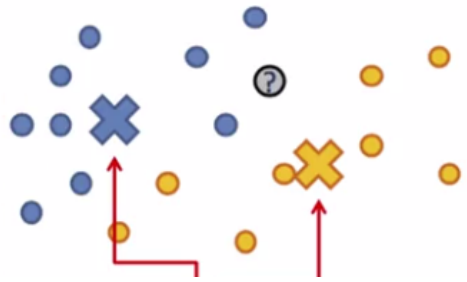

Рисунок 7 — Демонстрация метода *k*-NN

При решении задачи регрессии с помощью метода *k*-NN суммируются не просто веса обьектов, а веса, умноженные на значения функции, которую мы хотим приблизить, в этих объектах, нормировав ее на сумму всех весов.

В библиотеке sklearn метод *k*-NN реализуется с помощью модуля sklearn.Neighbors[4]

3.3.3. Метод опорных векторов (SVM)

Один из самых распространенных методов машинного обучения – машина опорных векторов (SVM – Support Vector Machine) – является развитием идей, предложенных в 1960–1970 гг. В. Н. Вапником и А. Я. Червоненкисом [25]. Данный метод представляет из себя линейный классификатор, использующий кусочно-линейную функцию потерь и L2-регуляризатор:

$$\sum_{i=1}^{l} L(M_i) + \gamma||w||^2 \to min_w \tag{10}$$

[4]http://scikit-learn.org/stable/modules/neighbors.html

где $L(M_i)$ - функция потерь, а $\gamma||w||^2$ - регуляризатор

Идею метода удобно проиллюстрировать на следующем простом примере: даны точки на плоскости, разбитые на два класса (Рисунок 8).

Проведем линию, разделяющую эти два класса. Далее, все новые точки (не из обучающей выборки) автоматически классифицируются следующим образом: точка выше прямой попадает в класс A, точка ниже прямой — в класс B.

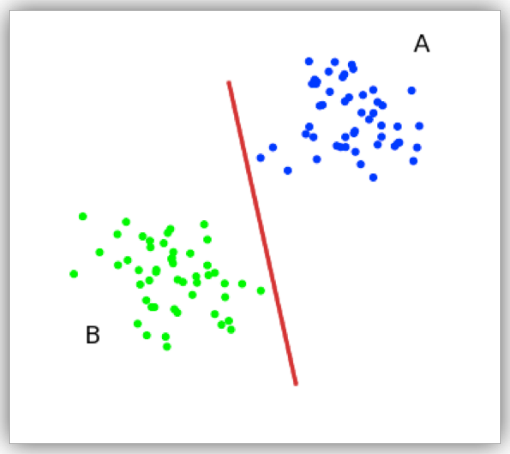

Рисунок 8 — Демонстрация метода опорных векторов

Такую прямую назовем разделяющей прямой. Однако в пространствах высоких размерностей прямая уже не будет разделять наши классы, так как понятие «ниже прямой» или «выше прямой» теряет всякий смысл. Поэтому вместо прямых необходимо рассматривать гиперплоскости — пространства, размерность которых на единицу меньше, чем размерность исходного пространства. В $\mathbb{R}^3$, например, гиперплоскость — это обычная двумерная плоскость. В нашем примере существует несколько прямых, разделяющих два класса (Рисунок 9):

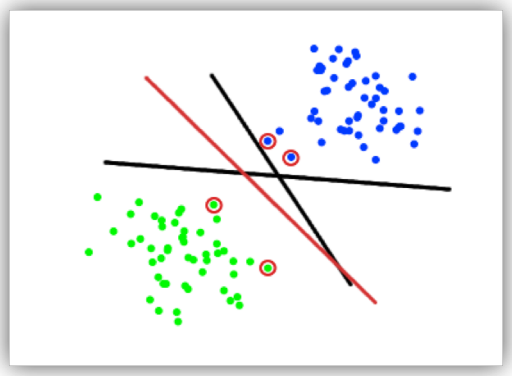

Рисунок 9 — Демонстрация метода опорных векторов

С точки зрения точности классификации лучше всего выбрать прямую, расстояние от которой до каждого класса максимально. Другими словами, выберем ту прямую, которая разделяет классы наилучшим образом (красная прямая на рис. 9). Такая прямая, а в общем случае — гиперплоскость, называется оптимальной разделя-

ющей гиперплоскостью. Другими словами, оптимальной разделяющей гиперплоскостью называется гиперплоскость, ортогональная отрезку, соединяющему ближайшие точки выпуклых оболочек двух классов, и проходящая через середину этого отрезка.

Вектора, лежащие ближе всех к разделяющей гиперплоскости, называются *опорными векторами* (support vectors). На Рисунке 9 они помечены красными кружочками.

Для дальнейшей формализации положим в математической постановке задачи обучения с учителем,$\mathbb{Y} = \{-1, 1\}, \mathbb{X} = \mathbb{R}^n$. Рассмотрим случай линейной разделимости. Пусть имеется обучающая выборка: $(x_1, y_1), \ldots, (x_m, y_m), x_i \in \mathbb{R}^n, y_i \in \{-1, 1\}$. Метод опорных векторов строит классифицирующую функцию в виде

$$f(x) = sign(w \times x + b) \tag{11}$$

где $w \times x$ — скалярное произведение, w — нормальный вектор к разделяющей гиперплоскости,b — вспомогательный параметр. Те объекты, для которых $f(x) = 1$ попадают в один класс, а объекты с $f(x) = -1$ — в другой. Выбор именно такой функции неслучаен: любая гиперплоскость может быть задана в виде $w \times x + b = 0$ для некоторых w и b.

Далее, мы хотим выбрать такие w и b которые максимизируют расстояние до каждого класса. Можно подсчитать, что данное расстояние равно $\frac{1}{||w||}$ (Рисунок 10) Проблема нахождения максимума $\frac{1}{||w||}$ эквивалентна проблеме нахождения минимума $||w||^2$. Запишем все это в виде задачи оптимизации:

$$\begin{cases} arg\ min_{(w,b)}\ ||w||^2, \\ y_i(w \times x + b) \geqslant 1, i = 1, \ldots, m \end{cases} \tag{12}$$

которая является стандартной задачей квадратичного программирования и решается с помощью множителей Лагранжа.

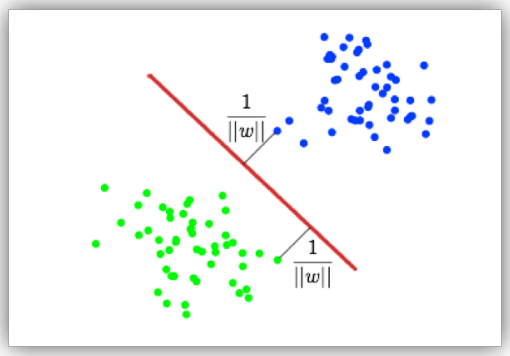

Рисунок 10 — Демонстрация метода опорных векторов

На практике случаи, когда данные можно разделить гиперплоскостью, или, как еще говорят, *линейно*, довольно редки. Пример линейной неразделимости можно видеть на Рисунок 11

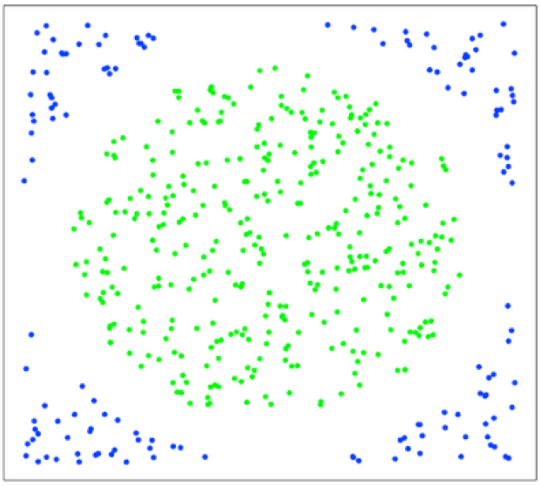

Рисунок 11 — Демонстрация метода опорных векторов

В этом случае поступают так: все элементы обучающей выборки вкладываются в пространство $\mathbb{Z}$ более высокой размерности с помощью специального отображения $\varphi : \mathbb{R}^n \to Z$. При этом отображение выбирается так, чтобы в новом пространстве Z выборка была *линейно* разделима. Классифицирующая функция f принимает вид

$$f(x) = sign(w \times \varphi(x) + b) \tag{13}$$

Выражение $K(x, x') = \varphi(x) \times \varphi(x')$ называется *ядром* классификатора. С математической точки зрения ядром может служить любая положительно определенная симметричная функция двух переменных. Положительная определенность необходимо для того, чтобы соответствующая функция Лагранжа в задаче оптимизации была ограничена снизу, т.е. задача оптимизации была бы корректно определена. Точность классификатора зависит, в частности, от выбора ядра. Чаще всего на практике встречаются следующие ядра:

1. Полиномиальное: $K(x, x') = (x \times x' + const)^d$
2. Радиальная базисная функция: $K(x, x') = e^{-\gamma|x-x'|^2}, \gamma > 0$
3. Гауссова радиальная базисная функция: $K(x, x') = e^{-\frac{|x-x'|^2}{(2\sigma^2)}}$
4. Сигмоид:$K(x, x') = \tanh\ (k(x \times x') + c), k > 0, c < 0$

Для того чтобы использовать метод опорных векторов для задачи классификации с числом классом $N > 2$, возможно также использовать следующие стратегии:

1. ”Каждый против каждого”: построить $N(N-1)/2$ классификаторов на всех возможных подзадачах бинарной классификации. Новый объект классифицируется всеми построенными решающими правилами, затем выбирается преобладающий класс.

2. ”Один против всех”: обучить N моделей на задачах бинарной классификации вида ”один класс против всех остальных”. Класс нового объекта выбирается по максимальному значению отступа.

В библиотеке sklearn алгоритм SVM реализуется с помощью модуля sklearn.svm.SVC[5]

3.3.4. Метод решающего дерева

Метод деревьев решений (decision trees) является одним из наиболее популярных методов решения задач классификации и прогнозирования. Иногда этот метод Data Mining также называют деревьями решающих правил, деревьями классификации и регрессии. Это семейство алгоритмов, которое очень сильно отличается от линейных моделей и в то же время занимает крайне важную роль в машинном обучениии [26].

Суть метода деревьев решений легко продемонстрировать на примере известной задачи определения судьбы пассажиров «Титаника» (см. Рис. 12)

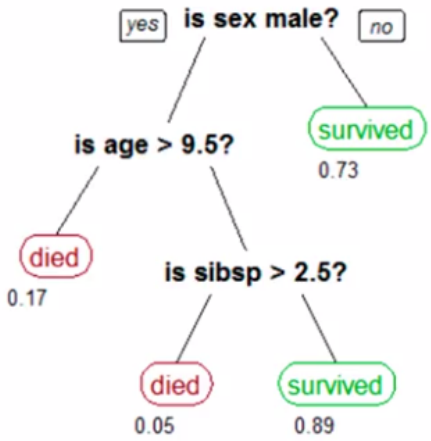

Рисунок 12 — Демонстрация метода деревьев решений

В первую очередь мы смотрим, какой пол у данного пассажира. Если это женщина, то мы сразу говорим, что она выживет, и этот ответ будет правильным в 73 % случаев. Если это мужчина, то мы смотрим, сколько ему лет. Если девять или меньше, то мы перейдем в следующую ветку, а если больше девяти, то сразу говорим, что пассажир погиб. Если пассажиру меньше девяти лет, то мы смотрим, сколько родственников у этого пассажира было на борту. Если три или больше, то говорим, что он погиб. Если меньше, то говорим, что он выжил.

В общем случае решающее дерево это некоторое бинарное дерево (не всегда), у которого в каждой внутренней вершине записано простое условие. В зависимости от того, верное оно или нет, мы будем идти либо вправо, либо влево от этой вершины. В каждом листе решающего дерева записан некоторый прогноз. Таким образом, мы

[5]http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

берем некий объект, стартуем из корня и движемся по дереву, проверяя условие в текущей вершине. В зависимости от его выполнения идем либо влево, либо вправо. В конце концов мы попадаем в лист, в котором записан прогноз, который и выдается в качестве ответа модели.

Можно строить и более сложные небинарные деревья, но, как правило, используются именно бинарные. Этого достаточно, чтобы решать большинство задач.

Условия во внутренних вершинах также используются крайне простые. Наиболее частый вариант — проверка, находится ли значение j-того признака левее, чем некоторый порог $[x^j \leqslant t]$. То есть мы берем у объекта j-тый признак, сравниваем с порогом t, и если оно меньше порога, мы идем влево, если больше порога, мы идем вправо.

Прогноз в листе будет вещественным, если это регрессия, и он будет пытаться как можно лучше приблизить истинный ответ. Если это классификация, то есть два варианта: дерево может выдавать либо номер класса (тогда в каждом листе будет записан просто тот или иной класс), либо распределение вероятности на классах. В этом случае в каждом листе будет записан некоторый вектор длины k, если k — это число классов, который будет говорить, насколько вероятно, что объект относится к тому или иному классу.

Посмотрим, как выглядят зависимости, которые восстанавливают решающие деревья. Рассмотрим задачу классификации с двумя признаками. В ней три класса, и видно, что решающее дерево может очень неплохо отделить каждый класс от всех остальных (см. Рис. 13)

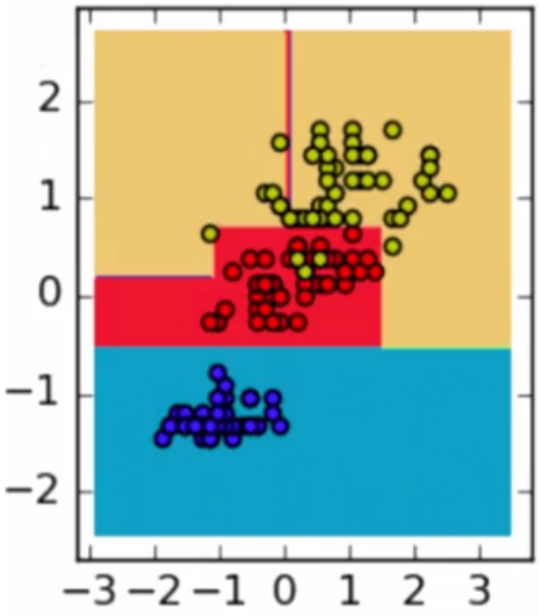

Рисунок 13 — Демонстрация метода деревьев решений

Видно, что разделяющая поверхность каждого класса кусочно-постоянная, и при этом каждая сторона разделяющей поверхности параллельна одной из осей координат из-за того, как именно мы выбрали условия. Каждое условие сравнивает значение ровно одного признака, ровно одной координаты с порогом.

В то же время решающее дерево может легко переобучиться. Его можно сделать настолько глубоким, что каждый лист решающего дерева будет соответствовать ровно одному объекту обучающей выборки (см. Рис. 14).

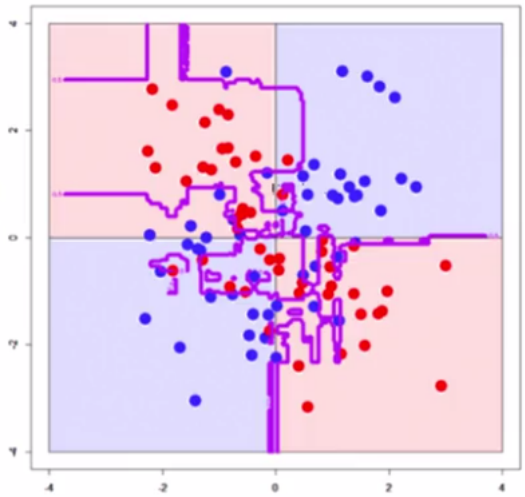

Рисунок 14 — Демонстрация метода деревьев решений

В этом случае, если мы запишем в каждом листе ответ соответствующего объекта, мы получим нулевую ошибку на обучающей выборке, но в то же время дерево будет явно переобученным.

Поскольку дерево может достичь нулевой ошибки на обучающей выборке, нам не подходит любое дерево. Теоретически, каждую задачу можно решить деревом, которое будет иметь нулевую ошибку и при этом не быть переобученным, и, скорее всего, это будет минимальное дерево из всех, у которых нулевая ошибка. Минимальным оно может быть, например, в смысле количества листьев, то есть можно поставить задачу построить такое решающее дерево, которое не ошибается на данной выборке и при этом имеет меньше всего листьев. К сожалению, эта задача NP-полная, то есть ее невозможно решить за разумное время, поэтому в машинном обучении пользуются гораздо более простым подходом: строят дерево жадно, последовательно, от корня к листьям, а именно мы начинаем с пустого дерева, дальше выбираем каким-то образом корень, который разбивает нашу выборку на две, дальше разбиваем потомков этого корня и так далее. Ветвим дерево до тех пор, пока не решим, что этого достаточно. Пусть в вершине m оказалась выборка X_m. Будем использовать некоторый критерий ошибки Q, который зависит от того, какие объекты попали в данную вершину, то есть X_m, и от параметров разбиения j и t, то есть на основе какого признака мы разбиваем и с каким порогом мы сравниваем значение этого признака.

Будем выбирать параметры j и t-разбиения так, чтобы они минимизировали данный критерий ошибки Q. Подбирать параметр j можно перебором, поскольку признаков у нас конечное число, а из всех возможных значений параметра t (порога) можно рассматривать только те, при которых получаются различные разбиения. Можно показать, что этих значений t столько, сколько различных значений признака j на обучающей выборке. Например, можно отсортировать все значения j-того

признака и брать пороги между этими значениями.

$$Q(X_m, j, t) \to min_{j,t} \tag{14}$$

После того как мы выбрали конкретное разбиение, выбрали оптимальные значения параметров j и t, мы разбиваем нашу вершину на две: левую и правую. При этом часть объектов, а именно те, на которых j-тый признак меньше или равен порогу t, отправляются влево (будем обозначать это подмножество как X_l), а часть объектов из X_m, те, у которых значение j-того признака больше порога t, отправляются вправо, и это подмножество обозначается как X_r:

$$X_l = x \in X_m | [x^i \leqslant t] \tag{15}$$

$$X_r = x \in X_m | [x^i > t] \tag{16}$$

Эту процедуру можно повторить дальше для двух дочерних вершин, тем самым углубляя наше дерево.

Процесс создания дерева происходит сверху вниз, т.е. является нисходящим. В ходе процесса алгоритм должен найти такой критерий расщепления, иногда также называемый критерием разбиения, чтобы разбить множество на подмножества, которые бы ассоциировались с данным узлом проверки. Каждый узел проверки должен быть помечен определенным атрибутом. Существует правило выбора атрибута: он должен разбивать исходное множество данных таким образом, чтобы объекты подмножеств, получаемых в результате этого разбиения, являлись представителями одного класса или же были максимально приближены к такому разбиению

Существуют различные критерии расщепления. Наиболее известные - мера энтропии и индекс Gini.

В некоторых методах для выбора атрибута расщепления используется так называемая мера информативности подпространств атрибутов, которая основывается на энтропийном подходе и известна под названием ”мера информационного выигрыша”(information gain measure) или мера энтропии.

Другой критерий расщепления, предложенный Брейманом (Breiman) и др., реализован в алгоритме CART и называется индексом Gini. При помощи этого индекса атрибут выбирается на основании расстояний между распределениями классов. Если дано множество T, включающее примеры из n классов, индекс Gini, т.е. $gini(T)$, определяется по формуле 17:

$$gini(T) = 1 - \sum_{i=1}^{n} p_i \tag{17}$$

где T - текущий узел, p_i - вероятность класса i в узле p, n - количество классов.

В какой-то момент нам все же придется остановиться. Критериев остановки очень много. Например, можно смотреть, сколько объектов находится в данной вершине. Если там всего один объект обучающей выборки, понятно, что дальше разбивать

не имеет смысла; если же больше, можно разбивать дальше. Или, например, можно смотреть, какие объекты попали в эту вершину: если они все относятся к одному классу в задаче классификации, можно прекратить разбиение; если же есть несколько классов, можно разбивать дальше. Или, например, можно следить за глубиной дерева и останавливать разбиение, если глубина превышает некоторый порог, например, 10.

Рассмотрим каким образом выбирать ответ в листе, если мы решили объявить вершину листом. Если мы решаем задачу классификации, то наиболее логичным выбором будет возвращать тот класс, который наиболее популярен в выборке X_m, которая попала в данный лист. То есть для каждого класса y мы считаем, сколько объектов этого класса попало в данную вершину, и возвращаем тот, который максимален:

$$a_m = arg\ max_y \sum_{i \in X_m} [y_i = y] \tag{18}$$

Если же мы хотим возвращать вероятности классов в данной вершине, это тоже очень легко сделать. Вероятность k-того класса оценивается как доля объектов k-того класса в данной вершине среди всех объектов, впавших в эту вершину, то есть среди всех объектов из X_m.

$$a_{mk} = \frac{1}{X_m} \sum_{i \in X_m} [y_i = k] \tag{19}$$

В библиотеке sklearn алгоритм дерева решений реализуется с помощью модуля sklearn.trees[6].

3.3.5. Метод случайных лесов

Решающее дерево слишком легко подгоняется под обучающую выборку и получается непригодным для построения прогнозов. Бороться с переобучением довольно сложно. Надо либо использовать критерии остановок, которые слишком простые и не всегда помогают, либо делать стрижку деревьев, которая, наоборот, слишком сложная. Однако решающие деревья очень хорошо подходят для объединения в композиции, для построения одного непереобученного алгоритма на основе большого количества решающих деревьев. Композиция алгоритмов — это объединение N алгоритмов в один. Алгоритм $a(x)$, который возвращает знак среднего (в случае задачи классификации) или просто среднее (в случае задачи регрессии) и называется композицией n алгоритмов. А алгоритмы $b1, ..., bn$, которые мы объединяем в композицию, называются базовыми алгоритмами:

[6] scikit-learn.org/stable/modules/tree.html

$$a(x) = sign\frac{1}{N}\sum_{n=1}^{N} b_n(x) \tag{20}$$

Для того чтобы строить композицию, нужно обучить n базовых алгоритмов. При этом понятно, что нельзя их обучать на всей обучающей выборке. Они получатся одинаковыми, и в их усреднении не будет никакого смысла. Нужно делать их немного различными. Например, с помощью рандомизации, то есть обучать их по разным подвыборкам обучающей выборки. Поскольку решающие деревья очень сильно меняются даже при небольших изменениях обучающей выборки, такая рандомизация с помощью подвыборок будет очень хорошо влиять на их различность. Один из популярных подходов по построению подвыборок — это бутстрап [27]. Предположим, что у нас есть полная обучающая выборка, состоящая из l объектов. Мы генерируем из нее l объектов с возвращением, то есть мы берем из нее некоторый случайный объект, записываем его в новую выборку и возвращаем обратно. То есть в какой-то момент мы можем снова его вытянуть и снова поместить в обучающую выборку. При этом новая выборка будет тоже иметь размер l, но при этом какие-то объекты будут в ней повторяться, а какие-то объекты исходной обучающей выборки не встретятся ни разу. Можно показать, что количество различных объектов в бутстрапированной выборке будет равняться $0,632 \times l$, то есть примерно 63 % объектов исходной выборки будет содержаться в бутстрапированной.

Есть и другой подход к рандомизации. Это просто генерация случайного подмножества обучающей выборки. Например, мы берем случайные 50 % объектов и на них обучаем базовый алгоритм. Этот подход чуть хуже, потому что в нем появляется гиперпараметр — размер подвыборки. В нашем примере это было 50 %. В случае же с бутстрапом никаких параметром нет. Он без какой-либо настройки выдает нам подвыборку, что гораздо удобнее.

Среди достоинств алгоритма случайных деревьев можно выделить:

1. высокое качество предсказания;
2. способность эффективно обрабатывать данные с большим числом классов и признаков;
3. внутреннюю оценку обобщающей способности модели;
4. легко построить параллельную высоко масштабируемую версию алгоритма;
5. метод обладает всеми преимуществами деревьев решений, в том числе о отсутствием необходимости предобработки входных данных, обработкой как вещественных, так и категориальных признаков, о поддержкой работы с отсутствующими значениями.

К недостаткам можно отнести:

6. склонность к переобучению на некоторых задачах, особенно на зашумленных задачах;

7. большой размер получающихся моделей. Требуется памяти для хранения модели, где — число деревьев.

В библиотеке sklearn модели «случайный лес» реализуются с помощью модуля sklearn.ensemble[7].

[7] http://scikit-learn.org/stable/modules/ensemble.html

4. Проведенные эксперименты

Для обучения было использовано 40% начальной выборки. В процессе оптимизации гиперпараметров для каждого классификатора использовалась кросс-валидация (CV-10). Кросс-валидация способствует минимизации риска переобучения, и следовательно, смещения в оценке качества классификатора. Оптимизация проводилась для обоих групп признаков (каждый признак был нормализован и принимал значение в диапазоне [-1, 1]). Кроме этого для настройки SVM было отобрано 30% наиболее значимых признаков по критерию хи-квадрат, поскольку это более оптимально с точки зрения вычислительной нагрузки.

Далее приведены результаты оптимизации гиперпараметров с помощью полного перебора по сетке.

4.1. Подбор параметров по сетке

4.1.1. Naive Bayes

Единственным параметром, по которому проходил поиск по сетке для наивного байесовского классификатора, это параметр *fit_prior*, который принимает булевые значения *True* или *False*. Суть этого параметра в использовании предыдущих вероятностей класса при обучении или их игнорировании.

Fit_Prior	**Accuracy**	**Std**
True	0.75619	0.00392
False	**0.75718**	**0.8549**

Таблица 2 — Подбор по сетке Naive Bayes

4.1.2. *k*-nearest neighbors

Для алгоритма *k*-NN оптимальной оказалась модель с $k = 13$ и применением весов, зависящих от расстояния до соседа (см. п. 3.3.2)

Number of neighbors *k*	Weights	Accuracy	Std
k = 1	uniform	0.8549	0.00398
k = 1	distance	0.8549	0.00398
k = 3	uniform	0.87634	0.00353
k = 3	distance	0.87596	0.00346
k = 5	uniform	0.88281	0.00304
k = 5	distance	0.88302	0.00295
k = 7	uniform	0.88481	0.00367
k = 7	distance	0.88525	0.00360
k = 9	uniform	0.88546	0.00424
k = 9	distance	0.88635	0.00401
k = 11	uniform	0.88560	0.00432
k = 11	distance	0.88737	0.00448
k = 13	uniform	0.88618	0.00394
k = 13	**distance**	**0.88844**	**0.00369**
k = 15	uniform	0.88612	0.00359
k = 15	distance	0.88802	0.00272
k = 17	uniform	0.88551	0.00345
k = 17	distance	0.88797	0.00300
k = 19	uniform	0.88481	0.00315
k = 19	distance	0.88802	0.00361

Таблица 3 — Подбор по сетке *k*-NN

4.1.3. SVM

Для алгоритма SVM оптимальной оказалась модель с сигмоидом в качестве ядра (см. пункт 3.3.3)

Kernel	Accuracy	Std
linear	0.88560	0.00432
poly	0.88481	0.00401
rbf	0.8549	0.00394
sigmoid	**0.88802**	**0.00398**
precomputed	0.8549	0.00300

Таблица 4 — Подбор по сетке *k*-NN

4.1.4. Decision tree

Для алгоритма Decision tree оптимальной оказалась модель с использованием меры энтропии в качестве критерия расщепления (см. пункт 3.3.4), без использования ограничений на глубину дерева, а также стратегией расщепления нацеленной на «наилучшее», а не случайное расщепление.

Criterion	Max depth	Splitter	Accuracy	Std
Gini	1	best	0.84177	0.00516
Gini	1	random	0.54702	0.03000
Gini	2	best	0.84177	0.00516
Gini	2	random	0.59500	0.02743
Gini	3	best	0.87258	0.00449
Gini	3	random	0.64052	0.07874
Gini	None	best	0.88475	0.00386
Gini	None	random	0.87360	0.00496
Entropy	1	best	0.84182	0.00480
Entropy	1	random	0.55109	0.03190
Entropy	2	best	0.84182	0.00480
Entropy	2	random	0.58652	0.03546
Entropy	3	best	0.86879	0.00419
Entropy	3	random	0.67764	0.06590
Entropy	**None**	**best**	**0.88884**	**0.00441**
Entropy	None	random	0.87409	0.00369

Таблица 5 — Подбор по сетке DT

4.1.5. Random forest

Для алгоритма Random forest оптимальной оказалась модель, состоящая из 19 деревьев, с использованием меры энтропии в качестве критерия расщепления (см. пункт 3.3.4), без использования ограничений на глубину деревьев и без использования бутстрапа (см. пункт 3.3.5).

Estimators	Criterion	Max Depth	Bootstrap	Accuracy	Std
1	entropy	None	False	0.88533	0.00387
2	entropy	None	False	0.87751	0.00397
3	entropy	None	False	0.91223	0.00430
4	entropy	None	False	0.90904	0.00313
5	entropy	None	False	0.91826	0.00300
6	entropy	None	False	0.91656	0.00358
7	entropy	None	False	0.92170	0.00378
8	entropy	None	False	0.92052	0.00251
9	entropy	None	False	0.92294	0.00340
10	entropy	None	False	0.92216	0.00363
11	entropy	None	False	0.92441	0.00347
12	entropy	None	False	0.92423	0.00263
13	entropy	None	False	0.92441	0.00258
14	entropy	None	False	0.92523	0.00323
15	entropy	None	False	0.92617	0.00309
16	entropy	None	False	0.92571	0.00254
17	entropy	None	False	0.92643	0.00405
18	entropy	None	False	0.92645	0.00305
19	**entropy**	**None**	**False**	**0.92685**	**0.00312**

Таблица 6 — Подбор по сетке Random Forest

4.2. Выбор классификатора

После оптимизации гиперпараметров каждый из алгоритмов оценивался на оставшихся 60% данных с использованием обоих групп признаков и CV-10. Также как и в случае поиска по сетке, каждый признак был нормализован, а для SVM использовалось 30% наиболее значимых признаков по критерию хи-квадрат.

В качестве метрик оценки классификаторов были выбраны полнота (R), точность (P) и F-мера. Метрика полноты представляет из себя отношение числа сообщений, корректно классифицированных как спам (True Positives) и общего числа спамовых сообщений (True Positives + False Negatives):

$$R = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{21}$$

Точность - это отношение числа сообщений, корректно классифицированных как спам (True Positives) и общего числа сообщений, классифицированных как спам (True Positives + False Positives):

$$P = \frac{TruePositives}{TruePositives + FalsePositives} \tag{22}$$

F-мера может быть проинтерпретирована как гармоническое среднее между метриками точности и полноты:

$$F1 = \frac{2 \times P \times R}{P + R} \tag{23}$$

Как видно из таблицы 7, классификаторы на основе дерева показали лучший результат. Random Forests обошел остальные классификаторы по показателю F-меры. Наилучший известный мне результат в данной задаче был достигнут у Lee [9] и составляет 98% по той же оценке F-меры. Однако Lee использовал историчные признаки, что является ограничением в нашем случае, следовательно, результат можно считать вполне неплохим. Это говорит о том, что лишь те признаки, которые можно извлечь непосредственно из твита и его метаданных, могут служить хорошей основой для классификатора.

Классификатор	**Точность**	**Полнота**	**F-мера**
Naive Bayes	76%	76%	76%
k-NN	86%	86%	86%
SVM	86%	86%	86%
Decision Tree	89%	89%	89%
Random Forest	**93%**	**93%**	**93%**

Таблица 7 — Сравнение показателей классификаторов

4.3. Оценка признаков

Для оценки значимости признаков были использованы их различные комбинации при обучении классификаторов. Результат однозначно показал, что наибольший вес с точки зрения F-меры имеют признаки из пользовательской группы. Комбинации с использованием признаков из этой группы с различными признаками из группы признаков контента дают лишь небольшой прирост в F-мере.

Еще одним важным аспектом, который желательно учитывать при отборе признаков, это время их вычисления, особенно если существует необходимость применения классификатора в режиме реального времени. Таблица 8 демонстрирует время вычисления каждой из групп признаков для 1000 твитов со следующими параметрами машины:

1. CPU: Intel Core i7
2. RAM: 16GB
3. OS: Ubuntu 16.04 64-bit

Группа признаков	Время на вычисление (сек.) для 1000 твитов
Пользовательские признаки	0.0057
Признаки контента	1.0448

Таблица 8 — Время вычисления признаков

Следует отметить, что обе группы признаков вычисляются за приемлемое время, что отвечает условиям поставленной задачи, однако такие признаки как *Количество спамовых слов (КСП)* и *Количество спамовых слов на одно слово (КСП/КС)* наиболее затратны с точки зрения вычисления и не приносят ощутимого прироста в точность модели, что можно предположительно можно обьяснить тем, что большинство из этих спамовых слов взяты из рекламных блогов и спамовых писем, для которых нехарактерна спамовая семантика коротких сообщений, поэтому этот признак можно исключить из выборки из соображений эффективности времени работы классификатора.

4.4. Анализ результатов

Наиболее качественный результат был получен с помощью алгоритма Random Forest. Из всех признаков наиболее весомыми с точки зрения точности классификатора показали себя пользовательские признаки, такие как: *Прирост Подписок, Количество Подписок, Репутация* и т.д. (см рис. 15)

Рисунок 15 — Полезность признаков в алгоритме Random Forest

Действительно, показатель *Прирост Подписок* сильно отличается у спамовых и легитимных аккаунтов (см. рис. 16). Это можно обьяснить тем, что спамовые аккаунты нацелены на охват максимально большой аудитории жертв за короткий промежуток времени.

Кроме этого, на рисунке 17 можно наблюдать определенную положительную корреляцию между количеством подписок и возрастом аккаунта у спамовых пользователей. У легитимных аккаунтов подобная зависимость отсутствует.

Посмотрим внимательнее на каких обьектах ошибается наш классификатор.

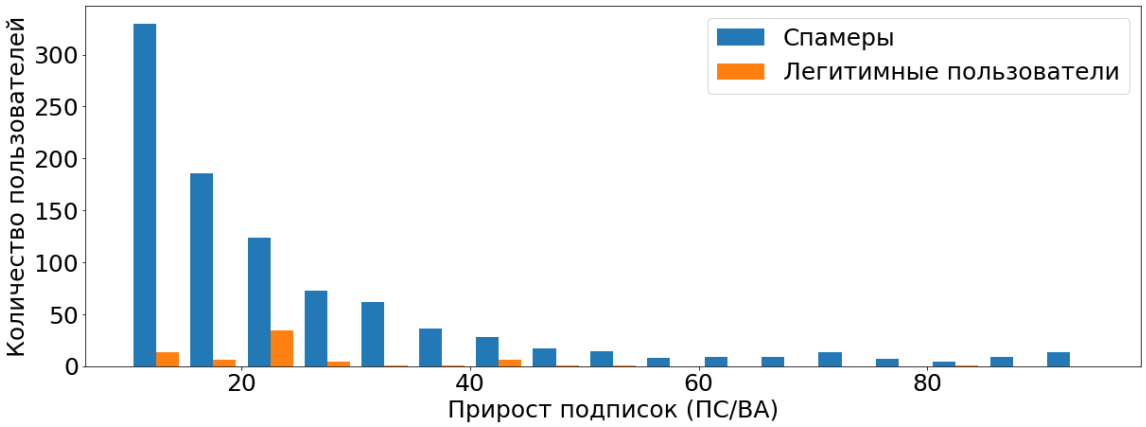

Рисунок 16 — Распределение пользователей по приросту подписок

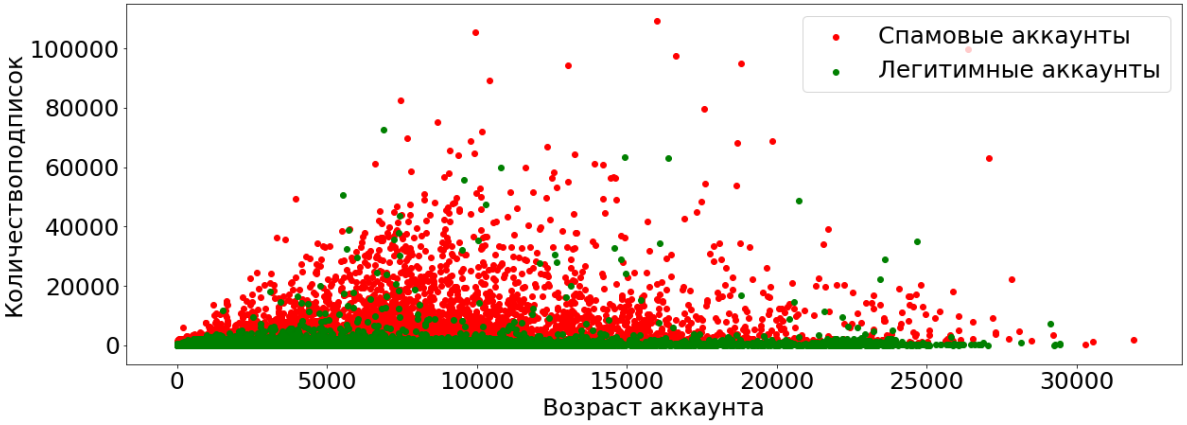

Рисунок 17 — Зависимость количества подписок от возраста аккаунта

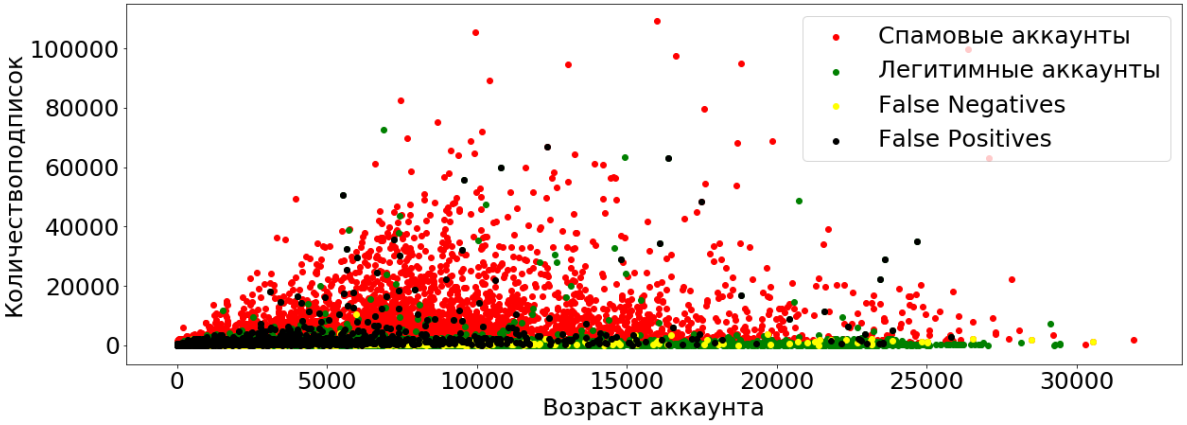

Рисунок 18 — Зависимость количества подписок от возраста аккаунта

Рисунок 18 позволяет убедится, что подавляющая доля ошибок происходит на пересечении классов.

Более чем в половине твитов, ошибочно классифицированных как спам, присутствуют ссылки (см. таблицу 9).

Количество ссылок	Доля в множестве FalsePositives
0	39.565628
1	58.545798
2	1.416431
5	0.094429

Таблица 9 — Наличие ссылок в ошибках

Примеры неверно классифицированных твитов приведены в таблицах 10 и 11.

#	Твит
1	If you aren't at SES Berlin, go to SES Chicago for 11 topics breaking http://tinyurl.com/yazwgcj
2	Hummingbird 2, by far the best marketing tool for twitter. More Info! http://bit.ly/2K5Zon
3	Need A Job? Hiring Today - Job Requires: Basic Computer Skills Your Own Home Computer. Weekly Pay : http://tinyurl.com/yzqyqoc
4	Need Parking at any UK Airport? We have used and recommend these guys: http://tinyurl.com/r6a6tm Secure, convenient and affordable.
5	We are recommending Essential Travel for affordable travel insurance throughout Europe from the UK: http://tinyurl.com/ozsjbw

Таблица 10 — Примеры FalsePositives

#	Количество Подписок	Количество подписчиков	Репутация	Прирост подписок	Твитов в день
1	1796.0	1040.0	0.366714	2.017978	2.453933
2	2807.0	2562.0	0.477184	0.976348	8.406261
3	17582.0	20181.0	0.534412	3.153156	13.635581
4	32596.0	30935.0	0.486928	5.770225	92.299522
5	35067.0	35122.0	0.500392	1.420867	23.990276

Таблица 11 — Примеры FalsePositives

Можно сделать вывод о том, что данные обьекты действительно очень похожи на спамовые, однако, в рамках политики пользования Twitter, они таковыми не являются.

Очевидно, что большую часть спама можно определить по пользовательским признакам (см. рис. 15), однако полное игнорирование признаков контента при классификации спама не является разумным решением, поскольку их использование дает определенный прирост в точности. Кроме того, использование признаков контента усложняет задачу имитации благонадежности для спамовых аккаунтов. К примеру, спамер может купить себе подписчиков, чтобы казаться более легитимным, но в его сообщениях все равно останется «спамовость», которая будет определена с помощью признаков контента.

Наиболее высокий известный мне результат в задаче классификации социального спама был достигнут Ли [9] и составляет 98% . В своей работе он протестировал 30 алгоритмов классификации, включая Naive Bayes, Logistic regression, SVM, Random Forest. Алгоритмы оценивались с помощью метрик полноты, точности и F-меры, а также кривой ошибок (ROC). Помимо взятых мною признаков в данной работе использовались историчные признаки, такие как: *Среднее количество ссылок (упоминаний, хештэгов) в твите*, *Среднее количество уникальных ссылок (упоминаний, хештэгов) в твите*, *Средняя схожесть контента твитов пользователя*,

Степень сжатия опубликованных твитов. Наиболее высокая точность была продемонстрирована с использованием алгоритма Random Forest с применением бустинга - 98%.

Сантош помимо довольно распространенных алгоритмов машинного обучения (Байесовская сеть, SVM, Random Forest, *k*-nn и т.д.) использовал классификаторы на основе сжатия текста (Динамические марковские цепи, Алгоритм сжатия PPM). Лучший результат также был получен с применением Random Forest - 96% [18].

Другой нестандартный метод классификации социального спама - алгоритм CIA [19] - не показал высоких результатов, и посему позиционируется лишь как дополнение к другим методам классификации.

Можно сделать вывод, что Random Forest является оптимальным выбором в вопросе определения классификатора социального спама.

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены различные подходы, которые применяются для построения систем определения социального спама в т.ч. с ограничением на число признаков. В результате работы были исследованы 5 алгоритмов машинного обучения, а именно наивный байесовский классификатор, метод *k* ближайших соседей, метод опорных векторов, решающее дерево и случайный лес. Лучший результат продемонстрировал алгоритм Random Forest - 93%.

Наиболее высокий известный мне результат в данной задаче составляет 98%, однако в нем отсутствуют какие-либо ограничения на историчность признаков, посему достигнутый результат можно считать довольно неплохим.

В силу отсутствия сьерьзной вычислительной нагрузки для извлечения использованных в работе признаков из твитов пользователей данный подход привлекателен своей потенциальной примененимостью в режиме реального времени.

Весь исходный код, написанный в процессе работы над классификатором, можно найти в открытом Git-репозитории[8].

[8] https://github.com/oy-vey/TwitterSpamClassifier

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Nguyen, H. Research report: 2013 state of social media spam. — 2013. http://nexgate.com/wp-content/uploads/2013/09/Nexgate-2013-State-of-Social-Media-Spam-Research-Report.pdf.

2. R, Kelly. Twitter study. — 2009. www.pearanalytics.com/wp-content/uploads/2012/12/Twitter-Study-August-2009.pdf.

3. Wang, A. H. Don't follow me: Spam detection in twitter / A. H. Wang // 2010 International Conference on Security and Cryptography (SECRYPT). — 2010. — P. 1–10.

4. @spam: The underground on 140 characters or less / Chris Grier, Kurt Thomas, Vern Paxson, Michael Zhang // Proceedings of the 17th ACM Conference on Computer and Communications Security. — CCS '10. — New York, NY, USA: ACM, 2010. — P. 27–37. http://doi.acm.org/10.1145/1866307.1866311.

5. Detecting spammers on twitter / Fabrício Benevenuto, Gabriel Magno, Tiago Rodrigues, Virgílio Almeida // In Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS. — 2010.

6. Sridharan, Vasumathi. Twitter games: how successful spammers pick targets. / Vasumathi Sridharan, Vaibhav Shankar, Minaxi Gupta // ACSAC / Ed. by Robert H'obbes' Zakon. — ACM, 2012. — P. 389–398. http://dblp.uni-trier.de/db/conf/acsac/acsac2012.html#SridharanSG12.

7. Suspended accounts in retrospect: An analysis of twitter spam / Kurt Thomas, Chris Grier, Dawn Song, Vern Paxson // Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference. — IMC '11. — New York, NY, USA: ACM, 2011. — P. 243–258. http://doi.acm.org/10.1145/2068816.2068840.

8. Martinez-Romo, Juan. Detecting malicious tweets in trending topics using a statistical analysis of language / Juan Martinez-Romo, Lourdes Araujo // Expert Systems with Applications. — 2013. — Vol. 40, no. 8. — P. 2992 – 3000.

9. Lee, Kyumin. Seven months with the devils: a long-term study of content polluters on twitter / Kyumin Lee, Brian David Eoff, James Caverlee // In AAAI Int'l Conference on Weblogs and Social Media (ICWSM. — 2011.

10. Twitter,. Twitter rules. — 2017. — [онлайн-источник; доступ 04.06.2017]. https://support.twitter.com/articles/18311-the-twitter-rules.

11. Twitter,. Twitter spammers list. — 2017. — [онлайн-источник; доступ 04.06.2017]. http://www.infochimps.com/datasets/twitter-spammers-list.

12. Caruana, Godwin. A survey of emerging approaches to spam filtering / Godwin Caruana, Maozhen Li // ACM Comput. Surv. — 2008. — Vol. 44, no. 2. — P. 9:1–9:27.

13. Almeida, Tiago A. Advances in Spam Filtering Techniques / Tiago A. Almeida, Akebo Yamakami // Computational Intelligence for Privacy and Security / Ed. by David A. Elizondo, Agusti Solanas, Antoni Martinez-Balleste. — Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. — P. 199–214. http://dx.doi.org/10.1007/978-3-642-25237-2_12.

14. McCord, M. Spam Detection on Twitter Using Traditional Classifiers / M. McCord, M. Chuah // Autonomic and Trusted Computing: 8th International Conference, ATC 2011, Banff, Canada, September 2-4, 2011. Proceedings / Ed. by Jose M. Alcaraz Calero, Laurence T. Yang, Félix Gómez Mármol et al. — Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. — P. 175–186. http://dx.doi.org/10.1007/978-3-642-23496-5_13.

15. Yang, Chao. Die Free or Live Hard? Empirical Evaluation and New Design for Fighting Evolving Twitter Spammers / Chao Yang, Robert Chandler Harkreader, Guofei Gu // Recent Advances in Intrusion Detection: 14th International Symposium, RAID 2011, Menlo Park, CA, USA, September 20-21, 2011. Proceedings / Ed. by Robin Sommer, Davide Balzarotti, Gregor Maier. — Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. — P. 318–337. http://dx.doi.org/10.1007/978-3-642-23644-0_17.

16. The rise of social bots / E. Ferrara, O. Varol, C. Davis et al. // CoRR. — 2014. — Vol. abs/1407.5225.

17. Twitter spammer detection using data stream clustering / Zachary Miller, Brian Dickinson, William Deitrick et al. // Inf. Sci. — 2014. — Vol. 260. — P. 64–73.

18. Twitter Content-Based Spam Filtering / Igor Santos, Igor Miñambres-Marcos, Carlos Laorden et al. // International Joint Conference SOCO'13-CISIS'13-ICEUTE'13: Salamanca, Spain, September 11th-13th, 2013 Proceedings / Ed. by Álvaro Herrero, Bruno Baruque, Fanny Klett et al. — Cham: Springer International Publishing, 2014. — P. 449–458. http://dx.doi.org/10.1007/978-3-319-01854-6_46.

19. Analyzing spammers' social networks for fun and profit: A case study of cyber criminal ecosystem on twitter / Chao Yang, Robert Harkreader, Jialong Zhang et al. // Proceedings of the 21st International Conference on World Wide Web. — WWW '12. — New York, NY, USA: ACM, 2012. — P. 71–80. http://doi.acm.org/10.1145/2187836.2187847.

20. Duplicate detection for identifying social spam in microblogs / Q. Zhang, H. Ma, W. Qian, A. Zhou // 2013 IEEE International Congress on Big Data. — 2013. — P. 141–148.

21. Samuel, Arthur L. Some studies in machine learning using the game of checkers / Arthur L. Samuel // IBM JOURNAL OF RESEARCH AND DEVELOPMENT. — 1959. — P. 71–105.

22. Mitchell, Thomas M. Machine Learning / Thomas M. Mitchell. — 1 edition. — New York, NY, USA: McGraw-Hill, Inc., 1997.

23. Wikipedia,. Naive bayes classifier. — 2017. — [онлайн-источник; доступ 04.06.2017]. https://en.wikipedia.org/wiki/Naive_Bayes_classifier.

24. Wikipedia,. K-nearest neighbors. — 2017. — [онлайн-источник; доступ 04.06.2017]. https://en.wikipedia.org/wiki/K-,nearest_neighbors_algorithm.

25. Wikipedia,. Support vector machine. — 2017. — [онлайн-источник; доступ 04.06.2017]. https://en.wikipedia.org/wiki/Support_vector_machine.

26. Wikipedia,. Decision tree. — 2017. — [онлайн-источник; доступ 04.06.2017]. https://en.wikipedia.org/wiki/Decision_tree.

27. Wikipedia,. Random forest. — 2017. — [онлайн-источник; доступ 04.06.2017]. https://en.wikipedia.org/wiki/Random_forest.