РЕФЕРАТ

Выпускная квалификационная работа содержит _ страниц, _ рисунка, _ использованных источников и _ приложение.

ЗАДАЧА КЛАССИФИКАЦИИ, МАШИННОЕ ОБУЧЕНИЕ, АНАЛИЗ ДАННЫХ, НАИВНЫЙ БАЙЕСОВСКИЙ КЛАССИФИКАТОР, ДЕРЕВЬЯ РЕШЕНИЙ, C4.5, СЛУЧАЙНЫЙ ЛЕС, PYTHON.

В данной работе реализовываются три алгоритма классификации: наивный байесовский классификатор, дерево решений по алгоритму C4.5 и случайный лес.

В теоретической части описывается задача классификации, рассмотрение различных методов ее решения. Подробно рассматриваются алгоритмы перечисленных классификаторов.

В практической части приводится спецификация конкретной задачи, описываются реализованные алгоритмы, проводится анализ эффективности методов классификации.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача классификации – одна из самых актуальных задач машинного обучения. Она встречается в медицине, экономике, геофизике, социологии, технике, биоинформатике и других отраслях. Классификация может быть разных типов, основные – двухклассовые и многоклассовые. Встречаются также задачи с нечёткими классами, классами с пересечением и без, с линейно разделимыми и неразделимыми классами.

В качестве примера прикладной задачи машинного обучения можно привести медицинскую диагностику. Объектом является пациент с определенным набором признаков – пол, рост, вес, симптомы (наличие слабости, тошноты, головной боли), результаты обследований (анализы крови), различные измерения (артериальное давление, пульс). На основе этих признаков можно определить вид заболевания, что является многоклассовой классификацией. Помимо этого, можно назначить способ лечения, оценить риск осложнений, предсказать эффективность терапии.

Другой пример – кредитный скоринг – является примером двухклассовой (бинарной) классификации. На основе признаков банку нужно принять решение, выдавать кредит данному физическому лицу или нет. В качестве признаков могут выступать доход, профессия, состав семьи, задолженности, кредитная история.

Также к задачам машинного обучения относятся распознавание символов, предсказание оттока клиентов, обнаружение спама, распознавание речи, предсказание месторождений полезных ископаемых, ранжирование, различное прогнозирование и т.д.

Таким образом, задачи классификации актуальны и часто встречаются в современном мире, их решение очень востребовано. Машинное обучение предлагает множество различных способов для решения подобных задач. К примеру, существует множество байесовских классификаторов: наивный байесовский классификатор, метод парзеновского окна, линейный дискриминант Фишера. Помимо перечисленных алгоритмов, эти задачи могут

быть решены с помощью персептрона и многослойных нейронных сетей, методом опорных векторов, решающими деревьями, логистической регрессией и множеством других методов.

В настоящей работе рассматривается классификация спортивных запросов с использованием хостовых признаков и логов поисковой системы. Цель работы – определить запросы, которые связаны с поиском расписания чемпионатов, турнирной таблицы, результатов игры и т.д. Распознав такие запросы, нужную информацию можно будет выводить сразу на странице выдачи, таким образом пользователю нет необходимости искать эту информацию по разным сайтам.

ОСНОВНАЯ ЧАСТЬ

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Задачи машинного обучения

Машинное обучение сейчас является одной из самых востребованных областей знаний. Исторически она развилась из области искусственного интеллекта. При попытке создать разумную машину обнаружилось, что запрограммировать её на решение некоторых задач вполне возможно, к примеру, на нахождение кратчайшего пути между двумя вершинами в графе. Однако решение других задач, таких как распознавание лиц, фильтрация спама, принятие решений и прочих, с которыми человек справляется без труда, запрограммировать не удается. Было предложено решать эти задачи путем самостоятельного обучения машины, когда алгоритм решения не программируется явно. В настоящее время машинное обучение используется в большом количестве прикладных задач, где присутствует большой объем данных и требуется решать задачи фильтрации, автоматизации принятия решений и т.д. Также оно решает задачи классификации, восстановления регрессии, кластеризации, идентификации, извлечения знаний, ранжирования, составления рекомендаций и многие другие.

1.2 Задача классификации

Одной из задач машинного обучения является задача классификации. Она заключается в том, чтобы разделить заданные объекты на заранее определенные классы. Изначально для обучения алгоритму даётся так называемая обучающая выборка – множество примеров, для объектов которого их классы заранее известны. Необходимо построить алгоритм, который будет определять класс произвольного объекта, основываясь на принадлежности к классам объектов из обучающей выборки.

Задачи классификации делятся на несколько типов. Один из самых простых – двухклассовая классификация. Бывают также многоклассовые классификации, пересекающиеся классы и непересекающиеся, нечёткие

классы. Помимо этого, выделяют линейно разделимые и линейно не разделимые классы.

Входные данные для классификатора могут быть представлены различными способами. Может быть дано изображение или аудио, матрица расстояний между элементами, временной ряд, текст. Но чаще всего данные представлены признаковым описанием. В этом случае каждому объекту ставятся в соответствие некоторые значения, чаще всего числовые, и признаки f_i выступают в роли такого отображения: $f_i: X \to D_j, j = 1, \dots n;$ X – множество объектов, D_j – область определения для j-го признака. Объект будет описывается вектором фиксированного размера, каждый элемент которого представляет собой определенный признак. Чаще всего встречаются бинарные признаки (принимающие только значения $+1$ или -1) и количественные (принимающие любое числовое значение). Бывают также номинальные (категориальные) и порядковые признаки, которые принимают значения из какого-то ограниченного множества, в последнем случае в этом множестве введено отношение порядка. Примером бинарного признака может служить пол человека, количественного – его рост, номинального – цвет глаз, порядкового – воинское звание.

Формальное определение задачи классификации: изначально даны множество объектов X и множество наименований классов (меток, ответов) Y. Существует неизвестная зависимость $y: X \to Y$, значения которой известны на объектах конечной обучающей выборки $X^l = (x_i, y_i)_{i=1}^l, y_i = y(x_i)$. Необходимо найти классификатор $a: X \to Y$ с минимальной вероятностью ошибки.

Признаковое описание объекта $x \in X$ выглядит следующим образом: $\big(f_1(x), \dots, f_n(x)\big)$, где $f_1, \dots, f_n$ – множество некоторых признаков. Множество таких описаний для всего множества объектов представляется в виде матрицы «объекты-признаки»:

$$F = \left\| f_j(x_i) \right\|_{l \times n} = \begin{pmatrix} f_1(x_1) & \cdots & f_n(x_1) \\ \vdots & \ddots & \vdots \\ f_1(x_l) & \cdots & f_n(x_l) \end{pmatrix} \quad (1)$$

Каждой строке этой матрицы соответствует ответ или метка класса. В случае бинарной классификации ответы могут принимать два возможных значения: $Y = \{+1, -1\}$. Для многоклассовой классификации есть несколько вариантов значений: $Y = \{1, \dots, M\}$.

1.3 Решение задач классификации

Для решения задач классификации существует множество методов: нейронные сети, сокращение размерности, байесовские классификаторы, деревья решений, линейные разделители, алгоритмическая композиция. Какой бы алгоритм не использовался, его работа делится на два этапа: этап обучения и этап применения (тестирования). На первом этапе по обучающей выборке строится функция, которая будет классифицировать новые объекты. На втором этапе даётся выборка из новых объектов, называемая тестовой выборкой, на которой полученная функция выдает ответы – предсказанные классы. Тестовая и обучающая выборки не пересекаются, так как иначе на объекты, присутствующие в обучающем множестве, классификатор будет выдавать ответы, которые были заранее известны на этапе обучения. На таких элементах точность будет близка к 100%, а классы элементов, которые классификатор раньше не видел, будут предсказаны неправильно. Такой эффект называется переобучением, когда алгоритм работает практически безошибочно на обучающей выборке, но с большим количеством ошибок на тестовой выборке с произвольными элементами.

В общем случае существует множество функций $A = \{a_i\}$, способных предсказать класс объекта, которые могут различаться между собой, например, значениями некоторых параметров. Необходимо каким-то образом выбирать из них наилучшую функцию, и для этого вводится функция потерь $L(a, x)$, которая характеризует величину ошибки алгоритма $a \in A$ на элементе $x \in X$. Для задачи классификации функция потерь имеет вид $L(a, x) = [a(x) \neq$

$y(x)]$, и равна 1, если алгоритм классифицировал неверно, или 0, если ответ правильный. Для оценки ошибки алгоритма на всем множестве X вводится функционал $Q(a, X^l) = \frac{1}{l}\sum_{i=1}^{l} L(a, x_i)$, называемый эмпирическим риском. Необходимо минимизировать этот функционал по параметрам модели, и таким образом получить наилучший алгоритм, который дает наименьшую ошибку.

1.4 Байесовские классификаторы

Рассмотрим байесовские классификаторы - класс методов, который состоит из алгоритмов, основанных на нахождении класса с максимальной апостериорной вероятностью. Апостериорная вероятность – условная вероятность при некотором фактически наблюдаемом условии. К примеру, вероятность того, что случайно взятый фрукт из корзины является апельсином, если известно, что взятый фрукт – круглый.

Для работы байесовских методов необходима полная априорная информация о классах, то есть функция правдоподобия для каждого из классов и вероятности появления классов (априорные вероятности). Функция правдоподобия – плотность совместного распределения выборки из параметрического распределения, рассматриваемая как функция параметра. Например, функция правдоподобия $P(x|y)$ для заданного y показывает вероятность того, что случится событие x при конкретном значении параметра y. По этим данным высчитываются апостериорные вероятности согласно формуле Байеса:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}, \qquad (2)$$

где y – предположительный класс, x – объект классификации, $P(y)$ – априорная вероятность класса y, $P(x|y)$ – функция правдоподобия для класса y, $P(y|x)$ – апостериорная вероятность класса y, $P(x)$ – априорная вероятность объекта x. Смысл этой формулы можно воспринимать как перестановку причины и следствия местами.

После расчета вероятностей для каждого класса необходимо найти класс y^*, который даёт наибольшую вероятность для заданного объекта x, т.е:

$$y^* = \arg\max_{y \in Y} \frac{P(x|y)P(y)}{P(x)}, \quad (3)$$

где Y – множество классов. На практике часто сталкиваются с тем, что плотности распределения классов и вероятности их появления неизвестны, поэтому их приходится находить с некоторой погрешностью по обучающей выборке. Для восстановления функции правдоподобия используются разные предположения и методы, которые порождают различные байесовские алгоритмы классификации.

1.5 Наивный байесовский классификатор

Одним из примеров таких алгоритмов является наивный байесовский классификатор. В его основе лежит предположение о том, что все признаки (вероятности) независимы и имеют одинаковый вклад в результат вычислений.

Представим классифицируемый объект в виде набора его признаков: $x = (x_1, x_2, \dots, x_n)$. В общем случае признаки могут быть зависимы друг от друга, и вероятность $P(x|y)$ представляет собой вероятность $P(x|y, x)$. В наивном байесовском классификаторе делается предположение о том, что признаки друг от друга не зависят, и в этом случае условную вероятность вектора признаков можно представить в виде произведения условных вероятностей отдельных признаков:

$$P(x|y) = P(x_1|y)P(x_2|y) \dots P(x_n|y) \quad (4)$$

Тогда формула Байеса принимает следующий вид:

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)}{P(x_1)P(x_2) \dots P(x_n)} \quad (5)$$

Так как знаменатель при вычислении с разными классами не меняется, а значит, не влияет на получающиеся вероятности, его можно не учитывать. Получаем:

$$P(y|x_1, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y) \quad (6)$$

После всех перечисленных упрощений задача классификации сводится к нахождению следующего класса y^*:

$$y^* = \arg\max_{y \in Y} P(y) \prod_{i=1}^{n} P(x_i|y) \quad (7)$$

Таким образом, для классификации нужно знать априорные вероятности $P(y)$ каждого класса, а также условные вероятности $P(x_i|y)$ каждого признака. В первом случае можно взять долю объектов из обучающей выборки, которые принадлежат этому классу. Однако частота определенного класса в выборке может отличаться от реального распределения, так как, к примеру, на этапе составления выборки могло быть взято одинаковое количество объектов каждого класса. Для подсчета $P(x_i = a|y = b)$, (a, b – конкретные значения признака и класса) можно взять отношение количества объектов из класса b с признаком $x_i = a$ к общему количеству объектов, принадлежащих этому классу: $P(x_i = a|y = b) = \frac{\sum obj(x_i=a, y=b)}{\sum obj(x, y=b)}$ (за $obj(x_i = a, y = b)$ примем объекты обучающей выборки, которые принадлежат классу b и имеют признак x_i равным a; за $obj(x, y = b)$ примем объекты, принадлежащие классу b).

Рассмотрим пример работы наивного байесовского классификатор. Пусть даны признаки в виде информации о погоде и решение (класс), стоит ли в такую погоду играть в гольф:

Погода, x_1	Ветер, x_2	Гольф, y
Дождливо	Нет	Нет
Дождливо	Есть	Нет
Солнечно	Нет	Да
Солнечно	Нет	Да
Солнечно	Есть	Нет

Дождливо	Нет	Нет
Дождливо	Нет	Да
Солнечно	Нет	Да
Дождливо	Есть	Да

Для каждого из признаков можно составить таблицу частот и правдоподобия:

Погода, x_1 \ Гольф, y	Нет	Да	
Солнечно	1	3	$P(x_1) = 4/9$
Дождливо	3	2	$P(x_1) = 5/9$
Всего	*4*	*5*	*9*
	$P(y) = 4/9$	$P(y) = 5/9$	

Ветер, x_2 \ Гольф, y	Нет	Да	
Есть	2	1	$P(x_2) = 3/9$
Нет	2	4	$P(x_2) = 6/9$
Всего	*4*	*5*	*9*
	$P(y) = 4/9$	$P(y) = 5/9$	

Предположим, что теперь необходимо посчитать, стоит ли играть в гольф, если погода солнечная и есть ветер. Для каждого класса по формуле Байеса получаем:

$$P(y = \text{Да} | x_1 = \text{Солнечно}, x_2 = \text{Есть}) =$$

$$= \frac{P(x_1 = \text{Солнечно} | y = \text{Да}) P(x_2 = \text{Есть} | y = \text{Да}) P(y = \text{Да})}{P(x_1 = \text{Солнечно}) P(x_2 = \text{Есть})} =$$

$$= \frac{3/5 * 1/5 * 5/9}{4/9 * 3/9} = 0.4$$

$$P(y = \text{Нет} | x_1 = \text{Солнечно}, x_2 = \text{Есть}) =$$

$$= \frac{P(x_1 = \text{Солнечно} | y = \text{Нет}) P(x_2 = \text{Есть} | y = \text{Нет}) P(y = \text{Нет})}{P(x_1 = \text{Солнечно}) P(x_2 = \text{Есть})} =$$

$$= \frac{1/4 * 2/4 * 4/9}{4/9 * 3/9} = 0.375$$

Получаем, что $P(y = \text{Да}|x) > P(y = \text{Нет}|x)$, поэтому ответом будет класс «Да».

На этапе классификации может возникнуть так называемая проблема нулевой частоты. Это происходит, когда на этапе тестирования какой-нибудь категориальный признак принимает значение, которого не было в обучающей выборке. Например, для приведенной выше задачи в качестве признака x_1 может попасться значение Облачно. В таком случае все условные вероятности будут равны нулю, что даст ноль при их перемножении, а значит и все классы будут иметь нулевую вероятность. Для решения этой проблемы применяют сглаживание Лапласа. Его идея заключается в том, что каждое слово считается встреченным на один раз больше, то есть к частотам слов прибавляется единица:

$$P(x_i = a|y = b) = \frac{\sum obj(x_i = a, y = b) + 1}{\sum[obj(y = b) + 1]} \quad (8)$$

В итоге условные вероятности для признаков, которые ранее не встречались, никогда не будут нулевыми.

В случае, когда объект выражается большим количеством признаков, приходится перемножать большое количество чисел, которые могут быть близки к нулю. Это может привести к исчезновению порядка, в результате чего все вероятности окажутся нулевыми. Чтобы этого избежать, можно использовать свойство логарифма произведения: $\log(a * b) = \log(a) + \log(b)$. Логарифм является монотонной функцией, поэтому переход к нему не изменит максимум функции, а значит, и искомый класс y^*. В результате все полученные слагаемые по модулю будут больше нуля, а значит их сумма скорее всего не будет давать настолько маленькое число, чтобы можно было столкнуться с исчезновением порядка.

После перехода к логарифму получается следующая формула:

$$y^* = \arg\max_{y \in Y} \left[\log P(y) + \sum_{i=1}^{n} \log P(x_i|y) \right] \quad (9)$$

Бывает несколько типов моделей наивных байесовских классификаторов, которые различаются предположениями относительно условных вероятностей $P(x|y)$ – распределений признаков. Один из них – мультиномиальный. Он используется для случаев, когда признаки принимают дискретные значения, и считается, что каждый признак имеет мультиномиальное распределение. Чаще всего используется для классификации документов, в этом случае подсчитывается частота появления слова в документе.

Следующий тип использует распределение Бернулли. Он похож на мультиномиальную модель, но признаки могут принимать только значения 0 или 1. Для классификации документов это может означать проверку факта, появляется слово в тексте или нет. Мотивация такого подхода в том, что наличие или отсутствие слова в тексте важнее частоты его появления, поэтому количество появлений слова можно не учитывать.

Еще одна модель основывается на распределении Гаусса. Она используется в случае, когда признаки принимают непрерывные значения, и предполагается, что они имеют нормальное распределение. На основе этого условная вероятность считается по формуле:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right), \quad (10)$$

где μ_y – математическое ожидание признака x_i в классе y, σ_y^2 – дисперсия.

Если непрерывный признак не имеет нормального распределения, можно попробовать различными методами и преобразованиями привести его к такому. Другим вариантом является разбиение значений по интервалам, в которых содержится примерно одинаковое количество различных значений признака, например, на квартили или децили. Каждое значение признака помечается номером соответствующего интервала, и признак считается

категориальным. Во время классификации признаки также относят к определенному интервалу, и исходя из этого рассчитывают необходимые вероятности. Также для улучшения работы наивного байесовского классификатора можно убрать зависимые признаки, чтобы они не перевешивали другие признаки по значимости. В задаче классификации документов (текстов) можно убрать так называемые стоп-слова, которые могут встречаться в любом тексте и не имеют отношения к каким-то определенным классам.

1.6 Дерево решений

Еще одним методом решения задач классификации является дерево решений. Этот метод приближен к той схеме, по которой человек оценивает, какое решение нужно принять. Уточняя различную информацию, получается постепенно прийти к какому-то ответу. Этот метод также можно сравнить с инструкциями, которые описывают, какие действия предпринимать в различных ситуациях.

Самым простым видом решающего дерева является бинарное дерево. Все его вершины делятся на два типа. Первый – внутренние вершины, из которых идут рёбра к двум дочерним вершинам. Второй тип – листья – конечные вершины, у которых нет потомков. С каждой внутренней вершиной связано решающее правило, которое разделяет случаи. Это функции (или предикаты), которые определяют, в какое поддерево необходимо отправить рассматриваемый объект. Решающие правила связаны с признаками объектов, и если признаки являются бинарными, то разделение на дочерние вершины происходит согласно значениям признака. Например, если признак принимает значение 1, то объект определяют в правое поддерево, если значение 0, то в левое. Если же признаки могут принимать несколько различных значений, то разделение происходит по какому-нибудь из порогов, отделяющие эти значения. Например, если множество допустимых значений какого-нибудь признака f_i равно целым числам от двух до семи, то есть смысл рассматривать

решающие правила «$f_i \geq 2$», «$f_i \geq 3$» и т.д. Для выбора лучшего порога используются различные методы, которые оценивают эффективность разделения при выборе порога.

С каждым листом связана метка класса, к которому пришли в результате разбиений с помощью решающих правил, либо плотность вероятности, если класс определяется неоднозначно.

На первом этапе метода по обучающей выборке строится само дерево, выбирая наиболее полезные признаки в качестве решающих правил. Построение продолжается до тех пор, пока в листе не останутся элементы одного класса. Допускаются также другие различные критерии остановки, например, минимальное допустимое количество элементов в листе, глубина дерева и т.д. При завершении построения каждой ветки в листе ставится вектор вероятности либо метка того класса, который имеют объекты, оказавшиеся в этом листе.

На этапе классификации объект пропускается по построенному решающему дереву, в каждой внутренней вершине переходя в правое или левое поддерево в зависимости от значения проверяемого признака. В итоге объект доходит до листа, и ему присваивается класс, который соответствует этому листу, либо класс с максимальной вероятностью, если листу соответствует плотность вероятности.

1.7 Алгоритм C4.5

Рассмотрим алгоритм построения решающего дерева под названием C4.5. Общую схему можно описать следующим псевдокодом:

1. Процедура **C4.5** ($U \subseteq X^l$):
2. **Если** все объекты из U принадлежат одному классу $c \in Y$ **то**
3. **Вернуть** новый лист $v, c_v \coloneqq c$;
4. Найти предикат с максимальной информативностью:
$$\beta \coloneqq \arg\max_{\beta \in B} I(\beta, U);$$
5. Разбить выборку на две части $U = U_0 \sqcup U_1$ по предикату β:
$$U_0 \coloneqq \{x \in U : \beta(x) = 0\};$$
$$U_1 \coloneqq \{x \in U : \beta(x) = 1\};$$

6. **Если** $U_0 = \emptyset$ или $U_1 = \emptyset$ **то**
7. **Вернуть** новый лист $v, c_v \coloneqq$ Мажоритарный класс(U);
8. Создать новую внутреннюю вершину v: $\beta_v \coloneqq \beta$;
   Построить левое поддерево: $L_v \coloneqq \boldsymbol{C4.5}(U_0)$;
   Построить правое поддерево: $R_v \coloneqq \boldsymbol{C4.5}(U_1)$;
9. **Вернуть** v;

Алгоритм принимает на вход обучающую выборку и выдаёт дерево, построенное на ней. Очевидно, если вся выборка состоит из элементов одного класса, то дерево будет состоять из листа v с меткой этого класса c_v. В противном случае необходимо найти признак β из множества признаков B с максимальной информативностью, то есть такой, который лучше всего отделяет некоторую группу классов либо один класс от всех остальных. Такой признак находится с помощью определенного критерия информативности I. В результате выборка разбивается на две непересекающиеся части U_1 и U_0: в одной из них находятся примеры, которые удовлетворяют признаку, в другой – все остальные. Они формируют правую и левую ветви дерева. Если какая-то из частей оказалась пустой, значит, не нашлось никакого эффективного разбиения, и нужно образовать листовую вершину с самым частым классом из этих частей. В основном случае после разбиения создается внутренняя вершина с найденным признаком β, а для построения правой и левой ветвей поддерева рекурсивно запускается этот же алгоритм **C4.5** на полученных после разбиения частях U_0 и U_1. После этого возвращается полученная вершина v.

После построения дерева таким способом можно столкнуться с проблемой переобучения, так как дерево строится до тех пор, пока существуют эффективные разбиения и пока не останутся элементы из одного класса. Для борьбы с этим к полученному дереву применяют процедуру усечения (pruning), которая ищет неэффективные ветвления и заменяет их на листья.

Рассмотрим алгоритм подробнее. Пусть имеется обучающая выборка X^l, каждый элемент которой описывается множеством признаков $f_1, \dots, f_n$. В

общем случае имеется k классов: $Y = \{C_1, C_2, \dots, C_k\}$. При выборе решающего правила есть n различных вариантов, и если признак принимает m различных значений, то есть еще m различных порогов. Для выбора лучшего варианта просматриваются все возможные разбиения, полученные при выборе признака и порога, и выбирается наилучший согласно критерию информативности.

Если дано множество X, то выражение

$$Entropy(X) = -\sum_{j=1}^{k} \frac{freq(C_j, X)}{|X|} * \log_2 \frac{freq(C_j, X)}{|X|} \quad (11)$$

называется *энтропией* и означает меру хаотичности в множестве X, то есть меру того, насколько разные элементы находятся в этом множестве. $freq(C_j, X)$ означает количество элементов из множества X, которые принадлежат классу C_J, $|X|$ - мощность (размер) множества X. Если в X присутствуют элементы только одного класса, то энтропия будет равна нулю. Если же там содержатся объекты разных классов, то энтропия будет высокой. Эту меру можно рассматривать как определенность, к какому классу можно отнести элементы множества. Если энтропия равна нулю, то класс определяется однозначно. Чем она больше, тем сложнее определить класс. Самый неоднозначный случай можно наблюдать, когда в множестве содержится одинаковое количество элементов разных классов. Например, имеется по 5 элементов из классов C_1, C_2 и C_3. В этом случае энтропия равна $Entropy(X) = -\frac{5}{15}\log_2\frac{5}{15} - \frac{5}{15}\log_2\frac{5}{15} - \frac{5}{15}\log_2\frac{5}{15} = -\frac{1}{3} * (-1.58) * 3 = 1.58.$

На основе энтропии можно оценить *прирост информации*, полученный при выборе какого-либо признака:

$$Gain(X, f) = Entropy(X) - \sum_{i=1}^{m} \frac{|X_{f^i}|}{|X|} Entropy(X_{f^i}), \quad (12)$$

где f – проверяемый признак, f^i – i-е значение признака f, X_{f^i} – элементы из множества X, у которых признак f принимает i-е его значение. Эта величина

показывает, как сильно разбиение по выбранному признаку уменьшает энтропию множества, то есть насколько более разделенным оно становится.

В некоторых методах в качестве решающего правила выбирается признак, который приводит к максимальному приросту информации. Однако в этом случае может быть выбран признак, который принимает индивидуальное значение в каждом примере. Это может привести к тому, что таким признаком будет отделятся только один элемент из обучающего множества. Поэтому вводится также величина, которая характеризует информативность разбиения по признаку:

$$SplitInfo(X, f) = -\sum_{i=1}^{m} \frac{|X_{f^i}|}{|X|} \log_2 \frac{|X_{f^i}|}{|X|} \quad (13)$$

Алгоритм C4.5 в качестве критерия информативности использует комбинацию этих величин:

$$GainRatio(X, f) = \frac{Gain(X, f)}{SplitInfo(X, f)} \quad (14)$$

В качестве решающего правила выбирается тот признак, который приводит к максимальному значению $GainRatio$.

Приведем пример работы этого алгоритма. Допустим, имеются следующие данные:

Погода, f_1	Ветер, f_2	Гольф, y
Дождливо	Нет	Нет
Дождливо	Есть	Нет
Солнечно	Нет	Да
Солнечно	Нет	Да
Солнечно	Есть	Нет
Дождливо	Нет	Нет
Дождливо	Нет	Да
Солнечно	Нет	Да
Дождливо	Есть	Да

В первую очередь нужно построить корневую вершину, для этого необходимо выбрать решающее правило. Признаки принимают следующие значения: $f_1^1 = \text{Дождливо}, f_1^2 = \text{Солнечно}, f_2^1 = \text{Нет}, f_2^2 = \text{Есть}$. Посчитаем все необходимые величины:

$$Entropy(X) = -\frac{5}{9} * \log_2 \frac{5}{9} - \frac{4}{9} * \log_2 \frac{4}{9} = 1.93$$

$$Entropy\left(X_{f_1^1}\right) = -\frac{3}{5} * \log_2 \frac{3}{5} - \frac{2}{5} * \log_2 \frac{2}{5} = 0.97$$

$$Entropy\left(X_{f_1^2}\right) = -\frac{3}{4} * \log_2 \frac{3}{4} - \frac{1}{4} * \log_2 \frac{1}{4} = 0.81$$

$$Gain(X, f_1) = Entropy(X) - \frac{5}{9} Entropy\left(X_{f_1^1}\right) - \frac{4}{9} Entropy\left(X_{f_1^2}\right)$$

$$= 1.93 - \frac{5}{9} * 0.97 - \frac{4}{9} * 0.81 = 1.03$$

$$SplitInfo(X, f_1) = -\frac{5}{9} \log_2 \frac{5}{9} - \frac{4}{9} \log_2 \frac{4}{9} = 1.93$$

$$GainRatio(X, f_1) = \frac{Gain(X, f_1)}{SplitInfo(X, f_1)} = \frac{1.03}{1.93} = 0.53$$

$$Entropy\left(X_{f_2^1}\right) = -\frac{4}{6} * \log_2 \frac{4}{6} - \frac{2}{6} * \log_2 \frac{2}{6} = 0.92$$

$$Entropy\left(X_{f_2^2}\right) = -\frac{2}{3} * \log_2 \frac{2}{3} - \frac{1}{3} * \log_2 \frac{1}{3} = 0.92$$

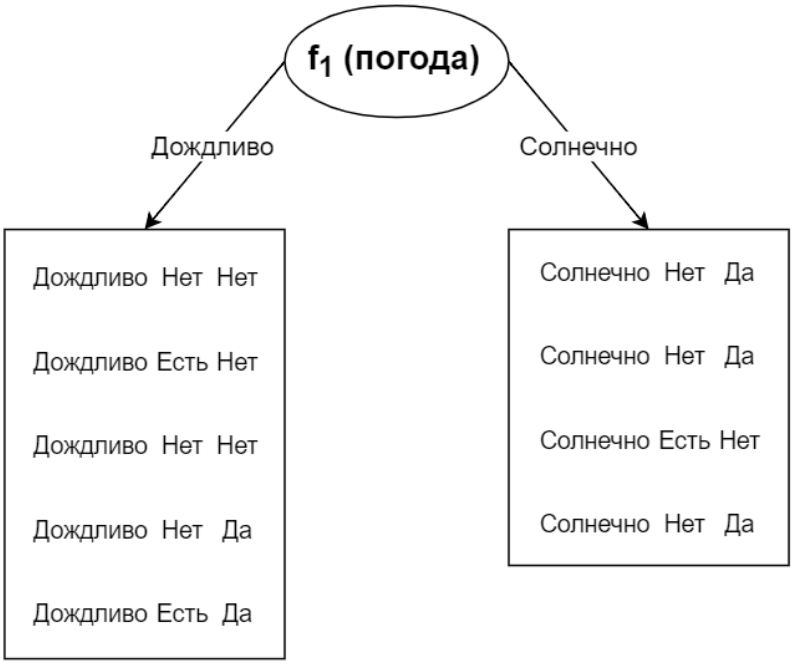
$$Gain(X, f_2) = Entropy(X) - \frac{6}{9} Entropy\left(X_{f_2^1}\right) - \frac{3}{9} Entropy\left(X_{f_2^2}\right)$$

$$= 1.93 - \frac{6}{9} * 0.92 - \frac{3}{9} * 0.92 = 0.24$$

$$SplitInfo(X, f_2) = -\frac{6}{9} \log_2 \frac{6}{9} - \frac{3}{9} \log_2 \frac{3}{9} = 0.92$$

$$GainRatio(X, f_2) = \frac{Gain(X, f_2)}{SplitInfo(X, f_2)} = \frac{0.24}{0.92} = 0.26$$

Максимально значение $GainRatio$ дал признак f_1, поэтому его берем в качестве решающего правила для корневой вершины. Получается следующее дерево:



Теперь нужно проделать такую же операцию для каждого из поддеревьев. В качестве множества *X* будут выступать те элементы, которые попали в рассматриваемую ветку.

Построение дерева данным методом может привести к переобучению, когда разбиения делаются до тех пор, пока в листе не окажется единственный элемент. Для борьбы с этим используется *усечение* (pruning) дерева. Оно бывает двух видов – до завершения построения дерева и после. В первом случае дерево строится до тех пор, пока получаются уместные разбиения. Этот способ работает быстрее, но его сложнее реализовать. Второй метод заключается в том, что после построения дерева убираются некоторые неэффективные ветки. Это можно осуществить, заменяя некоторые поддеревья листьями. Алгоритм C4.5 использует именно такой подход.

Метод усечения в C4.5 называется пессимистичным усечением (pessimistic pruning). Он заключается в том, что по доверительным интервалам оценивается реальная вероятность ошибки классификации в поддереве, которая может быть получена на тестовом множестве. Если замена поддерева листом снижает вероятность ошибки, то поддерево усекается. Этот метод усечения хорош тем, что нет необходимости выделять контрольное множество для оценки ошибок, алгоритм работает с уже построенным деревом.

Сначала необходимо найти доверительные интервалы для вероятности ошибки. Доверительным интервалом называется такой интервал, в который с заранее заданной вероятностью (называющаяся доверительной вероятностью) будет попадать неизвестный параметр распределения. В данном случае в качестве такого параметра рассматривается вероятность ошибки классификации в поддереве на произвольных элементах (не из обучающей выборки). Принимается, что количество ошибок имеет биномиальное распределение. Так как к биномиальному распределению применима центральная предельная теорема, доверительный интервал можно найти через формулу вероятности принадлежности значения нормальной случайной величины интервалу:

$$P\left(-Z_\alpha < \frac{n\hat{p} - np}{\sqrt{n\hat{p}(1-\hat{p})}} < Z_\alpha\right) = 1 - \alpha, \tag{15}$$

где n – общее количество классифицированных элементов, $\hat{p} = \frac{count(errors)}{n}$ – наблюдаемая вероятность ошибки, p – искомая вероятность ошибки, α – доверительная вероятность, Z_α – табличное значение, являющееся аргументом функции Лапласа, при котором значение функции равно $\frac{\alpha}{2}$ (т.е. $\Phi(Z_\alpha) = \frac{\alpha}{2}$). Тогда:

$$P\left(\hat{p} - Z_\alpha\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} < p < \hat{p} + Z_\alpha\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}\right) = 1 - \alpha, \tag{16}$$

Отсюда получаем границы интервала для вероятности ошибки: $\hat{p} \pm Z_\alpha\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$. Из них для оценки вероятности берется большая граница, отсюда и название метода – пессимистичное усечение.

В итоге для оценивания ошибки в листе используется формула:

$$\varepsilon(L,S) = \hat{\varepsilon}(L,S_L) + Z_\alpha\sqrt{\frac{\hat{\varepsilon}(L,S_L)\left(1-\hat{\varepsilon}(L,S_L)\right)}{|S_L|}}, \tag{17}$$

где L – рассматриваемый лист, S_L – множество классифицируемых листом объектов, $\hat{\varepsilon}(L, S_L)$ – доля ошибочно классифицированных объектов из S_L, $|S_L|$ - количество элементов множества S. В качестве ошибки поддерева, содержащего листья, берется взвешенная сумма ошибок по всем листья, т.е.:

$$\varepsilon(T, X) = \sum_{L \in Leafs(T)} \frac{|S_L|}{|X|} \hat{\varepsilon}(L, S_L), \bigcup S_L = X, \quad (18)$$

$Leafs(T)$ – листья из поддерева T. Если эта сумма меньше доли ошибочных элементов по листьям поддерева, то поддерево заменяется листом с самым частым классом среди листьев.

1.8 Random Forest

При использовании деревьев решений часто можно столкнуться с переобучением, особенно когда обучающее множество мало и состоит из сильно различающихся объектов. Если же ввести жесткие ограничения на построение дерева, может оказаться так, что даже на обучающем множестве дерево классифицирует объекты с большим количеством ошибок. Для улучшения качества работы решающих деревьев был придумал алгоритм Random Forest – «случайный лес», которой представляет из себя ансамбль решающих деревьев. Учитывая предсказания многих деревьев, получается добиться более точного результата.

Для построения случайного леса создается множество деревьев решений, каждое из которых построено на подмножестве обучающего множества и с использованием подмножества признаков. Размер обучающей выборки для конкретного дерева равен размеру оригинального обучающего множество, но объекты выборки для дерева берутся из оригинального с повторением. Этот подход называется бутстрэпом (bootstrap), который заключается в генерации различных псевдовыборок по одной имеющейся выборке. Он используется для построения эмпирического распределения и последующего анализа различных статистик. Хотя каждое дерево будет лучше

классифицировать свое обучающее подмножество, в целом лес деревьев будет работать точнее, чем обычное дерево решений.

Другим важным концептом случайного леса является рассматривание только подмножества всех признаков для разделения вершины. Чаще всего для каждой вершины признаки рассматриваются в количестве $\sqrt{n}$, где n – количество всех признаков.

На этапе классификации тестовое множество пропускается через все деревья леса, а итоговый класс выбирается путем голосования – берется самый частый класс среди ответов деревьев. Если деревьев в лесу достаточно много, то переобучения не произойдет.

1.9 Оценка качества классификатора

После выполнения классификации необходим способ оценивания, насколько хорошо алгоритм справился с задачей. Для этого необходимо сравнить получившееся решение на тестовой выборке с известными результатами. При этом, чтобы оценить качество классификатора, нужно выбрать подходящую метрику.

В самом простом случае можно считать отношение правильно классифицированных объектов к общему их количеству. Недостатками этого метода является плохая работа в случае несбалансированных классов, когда к одному классу принадлежит в несколько раз больше объектов, чем к другому. Полученная корректность может быть высокой за счет правильной классификации внутри большого класса, так как в нем содержится больше данных для анализа. Внутри же маленького класса алгоритм может работать неправильно, но из-за небольшого количества таких примеров это почти не испортит оценку точности классификатора.

В качестве меры также можно использовать полноту (recall) и точность (precision). Их можно использовать отдельно или как основу для других метрик. При сравнении результатов работы алгоритма с правильными ответами, можно выделить четыре варианта их соотношений:

- TP (true positive) – истинно-положительный результат;
- TN (true negative) – истинно-отрицательный результат;
- FP (false positive) – ложно-положительный результат;
- FN (false negative) – ложно-отрицательный результат.

Это можно представить в виде таблицы следующим образом:

		Правильный ответ	
		Положительный	Отрицательный
Ответ алгоритма	Положительный	TP	FP
	Отрицательный	FN	TN

То есть истинно-положительный результат получается, когда алгоритм отнес объект к правильному классу, ложно-отрицательный – когда алгоритм не отнес объект к правильному классу и т.д.

Полнота и точность высчитываются по следующим формулам:

$$Recall = \frac{TP}{TP + FN}; \quad (19)$$

$$Precision = \frac{TP}{TP + FP}. \quad (20)$$

Таким образом, полнота означает долю тех объектов, которые классификатор отнес к классу, относительно реального числа объектов этого класса. Она характеризует способность алгоритма выявлять положительные ответы, то есть какая доля правильных ответов была найдена. Точность внутри класса означает долю правильно классифицированных объектов относительно объектов, отнесенных алгоритмом к этому классу. Она характеризует, сколько положительных ответов классификатора являются правильными.

Так как невозможно одновременно достичь максимальные полноту и точность, вводится мера, которая объединяет эту информацию, выражает баланс. Она называется F1-мерой и высчитывается следующим образом:

$$F1 = \frac{2 * Recall * Precison}{Recall + Precision} \quad (21)$$

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Описание задачи

В качестве исходных данных дан файл в формате tsv с признаковым описанием запросов к поисковой системе. Этот формат представляет собой таблицу, в которой каждое поле отделяется символом табуляции. Файл с данными содержит текст запроса “query”, поле с вещественными признаками “factors”, поле с документами (ссылками на сайты) из выдачи “urls”, факт нажатия на большой спортивный ответ на странице выдачи “target” и массив с фактами перехода по каждой из ссылок “clicks”. Каждая строка файла является примером. Последний массив необходим для добавления дополнительных признаков и выявления того, как они влияют на качество классификации. Необходимо определить, является ли запрос спортивным (в частности, связан ли он с футбольными матчами) и нужно ли по нему выдавать большой спортивный ответ, расписание матчей, результаты чемпионатов или прочую информацию.

Для выполнения задачи применяются наивный байесовский классификатор, дерево решений с ограниченной глубиной и случайный лес. Необходимо подобрать оптимальную глубину дерева, для случайного леса – оптимальное количество деревьев и прочие параметры метода. Качество классификатора оценивается F1-мерой.

2.2 Описание программы

Для реализации классификаторов был использован язык программирования Python версии 3.7.0. Программа делится на несколько модулей: «naive_bayes», «c45» и «random_forest».

В первом модуле реализован алгоритм наивного байесовского классификатора. Программа считывает данные из файла формата tsv, сохраняет их и потом делит на обучающее и тестовое подмножества. Долю примеров в обучающей выборке можно задавать, проставив необходимое значение в переменную `split_ratio`. После этого подмножества

модифицируются: вещественные значения признаков разбиваются на заданное количество интервалов, после чего значение конкретного признака заменяется на соответствующий ему номер интервала. Количество таких интервалов задается с помощью переменной `NUM_OF_BUCKETS`. Такое разбиение необходимо для того, чтобы наивный байесовский классификатор смог посчитать частоту, с которой встречается данное значение признака. Но так как значения вещественные и могут встречаться без повторений, необходимо их дискретизировать. Далее на модифицированных подмножествах запускается алгоритм наивного байесовского классификатора. Внутри него считается количество истинно-положительных, ложно-положительных и ложно-отрицательных ответов для дальнейшей оценки качества классификатора. Все необходимые частоты для классов и значений признаков высчитываются заранее для ускорения работы программы.

В модуле «c45» реализован метод построения дерева решений C4.5. Она считывает тот же файл и сохраняет данные. Значения признаков разбиваются на интервалы для более быстрой работы алгоритма, их количество задается переменной `NUM_OF_BUCKETS`. Значения признаков заменяются на соответствующие номера интервалов, по модифицированному обучающему подмножеству рекурсивно строится дерево решений ограниченной глубины. Максимальная глубина дерева задается переменной `MAX_DEPTH`. После этого для работы усечения дерева считается количество ошибок на обучающем подмножестве, затем запускается само усечения. По полученному дереву классифицируется каждый элемент тестового множества, считаются необходимые для оценки качества классификатора величины.

В третьем модуле реализован алгоритм построения случайного леса. Для построения дерева используется алгоритм C4.5 без усечения. Каждое дерево ограничено по глубине, максимальная глубина задается переменной `MAX_DEPTH`. Количество деревьев в лесу задается переменной `NUM_OF_TREES`. После построения деревьев каждый пример из тестового множества подается

каждому дереву, после чего в качестве итогового класса принимается самый частый ответ.

Для хранения кода и контроля версий был использован GitHub - крупнейший веб-сервис для хостинга IT-проектов. Для удобства написания кода использовалась среда программирования PyCharm, которая предоставляет средства для анализа кода, графический отладчик и многие другие инструменты.