РЕФЕРАТ

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc9899740)

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc9899741)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_Toc9899742)

[1.1 Графы 4](#_Toc9899743)

[1.1.1 Основные определения 4](#_Toc9899744)

[1.1.2 Простые алгоритмы на графах 6](#_Toc9899745)

[1.2 Классификаторы 10](#_Toc9899746)

[1.2.1 Нормализация входных данных 12](#_Toc9899747)

[1.2.2 Метод k-ближайших соседей 13](#_Toc9899748)

[1.2.2 Градиентный бустинг 16](#_Toc9899749)

[1.2.3 Дерево решений 16](#_Toc9899750)

[1.2.4 Случайный лес 17](#_Toc9899751)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 17](#_Toc9899752)

[2.1 Описание данных 17](#_Toc9899753)

[2.2 Предварительный анализ 17](#_Toc9899754)

[2.3 Исследование признаков (feature engineering) 17](#_Toc9899755)

[2.4 Эксперименты на моделях 17](#_Toc9899756)

[2.5 Результаты 17](#_Toc9899757)

[Заключение 17](#_Toc9899758)

# ВВЕДЕНИЕ

Последние десятилетия в результате стремительного развития вычислительной техники

Каждый день в мире совершаются миллионы финансовых транзакций. Хоть деньги и частично утратили свой физический, ощущаемый облик, мошенники, желающие их похитить, остались.

В связи с огромным количеством совершаемых транзакций, человеку будет практически нереально анализировать весь этот поток. На помощь ему приходят вычислительные машины, которые в последние годы стремительно развиваются и способны выполнять невероятные по объему и скорости вычисления.

В связи с этим набирает закономерную популярность такая наука как машинное обучение. С помощью предлагаемых ею методов возможно предсказыватьрезультат тех или иных задач.

В данной работе я постараюсь обучить модель, способную отличать мошеннические транзакции от не мошеннических.

По соображениям конфиденциальности личных данных, в открытом доступе не имеется наборов реальных финансовых транзакций. Поэтому был выбран набор, созданный синтетически на основе реальных.

# ПОСТАНОВКА ЗАДАЧИ

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Графы

### 1.1.1 Основные определения

В данной работе графы будет использоваться в двух местах: для визуализации и анализа, а также при построении деревьев решений.

**Определение 1.1.** Графом называется упорядоченная пара , где - непустое множество вершин (узлов) графа , а  - множество ребер графа .

Далее будут обозначены некоторые вспомогательные определения:

* количество вершин графа называется его порядком и обозначается  В данной работе будут рассматриваться только конечные графы, т.е. множества вершин и ребер принимают конечное число значений;
* количество рёбер графа называется его размером и обозначается 
* ребро  соединяет вершины, называющиеся концевыми (концами)  и ;
* соседними вершинами называются такие концы, которые соединены одним и тем же ребром;
* смежными называются ребра, имеющие общую концевую вершину;
* вершина называется изолированной, если она не является конечной ни для одного из ребер;
* висячей вершиной или листом называется вершина, которая является концом ровно одного ребра.

Если не говорится об обратном, то граф считается неориентированным (пример приведен на рис 1.1), это означает, что каждое его ребро имеет два конца, порядок которых не имеет значения.



Рис. .1. Неориентированный граф

Пусть  называется дугой. Тогда вершину  называют её началом, а   — концом. Можно сказать, что дуга  ведёт от вершины  к вершине .

**Определение 1.2.** Граф называется ориентированным (орграфом), если  — упорядоченная пара, где — непустое множество вершин (узлов) графа , а  — множество упорядоченных пар различных вершин, которые называются дугами. Пример такого графа приведен на рис. 1.2.

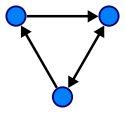


Рис. 1.2. Ориентированнный граф

Вспомогательные определения для орграфов:

* маршрутом в графе называется конечная последовательность вершин, в которой каждая вершина (исключая последнюю) соединяется со следующей в последовательности вершиной ребром;
* путем в графе называют конечную последовательность вершин и дуг, в которой каждый элемент соединен с предыдущем и последующим;
* цепью называется маршрут без повторяющихся ребер;
* циклом называют цепь, в которой начальная и конечная вершины совпадают;
* цикл называют простым, если ребра в нем не повторяются;
* цикл называют элементарным, если он простой и вершины в нем не повторяются.

**Определение 1.3.** Граф  называется подграфом графа , если  и .

**Определение 1.4.** 1) Вершины *a* и *b* графа *G* называются связанными, если в графе существует путь между ними.

2) Граф называется связным, если любые две его вершины связаны.

Существует много несколько разновидностей графов. В данной работе будет использоваться такие структуры как дерево и лес.

**Определение 1.5.** 1) Деревом называется связный граф без циклов.

2) Лесом называется упорядоченное множество деревьев.

**Определение 1.6.** Двоичным (бинарным) деревом называется ориентированное дерево, в котором число исходящих из каждой вершины ребер не превосходит 2.

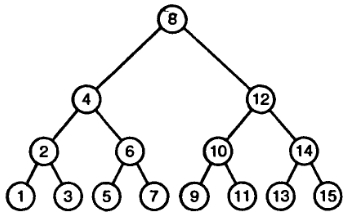


Рис. 1.3. Пример бинарного дерева

### 1.1.2 Простые алгоритмы на графах

В этом разделе будет приведено несколько основных алгоритмов обхода графов. Обходя граф, мы двигаемся по ребрам и проходим все вершины. При этом накапливается довольно много информации, которая полезна для дальнейшей обработки графа.

#### 1.1.2.1 Поиск в ширину

Поиск в ширину (breadth-first search) – один из базисных алгоритмов, составляющий основу многих других.

Пусть задан граф  и фиксирована начальная вершина *s.* Алгоритм поиска в ширину перечисляет все достижимые из *s* вершины, доступные при проходе по ребрам, в порядке возрастания расстояния от *s.* Расстоянием считается длина минимального пути из начальной вершины. Алгоритм применим как к ориентированным графам, так и к неориентированным.

Такое название объясняется тем, что в процессе поиска мы идем вширь, а не вглубь, т.е. сначала просматриваем все соседние вершины, затем соседей соседей и так далее.

**Алгоритм 1.1.**

1. Начинаем обход из фиксированной начальной вершины *s.* Пометить ее как посещенную. Добавить вершину в изначально пустую очередь.
2. Извлечь из начала очереди вершину *u.*
   1. Если она является уже посещенной повторить шаг 1.
   2. Добавить все соединенные с *u* не посещённые вершины в очередь
3. Если очередь пустая, закончить алгоритм. В противном случае перейти к шагу 1.

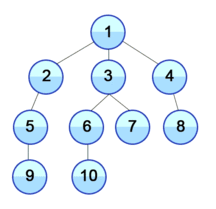


Рис. 1.4. Порядок обхода графа при поиске в ширину

На рис. 1.4. приведен пример работы алгоритма поиска в ширину. Цифры являются порядковым номером посещения вершины в процессе работы алгоритма.

#### 1.1.2.1 Поиск в глубину

Поиск в глубину (depth-first search) наряду с вышеописанным алгоритмом также является одним из базисных методов обхода графа.

Он имеет следующую стратегию: как и в поиске в ширину, фиксируем начальную вершину *s* и начинаем от нееидти «вглубь», пока имеется такая возможность, т.е. пока существуют не пройдённые ребра, затем возвращаться и искать иной путь, в случае, когда таких ребер не осталось. Алгоритм работает, пока не обнаружит все вершины, достижимые из исходной.

**Алгоритм 1.2.**

1. Начинаем обход из фиксированной начальной вершины *s.*
2. Пометить текущую вершину как посещенную.
3. Если есть соседние не посещенные вершины, перейти к одной из них и выполнить для нее алгоритм начиная с шага 1. Если все соседние вершины посещены, либо отсутствуют вовсе - закончить алгоритм.

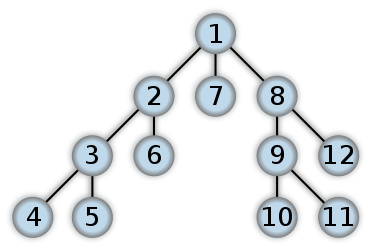


Рис. 1.5. Порядок обхода графа при поиске в глубину

На рис. 1.5. представлен пример работы алгоритма поиска в глубину. Аналогично рис. 1.4. цифры соответствуют порядковому номеру вершины при обходе графа алгоритмом 1.2.

## 1.2 Классификаторы

Классификацией называют один из разделов [машинного обучения](http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), посвященный решению следующей задачи. Имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется [обучающей выборкой](http://www.machinelearning.ru/wiki/index.php?title=%D0%92%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B0). Классовая принадлежность остальных объектов неизвестна. Требуется построить [алгоритм](http://www.machinelearning.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC), способный классифицировать произвольный объект из исходного множества.

Классифицировать объект — значит, указать номер (или наименование класса), к которому относится данный объект.

Классификация объекта — номер или наименование класса, выдаваемый [алгоритмом](http://www.machinelearning.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) классификации в результате его применения к данному конкретному объекту.

Существуют несколько типов классов:

* двухклассовый, число классов равно двум;
* многоклассовый, когда число классов достигает многих тысяч (при распознавании иероглифов или слитной речи).
* непересекающиеся классы.
* пересекающиеся классы. Объект может относиться одновременно к нескольким классам.
* нечёткие классы. Требуется определять степень принадлежности объекта каждому из классов, обычно это действительное число от 0 до 1.

В данной работе будет рассматриваться двухклассовый случай.

Введем некоторые базовые определения, которые будут использованы в дальнейшем.

**Определение 1.7.** Признаком (feature) называется результат измерения некоторой характеристики объекта. Можно сказать, что признак есть отображение  где - множество допустимых значений признака.

В зависимости от природы этого множества, признаки делятся на нижеперечисленные типы:

* бинарный признак 
* номинальный признак  — конечное множество;
* порядковый признак:  — конечное упорядоченное множество;
* количественный признак: 

Если все признаки имею одинаковый тип, то исходные данные называются однородными, в ином случае – разнородными.

**Определение 1.8.** Пусть имеется набор признаков . Признаковым описанием объекта  называют вектор , составленный из значений фиксированного набора признаков на данном объекте.

В задачах машинного обучения не делается различия между объектом и его признаковым описанием. Полагается, что 

Постановка задачи классификации выглядит следующим образом. Пусть  — множество описаний (признаков) объектов. Чем является объект, определяется спецификой предметной области. Например, в задачах спортивного менеджмента объектами являются спортсмены.

Пусть  — конечное множество номеров (имён, меток) классов. Существует неизвестная целевая зависимость — отображение , значения которой известны только на объектах конечной [обучающей выборки](http://www.machinelearning.ru/wiki/index.php?title=%D0%92%D1%8B%D0%B1%D0%BE%D1%80%D0%BA%D0%B0) . Требуется построить [алгоритм](http://www.machinelearning.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) , способный классифицировать произвольный объект .

### 1.2.1 Нормализация входных данных

Иногда, чтобы достичь адекватности работы модели, необходимо нормализовать (масштабировать) входные данные. Как будет видно далее работоспособность некоторых моделей зависит расстояния между объектами, в следствие чего возникает необходимость проведения данной процедуры. Проблема заключается в разных измерениях признаков. Например, если рассматривать погоду, то такие ее признаки как температура, давление, скорость ветра и т.д. измеряются в различных физических величинах, а их числовые значения могут на порядки отличаться.

Нормализовать данные можно разными способами, вот два основных.

Минимаксная нормализация:



Данный метод осуществляет переход от абсолютных значений признаков к относительным. Новые переменные будут принимать значения в диапазоне от 0 до 1.

Z-нормализация:

,

где - выборочное среднее,  - выборочное среднеквадратичное отклонение.

Поскольку не все признаки имеют количественные значения, может применяться создание фиктивных переменных (dummy coding). В этом случае категориальные признаки заменяются бинарными. Например, заменяется признак «пол» на два новых: «пол мужской», «пол женский» со значения 0 и 1.

### 1.2.2 Метод k-ближайших соседей

Алгоритмы, основанные на анализе сходства объектов, часто называют метрическими.

Метрическим классификатором (similarity-based classifier) называют алгоритм классификации, основанный на вычислении оценок сходства между объектами. Чтобы формализовать понятие сходства вводится функция расстояния между объектами  в пространстве объектов *X*. Следует заметить, что данная функция может не всегда удовлетворят всем аксиомам метрики. Например, довольно часто не выполняется неравенство треугольника.

Метрические классификаторы опираются на гипотезу компактности. Она, в свою очередь, предполагает, что схожие объекты гораздо чаще лежат в одном классе, чем в разных. Можно сказать, что классы образуют компактно локализованные подмножества в пространстве объектов. То есть граница между классами имеет довольно простую форму.

Метод ближайшего соседа позиционируется, как один из простейших метрических классификаторов. Классифицируемый объект  относится к тому классу, которому принадлежат ближайший к нему объект обучающей выборки .

Метод *k* ближайших соседей ( k-nearest neighbors algorithm, k-NN) для повышения надёжности классификации относит объект к тому классу, которому принадлежит большинство из его соседей, то есть  k ближайших к нему объектов обучающей выборки . В задачах с двумя классами число соседей берут нечётным, чтобы не возникало ситуаций неоднозначности, когда одинаковое число соседей принадлежат разным классам.

Пусть задана обучающая выборка  и на множестве объектов задана функция расстояния . Данная функция должна быть достаточно адекватной моделью сходства объектов, для этого можно провести процедуру нормализации, описанную в разделе 1.2.1. Чем больше значение этой функции, тем менее схожими являются два объекта .

Для произвольного объекта *u* расположим объекты обучающей выборки  в порядке возрастания расстояний до *u:* , где  обозначает объект обучающей выборки, который является *i*-ым соседом объекта *u.* Аналогичное обозначение введём и для ответа на *i*-ом соседе: . Таким образом, произвольный объект *u* порождает свою перенумерацию выборки. В наиболее общем виде [алгоритм](http://www.machinelearning.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) ближайших соседей выглядит так:

,

где - заданная весовая функция, которая оценивает степень важности *i*-го соседа для классификации объекта *u*. Эта функция неотрицательна и не возрастает по *i.*

Различно задавая весовую функцию, получаются различные варианты методы ближайших соседей.

*  - простейший метод ближайшего соседа;
* - метод *k* ближайших соседей;
* - метод *k* экспоненциально взвешенных ближайших соседей, где предполагается  (обычно используется в случае 3-х и более классов).



Рис. 1.6. Метод *k* ближайших соседей

На рис. 1.6. приведена ситуация классификации объекта, в данном случае зеленого круга. Круг должен быть классифицировать как синий квадрат, либо как красный треугольник (класс 1 и класс 2 соответственно). На иллюстрации видно два круга.

Круг, обведенный сплошной линией, показывает поведение алгоритма при *k=3.* В этом случае объект будет классифицирован как 2-ой класс, так как внутри круга находятся 2 треугольник и 1 квадрат, треугольников больше, а значит и решение принимается в сторону этого класса.

В кругу, обведенном штрихом, *k=5.* Тогда ситуация меняется, потому что количество квадратов начало превалировать над треугольниками. Соответственно и объект будет классифицирован как синий квадрат, то есть 1-й класс.

### 1.2.2 Дерево решений

Решающими деревьями называется семейство моделей, которые позволяют восстанавливать нелинейные зависимости произвольной сложности. Они воспроизводят логические схемы, позволяющие получить окончательное решение о классификации объекта с помощью ответов на иерархически организованную систему вопросов. Причем вопрос, задаваемый на последующем иерархическом уровне, зависит от ответа, полученного на предыдущем уровне.

Каждой из вершин дерева за исключением листьев соответствует некоторый вопрос, подразумевающий несколько вариантов ответов, соответствующих выходящим ребрам. В зависимости от выбранного варианта ответа осуществляется переход к вершине следующего уровня. Листьям поставлены в соответствие метки, указывающие на отнесение распознаваемого объекта к одному из классов.

Решающее дерево называется бинарным, если каждая внутренняя или корневая вершина инцидентна только двум выходящим рёбрам.

**Определение 1.9** Рассмотрим бинарное дерево, в котором:

* каждой внутренней вершине ν приписана функция (или предикат) 
* каждый листовой вершине ν приписан прогноз . В случае классификации листу может быть приписан вектор вероятностей.

Теперь рассмотрим алгоритм *a(x),* который стартует из корневой вершиныи вычисляет значение функции . Если оно равно нулю, то алгоритм переходит в левую дочернюю вершину, иначе в правую, вычисляет значение предиката в новой вершине и делает переход или влево, или вправо. Процесс продолжается, пока не будет достигнута листовая вершина; алгоритм возвращает тот класс, который приписан этой вершине. Такой алгоритм называется бинарным решающим деревом.

На практике в большинстве случаев используются одномерные предикаты , которые сравнивают значение одного из признаков с порогом:.

Легко убедиться, что для любой выборки можно построить решающее дерево, не допускающее на ней ни одной ошибки — даже с простыми одномерными предикатами можно сформировать дерево, в каждом листе которого находится ровно по одному объекту выборки. Скорее всего, это дерево будет переобученным и не сможет показать хорошее качество на новых данных. Можно было бы поставить задачу поиска дерева, которое является минимальным (с точки зрения количества листьев) среди всех деревьев, не допускающих ошибок на обучении — в этом случае можно было бы надеяться на наличие у дерева обобщающей способности.

Опишем базовый жадный алгоритм построения бинарного решающего дерева. Начнем со всей обучающей выборки  и найдем наилучшее ее разбиение на две части  и с точки зрения заранее заданного функционала качества Q(X,j,t). Найдя наилучшие значения j и t, создадим корневую вершину дерева, поставив ей в соответствие предикат [xj < t]. Объекты разобьются на две части — одни попадут в левое поддерево, другие в правое. Для каждой из этих подвыборок рекурсивно повторим процедуру, построив дочерние вершины для корневой, и так далее. В каждой вершине мы проверяем, не выполнилось ли некоторое условие останова — и если выполнилось, то прекращаем рекурсию и объявляем эту вершину листом. Когда дерево построено, каждому листу ставится в соответствие ответ. В случае с классификацией это может быть класс, к которому относится больше всего объектов в листе, или вектор вероятностей (скажем, вероятность класса может быть равна доле его объектов в листе). Для регрессии это может быть среднее значение, медиана или другая функция от целевых переменных объектов в листе. Выбор конкретной функции зависит от функционала качества в исходной задаче. Решающие деревья могут обрабатывать пропущенные значения — ситуации, в которых для некоторых объектов неизвестны значения одного или нескольких признаков. Для этого необходимо модифицировать процедуру разбиения выборки в вершине, что можно сделать несколькими способами. После того, как дерево построено, можно провести его стрижку (pruning) — удаление некоторых вершин с целью понижения сложности и повышения обобщающей способности. Существует несколько подходов к стрижке, о которых мы немного упомянем ниже. Таким образом, конкретный метод построения решающего дерева определяется:

* Видом предикатов в вершинах;
* Функционалом качества Q(X,j,t);
* Критерием останова;
* Методом обработки пропущенных значений;
* Методом стрижки.

Также могут иметь место различные расширения, связанные с учетом весов объектов, работой с категориальными признакам и т.д. Ниже мы обсудим варианты каждого из перечисленных пунктов.

### 1.2.3 Случайный лес

### 1.2.4 Градиентный бустинг

# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1 Описание данных

## 2.2 Предварительный анализ

## 2.3 Исследование признаков (feature engineering)

## 2.4 Эксперименты на моделях

## 2.5 Результаты

# Заключение