Лекция 4

ЗАДАЧА КЛАССИФИКАЦИИ. МЕТОД ДЕРЕВЬЕВ РЕШЕНИЙ

Основные положения метода

Метод деревьев решений (***decision tree***) для задачи классификации состоит в том, чтобы осуществлять процесс деления исходных данные на группы, пока не будут получены однородные (или почти однородные) их множества. Совокупность правил, которые дают такое разбиение (***partition***), позволят затем делать прогноз (определять наиболее вероятный номер класса) для новых данных.

Метод деревьев решений применим для решения задач классификации, возникающих в самых разных областях, и считается одним из самых эффективных.

Примеры практических задач классификации:

- *скоринговые модели* кредитования;
- *маркетинговые исследования*, направленные на выявление предпочтений клиента или степени его удовлетворённости;
- *диагностика* (медицинская или техническая).

Основные понятия

Дерево решений *– это модель, представляющая собой совокупность правил для принятия решений*.

Графически её можно представить в виде древовидной структуры, где моменты принятия решений соответствуют так называемым ***узлам*** (***decision nodes***).

В узлах происходит **ветвление** процесса (***branching***), т.е. деление его на так называемые **ветви** (***branches***) в зависимости от сделанного выбора.

Конечные (или, терминальные) узлы называют ***листьями*** (***leafs***, ***leaf nodes***), каждый лист – это конечный результат последовательного принятия решений.

Данные, подлежащие классификации, находятся в так называемом «***корне***» дерева (***root***).

В зависимости от решения, принимаемого в узлах, процесс в конце концов останавливается в одном из листьев, где переменной отклика (искомому номеру класса) присваивается то или иное значение.

Идея метода

Метод деревьев решений реализует принцип так называемого «***рекурсивного деления***» (recursive partitioning).

Эта стратегия также называется «***Разделяй и властвуй***» («***Divide and conquer***»).

В узлах, начиная с корневого, выбирается признак, значение которого используется для разбиения всех данных на 2 класса.

Процесс продолжается до тех пор, пока не выполнится *критерий остановки*:

- Все (или почти все) данные данного узла принадлежат одному и тому же классу;
- Не осталось признаков, по которым можно построить новое разбиение;
- Дерево превысило заранее заданный «лимит роста» (если таковой был заранее установлен).

Пример

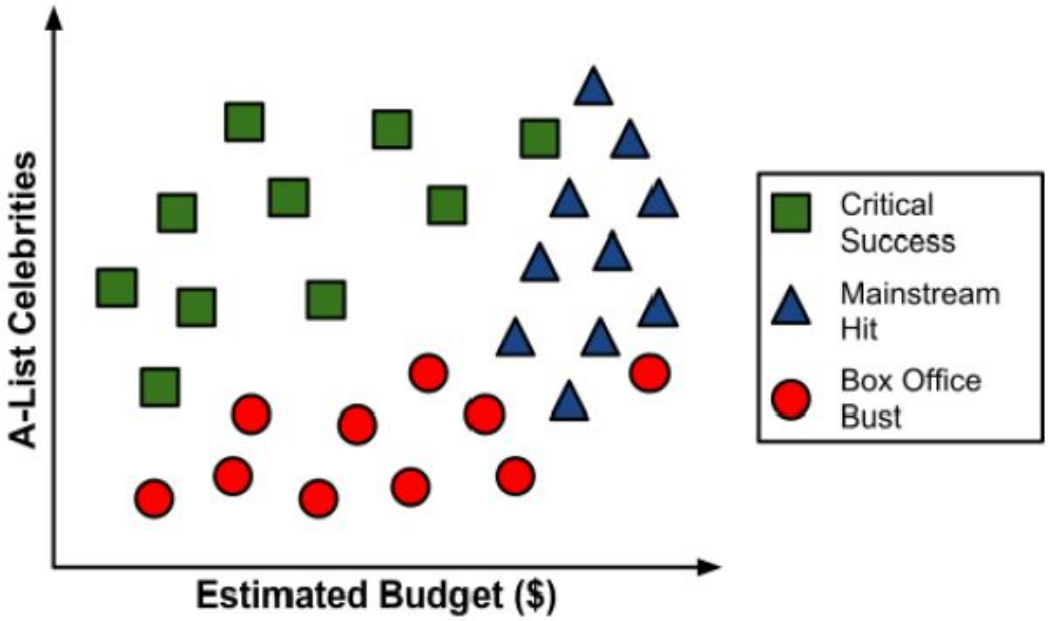
В кинокомпании стол редактора завален сценариями кинофильмов, нужно разложить их по трём ящикам:

- Популярные («mainstream hits»);
- Не популярные у зрителей, но получившие высокую оценку критиков;
- Не имеющие успеха.

Не прочитывая каждый сценарий нужно разработать алгоритм классификации сценариев по трем классам.

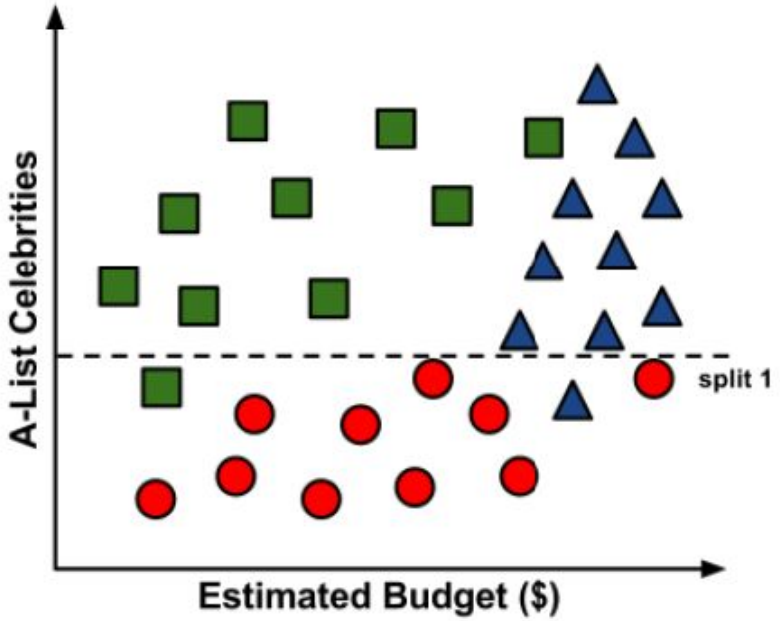
В архиве киностудии собраны данные о киносценариях за последние 10 лет.

После просмотра 30 киносценариев замечено, что фильмы с высоким бюджетом привлекают больше звёзд.



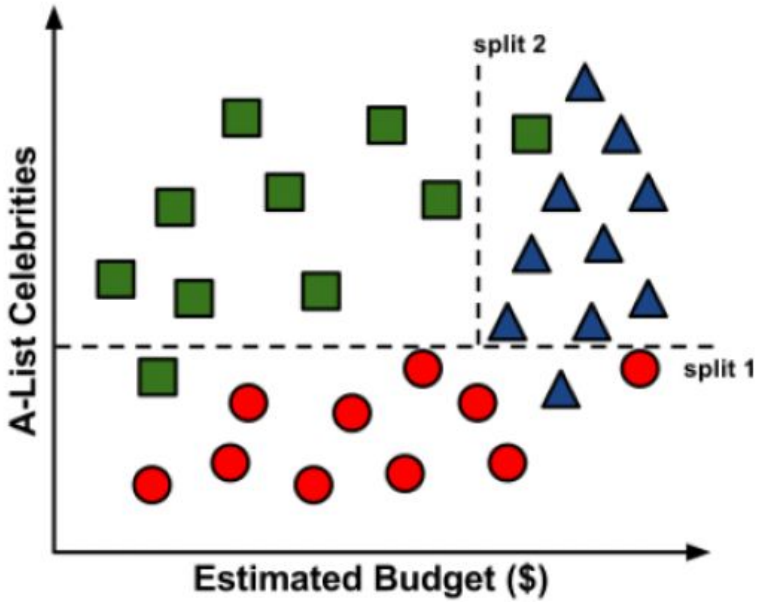
Пример

1) Количество снимавшихся в фильме звёзд как первый из признаков, по которому производится разбиение данных



2) Рассмотрим группу сценариев с большим числом задействованных звёзд.

Разобьём её на 2 подгруппы по признаку «размер бюджета»



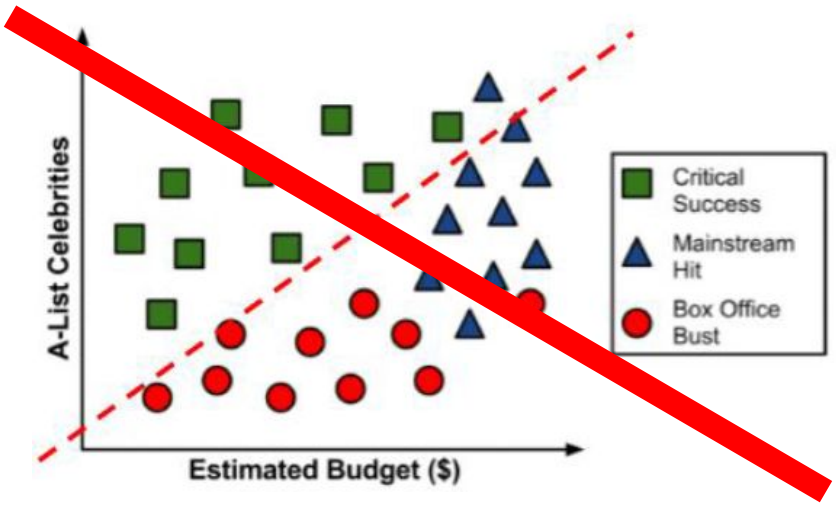
Левая верхняя группа - фильмы, которые были высоко оценены критиками (большое число занятых звёзд, но относительно небольшой бюджет).
Группа в правом верхнем углу – фильмы с большим бюджетом и большим числом звёзд;
Третья группа – состоит преимущественно из фильмов, не имевших успеха (число звёзд, занятых в них, невелико, а их бюджет варьируется от очень малого до очень большого).

Пример

Продолжать процесс разделения данных можно и дальше, пока не получим очень «мелкое» разделение (может оказаться, что каждая группа будет содержать лишь по одному элементу), однако понятно, что смысла в такой классификации нет.

Ограничим ветвление дерева – например, остановим процесс, когда *каждая группа хотя бы на 80% будет состоять из элементов одного и того же класса*.

Заметим, что в данном случае мы говорим лишь о разбиениях данных (точек в Евклидовом пространстве) прямыми (в общем случае – гиперплоскостями), параллельными осям координат.



Метод деревьев решений не позволяет строить такие разбиения, поскольку для разделения данных используются условия вида:

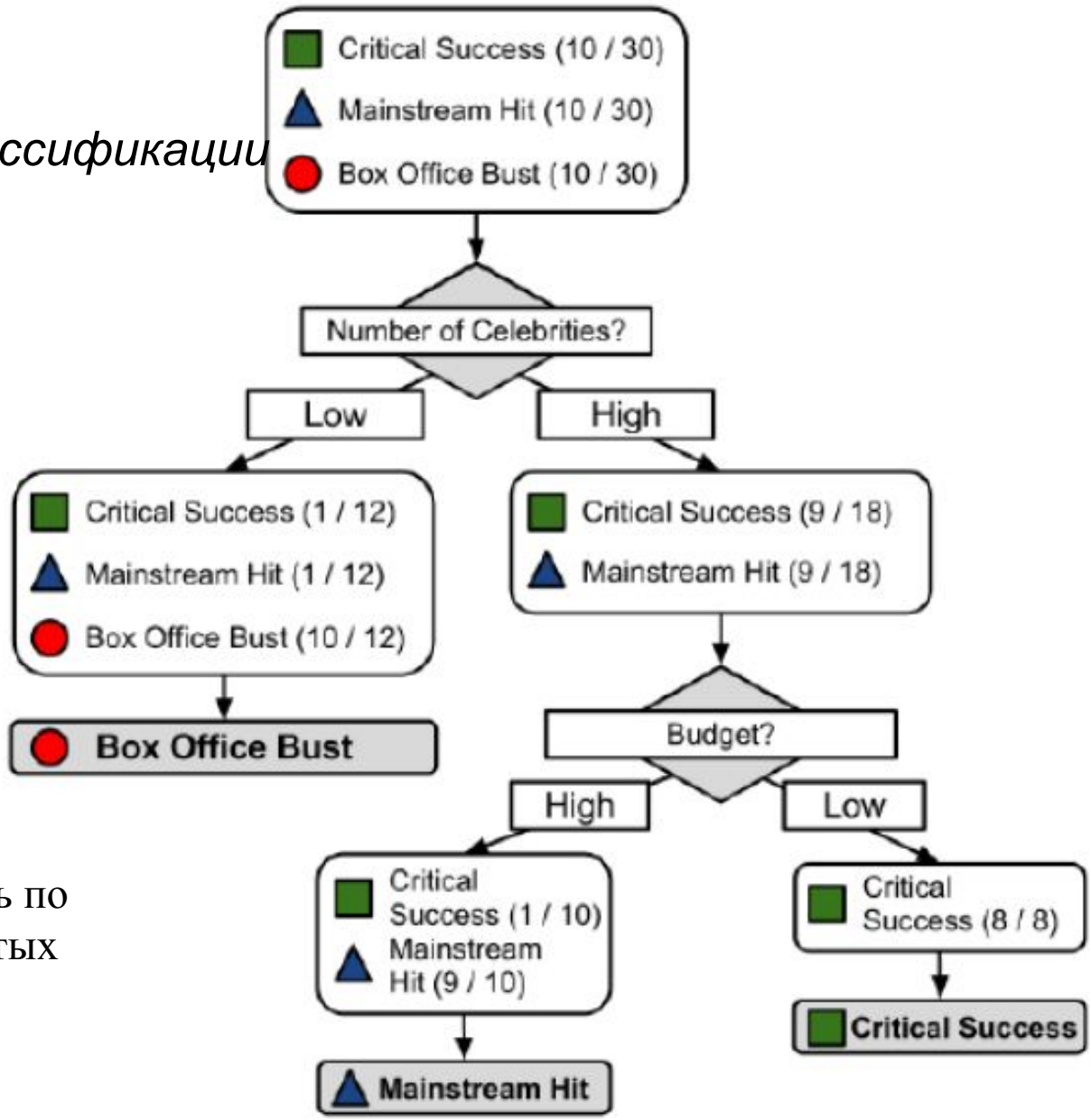
$$\{x < a\}, \{x > a\},$$

где x – значение фактора, a – некоторое фиксированное число.

«***axis-parallel splits***», т.е. «**разбиения, параллельные осям**».

Пример

Дерево решений для классификации киносценариев



Классификация сценариев лишь по 2-м факторам (количество занятых звёзд и бюджет) - дерево получилось небольшим.

Численные алгоритмы метода деревьев решений, допускающие компьютерную реализацию

Существуют различные численные алгоритмы построения деревьев решений: ***CART*, C4.5, CHAID, CN2, NewId, ITrule** и другие.

Алгоритм CART (***Classification and Regression Tree***) очевидно решает задачи классификации и регрессии.

Разработан в 1974-1984 годах четырьмя профессорами статистики - *Leo* Breiman (Berkeley), Jerry Friedman (Stanford), Charles Stone (Berkeley) и Richard Olshen (Stanford).

Атрибуты набора данных могут иметь как дискретное, так и числовое значение.

Алгоритм *CART* предназначен для построения *бинарного дерева решений*.

Другие особенности алгоритма *CART*:

- функция оценки качества разбиения;
- механизм отсечения дерева;
- алгоритм обработки пропущенных значений;
- построение деревьев регрессии.

Алгоритм CART

Функция оценки качества разбиения, которая используется для выбора оптимального *правила,* - индекс *Gini* . Данная оценочная функция основана на идее уменьшения неопределенности в узле:

где T – текущий узел, p_j – вероятность класса j в узле T, n – количество классов.

Допустим, есть узел, и он разбит на два класса. Максимальная неопределенность в узле будет достигнута при разбиении его на два подмножества по 50 примеров, а максимальная определенность - при разбиении на 100 и 0 примеров.

Правила разбиения. В каждом узле разбиение может идти только по одному атрибуту. Если атрибут является числовым, то во *внутреннем узле* формируется *правило* вида *xi* <= *c*.

Значение *c* в большинстве случаев - среднее арифметическое двух соседних упорядоченных значений переменной *xi* обучающего набора данных.

Если же атрибут относится к категориальному типу, то во внутреннем узле формируется правило xi V(xi), где V(xi) - некоторое непустое подмножество множества значений переменной xi в обучающем наборе данных.

Алгоритм CART

Механизм отсечения - *minimal cost-complexity tree pruning*, алгоритм *CART* принципиально отличается от других алгоритмов конструирования деревьев решений.

В рассматриваемом алгоритме отсечение - это компромисс между получением дерева "подходящего размера" и получением наиболее точной оценки классификации.

Метод заключается в получении последовательности уменьшающихся деревьев, но деревья рассматриваются не все, а только "лучшие представители".

Перекрестная проверка (*V-fold cross-validation*) является наиболее сложной и одновременно оригинальной частью алгоритма *CART* - путь выбора окончательного дерева, при условии, что набор данных имеет небольшой объем или же записи набора данных настолько специфические, что разделить набор на обучающую и тестовую выборку не представляется возможным.

Алгоритм C4.5

разработан **Джоном Квинланом** (*John Ross Quinlan*).
C4.5 является усовершенствованной версией алгоритма ID3 того же автора.

Алгоритм C4.5 строит дерево решений с неограниченным количеством *ветвей* у узла. Данный алгоритм может работать только с дискретным зависимым атрибутом и поэтому может решать только задачи классификации.

C4.5 считается одним из самых известных и широко используемых алгоритмов построения деревьев классификации.

Для работы алгоритма C4.5 необходимо соблюдение следующих требований:

- Каждая запись набора данных должна быть ассоциирована с одним из предопределенных классов, т.е. один из атрибутов набора данных должен являться меткой класса.
- Классы должны быть дискретными. Каждый пример должен однозначно относиться к одному из классов.
- Количество классов должно быть значительно меньше количества записей в исследуемом наборе данных.

Версия алгоритма - алгоритм C4.8 - реализована в инструменте Weka как J4.8 (Java). Коммерческая реализация метода: C5.0, разработчик RuleQuest, Австралия.

Алгоритм C4.5 медленно работает на сверхбольших и зашумленных наборах данных.

Алгоритм (C5.0) автоматизированного построения дерева решений

Фактически алгоритм C5.0 представляет собой стандарт процедуры построения деревьев решений.

Эта программа реализуется на коммерческой основе (http://www.rulequest.com/), но версия, встроенная в пакет R (и некоторые другие пакеты) доступны бесплатно.

Выбор признака, по которому будет осуществляться разбиение: ищем такой признак (для построения разбиения по нему), который позволил бы получить как можно более чистые группы.

Группы, полностью состоящие из элементов одного класса называют «***чистыми***» («***pure***»).

Для измерения степени чистоты группы существует несколько способов. Алгоритм C5.0 использует в качестве меры чистоты группы пон

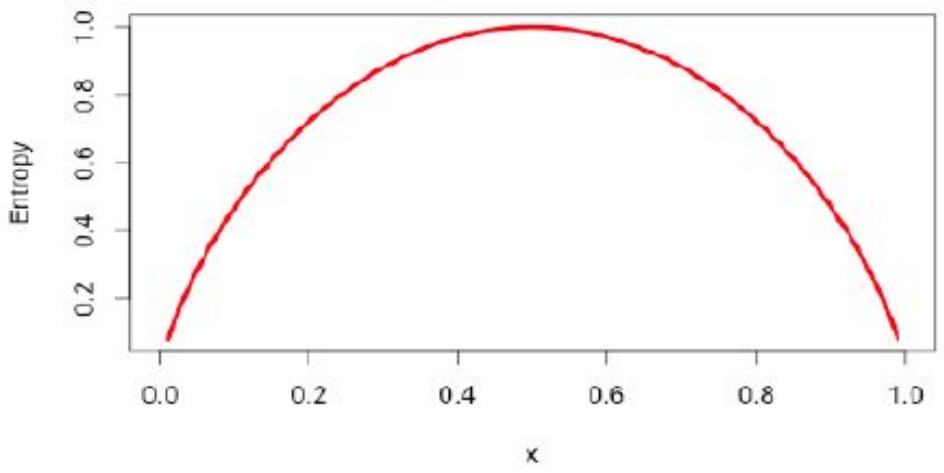
$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

– для системы, допускающей c возможных состояний.

Здесь pi – вероятность нахождения системы в состоянии i.

Энтропия как мера чистоты групп

Если у системы всего 2 возможных состояния, то её энтропия – функция одной переменной *p* , график которой имеет вид:




```
> curve(-x * log2(x) - (1 - x) * log2(1 - x),
        col="red", xlab = "x", ylab = "Entropy", lwd=4)
```

Энтропия максимальна, когда *p*=0,5. Нетрудно доказать аналитически, что для любого числа состояний максимум энтропии достигается в том случае, когда все они равновероятны.

Нулевая энтропия означает отсутствие неопределённости (применительно к деревьям решений это означает, что группа является полностью однородной, т.е. чистой).

Единичная энтропия означает полную неопределённость (в нашем случае – неопределённость относительно состава группы).

Алгоритм (C5.0) автоматизированного построения дерева решений

Алгоритм может выбрать тот признак, разбиение по которому даст самую чистую группу (т.е. группу, имеющую наименьшую энтропию). Эти вычисления называются «*information gain*» (буквально «усиление информации»).

Этот признак определяется методом перебора. Для каждого признака F («feature» – признак, свойство, характеристика) значение *information gain* вычисляется как разность энтропий группы до разбиения и после него:

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

Здесь Entropy(S2) – суммарная энтропия групп, полученных в результате разбиения.

Чем больше значение *information gain* для выбранного признака F, тем лучше этот признак F подходит для того, чтобы провести по нему разбиение.

Если для выбранного признака F значение *information gain* = 0, это означает, что разбиение по этому признаку бесперспективно, т.к. оно не приведёт к снижению энтропии.

С другой стороны, максимально возможное значение величины *information gain* = величине энтропии до разбиения (энтропия после разбиения будет = 0, т.е. полученные в результате разбиения группы будут полностью однородными - чистыми).

«Обрезка» дерева решений

Может возникнуть ситуация, когда группы окажутся слишком мелкими, а точек ветвления будет слишком много – в этом случае говорят, что модель «*is overfitted*», т.е. переопределена.

Пользоваться такими деревьями решений на практике бывает неудобно. Чтобы избежать этого, осуществляют так называемую «обрезку» (***pruning***) дерева решений. Результат «обрезки» - уменьшение размера дерева решений.

«***Pre-pruning***» - «*предварительная обрезка*»: состоит в том, чтобы *заранее* условиться о том, что процесс ветвления будет остановлен по достижении заданного числа решений. Как любой метод без обратной связи, он чреват тем, что будут пропущены важные, но трудно обнаруживаемые данные.

«***Post-pruning***» - «последующая обрезка»: сначала строится всё дерево решений, а потом по определённым правилам производится его «обрезка», т.е. отсечение лишних ветвей. Этот подход считается более надёжным - он гарантирует, что никакая важная информация не будет потеряна.

Иногда вместо того, чтобы «отсекать» ветви, их переносят выше по дереву или же заменяют более простыми - «subtree raising» («подъём поддерева») и «subtree replacement» («замена поддерева»).

Сильные стороны алгоритма C5.0	Слабые стороны алгоритма C5.0
Алгоритм C5.0 является универсальным, хорошо решает задачи классификации из разных областей.	Алгоритм C5.0 «тяготеет» к разбиениям по признакам с большим количеством уровней (Decision tree models are often biased toward splits on features having a large number of levels)
Алгоритм C5.0 может применяться для анализа не только числовых, но и номинальных данных; обеспечивает обработку пропущенных данных.	Модель (т.е. дерево решений) может оказаться как недоопределённой, так и переопредёлённой (It is easy to overfit or underfit the model)
Алгоритм C5.0 для построения дерева решений использует только самые важные признаки объектов (выбирает из множества факторов только те, которые сильно влияют на результат классификации).	Неточности классификации могут возникнуть из-за того, что используются только «параллельные осям» разбиения (axis-parallel splits)
Алгоритм C5.0 требует относительно небольшого объёма обучающей выборки.	Алгоритм C5.0 очень чувствителен к обучающей выборке – даже небольшие изменения в ней могут сильно повлиять на результат.
Интерпретация результатов работы алгоритма C5.0 не требует специальной математической подготовки.	Деревья решений иногда получаются очень большими, что затрудняет их интуитивное понимание.
Алгоритм C5.0 считается более эффективным чем другие алгоритмы классификации.	

ЗАДАЧА КЛАССИФИКАЦИИ. ДИСКРИМИНАНТНЫЙ АНАЛИЗ

Дискриминантный анализ

Дискриминантный анализ является разделом многомерного стати стического анализа, который включает в себя методы классификации многомерных наблюдений по принципу максимального сходства при наличии обучающих признаков.

В кластерном анализе рассматриваются методы многомерной классификации без обучения.

В дискриминантном анализе новые кластеры не образуются, а формулируется правило, по которому объекты подмножества подлежащего классификации относятся к одному из уже существующих (обучающих) подмножеств (классов), на основе сравнения величины дискриминантной функции классифицируемого объекта, рассчитанной по дискриминантным переменным, с некоторой константой дискриминации.

Дискриминантный анализ

Дискриминантный анализ – это общий термин, относящийся к нескольким тесно связанным статистическим процедурам.

Эти процедуры можно разделить на методы ***интерпретации межгрупповых различий – дискриминации*** и методы ***классификации наблюдений*** по группам.

Задачи дискриминантного анализа

Задачи первого типа – задачи дискриминации (пример – в медицинской практике).

Второй тип задачи относится к ситуации, когда признаки принадлежности объекта к той или иной группе потеряны, и их нужно восстановить.

Задачи третьего типа связаны с предсказанием будущих событий на основании имеющихся данных.

Дискриминация

Основной целью дискриминации является нахождение такой линейной комбинации переменных (в дальнейшем эти переменные будем называть ***дискриминантными переменными***), которая бы оптимально разделила рассматриваемые группы. Линейная функция

$$d_{km} = \beta_0 + \beta_1 x_{1km} + \ldots + \beta_p x_{pkm}, \quad m = 1,\ldots,n, \; k = 1,\ldots,g$$

называется ***канонической дискриминантной функцией*** с неизвестными коэффициентами β_i

d_{km} – значение дискриминантной функции для ***m***-го объекта в группе ***k***

x_{ikm} – значение дискриминантной переменной X_i для ***m***-го объекта в группе ***k***.

С геометрической точки зрения дискриминантные функции определяют гиперповерхности в ***p***-мерном пространстве.

Дискриминация

Коэффициенты β_i первой канонической дискриминантной функции d выбираются таким образом, чтобы центроиды различных групп как можно больше отличались друг от друга.

Коэффициенты второй группы выбираются также, но при этом налагается дополнительное условие, чтобы значения второй функции были некоррелированы со значениями первой.

Аналогично определяются и другие функции.

Отсюда следует, что ***любая каноническая дискриминантная функция имеет нулевую внутригрупповую корреляцию с*** $d_1, d_2, \ldots, d_{g-1}$

Если число групп равно g, то число канонических дискриминантных функций будет на единицу меньше числа групп. Однако по многим причинам практического характера полезно иметь одну, две или же три дискриминантных функций. Тогда графическое изображение объектов будет представлено в одно–, двух– и трехмерных пространствах.