МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ДГТУ)

Факультет «Автоматизация, мехатроника и управление»

Кафедра «Автоматизация производственных процессов»

«Основы разработки СППР на языке Python»

Конспект лекций

Ростов-на-Дону

2022

УДК 531

Составитель: Быкадор В.С.

Конспект лекций. – Ростов-на-Дону : Донской гос. техн. ун-т, 2022. – 21 с.

Конспект лекций по дисциплине «Основы разработки СППР на языке Python» предназначены для студентов очной и заочной форм обучения по направлению подготовки 15.03.04 «Автоматизация технологических процессов и производств» профиль «Интеллектуальные системы сбора и анализа больших данных».

УДК 531

Печатается по решению редакционно-издательского совета
Донского государственного технического университета

В печать ___.___.20__г.

Формат 60x84/16. Объем ____ усл. п. л.

Тираж___ экз. Заказ № ___.

Издательский центр ДГТУ
Адрес университета и полиграфического предприятия:
344000, г. Ростов-на-Дону, пл. Гагарина, 1

СОДЕРЖАНИЕ

Метод k ближайших соседей

Общая характеристика метода k ближайших соседей

Метода k ближайших соседей является, пожалуй, одним из самых простых методов машинного обучения. Вся суть данного метода заключается в том, что запоминается весь обучающий, размеченный набор данных. Для того чтобы отнести новый объект к одному из известных классов, алгоритму достаточно найти ближайших, к этому объекту, элементов известных классов, то есть найти ближайших соседей.

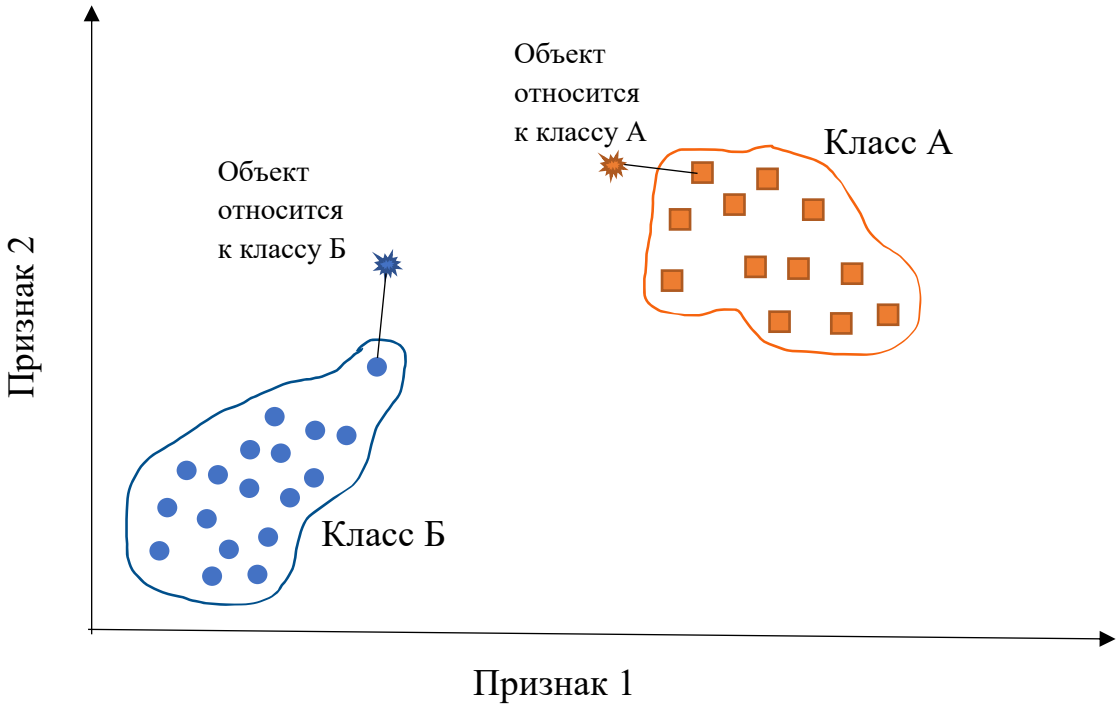
Алгоритм классификации с помощью k соседей

Рассмотрим простейший случай алгоритма k ближайших соседей, который выполняет классификацию представленного объекта по одному объекту из обучающего набора данных. Будем рассматривать задачу на плоскости, то есть каждый объект будем рассматривать как точку. Тогда количество признаков будет равно двум.

Пусть имеется обучающий набор данных, отражающий объекты двух классов – класс А и класс Б. Каждый из этих объектов характеризуется двумя измеряемыми признаками – признак 1 и признак 2, то есть каждый признак может быть оценён численно, так как показано в таблице ниже.

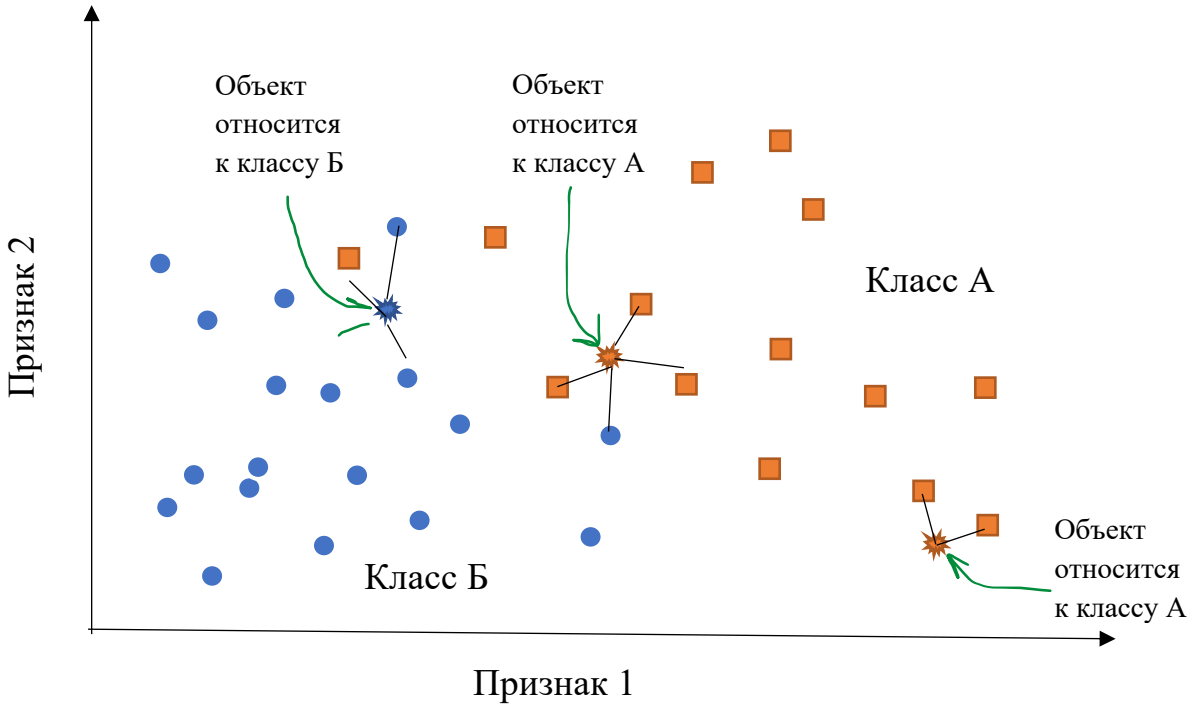
Признак 1	Признак 2	Класс
10,1	20,4	А
23,2	32,2	А
12,5	25,6	А
5,6	10,3	Б
2,3	11,2	Б
7,8	15,8	Б

10,7	31,4	A
…	…	…
…	…	…



В данном случае мы выполнили классификацию по одному ближайшему соседу. Но можем выполнить и по большему количеству соседей, не даром алгоритм называется «k ближайших соседей».

В случае, когда рассматривается количество «соседей» больше, чем один, то для принятия решения о том, к какому классу относится исследуемый объект, используется процедура «голосования» (voting), по сути, это означает подсчёт ближайших соседей, к исследуемому объекту, по каждому классу. Затем происходит присваивание исследуемого объекта к тому классу, точек которого будет больше около исследуемого объекта.



Следует отметить, что для полноты представления о методе k ближайших соседей, необходимо упомянуть, что количество классов может быть больше двух.

В случае двух классов мы имеет дело с дифференциальной классификацией, а если классов будет больше, то будет иметь дело с мультиклассовой классификацией.

Линейные модели классификации

Общие представления о линейных моделях машинного обучения

Линейные модели широко применяться в машинном обучении, а вообще этот тип моделей уже давно используется в научной сфере и различных областях народного хозяйства.

Часто линейные модели используются для прогнозирования какого-либо показателя. Формула линейной модели регрессии выглядит так:

$$\hat{y} = b + \sum_{i=0}^{N} \omega_i \cdot x_i \qquad (1)$$

где $\hat{y}$ – прогноз, выдаваемый моделью.

ω_i, b – параметры модели, которые оцениваются при её обучении;

x_i – i-ый признак для отдельной точки данных.

Линейные модели для задач классификации

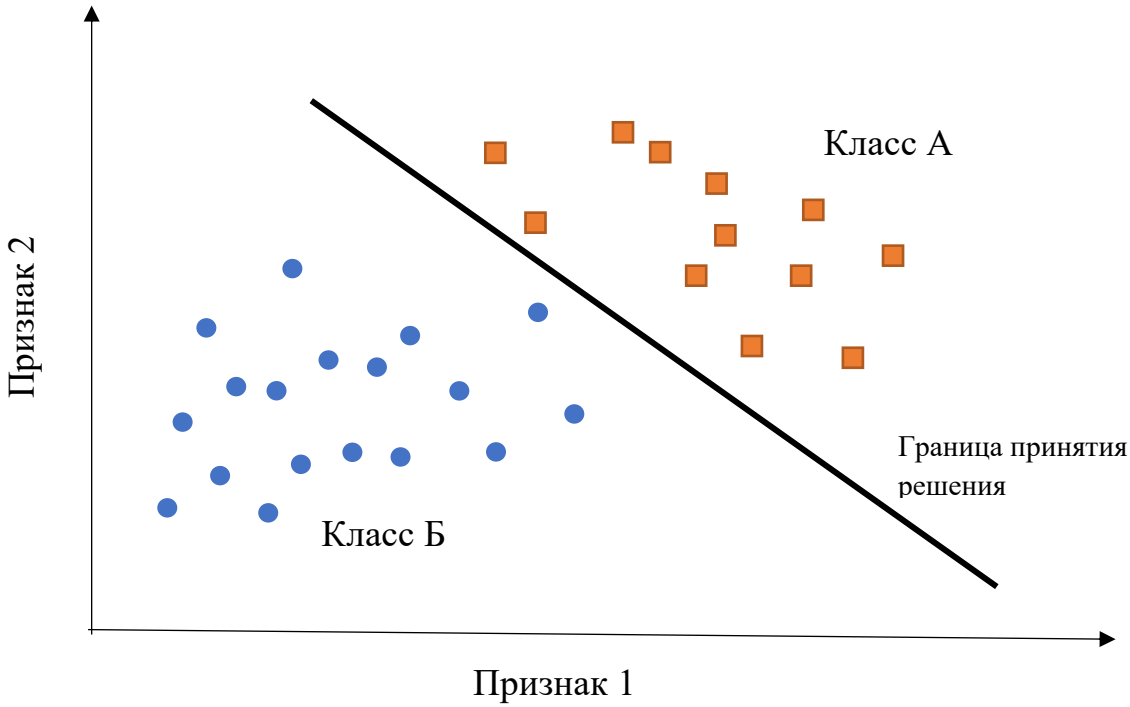
Кроме прогнозирования линейные модели достаточно широко применяются для классификации объектов.

Начнём рассмотрение линейных моделей с бинарной (дифференциальной) классификации. При этом формула (1) будет преобразована к следующему виду:

$$\hat{y} = \left(b + \sum_{i=0}^{N} \omega_i \cdot x_i \right) > 0 \qquad (2)$$

В отличии от формулы для прогнозирования, в выражении для классификации вводится некоторая граница, равная нулю. Теперь, если

значение функции $(\hat{y} > 0) \rightarrow$ класс "$+1$", а если $(\hat{y} < 0) \rightarrow$ класс "-1". Если мы обратимся к линейной классификации, то получаем границу принятия решений (decision boundary). Если мы обратимся к геометрической интерпретации данного подхода, то для двумерного случая мы получим некоторую линию, которая разделяет объекты на два класса, как это показано на рисунке ниже. В общем случае, когда размерность признаков больше двух, за место линии мы получим гиперплоскость.



Существуют два основных алгоритма для нахождения границы принятия решения:

1) Логистическая регрессия (logistic regression);
2) Линейный метод опорных векторов (linear support vector machines).

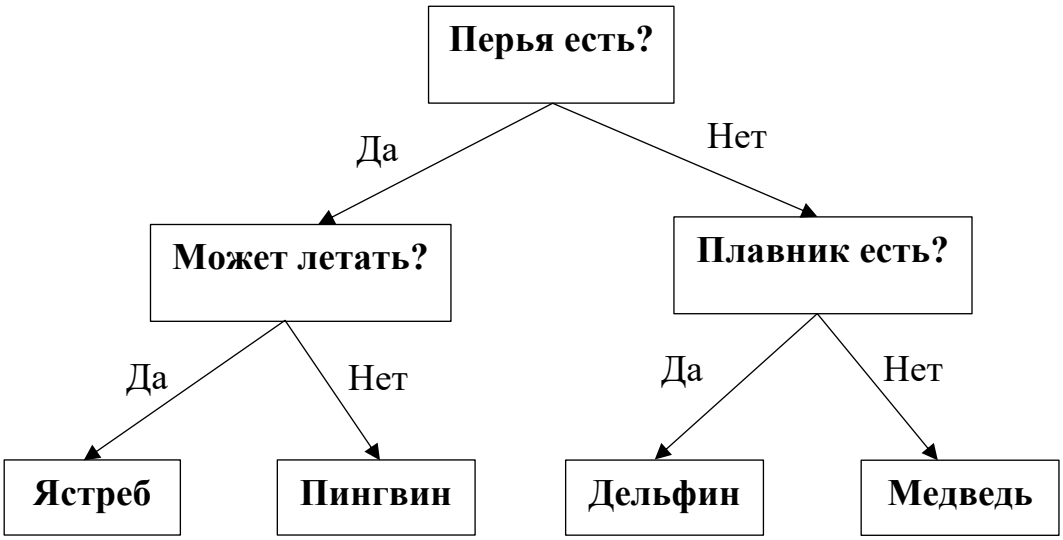
Деревья решений

Общие представления о деревьях решений

Деревья решений широко применяются для задач классификации и по сути, решение об отнесении исследуемого объекта к одному из заранее известных классов строится на последовательности вопросов «если …, то …». Формируя последовательно ответы на поставленные вопросы выполняется продвижение по структуре и в конечном счёте выявляется класс, к которому принадлежит исследуемый объект.

Принцип классификации объектов на основе дерева решений

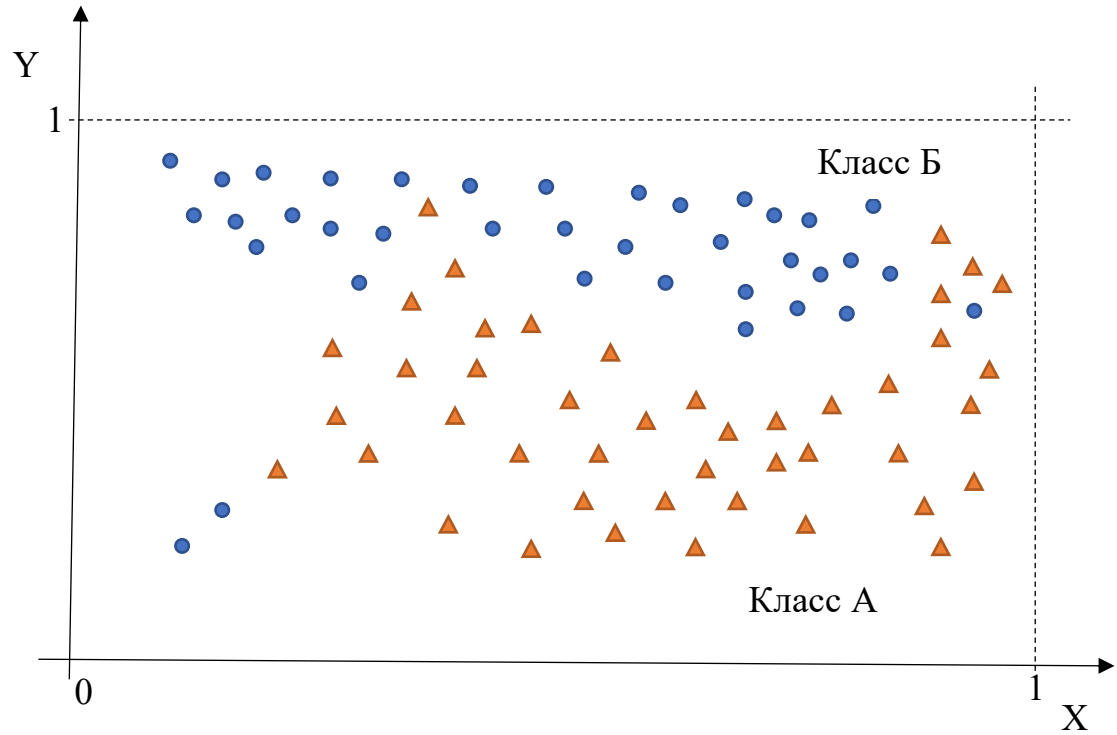
Допустим, что необходимо различать между собой четыре вида животных: медведей, ястребов, пингвинов и дельфинов. Для правильной классификации необходимо составить дерево решений, продвигаясь по которому мы сможем выполнить классификации исследуемого объекта. Такое дерево решений может выглядеть как показано на рисунке ниже.



На данной схеме показаны за ранее определенные классы: ястреб, пингвин, дельфин и медведь. А также ряд вопросов, ответы на которые приводит к определению класса исследуемого объекта. При построении дерева решений, необходимо провильно сформулировать вопросы, которые приведут к соответствующему классу. Для корректного

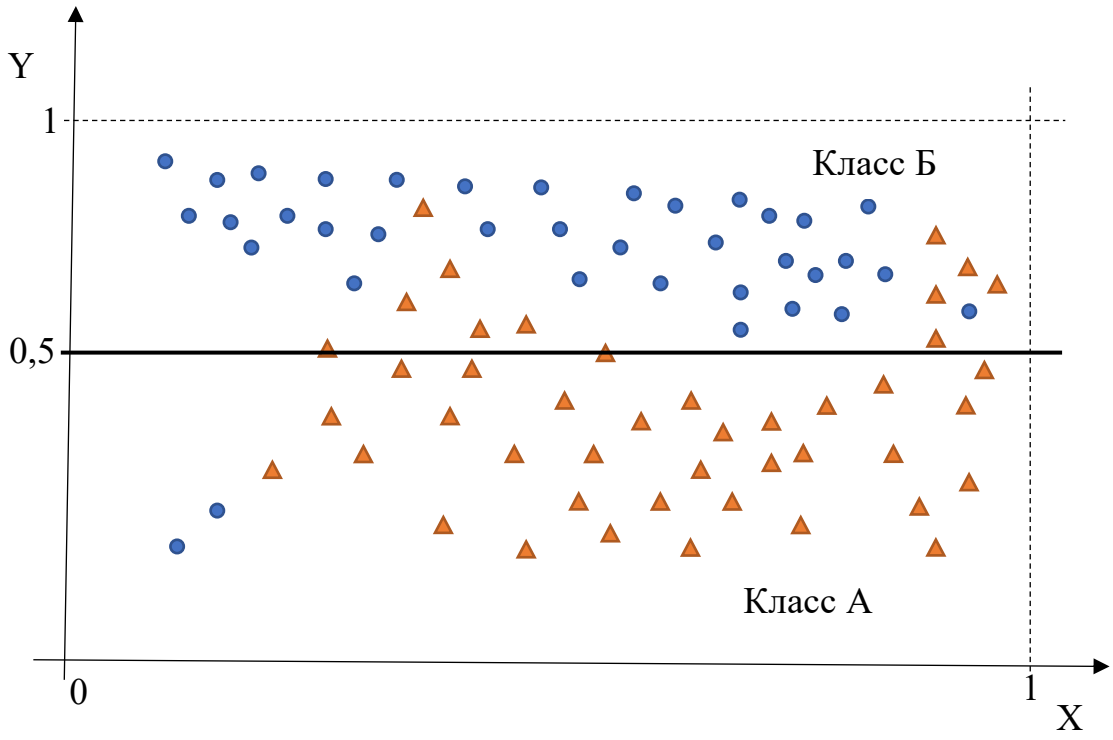
формылирования вопроса требуется определиться с признаками по которым можно различить классы между собой. Данный процесс имеет определённый формализм. Так, например, в рассматриваемом примере, вначале выбирается признак, который однозначно отделить птиц от млекопитающих – «Перья есть?». То есть сначала определяется укрупнённая группа, но тем, не менее ответ на данный вопрос уже отсекает два класса (в данном примере). Далее задается вопрос по признаку, который однозначено выведет правильный класс объекта. Ещё раз следует отметить, что как вопросы, так и признаки для вопросов формируются в системе ранее определённых классов.

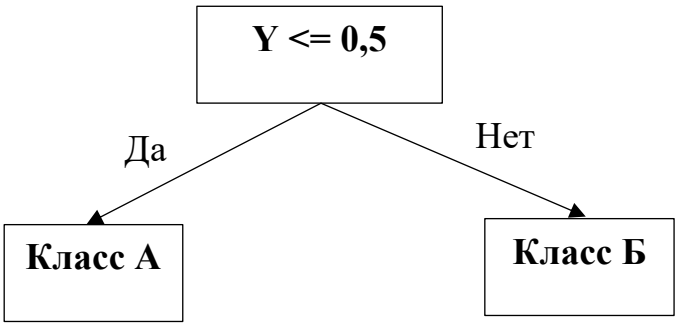
В общем случае, данные могут быть представленны не только в виде «Да/Нет», а так же в виде непрерывных признаков, например, как показано на рисунке ниже.



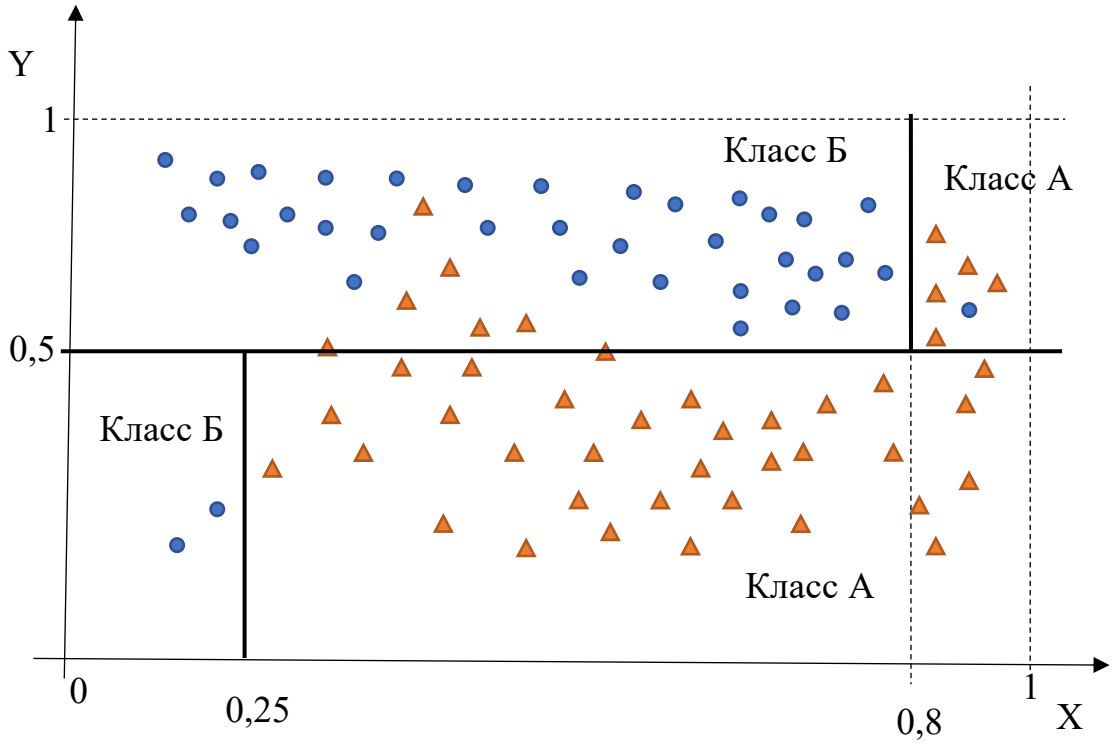
Для непрырыных данных, тесты (то есть вопросы) имеют следующий вид: «Признак i больше некоторого значения a?». Для построения дерева решений, перебираются все возможные варианты или так называемые

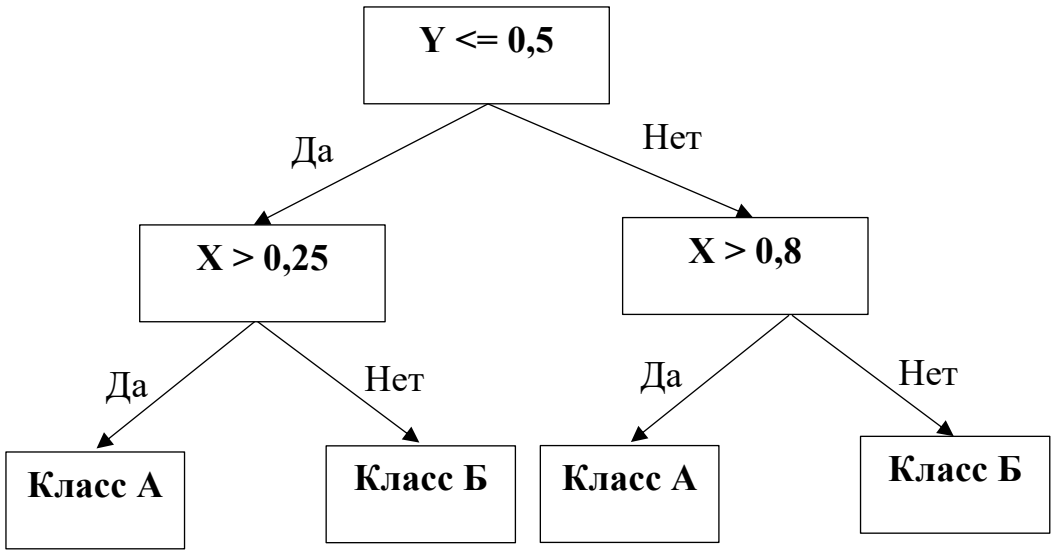
тесты (не путать с тестовым набором данных) и выбирается наиболее информативный, то есть такой, который позволяет с наименьшей ошибкой определить класс исследуемого объекта. Например, для выше приведенного рисунка данных мы можем получить следующий тест: **Y <= 0,5 => «Класс А»**.





Первый тест довольно неплохо разделил два класса, тем не менее, как можно видеть в каждом классе, имеются точки, которые относятся к другому классу. Можно улучшить модель классификации на основании дерева решений, за счёт добавления дополнительных тестов. Например, так:





Вводя новые тесты, можно повышать точность модели классификации объектов, но следует учитывать, что, повышая точность модели классификации повышается и сложность дерева решений. В идеальном случае можно добиться наличия в дереве, так называемых «чистых листьев», то есть, когда модель выдаёт 100%-но правильные решения на обучающем наборе данных. А это в свою очередь может привести к переобучению модели.

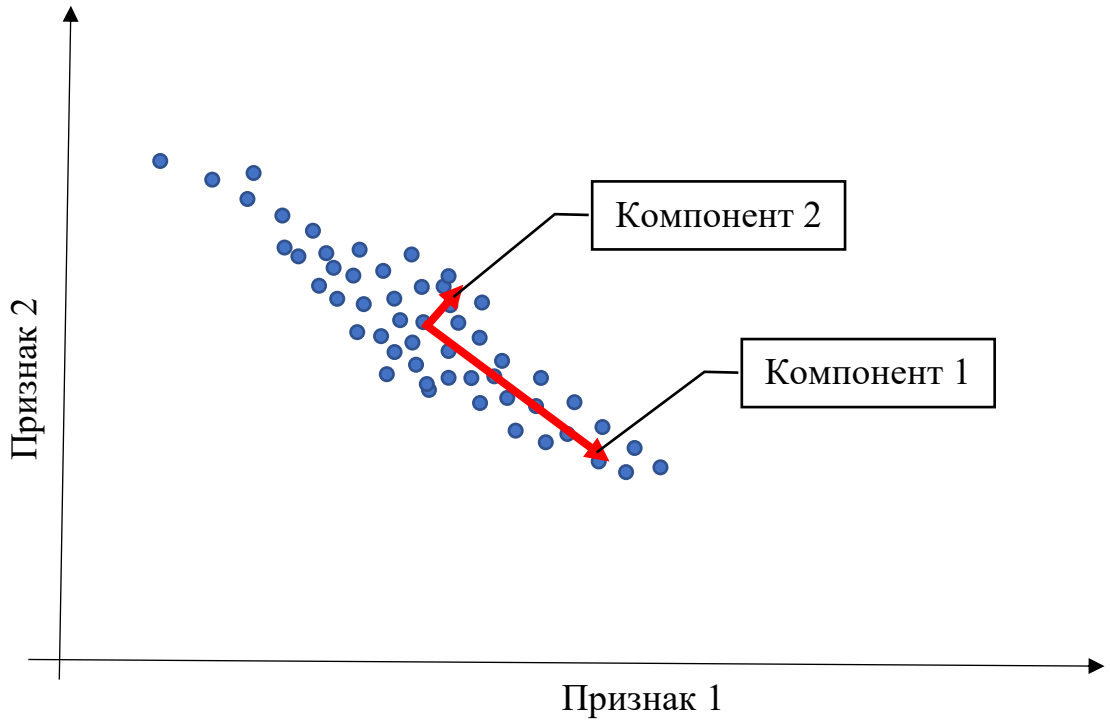
Методы снижения размерности, выделения признаков и множественного обучения

Основная задача

В машинном обучении может использоваться значительное количество данных, а также достаточно большое количество признаков. Для снижения количества данных и уменьшения количества признаков – снижение размерности задачи обучения можно использовать различные методы, ниже будет рассмотрен один из них, который позволяет изучить данную методологию.

Анализ главных компонент (PCA)

Основой метода анализа компонент является вращение данных с целью получения преобразованных признаков, которые не должны коррелировать между собой. Обычно вращение признаков приводит к выбору подмножества новых признаков основанной на их важности с точки зрения интерпретации данных. Для пояснения преобразования данных по методу главных компонент рассмотрим рисунок, приведённый ниже.



На рисунке показаны исходные данные (точки). Алгоритм работает следующим образом:

1) Находит направление максимальной дисперсии, которая обозначена как «Компонент 1». То есть выбирается направление (вектор) данных, который содержит большую часть информации. По-другому можно сказать, что выбирается направление, вдоль которого признаки коррелируют с друг другом больше всего.
2) Затем алгоритм находит направление, которое содержит набольшее количество информации, но при этом оно ортогонально первому найденному направлению.

Следует отметить, что в двумерном случае (как показано на рисунке выше) возможно только одно ортогональное направление по отношению к компоненту 1, но для многомерного случая возможно несколько ортогональных направлений.

Направления, найденные с помощью данного алгоритма, называются **главными компонентами**. В общем случае максимальное количество главных компонент равно количеству исходных признаков.

Таким образом, используя алгоритм выделения главных компонент мы можем снизить размерность задачи машинного обучения, оставив только один главный компонент – «Компонент № 1» (для этой двумерной задачи). А также уменьшить количество данных для анализа алгоритмами машинного обучения.

Следует отметить, что данный алгоритм может использоваться для удаления «шума» из данных.

Методы кластеризации

Основы методов кластеризации

Задачей кластеризации является разделение точек данных на группы, которые называются **кластерами**. Основной целью разделения данных является отнесение близких точек в один кластер.

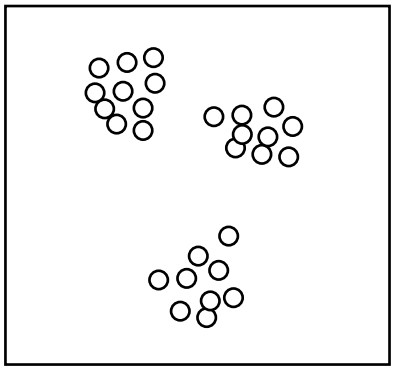
Кластеризация по методу k-средних

Суть метода кластеризация по методу k-средних состоит в следующих шагах:

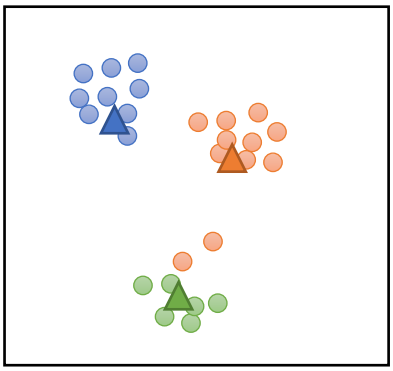
1) Выбор необходимого числа k кластеров.
2) отбор точек, которые будут представлять центры кластеров.
3) Вычисление для каждой точки евклидова расстояния до центра каждого кластера.
4) Отнесение каждой точки к тому или иному центру (кластеру) на основании значения вычисленного кластера.
5) После того как все точки будут распределены по кластерам, будут вычислены так называемые центроиды – центры тяжести кластеров. Каждый центроид является вектором, элементы которого представляют собой средние значения характеристик по всем точкам кластера.
6) После вычисления центроидов алгоритм метода переносит в них центры кластеров.
7) Выполнятся переназначение точек по данным новых центров кластеров.

Алгоритм пересчета центров кластеров и переназначения точек каждому кластеру повторяется до тех пор пока расположение точек центроидов не перестанет меняться, а это возможно когда одни и те же точки в i-ой итерации будут попадать в те же кластеры, что и на (i-1)-ой итерации.

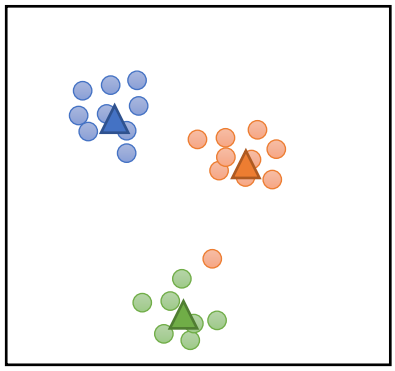
Исходные данные



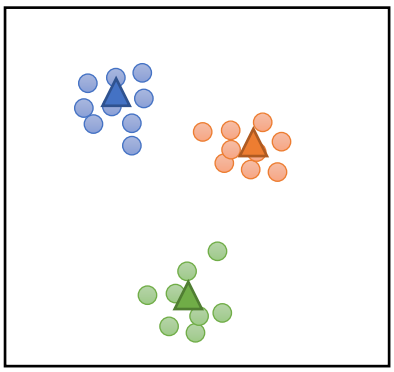
Первичные центроиды



Пересчёт № 1



Пересчёт № 2



Модель текстовых данных в виде "мешка слов"

Общее представление об обработке текстовых данных методами машинного обучения

Кроме представления данных в виде численных данных, для машинного обучения также широко используются данные в виде текста. Например, часто требуется анализ сообщений электронной почты или отзывов, даже возможна обработка таким образом текстовых документов.

Модель текстовых данных в виде «мешка слов»

Рассматриваемая модель текстовых данных является одной из самых простых, но эффективных. Данная модель, позволяет подготовить данные для машинного обучения. Название «мешок слов» используется потому, что из текста удаляется его структура, например, удаляется разделение текста на главы, параграфы, удаляются знаки препинания и форматирование текста.

В модели текстовых данных «мешок слов» важным является частота, с которой встречается каждое слово в тексте. Формально алгоритм получения «мешка слов» будет выглядеть следующим образом:

1) Токенизация – разбиение текстового документа на слова, которое в нём встречаются (получаем так называемые токены).
2) Построение словаря – собираются все слова, которые встречаются в текстовом документе и пронумировываются, например, можно слова собирать в алфавитном порядке.
3) Создание разряженной матрицы – подсчитываем насколько часто каждое из слов, записанное в ранее созданный словарь, встречается в текстовом документе.

Следует отметить, что можно составлять один общий словарь для нескольких текстовых документов, дополняя словарь новыми встречающимися словами в текущем текстовом документе.

На рисунке ниже показан образный пример перехода от текстового документа к «мешку слов».

Таким образом, каждое слово можно представить в качестве отдельного признака, а его частота появления в конкретном тексте будет характеризовать величину данного признака. Тогда текст мы можем рассматривать как вектор в пространстве слов. Далее мы можем разметить каждую область пространства слов и наблюдать куда указывает исследуемый вектор (текст), тем самым делать предположения о характере информации содержащейся в конкретном тексте.

Масштабирование данных методом tf- idf

Общее представление о методе tf - idf

Метод «частота термина – обратная частота документа» (term frequency – inverse document frequency или сокращённо tf-idf). Идея этого метода заключается в том, чтобы присвоить больший все термину, который часто встречается в конкретном документе, но при этом редко встречается в остальных документах, входящих в выборку для анализа. Если слова часто встречается в документе, но при этом редко встречается в других документах, то скорей всего это слово будет лучше описывать содержимое конкретного документа.

Метод tf - idf

Для реализации метода «частота термина – обратная частота документа» (tf-idf) необходимо каким-то образом выполнять взвешивание частот слов.

Существует несколько вариантов взвешивания частот, рассмотрим один из них.

Значение метрики tf-idf для слова w в документе d вычисляется с помощью выражения:

$$tfidf(w,d) = log\left(\frac{N+1}{N_w+1}\right)+1$$

где N – количество документов в обучающей наборе;

N_w – количество документов обучающего набора, в котором встретилось слово w.

Дадим некоторые пояснения по методу tf-idf.

Признаки и низкими значениями метрики tf-idf представляют собой признаки, которые либо встречаются во многих документах, либо используются редко.

Модель текстовых данных в виде "мешка слов" для n-грамм

Одним из недочётов представления в модели «мешок слов» заключается в полном игнорировании порядка слов. По этой причине выражения «Это не плохо, а хорошо» и «Это не хорошо, а плохо» в модели «мешок слов» будет представлены одинаково. И действительно, словарь будет выглядеть следующим образом, как для одного, так и для другого выражения:

```
['Это', 'не', 'плохо', 'а', 'хорошо']
```

Хотя в действительности вышеприведенные выражения имеют абсолютно противоположный смысл. И данный недостаток модели «мешка слова» можно устранить, если фиксировать не только частоты слов, но и из пары, тройки и так далее, которые появляются рядом друг с другом. Для совместно фиксируемых слов, а в терминах модели «мешок слов» - токенов, существуют названия. Для одиночный токен называется юниграммой, пара токенов называется биграммой, тройка токенов называется триграммой, а в общем случае - n токенов называется n-граммой.

Отметим следующие важные для практики случаи.

В большинстве прикладных задач используются юниграммы, то есть обычная модель «мешка слов». Менее часто, но тем не менее достаточно часто, чтобы об этом сказать используются биграммы. Использование n-грамм, при $n > 2$ приводит к резкому росту количества признаков, а это в свою очередь может привести к переобучению модели. Так, например, количество биграмм равно количеству юниграмм возведённых в квадрат, а количество триграмм равно количеству юниграмм возведённых в куб. Но в данном случае следует отметить, что реальное количество сгенерированных n-грамм будет определяться ещё и структурой естественного языка на котором составлен текст анализируемого документа.

Список использованных источников

1. Мюллер А., Гвидо С. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными, СПб.: ООО «Альфа-книга» – 2017, 480 с. : ил.