Деревья решений. Классификация

Занятие 2.1

Цели занятия

В конце занятия вы научитесь:

1. Применять алгоритм классификации, принятие решений которого можно проинтерпретировать
2. Измерять качество решений в задачах классификации
3. Оценивать важность фичей
4. Понимать основу продвинутых алгоритмов, таких как Random Forest, XG Boost, LGBM, etc..

О чём поговорим и что сделаем

О чём поговорим и что сделаем

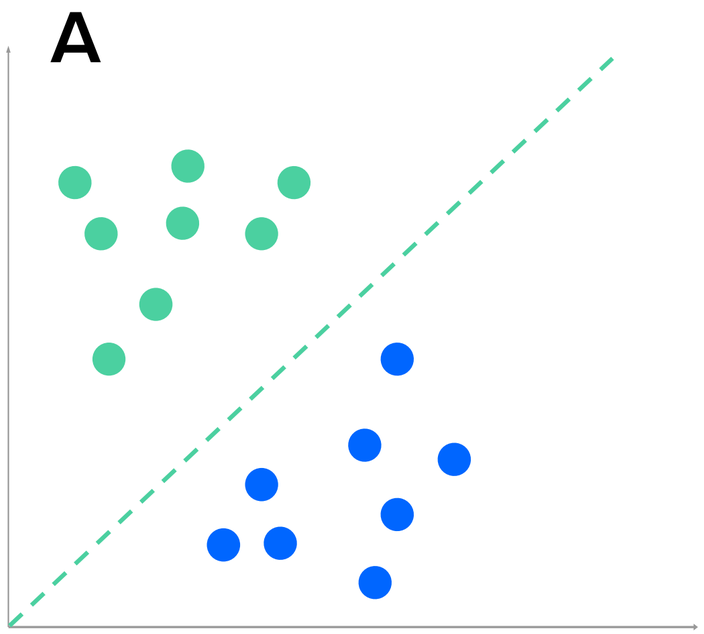
1. Задача классификации: постановка и примеры
2. Дерево решений: как его построить? Обзорsklearn.tree.DecisionTreeClassifier
3. Достоинства и недостатки деревьев решений.
4. Визуализируем принятие решений и предсказания алгоритма; примем участие в соревновании Kaggle
5. Метрики качества в задачах классификации
6. Оценим решение Kaggle; классифицируем статьи Ведомостей: политика или финансы?

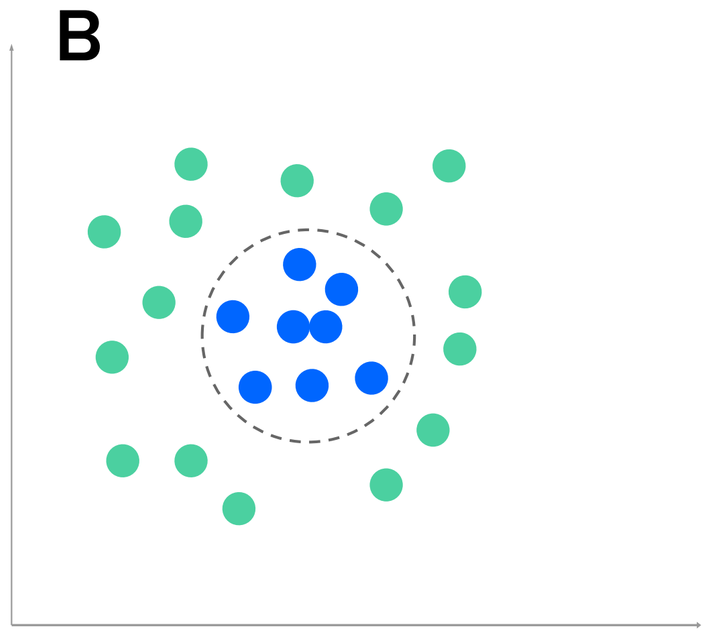
Задача классификации

1

Типы задач

- **классификация**
- ранжирование
- регрессия
- кластеризация





Примеры задач классификации

- **Скоринг.** Вернёт ли клиент кредит? (banking, insurance)
- **Отток.** Перестанет ли пользоваться клиент услугами компании? Перестанет ли, если дать ему бонус?(marketing)
- **Intent recognition.** О чём говорит пользователь в своем обращении? (может быть несколько intent’ов, может быть древовидная структура) (API.AI)
- **Image recognition.** Что на картинке? (Google,FindFace)

Постановка задачи

Задача восстановления зависимости *y*: $X \to Y, \ |Y| < \infty$
по точкам *обучающей выборки* $(x_i, y_i), \ i = 1, ..., l$:

Дано: векторы $x_{i=}(x_i^1, \dots x_i^n)$ - объекты обучающей выборки, $y_i = y(x_i)$ - классификации, ответы учителя, $i = 1, \dots, l$:

$$\begin{pmatrix} x_1^1 & \dots & x_1^n \\ \dots & \dots & \dots \\ x_\ell^1 & \dots & x_\ell^n \end{pmatrix} \xrightarrow{y^*} \begin{pmatrix} y_1 \\ \dots \\ y_\ell \end{pmatrix}$$

Найти: функцию *a (x)*, способную классифицировать объекты произвольной *тестовой выборки* $\tilde{x}_i = (\tilde{x}_i^1, \dots, \tilde{x}_i^n), \ i = 1, \dots, k$:

$$\begin{pmatrix} \tilde{x}_1^1 & \dots & \tilde{x}_1^n \\ \dots & \dots & \dots \\ \tilde{x}_k^1 & \dots & \tilde{x}_k^n \end{pmatrix} \xrightarrow{a?} \begin{pmatrix} a(\tilde{x}_1) \\ \dots \\ a(\tilde{x}_k) \end{pmatrix}$$

* из материалов к курсу Воронцова на Coursera

Построение дерева решений

2

Цветки ириса: задача

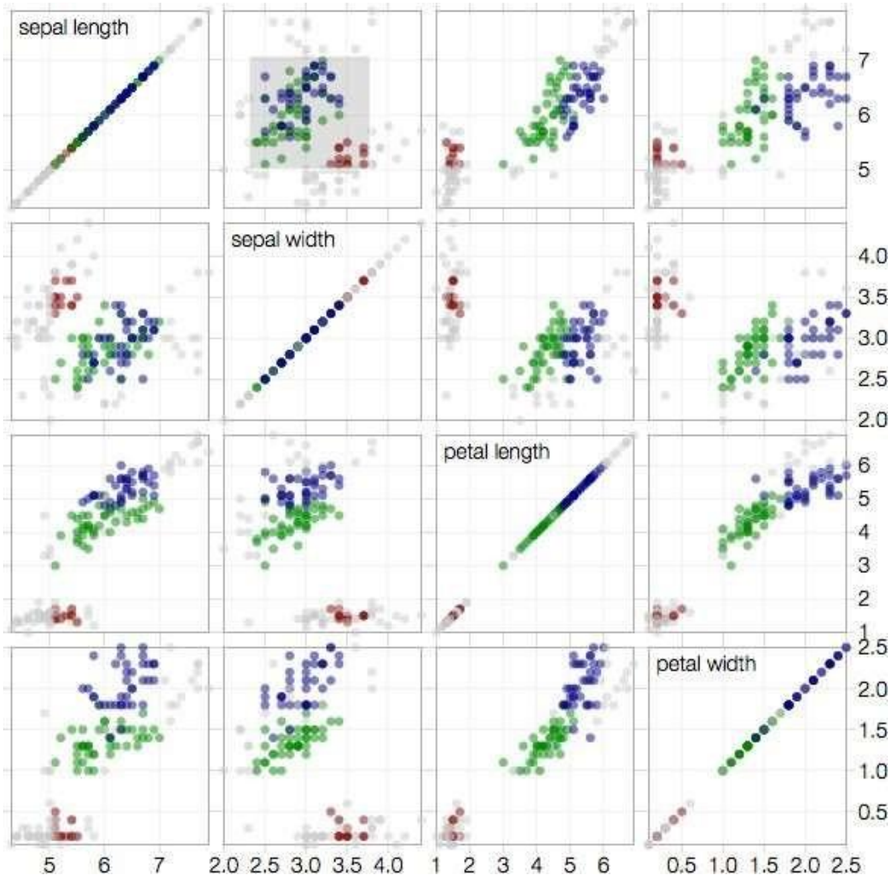
Ирис щетинистый
(*Iris setosa*)

Ирис разноцветный
(*Iris versicolor*)

Ирис виргинский
(*Iris virginica*)

* Ирисы Фишера, 1936г.

Цветки ириса: данные



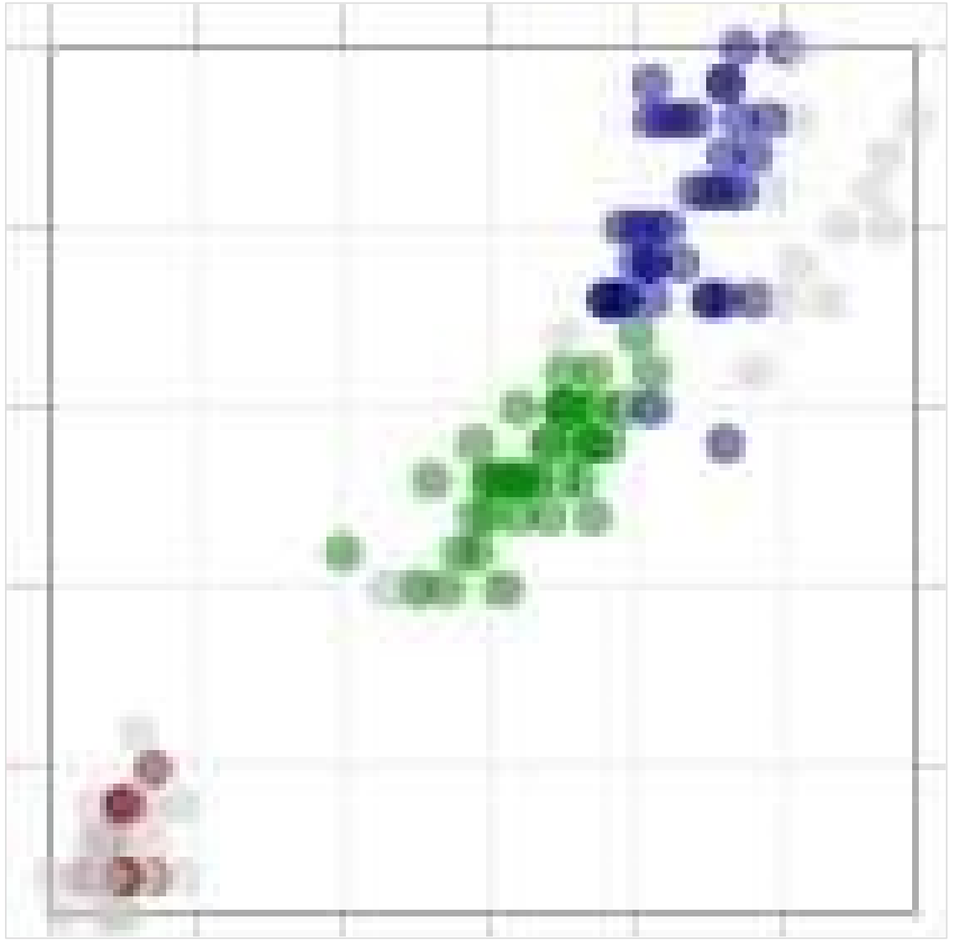
Дано:

- 3 вида цветков ириса
- 4 параметра: 2 длины и 2 ширины листа
- по 50 наборов значений на каждый вид

Найти:

- тип цветка по 4 параметрам

Цветки ириса: данные



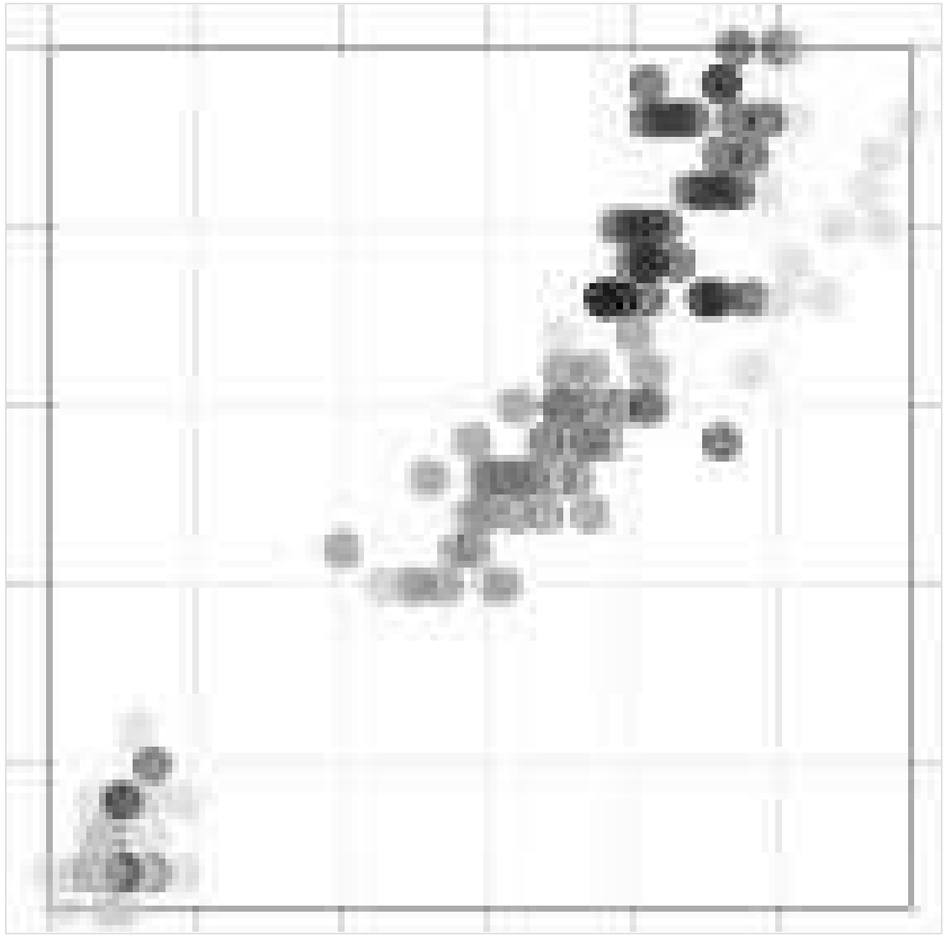
Дано:

- 3 вида цветков ириса
- 4 параметра: 2 длины и 2 ширины листа
- по 50 наборов значений на каждый вид

Найти:

- тип цветка по 4 параметрам

Цветки ириса: данные



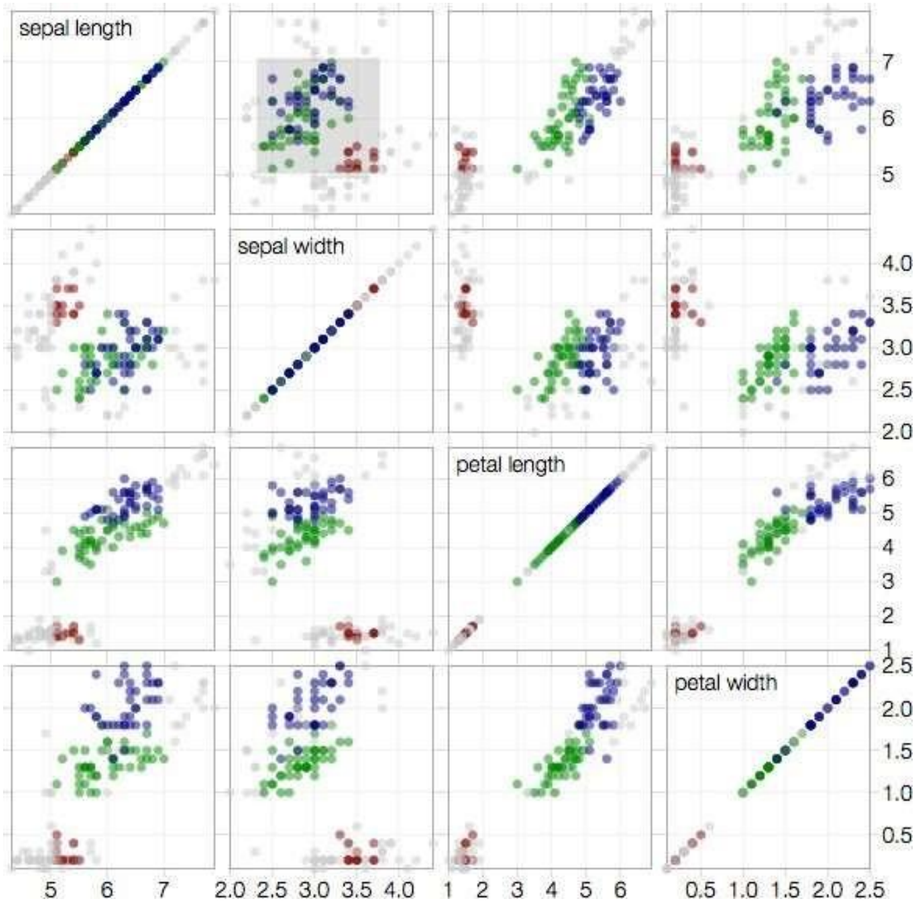
Дано:

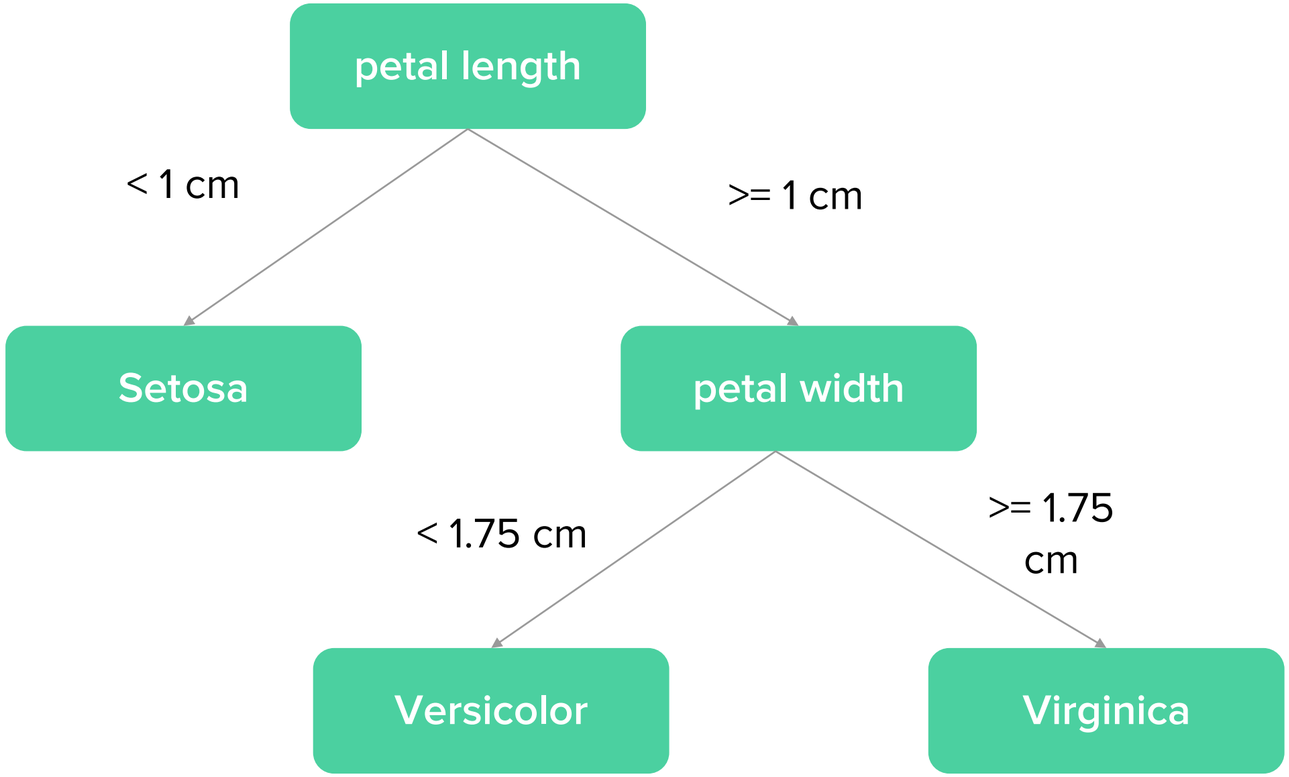
- 3 вида цветков ириса
- 4 параметра: 2 длины и 2 ширины листа
- по 50 наборов значений на каждый вид

Найти:

- тип цветка по 4 параметрам

Цветки ириса: решающее дерево

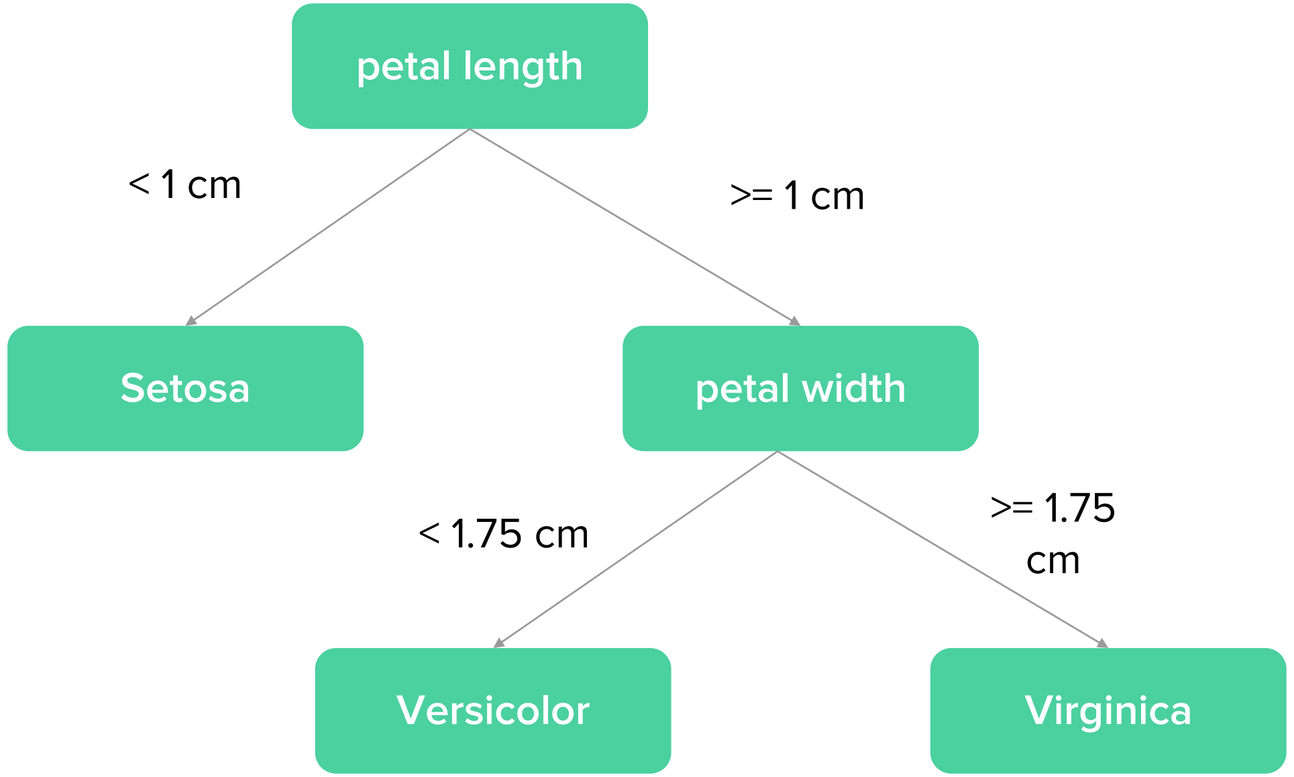




Как построить дерево?

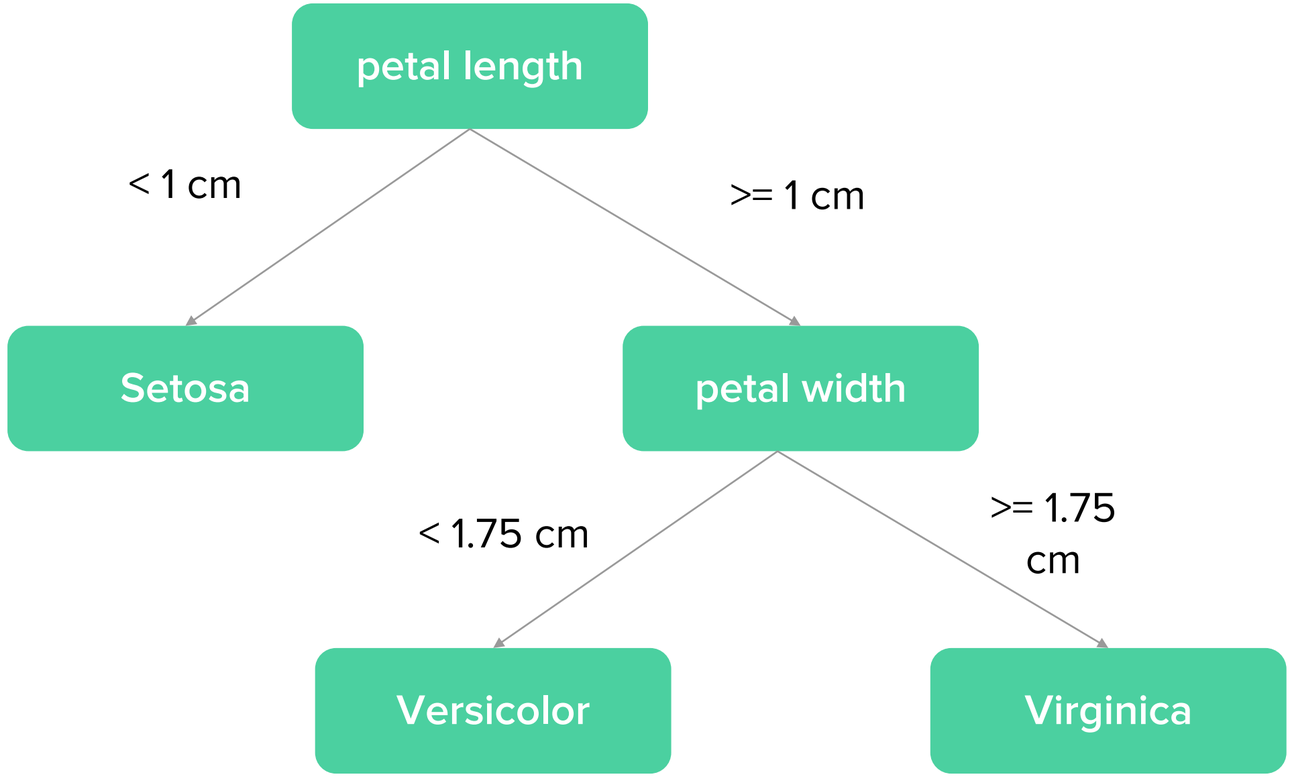
Определить:

- вид правила разбиения
- критерий информативности разбиения
- критерий останова
- метод стрижки
- обработка пропусков



Вид правила для разбиения

- **одномерное:** сравнивается значение одной фичи векторах
- **линейное:** сравнивается линейная комбинация фичей
- **метрическое:** расстояние до точки признакового пространства



Здесь используется одномерный предикат: сравнение идёт лишь по одной фиче из вектора признаков

Функционал качества разбиения

Идея:

- взять признак
- отсортировать его по возрастанию
- в зависимости от целевой переменной установить порог разделения выборки на две, максимально снижая численно выражаемый разброс внутри каждой из 2 групп
- подобрать лучшее с точки зрения улучшения разбиение

Вопрос: а как измерить улучшение?

Измерение поэтапного улучшения

1 2 3 4 5 6 7 8 9

Есть 1 группа, в ней 2класса.

Пусть H(R) -«критерии информативности»группы,
больше разнообразия -больше H(R) -хуже для классификатора
Будем измерять улучшение разбиения по функционалу вида:

$$IG(R) = H(R) - q_{left}*H(R_{left}) - q_{right}*H(R_{right}),$$

где q_{left} и q_{right} - доли объектов, попавших в левый или правый класс соответственно

Измерение поэтапного улучшения

$$IG(R) = H(R) - q_{left}*H(R_{left}) - q_{right}*H(R_{right})$$

$$H(R) = x > 0$$

$$H(R_{left}) = 0$$

$$H(R_{right}) = 0$$

$$IG(R) = x - 5/9*0 - 4/9*0 = x > 0$$

Измерение поэтапного улучшения

$$H(R) = \sum_{k=1}^{K} p_k(1 - p_k)$$

K -количество классов
pk -доля класса в выборке

$IG(R) = H(R) - q_{left}*H(R_{left}) - q_{right}*H(R_{right})$

$H(R) = 4/9*(1-4/9) + 5/9*(1-5/9) = 0.494$ $H(R_{left}) = 3/4*(1-3/4) + 1/4*(1-1/4) = 0.375$ $H(R_{right}) = 1/5*(1-1/5) + 4/5*(1-4/5) = 0.32$

$IG(R) = 0.494 - 4/9*0.375 - 5/9*0.32 =$ **0.15**

Энтропийный критерий

$$H(R) = -\sum_{k=1}^{K} p_k \log p_k$$

K -количество классов
pk -доля класса в выборке

IG(R) = H(R) - qleft*H(Rleft) - qright*H(Rright)

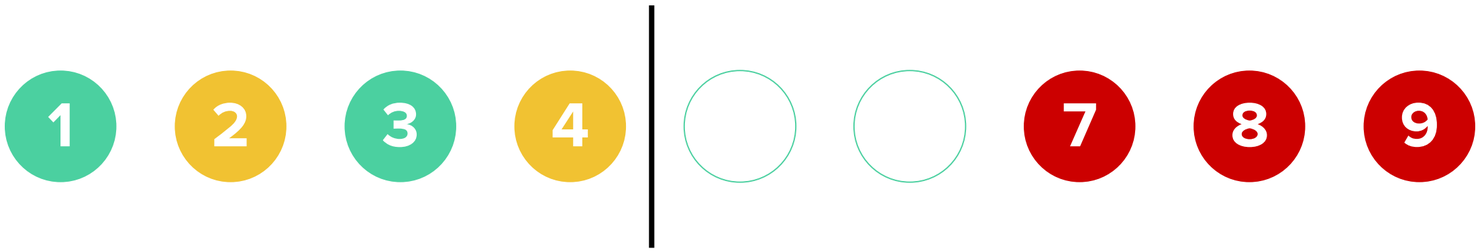
H(R) = -4/9*log2(4/9) - 5/9*log2(5/9) = 0.991
H(Rleft) = -3/4*log2(3/4) - 1/4*log2(1/4) = 0.81

H(Rright) = -1/5*log2(1/5) - 4/5*log2(4/5) = 0.72

IG(R) = 0.991 - 4/9*0.811 - 5/9*0.722 =**0.22**

Критерий Джини

IG(R) = ?



Критерий Останова

- Останов, когда в каждом листе объекты только одного класса
- Ограничение max глубины дерева
- Ограничение min число объектов в листьях
- Требование улучшения функционала качества при дроблении не менее, чем x или на x%

Проблема пропусков

- Выкинуть объекты с пропусками из обучающей (что на тестовой?)
- Замена на значения вне средние, медианные..
- Заменить на значения вне области значений фич
- Модифицировать алгоритм построения и работы дерева: включать элементы с пропусками в обе ветки дерева, но взвешивать качество разбиения по объёму пропусков

Стрижка деревьев (Pruning)

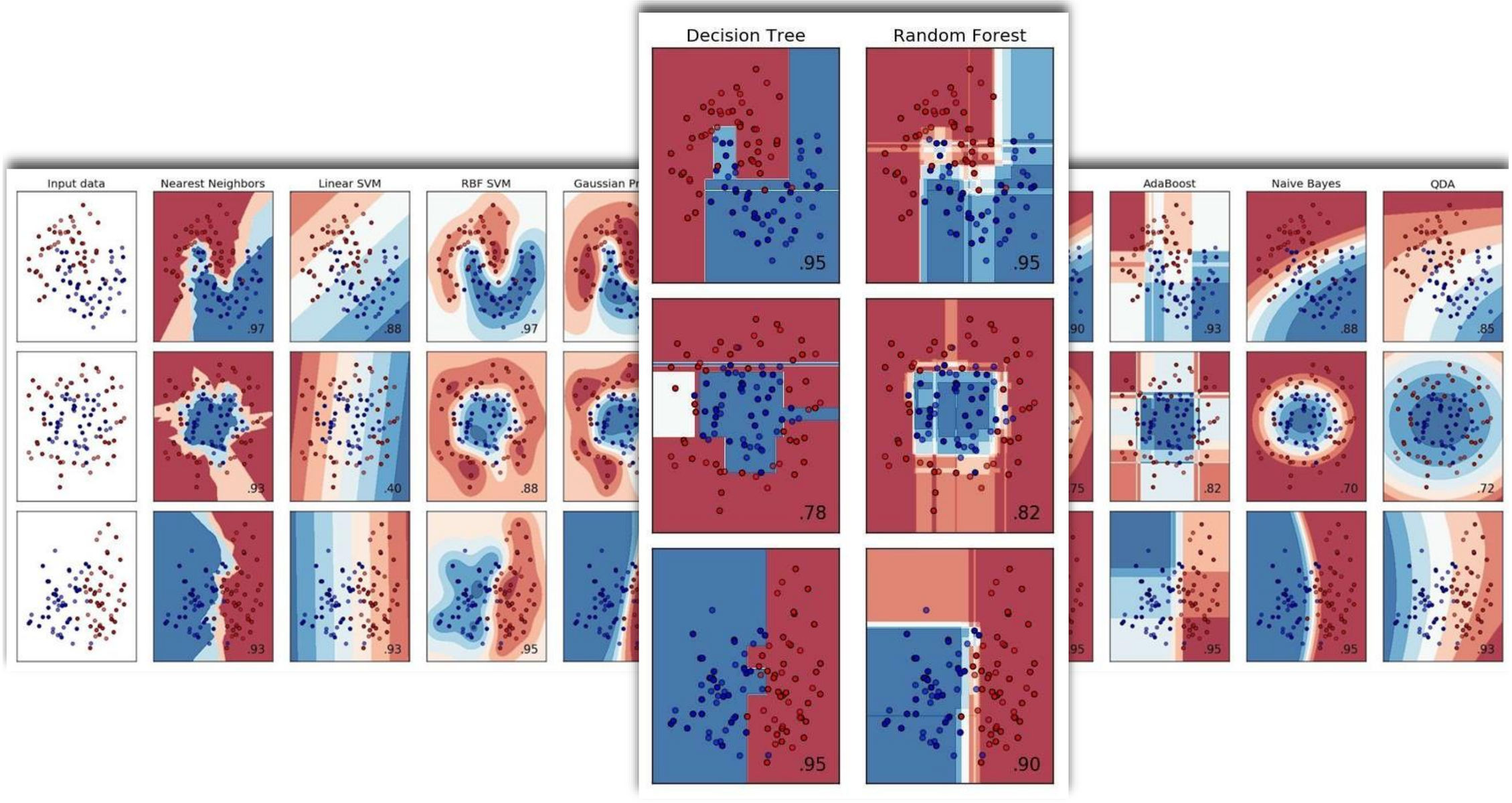
- Стрижка из полностью построенного дерева убирает наименее информативные листья
- Стрижка работает лучше раннего останова
- Редко используется, т.к. деревья не используются самостоятельно, а в ансамблях она излишняя (там либо нужно переобучение, либо используется ограничение глубины)
- В основе идея регуляризации: в функционале качестве под дерева линейно штрафуется количество листьев

Популярные методы построения

Деревья в силу дискретности не сводятся к оптимизации в аналитическом виде, поэтому все методы их построения являются эвристическими и жадными

Популярные методы отличаются ранее рассмотренными параметрами построения дерева:

- ID3: энтропийный критерий, максимально жадный, требуется стрижка(1986)
- C4.5, C5.0: нормированный энтропийный критерий
- CART: критерий Джини-используется в sklearn (optimized)



* sklearn, сравнение классификаторов

Реализация BSKLEARN

sklearn.tree.DecisionTreeClassifier

*splitter='best'
* max_depth=None
* min_samples_split=2
* min_samples_leaf=1
* min_weight_fraction_leaf=0.0
* max_features=None
* random_state=None
* max_leaf_nodes=None
* min_impurity_split=1e-07
* class_weight=None
* presort=False

Основные характеристики

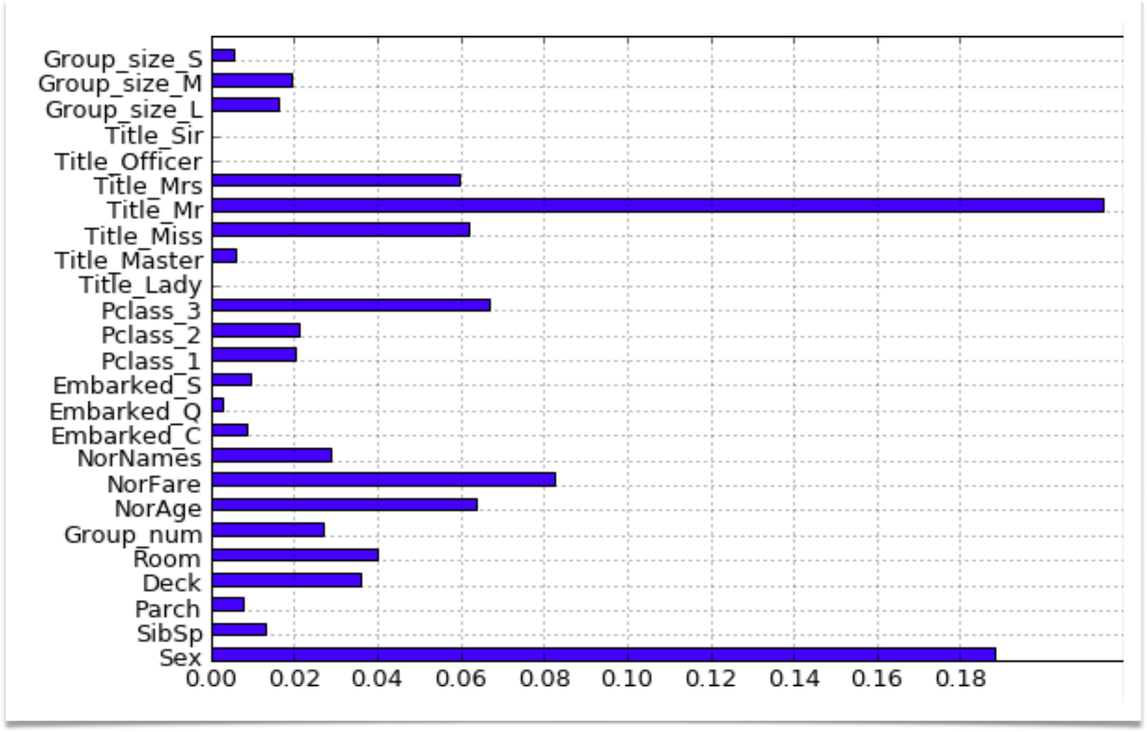
- 12 параметров
- Функционал качества: Джини /энтропия
- Реализованы различные простые критерии останова: кол-во объектов, улучшение качества..
- Не реализована стрижка дерева

Основные методы

- fit
- predict, predict_proba

Реализация BSKLEARN. Бонус

Деревья могут оценивать важность фичей



Например, судя по решению, на выживаемость на Титанике сильнее всего влияли:

- наличие в обращении «Mr.»
- пол
- уровень дохода
- проживание в 3 классе
- возраст
- наличие в обращении «Mrs» / «Miss»

Достоинства и недостатки деревьев решений

3

Достоинства

- Легко интерпретировать, визуализировать, «белый ящик»
- Простота подготовки данных: не требуется нормализация, dummy переменные, возможны пропуски
- Скорость работы

Недостатки

- Острая проблема переобучения
- Неустойчивость
- Не учитывает нелинейные зависимости или даже простые линейные, которые идут не по осям координат (f.e., представьте дерево для классификатора вида y>x)
- Чувствителен к несбалансированным классам
- Хорошо интерполирует, плохо экстраполирует

Практическое задание 2

Оценка качества классификации

4

Матрица ошибок

confusion matrix	y = 1	y = 0
a = 1	True Positive	False Positive
a = 0	False Negative	True Negative

На тестовой выборке имеем:

- y - вектор истинных значений
- a - вектор предсказаний классификатора

Будем раскладывать все пары (предсказание, истина) по ячейкам матрицы ошибок

Accuracy

confusion matrix	y = 1	y = 0
a = 1	0	0
a = 0	2	998

Accuracy, Доля верных ответов *(в просторечии точность, но не путать с точностью из ML!)* Простая метрика, но абсолютно не показательна в задачах с несбалансированными классами

Пример:
определение качества теста на рак. Тест постоянно предсказывает отсутствие рака.
Доля верных ответов: 99.8%

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision

confusion matrix	y = 1	y = 0
a = 1	170	100
a = 0	30	700

Precision, Точность

отсутствие ложных срабатываний

Пример:

правильное распознавание намерения пользователя: лучше переспросить пользователя, чем сделать не то, что нужно

Точность = 170 / (170+100) = 0.629

$$precision = \frac{TP}{TP + FP}$$

Recall

confusion matrix	y = 1	y = 0
a = 1	170	100
a = 0	30	700

Recall, Полнота

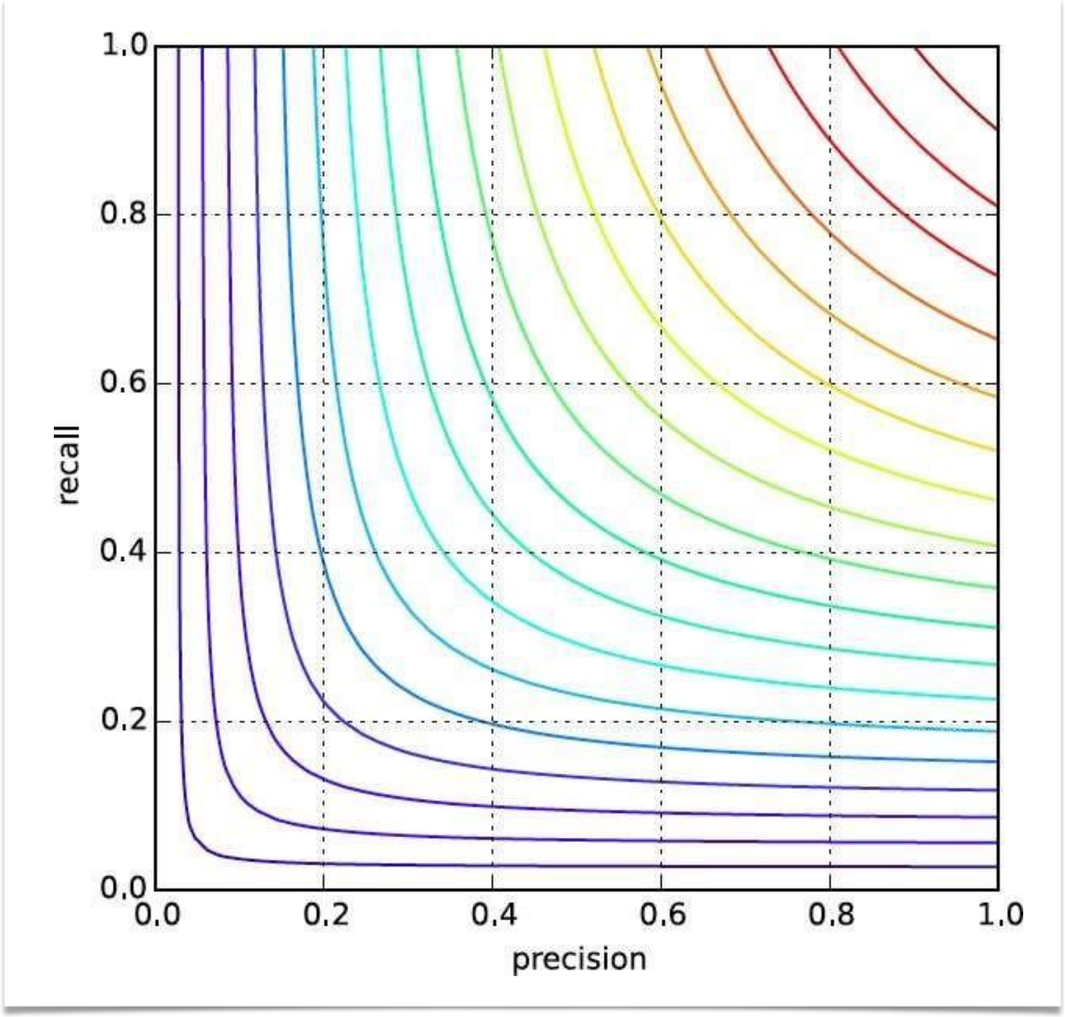
отсутствие ложных пропусков

Пример:
определение мошеннических действий в банке:
лучше найти лишнее и проверить, чем не найти

Точность = 170 / (170+30) = 0.85

$$recall = \frac{TP}{TP + FN}$$

F1 -Мера



F1-мера

комбинация точности и полноты в одну метрику

Пример:

правильное распознавание намерения пользователя. Насколько мы уверены в том, что правильно поняли? Надо ли уточнить?

F1 = 2 * 0.629 * 0.85 / (0.629 + 0.85) = 0.723

$$F = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

* из материалов к курсу Воронцова на Coursera

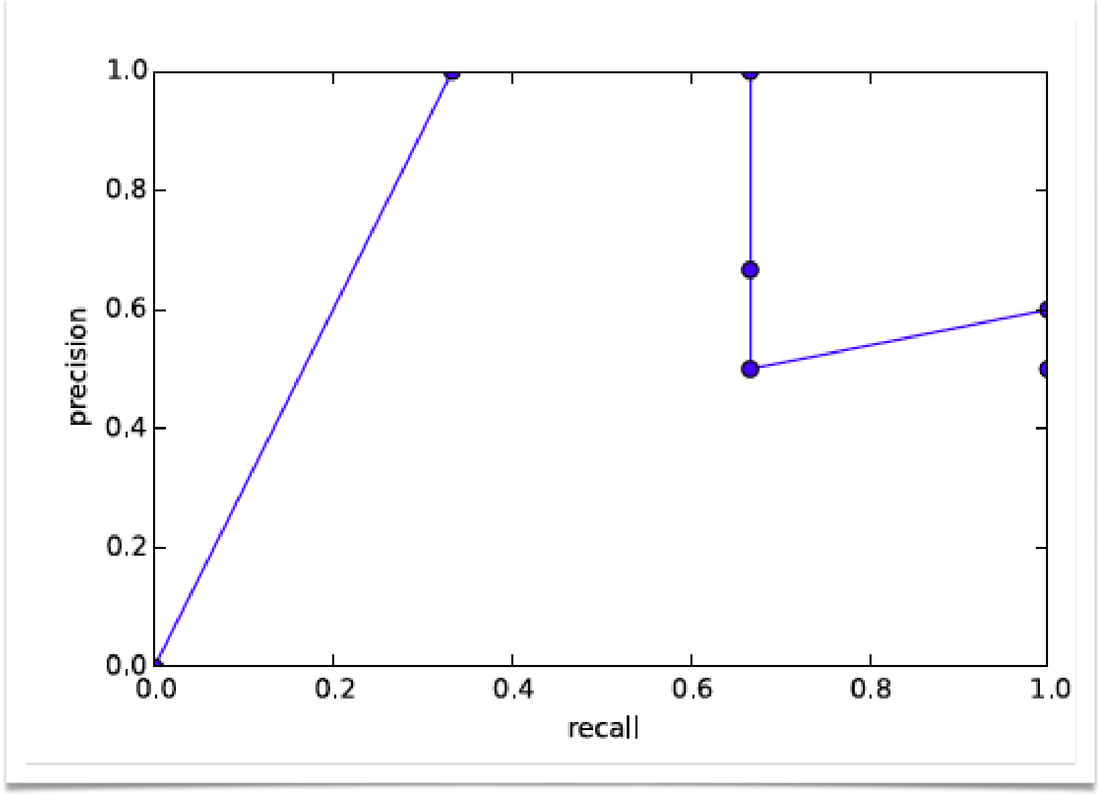
Going deeper

Классификаторы обычно работают в 2 этапа:

- оценка вероятности принадлежности к класс ама(x)
- выбор порога отсечения, при котором идет распределение в тот или иной класс

Это 2 отдельные задачи, после получения оценки вероятности можно отсортировать объекты и в различные периоды времени использовать разные пороги

AUC-PRC



Precision - Recall (PR curve)
мера качества разделения на классы

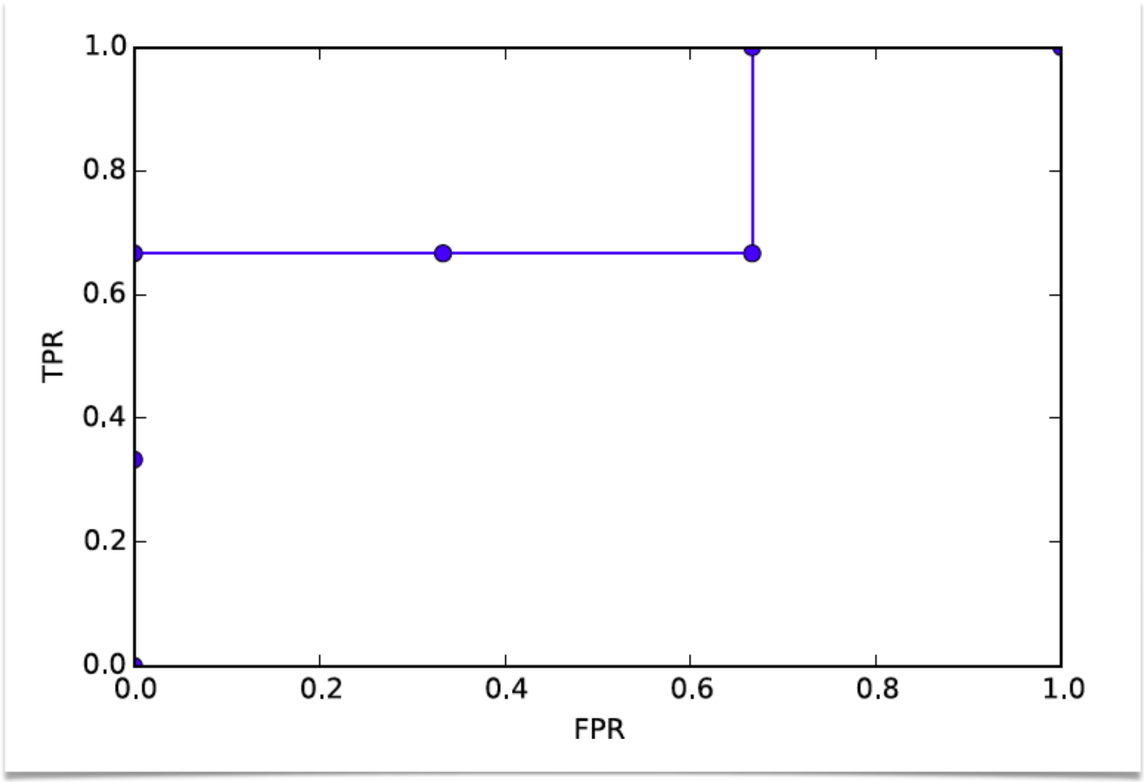
Построение:

1. Считаем вероятности принадлежности классу
2. Сортируем объекты по вероятности
3. Для каждого порога отсечения между объектами считаем precision и recall и последовательно наносим на график
4. Считаем площадь под кривой

AUC - PRC (area under curve: precision recall curve) итоговая метрика. Больше - лучше

* из материалов к курсу Воронцова на Coursera

AUC-ROC



Receiver Operating Characteristic curve
мера качества разделения на классы

Построение:
аналогично PR-кривой, но наносятся точки

FPR = FP / (FP + TN)
1-FPR - специфичность алгоритма
TPR = TP / (TP + FN)
TPR - чувствительность алгоритма
AUC - ROC
итоговая метрика. Больше - лучше

* из материалов к курсу Воронцова на Coursera

AUC-PRC VS AUC-ROC

- AUC-PRC лучше использовать в несбалансированных классах
- AUC-ROC можно использовать, когда алгоритм будет оцениваться на одних данных, но с разным соотношением классов

Многоклассовая классификация

Задача: a(x) ∈ {1, . . . ,K}
Сводится к K задачам отделения класса N от остальных

Как усреднить качество K задач?

Многоклассовая классификация

micro-averaging:

- вычислим confusion matrix для каждой задачи
- усредним по задачам
- вычислим итоговую метрику

классы делают вклад, пропорциональный размеру

macro-averaging:

- вычислим итоговую метрику для каждой задачи
- усредним по задачам

все классы делают равный вклад

Реализация BSKLEARN

sklearn.metrics

Some of these are restricted to the binary classification case:

`matthews_corrcoef` (y_true, y_pred[, ...])	Compute the Matthews correlation coefficient (MCC) for binary classes
`precision_recall_curve` (y_true, probas_pred)	Compute precision-recall pairs for different probability thresholds
`roc_curve` (y_true, y_score[, pos_label, ...])	Compute Receiver operating characteristic (ROC)

Others also work in the multiclass case:

`cohen_kappa_score` (y1, y2[, labels, weights])	Cohen's kappa: a statistic that measures inter-annotator agreement.
`confusion_matrix` (y_true, y_pred[, labels, ...])	Compute confusion matrix to evaluate the accuracy of a classification
`hinge_loss` (y_true, pred_decision[, labels, ...])	Average hinge loss (non-regularized)

Some also work in the multilabel case:

`accuracy_score` (y_true, y_pred[, normalize, ...])	Accuracy classification score.
`classification_report` (y_true, y_pred[, ...])	Build a text report showing the main classification metrics
`f1_score` (y_true, y_pred[, labels, ...])	Compute the F1 score, also known as balanced F-score or F-measure
`fbeta_score` (y_true, y_pred, beta[, labels, ...])	Compute the F-beta score
`hamming_loss` (y_true, y_pred[, labels, ...])	Compute the average Hamming loss.
`jaccard_similarity_score` (y_true, y_pred[, ...])	Jaccard similarity coefficient score
`log_loss` (y_true, y_pred[, eps, normalize, ...])	Log loss, aka logistic loss or cross-entropy loss.
`precision_recall_fscore_support` (y_true, y_pred)	Compute precision, recall, F-measure and support for each class
`precision_score` (y_true, y_pred[, labels, ...])	Compute the precision
`recall_score` (y_true, y_pred[, labels, ...])	Compute the recall
`zero_one_loss` (y_true, y_pred[, normalize, ...])	Zero-one classification loss.

And some work with binary and multilabel (but not multiclass) problems:

`average_precision_score` (y_true, y_score[, ...])	Compute average precision (AP) from prediction scores
`roc_auc_score` (y_true, y_score[, average, ...])	Compute Area Under the Curve (AUC) from prediction scores

Основные характеристики

19 функций

Схожий интерфейс: функции от y, y_pred

Пример использования f1_score

```
>>> from sklearn.metrics import f1_score
>>> y_true = [0, 1, 2, 0, 1, 2]
>>> y_pred = [0, 2, 1, 0, 0, 1]
>>> f1_score(y_true, y_pred, average='macro')
0.26...
>>> f1_score(y_true, y_pred,
average='micro')  0.33...
>>> f1_score(y_true, y_pred, average='weighted')
0.26...
>>> f1_score(y_true, y_pred,
average=None)  array([ 0.8, 0. , 0. ])
```

* из материалов к курсу Воронцова на Coursera

Практическое задание 3

Что мы сегодня узнали

Что мы сегодня узнали

1. Деревья решений, объединённые в «лес», составляют одни из наиболее сильных алгоритмов. По одиночке же они являются слабыми, зато очень легко интерпретируемыми и визуализируемыми алгоритмами
2. Деревья позволяют оценивать важность признаков
3. Метрик качества много, они разные по смыслу, для своих задач надо выбирать подходящую

Полезные материалы

Полезные материалы

1. Документация sklearn по деревьям
2. Open Data Science, habrahabr: Классификация, дерево решений и метод ближайших соседей
3. Лекция Евгения Соколова на ФКН ВШЭ по деревьям
   Конспект ; ipynb тетрадка
4. Метрики sklearn
5. Метрики kaggle
6. Объяснение метрик на курсе Coursera от Соколова & Воронцова

Спасибо за внимание!