Анализ данных.
Лекция. Неделя 1-2.
Машинное обучение. Классификация: кредитный скоринг.

Михайлов Даниил

28 сентября 2022 г.

Содержание

1 Предобработка данных

1.1 Изучение исходных данных

- Размер train и test выборки.
- Проверка однородности train и test выборки - дисперсия и мат.ожидание.
- Исследуемая переменная: какой тип данных, какие значения принимает, какое распределение?

1.2 Подготовка данных

- Работа с пропущенными значениями. Их подсчет и замена на среднее, моду. Также можно заменять на нетипичное для остальной выборки значение (например, -99999) - в случае если пропуск в данных является дополнительной информацией. В таком случае возможно добавление новой переменной - есть ли пропуск в данных или нет.
- Из данных можно извлечь дополнительные признаки: длина строки, частота встречи в выборке, и т.д.
- Из переменных с форматом даты можно извлечь следующие признаки: время года, год, месяц, день, утро/вечер, является ли день выходным или праздником, и т.д.
- Перевод категориальных признаков в признаки, принимающие значения 0 и 1 - one hot кодирование.

gender
female
male
female

$\longrightarrow$

male	female
0	1
1	0
0	1

- Стандартизация данных. Приведение к нулевому мат.ожиданию и единичной дисперсии. Таким образом, делаем разницу двух значений у разных параметров сравнимой. (Метрические алгоритмы чувствительны к нормировке, в отличие от алгоритмов, построенных на деревьях).
- Добавление признаков, которые являются перемножением уже существующих - для поиска квадратичной зависимости (Но надо помнить о "проклятии размерности").
- Оптимизация хранения данных. Перевод бинарных признаков в int8.
- Сохранение данных с их форматами.

2 Построение алгоритмов

- Деление выборки на обучающую и тестовую (train-test split). На первой мы будем строить и тренировать модель, на второй - валидировать.

2.1 Метрики для бинарной классификации

- **log_loss**.
  Пусть y_{true} - реальное значение (0 или 1), а y_{pred} - предсказанная нами вероятность принятия значения 1.
  Тогда
  $$log\_loss = y_{true} * log(y_{pred}) + (1 - y_{true}) * log(1 - y_{pred})$$
  log_loss для всей выборки рассчитывается, как среднее log_loss по каждому измерению.
  Чем ниже log_loss у модели, тем она точнее.
  Для многоклассовой классификации можно использовать multi class log loss.

- Confusion Matrix.
  Перед построением матрицы и использованием дальнейших метрик необходимо подобрать порог, начиная с которого мы будем предсказывать 1. Например, 0.5. Тогда набор полученных моделью значений вероятностей (0.64, 0.38, 0.86, 0.12) превратиться в предсказания (1, 0, 1, 0).

Предсказания → / Реальность ↓	0	1
0	True Negative	False Positive
1	False Negative	True Positive

- **Accuracy**.
  $$(TN + TP)/(TN + TP + FN + FP)$$
  То есть отношение верно сделанных предсказаний ко всем.

- **Precision**.
  $$TP/(TP + FP)$$
  Какая доля единиц, которые мы предсказали, верна.

- **Recall**.
  $$TP/(TP + FN)$$
  Какую часть настоящих единиц мы покрыли нашими предсказаниями?

- **f1 score**.
  $$(2 * Precision * Recall)/(Precision + Recall)$$
  Баланс между Precision и Recall.

- **ROC curve**.
  Процесс построения:

  1) Сортируем значения по предсказанной нами вероятности

  2) Получаем вектор реальных значений в определенном порядке

Модель	True
0.4	1
0.2	0
0.1	0
0.5	1
0.6	0
0.7	1

$\longrightarrow$

Модель	True
0.1	0
0.2	0
0.4	1
0.5	1
0.6	0
0.7	1

$\longrightarrow \quad \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$

3) Идем по порядку значений в этом векторе. Ноль означает сдвиг вверх на один шаг, единица - сдвиг вправо на один шаг.

4) Получаем ступенчатую фигуру. ROC AUC score - площадь под этой кривой.

Идеальный случай, когда у нас сначала все нули, потом все единицы - тогда площадь равна единице. В обратном случае она равну нулю.
Случайным предсказаниям соответствует ROC AUC равный примерно 0.5.

- **top k Precision**.
Мы сортируем предсказания по вероятности (так же, как для ROC), и берем только k предсказаний с самой высокой вероятностью, на которых будем использовать Precision.

  Пример задач с лекции:

  - Для зенитчика нужно определить, дружеский (0) или вражеский (1) самолет в небе, какую метрику оптимизировать?

    Ответ: Precision, так как подбить дружеский самолет хуже, чем не подбить вражеский.

  - Нужно построить алгоритм, который будет искать мошеннические (1) транзакции среди обычных (0) для отправки их на дополнительную проверку. Какую метрику оптимизировать?

    Ответ: Recall (полнота), так как важно найти все плохие транзакции; то, что будут лишний раз проверены некоторые обычные не так важно.

  - Мы хотим отправить смс сообщения тем 1000 клиентам из нашей базы в 10000000 людей, которые с наибольшим шансом откликнуться на них - какую метрику оптимизировать?

    Ответ: Roc_auc - тогда отправим топ 1000 после сортировки по вероятностям. Аналогично можно поступить с top_k_precision.

2.2 Бинарная классификация

2.2.1 Предсказание константой

- В качестве предсказаний мы можем всегда давать одно и то же число: например, 0.5 при бинарной классификации (0 или 1).
- Поиск лучшей константы зависит от метрики качества и баланса классов в выборке.
- Константа хорошо подходит в качестве базового решения, с которым будут сравниваться последующие решения.

2.2.2 Модели

- Для осуществления бинарной классификации можно использовать модели Логистической регрессии, Решающего дерева или kNN (Ближайшие соседи).
- Последний, в отличие от остальных является метрическим, а не параметрическим алгоритмом. Это значит, что у него нет параметров и он ничего не обучает (как, например, логистическая регрессия в процессе обучения находит параметры - коэффициенты) - kNN имеет только гиперпараметры - то есть те, которые задаются нами, а не определяются моделью при оптимизации.
- Помимо этого, для kNN очень важно, чтобы исходные данные были приведены к нормальному виду (нулевое мат.ожидание и единичная дисперсия).

2.3 Кросс-валидация

- Разбиваем выборку на сколько-то частей [X1, X2, X3, X4]; обучаемся на [X1, X2, X3] и смотрим качество на [X4], - повторяем для всех возможных комбинаций и определяем среднюю точность.
- Используя кросс-валидацию, можно подбирать гиперпараметры модели - то есть те, которые задаются нами, а не определяются моделью при оптимизации.
- Просто перебрав какое-то количество возможных гиперпараметров в кроссвалидации, выберем тот, при котором точность максимальна.

3 Blending

- усреднение результатов нескольких моделей может привести к улучшению их точности.
- Пусть имеем две модели с предсказаниями y_{pred_1} и y_{pred_2}.
  Тогда
  $$y_{pred} = \alpha * y_{pred\_1} + (1 - \alpha) * y_{pred\_2}$$
  Оптимальный параметр альфа можно найти с помощью кросс-валидации.

4 Pipeline

- Pipeline - это алгоритм, последовательность действий, которая полностью преобразует исходные данные в предсказания.
- В pipeline мы сразу же подставляем подобранные гиперпараметры, с которыми модели работали точнее всего.