

Пропущенные значения.  
Обработка текстов.

Артём Филатов

Пропущенные значения.




# Типы пропусков

- › Пропуск появился абсолютно случайно.
- › Вероятность появления пропуска зависит от наблюдаемых признаков.
- › Вероятность появления пропуска зависит от всех величин.

# Простые методы

- › Удалить все наблюдения/признаки с пропущенными значениями.
- › Заменить пропущенные значения на среднее/моду/медиану/ноль.



Данные методы сильно смещают выборку и могут негативно сказаться на качестве модели.

# Продвинутые методы

■ Воспользуемся SVD!

- › Заполним пропуски средним и применим SVD.
- › Заполним пропуски  $k$ -ранговой аппроксимацией.
- › Повторим шаги.

# Продвинутые методы

Идея: посмотреть на ближайших соседей и взять среднее.

- › Для каждого объекта необходимо найти  $k$ -ближайших соседей без пропущенных признаков.
- › После этого мы можем усреднить пропущенные значения по соседям.

# Продвинутые методы

Построим модель по признакам без пропущенных значений и будем предсказывать пропущенные значения.

› Высокое качество, особенно с продвинутыми моделями.

Обработка текстов.  
Категориальные  
признаки.





# Мотивация

Достаточно часто возникает необходимость построения моделей над текстами.

- › Siri
- › Машинный перевод
- › Генерация текста
- › Чат-боты

# Задача

Нам дан корпус текстов  $X$ , где каждое наблюдение это отдельный текст. Требуется предсказать `target` для каждого текста.

› Оценку качества будем производить стандартными методами.

# Bag of Words

Самым 'наивным' подходом является мешок слов:

- › Создаем колонку для каждого слова.
- › Вставляем единицу, если слово встречается в тексте.
- › Обучаем модель по построенному датасету.

# Hello, I am student.

Для данного текста мы получим 4 признака, где в каждом будет стоять единица.

- › hello
- › I
- › am
- › Student

Но мы не учитываем контекст!

# N-grams

- › Возьмем теперь не каждое слово, а пары, тройки и т.д.
- › Получим N-граммную модель, где N – это максимальная длина последовательности слов.

# Hello, I am student.

Рассмотрим 2-граммную модель. Получим следующие признаки:

- › hello I
- › I am
- › am student
- › + bag of words


# TF-IDF

Вместо бинарного значения будем приписывать каждому слову вес в документе.

- › НО! Сделаем это по умному: учтём также, как часто слово встречается вообще.

# TF-IDF

- › TF (Term Frequency) : логарифм количества слова  $w$  в документе  $d$ .
- › IDF (Inverse Document Frequency) : логарифм обратной доли документов, в которых встретилось слово.


$$\text{TF-IDF} = \text{TF} * \text{IDF}$$



# Косинусная мера

Сопоставив текстам векторы, мы можем мерить похожесть текстов.

- › Вычислим косинус угла между векторами двух текстов.
- › Чем меньше значение, тем больше похожесть текстов.

# Hashing trick



# ЛикБез

В компьютерных науках хеш-функцией называется функция (как правило случайная), отображающая вход (слово, число, объект) в заданный интервал.

Пример:  $(ax + b) \% p \% m$

Применение: алгоритмы и структуры данных, криптография

# Hashing trick

- › Вычислим хэш от всех N-грамм и перезапишем их в соответствующий столбец.
- › Уменьшает количество признаков.
- › Каждый столбец теперь является представлением ни одной, а нескольких N-грамм.

Что дальше?



# Natural Language Processing

- › Представление слов в виде векторов: word2vec, авто-кодировщики.
- › Языковые модели: как построить вероятностное распределение над корпусом?
- › Как определить язык текста?
- › Как перевести текст?