Билет 9

1. Сбор и подготовка данных. Виды. Источники данных.

   Подготовка данных включает такие процессы:

   - получение,
   - очистка,
   - нормализация,
   - превращение в оптимизированный набор данных.
   - Обычно это табличная форма, которая идеально подходит для этих методов, что были запланированы на этапе проектировки.

   Источник данных – это контейнер, содержащий данные, которые вы загружаете в Аналитику. Источники данных определяют, как загруженная информация объединяется с существующей. Источники настраиваются на уровне ресурса. Загружать данные из одного и того же источника данных можно многократно.

   Примеры: Радио, телепередачи, газеты, журналы, выставки, конференции, сайты Интернета.

2. Три основных этапа сбора и подготовки данных

Нормирование

Нормировка – это корректировка значений в соответствии с некоторыми функциями преобразования, с целью сделать их более удобными для сравнения.

Нормировка данных требуется, когда несовместимость единиц измерений переменных может отразиться на результатах и рекомендуется, когда итоговые отчеты могут быть улучшены, если выразить результаты в определенных понятных/совместимых единицах. Например, время реакции, записанное в миллисекундах, легче интерпретировать, чем число тактов процессора, в которых были получены данные эксперимента.

Например, разделив набор измерений о росте людей в дюймах на 2.54, мы получим значение роста в метрической системе.

Форматирование

Операции **форматирования** включают в себя выбор выравнивания абзацев, отступов и отбивок между абзацами, обтекания отдельных абзацев, а также видов и начертаний шрифтов. Эти операции выполняются различными текстовыми процессорами с разной степенью автоматизации.

Для удобства просмотра данных можно установить отображение отрицательных чисел красным цветом. Знак минус (-) при этом можно отображать или не отображать.

Кодирование – это представление категориальных данных в числовой форме. Например, при бинарной классификации один из классов можно представить числом `0`, а другой класс – числом `1`.

Кодирование

LabelEncoder

1. Категориальная особенность является порядковой (например, низкий, средний, высокий)
2. Количество категорий довольно велико, и **One-Hot-Encoding** может привести к высокому потреблению памяти.

Этот подход очень прост и включает преобразование каждого значения в столбце в число. Рассмотрим набор данных, в котором описываются категориальные признаки мостов:

Тип моста
Arch
Beam
Truss
Cantilever
Tied Arch
Suspension
Cable

Теперь закодируем текстовые значения. Позиция текстового значения в наборе данных будет являться его числовым значением:

Тип моста (Текст)	**Тип моста (Числовое значение)**
Arch	0
Beam	1
Truss	2
Cantilever	3
Tied Arch	4
Suspension	5
Cable	6

Проблема использования такого метода состоит в том, что они вводят сравнение между ними. Нет никакой связи между различными типами мостов, но, глядя на эти числа, можно подумать, что тип моста «Cable» имеет более высокий приоритет над типом моста «Arch». Алгоритм может неправильно понимать, что данные имеют какой-то порядок 0 <1 <2… <6 и могут придать «Cable» в 6 раз больший вес, чем типу моста «Arch».

Уровень безопасности (Текстовое значение)	**Уровень безопасности (Числовое значение)**
Никакой	0
Низкий	1
Средний	2
Высокий	3

Уровень безопасности (Текстовое значение)	Уровень безопасности (Числовое значение)
Очень высокий	4

Кодирование Label-Encoder в этом столбце также вызывает приоритет по числу, но правильным способом. Здесь числовой порядок не выглядит нестандартным, и будет иметь смысл, если алгоритм интерпретирует порядок безопасности 0 < 1 < 2 < 3 < 4, т. е. никакой <низкий <средний <высокий <очень высокий.

One-Hot Encoder

1. Категориальная особенность не является порядковой (как типы мостов в примере)
2. Небольшое количество категорий.

Хотя **Label-Encoder** является прямолинейным, но у него есть недостаток, заключающийся в том, что числовые значения могут быть неверно истолкованы алгоритмами как имеющие некоторый порядок. Эта проблема упорядочения решается с помощью другого распространенного альтернативного подхода, называемого «One-Hot-Encoding». В этом алгоритме каждое значение категории преобразуется в новый столбец, и столбцу присваивается значение `1` или `0` (обозначение `true` / `false`). Давайте рассмотрим предыдущий пример типов мостов и уровней безопасности с **One-Hot-Encoding**.

Тип моста (Текст)	Тип моста (Arch)	Тип моста (Beam)	Тип моста (Truss)	Тип моста (Cantileve)	Тип моста (Tied Arch)	Тип моста (Suspension)	Тип моста (Cable)
Arch	1	0	0	0	0	0	0
Beam	0	1	0	0	0	0	0
Truss	0	0	1	0	0	0	0
Cantilever	0	0	0	1	0	0	0
Tied Arch	0	0	0	0	1	0	0
Suspension	0	0	0	0	0	1	0
Cable	0	0	0	0	0	0	1

Уровень безопасности (Текст)	Уровень безопасности (Никакой)	Уровень безопасности (Низкий)	Уровень безопасности (Средний)	Уровень безопасности (Высокий)	Уровень безопасности (Очень высокий)
Никакой	1	0	0	0	0
Низкий	0	1	0	0	0
Средний	0	0	1	0	0
Высокий	0	0	0	1	0
Очень высокий	0	0	0	0	1

Строки с первым значением столбца (Arch/Никакой) будут иметь значение `1` (указывает на истинное значение), а столбцы других значений будут иметь значение `0` (указывающее на ложное значение). Аналогично для других строк сопоставляется значение со значением столбца.

Хотя этот подход устраняет проблемы порядка, но имеет недостаток в добавлении большего количества столбцов в набор данных. Это может привести к значительному увеличению количества столбцов, если у вас есть много уникальных значений в столбце категории. В приведенном выше примере это было управляемо, но управлять им будет действительно сложно, когда кодирование дает много столбцов

Практика

```
class DataAnalysis:
    '''
    Даны три целых числа. Выбрать из них те, которые принадлежат интервалу [1, 3].
    '''

    def __init__(self, lowerLimit=None, upperLimit=None, *nums):
        self._lowerLimit = lowerLimit
        self._upperLimit = upperLimit
        self._num1, self._num2, self._num3 = nums
        self._numsInBetween = []

    def getNumsInBetween(self):
        for num in [self._num1, self._num2, self._num3]:
            if num >= 1 and num <= 3:
                self._numsInBetween.append(num)
        return self._numsInBetween


task = DataAnalysis(1, 3, 1, 4, 6)
print(task.getNumsInBetween())
```