Предобработка данных

Синтаксис

Метод set_axis() для изменения названий столбцов

In
```
df.set_axis(['a','b','c'],axis = 'columns',inplace = True)

# аргументы – список новых названий столбцов,
# axis со значением columns для изменений в столбцах,
# inplace со значением True для изменения структуры данных
```

Методы isnull() и isna() для определения пропущенных значений

In
```
df.isnull()
df.isna()

# В сочетании с методом sum() –
# подсчёт пропущенных значений
df.isnull().sum()
df.isna().sum()
```

Метод fillna() для заполнения пропущенных значений

In
```
df = df.fillna(0)
# аргумент – значение, на которое
# будут заменены пропущенные значения
```

Метод dropna() для удаления пропущенных значений

In
```
df.dropna()
# удаление всех строк, где есть
# хотя бы одно пропущенное значение
```

In
```
df.dropna(subset = ['a','b','c'],
inplace = True)
# аргумент subset – названия столбцов,
# где нужно искать пропуски
```

In
```
df.dropna(axis = 'columns',
inplace = True)
# аргумент axis со значением 'columns'
# для удаления столбцов с хотя бы
# одним пропущенным значением
```

Метод duplicated() для нахождения дубликатов

In
```
df.duplicated()

# В сочетании с методом sum() –
# возвращает количество дубликатов
df.duplicated().sum()
```

Метод unique() для просмотра всех уникальных значений в столбце

In
```
df['column'].unique()
```

Метод drop_duplicates() для удаления дубликатов

In
```
df.drop_duplicates().reset_index(drop
 = True)
# аргумент drop со значением True,
# чтобы не создавать столбец со
# старыми значениями индексов

'''
При вызове метода drop_duplicates()
вместе с повторяющимися строками
удаляются их индексы, поэтому
используется с методом reset_index()
'''
```

Метод replace() для замены значений в таблице или столбце

In
```
df.replace('first_value', 'second_value')

# первый аргумент – текущее значение
# второй аргмуент – новое значение
```

Словарь

Предобработка
Процесс подготовки данных для дальнейшего анализа. Его суть заключается в поиске и устранении возможных ”проблем”в данных

GIGO (от англ. garbage in — garbage out, буквально «мусор на входе — мусор на выходе»)
Принцип, утверждающий, что при неверных входных данных даже правильный алгоритм анализа выдаёт неверные результаты

Таблица, удобная для анализа данных:

- в каждом столбце хранятся значения одной переменной
- каждая строка содержит одно наблюдение, к которому привязаны значения разных переменных

Названия столбцов:

- без пробелов в начале, середине и конце
- несколько слов разделяются нижним подчеркиваем
- на одном языке и в одном регистре
- отражают в краткой форме, какого рода информация содержится в каждом столбце

Пропущенные значения бывают разные:

- чаще всего это **None** или **NaN**
- плейсхолдеры (тексты-заполнители) какого-то общепринятого стандарта, иногда неизвестного вам, но которого придерживаются составители. Чаще всего это **n/a**, **na**, **NA**, и **N.N.** либо **NN**
- произвольное значение, которое по договорённости между собой используют создатели исходной таблицы данных

Пропущенные значения можно как удалять, так и заполнять на основе известных данных:

- плюс удаления данных в том, что это простой процесс. Также можно быть уверенным, что те данные, которые остались, хорошие и отвечают всем требованиям. Потенциальные минусы: потеря важной информации и снижение точности
- заполнение позволяет сохранить наибольшее количество данных. Очевидный минус — могут получиться плохие результаты на основе уже существующих данных

Дубликаты (дублированные записи)
могут быть следующего вида:

- две и более одинаковых строки с идентичной информацией. Большое количество повторов раздувает размер таблицы, а значит, увеличивает время обработки данных
- одинаковые по смыслу категории с разными названиями, например, «Политика» и «Политическая ситуация». Замаскированные повторы — источник серьёзных и с трудом обнаруживаемых ошибок в анализе