Data preprocessing

Syntax

The set_axis() method modifying column names

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

# arguments are a list of new column names,
# axis with the 'columns' value for changes in columns,
# 'inplace' with the value True for changes to the data structure
```

The isnull() и isna() methods for finding missing values

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

# delete all rows containing –
# at least one missing value
df.isnull().sum()
df.isna().sum()
```

The fillna() method for filling in missing values

In
```
df = df.fillna(0)
# the argument with the new value that
# will replace all the missing values
```

The dropna() method for deleting missing values

In
```
df.dropna()
# delete all rows containing
# at least one missing value
```

In
```
df.dropna(subset = ['a','b','c'],
inplace = True)
# the 'subset' argument is the names
# of the columns, in which you need
# to find missing values
```

In
```
df.dropna(axis = 'columns',
inplace = True)
# axis argument with the value ‘columns’
# for deleting columns with at least
# one missing value
```

The duplicated() method for finding duplicates

In
```
df.duplicated()

# Along with the method sum() – returns
# the total number of duplicates
df.duplicated().sum()
```

The unique() method for seeing all unique values in a column

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

The drop_duplicates() method for finding duplicates

In
```
df.drop_duplicates().reset_index(drop
 = True)
# the argument drop with the value True,
# so you avoid creating a column with
# old index values

'''
When calling the method drop_duplicates()
along with repeating rows, their indices
are deleted, at which point the method
reset_index() is used.
'''
```

The replace() method for replacing values in a table or column:

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

# the first argument is the current value
# the second argument is the new value
```

Glossary

Preprocessing
Preparing data for subsequent analysis. The idea is to find and eliminate potential problems in the data.

GIGO (GIGO (garbage in, garbage out))
The principle that when you have poor input data , even the best analytical algorithm will return poor results.

Таблица, удобная для анализа данных:

- each column stores the values for one variable
- each row contains one observation the values, for different variables are tied to

Column names

- without spaces at the beginning, at the end, or in the middle
- multiple words are separated by underscores
- in the same language and case
- briefly describe the kind of information each column contains

Missing values can be deleted or filled in using available data:

- most often, they're **None** or **NaN**
- Placeholders of a generally accepted standard, sometimes one you don't know about, but which the compilers stick to. Most often, they're **n/a**, **na**, **NA**, и **N.N.** либо **NN**
- a random value the creators of a source data table have decided to use.

Missing values can be deleted or filled in using available data:

- the upside to deleting them is that it's a simple process. That also makes sure that the remaining data is clean and matches all the requirements. Potential downsides: losing important information and reducing accuracy.
- filling in missing values lets you save the most data. An obvious drawback is that you can get poor results based on existing data.

Duplicates (There are different kinds) of duplicates:

- two or more rows containing identical information. Lots of repetitions pad out tables, forcing us to spend more time processing data.
- categories with different names but identical subject matters (Politics and Political Situation, for instance). Disguised repetitions can cause serious roadblocks for analysis that are difficult to pinpoint.