Министерство науки и высшего образования Российской Федерации
Российский химико-технологический университет имени Д. И. Менделеева

Факультет цифровых технологий и химического инжиниринга
Кафедра информационных компьютерных технологий

**ОТЧЕТ**

ПО ЗАЧЕТНОЙ РАБОТЕ

**«Информационные технологии»**

# Библиотека Pandas

**ВЫПОЛНИЛ:** студент группы КС-20 Шишко Р.Г.

**ПРОВЕРИЛ:** ст. преподаватель Васецкий А. М.

**Москва**
**2020**

# Библиотека Pandas

Цель данной зачетной работы — изучить библиотеку и на практических примерах программного кода проиллюстрировать её использование. Использоваться будет библиотека Pandas. Pandas - библиотека, ориентированная на обработке и анализе данных, поэтому для иллюстрации понадобятся исходные данные для анализа.

Мы будем использовать три датасета для разных целей:

- Данные о пассажирах Титаника, обработав которые через Pandas, в связке с библиотекой sklearn, мы сможем попробовать предсказать вероятность выживания для заданного пассажира, используя машинное обучение
- Данные об исследуемой экосистеме в южной Аризоне, которые мы обработаем и визуализируем в связке с библиотекой matplotlib/seaborn
- Данные о книгах, находящихся в Британской Библиотеке, в которых нужно будет откорректировать некачественные/неверные данные

Краткий список команд, использованных в данной работе:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| read_csv | head | shape | info | groupby | sum | sort_values | tail |
| mean | describe | index | loc | value_counts | sample | to_datetime | dropna |
| fillna | unique | nunique | crosstab | to_numeric | isnull | drop | median |

Для импорта данных будет использоваться функция **read_csv()**, которая принимает путь к файлу, а также может принимать параметры: - *sep* — отвечает за разделитель, используемый в документе - *index_col* — отвечает за номер колонки, которая будет использоваться для индексации данных - *skip_blank_lines* — отвечает за пропуск пустых строк, принимает значение True/False - *encoding* — отвечает за кодировку файла, принимает её имя ...и прочие;

*read_csv ( ), head ( )*

*Сама функция read_csv() возвращает объект типа dataframe, который является табличной структурой данных (каждая строка/столбец является объектом типа Series), а метод .head() отвечает за вывод шапки исходных данных*

## Возьмем первый массив данных

*Этот датасет содержит данные о всех пассажирах Титаника, находящихся на борту во время катастрофы*

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn.ensemble as ensemble
import sklearn.model_selection as model_selection
import joblib as joblib
import seaborn
```

```
dataset = pd.read_csv('./data/1_titanic/titanic.csv', encoding='utf-8', index_col=[1], parse_dates=True)
dataset.head()
```

| row.names | id | pclass | survived | name | age | embarked | home.dest | room | ticket | boat | sex |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **999** | 998 | 3rd | 1 | McCarthy, Miss Katie | NaN | NaN | NaN | NaN | NaN | NaN | female |
| **180** | 179 | 1st | 0 | Millet, Mr Francis Davis | 65.0 | Southampton | East Bridgewater, MA | NaN | NaN | (249) | male |
| **557** | 556 | 2nd | 0 | Sjostedt, Mr Ernst Adolf | 59.0 | Southampton | Sault St Marie, ON | NaN | NaN | NaN | male |
| **175** | 174 | 1st | 0 | McCaffry, Mr Thomas Francis | 46.0 | Cherbourg | Vancouver, BC | NaN | NaN | (292) | male |
| **1233** | 1232 | 3rd | 0 | Strilic, Mr Ivan | NaN | NaN | NaN | NaN | NaN | NaN | male |

*.shape*

*Возвращает размерность массива данных:*

```
dataset.shape
```

```
(919, 11)
```

*Датасет является таблицей, состоящей из 919 строк и 11 столбцов*

Мы также можем получить более подробную информацию о типах данных, как всего массива данных, так и конкретных столбцов:

```
type(dataset)
```

```
pandas.core.frame.DataFrame
```

```
type(dataset['name'])
```

```
pandas.core.series.Series
```

*.info( )*

*Возвращает данные по столбцам:*

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 919 entries, 999 to 671
Data columns (total 11 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   id         919 non-null    int64
 1   pclass     919 non-null    object
 2   survived   919 non-null    int64
 3   name       919 non-null    object
 4   age        439 non-null    float64
 5   embarked   571 non-null    object
 6   home.dest  531 non-null    object
 7   room       48 non-null     object
 8   ticket     46 non-null     object
 9   boat       233 non-null    object
 10  sex        919 non-null    object
dtypes: float64(1), int64(2), object(8)
memory usage: 86.2+ KB
```

*.drop( )*

*Удаляет столбец массива:*

***Удалим столбец "id", дабы оставить только индексы, которые указывают на номер каюты:***

```
dataset = dataset.drop('id', 1)
dataset.head(5)
```

| row.names | pclass | survived | name | age | embarked | home.dest | room | ticket | boat | sex |
|---|---|---|---|---|---|---|---|---|---|---|
| **999** | 3rd | 1 | McCarthy, Miss Katie | NaN | NaN | NaN | NaN | NaN | NaN | female |
| **180** | 1st | 0 | Millet, Mr Francis Davis | 65.0 | Southampton | East Bridgewater, MA | NaN | NaN | (249) | male |
| **557** | 2nd | 0 | Sjostedt, Mr Ernst Adolf | 59.0 | Southampton | Sault St Marie, ON | NaN | NaN | NaN | male |
| **175** | 1st | 0 | McCaffry, Mr Thomas Francis | 46.0 | Cherbourg | Vancouver, BC | NaN | NaN | (292) | male |
| **1233** | 3rd | 0 | Strilic, Mr Ivan | NaN | NaN | NaN | NaN | NaN | NaN | male |

*.groupby( )*

*Группирует данные в таблице по значению индекса:*

*.sum( )*

*Суммирует данные в таблице по значению индекса:*

*.sort_values( )*

*Сортирует данные:*

***Посмотрим, какое количество пассажиров выжило в каждом классе:***

```
dataset.groupby(['pclass'])['survived'].sum().sort_values()
```

```
pclass
2nd     86
3rd     97
1st    132
Name: survived, dtype: int64
```

*.value_counts( )*

*Подсчитывает какое количество раз встречалось то или иное значение:*

***Выведем общее количество пассажиров по классам***

```
dataset['pclass'].value_counts().head(5)
```

```
3rd    494
1st    228
2nd    197
Name: pclass, dtype: int64
```

*.tail( )*

*Возвращает последние элементы массива данных*

***Выделим в отдельный датасет всех мужчин:***

```
men = dataset[dataset['sex'] == 'male']
men.tail(3)
```

| row.names | pclass | survived | name | age | embarked | home.dest | room | ticket | boat | sex |
|---|---|---|---|---|---|---|---|---|---|---|
| **1267** | 3rd | 0 | Van Billiard, Mr Austin Blyler | NaN | NaN | NaN | NaN | NaN | NaN | male |
| **1247** | 3rd | 0 | Thomas, Mr Charles | NaN | NaN | NaN | NaN | NaN | NaN | male |
| **671** | 3rd | 0 | Bengtsson, Mr John Viktor | 26.0 | Southampton | Krakudden, Sweden Moune, IL | NaN | NaN | NaN | male |

*.mean( )*

*Возвращает средние значения:*

```
men.mean()
```

```
survived     0.182566
age         31.064448
dtype: float64
```

***Заметим, что выжило всего 18% мужчин, а средний возраст мужчин был 31***

*.describe( )*

*Возвращает численную статистику массива данных*

```
men['age'].describe()
```

```
count    278.000000
mean      31.064448
std       15.085240
min        0.333300
25%       22.000000
50%       28.500000
75%       40.000000
max       71.000000
Name: age, dtype: float64
```

***Максимальный возраст был 71 год, а минимальный был 4 месяца***

*.index*

*Возвращает значения, по которым индексирован наш массив данных.*

```
men.index
```

```
Int64Index([ 180,  557,  175, 1233,  731,  118,  755,  162,  535,  337,
             ...
             457,  431, 1148,  355, 1212,  825,  635, 1267, 1247,  671],
            dtype='int64', name='row.names', length=608)
```

***Так как наши данные проиндексированы по номеру каюты, можем вывести данные, по этому номеру:***

*.loc[ ]*

*Ищет элементы массива данных по индексу*

```
men.loc[180]['age']
```

```
65.0
```

*Как мы видим, пассажиру из 180-ой каюты было 65 лет*

***Также Pandas позволяет как добавлять:***

```
dataset['Column-1'] = 'Custom Column'
dataset.head(10)
```

| row.names | pclass | survived | name | age | embarked | home.dest | room | ticket | boat | sex | Column-1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 999 | 3rd | 1 | McCarthy, Miss Katie | NaN | NaN | NaN | NaN | NaN | NaN | female | Custom Column |
| 180 | 1st | 0 | Millet, Mr Francis Davis | 65.0 | Southampton | East Bridgewater, MA | NaN | NaN | (249) | male | Custom Column |
| 557 | 2nd | 0 | Sjostedt, Mr Ernst Adolf | 59.0 | Southampton | Sault St Marie, ON | NaN | NaN | NaN | male | Custom Column |
| 175 | 1st | 0 | McCaffry, Mr Thomas Francis | 46.0 | Cherbourg | Vancouver, BC | NaN | NaN | (292) | male | Custom Column |
| 1233 | 3rd | 0 | Strilic, Mr Ivan | NaN | NaN | NaN | NaN | NaN | NaN | male | Custom Column |
| 816 | 3rd | 1 | Georges, Mrs Shahini Weappi | NaN | Cherbourg | Youngstown, OH | NaN | NaN | NaN | female | Custom Column |
| 1172 | 3rd | 0 | Sage, Miss Constance | NaN | NaN | NaN | NaN | NaN | NaN | female | Custom Column |
| 1220 | 3rd | 1 | Smyth, Miss Julia | NaN | NaN | NaN | NaN | NaN | NaN | female | Custom Column |
| 731 | 3rd | 0 | Connors, Mr Patrick | NaN | Queenstown | NaN | NaN | NaN | (171) | male | Custom Column |
| 118 | 1st | 1 | Goldenberg, Mr Samuel L. | 49.0 | Cherbourg | Paris, France / New York, NY | NaN | NaN | 5 | male | Custom Column |

***...так и удалять колонки:***

```
dataset = dataset.drop('Column-1', 1)
dataset.head(10)
```

| row.names | pclass | survived | name | age | embarked | home.dest | room | ticket | boat | sex |
|---|---|---|---|---|---|---|---|---|---|---|
| 999 | 3rd | 1 | McCarthy, Miss Katie | NaN | NaN | NaN | NaN | NaN | NaN | female |
| 180 | 1st | 0 | Millet, Mr Francis Davis | 65.0 | Southampton | East Bridgewater, MA | NaN | NaN | (249) | male |
| 557 | 2nd | 0 | Sjostedt, Mr Ernst Adolf | 59.0 | Southampton | Sault St Marie, ON | NaN | NaN | NaN | male |
| 175 | 1st | 0 | McCaffry, Mr Thomas Francis | 46.0 | Cherbourg | Vancouver, BC | NaN | NaN | (292) | male |
| 1233 | 3rd | 0 | Strilic, Mr Ivan | NaN | NaN | NaN | NaN | NaN | NaN | male |
| 816 | 3rd | 1 | Georges, Mrs Shahini Weappi | NaN | Cherbourg | Youngstown, OH | NaN | NaN | NaN | female |
| 1172 | 3rd | 0 | Sage, Miss Constance | NaN | NaN | NaN | NaN | NaN | NaN | female |
| 1220 | 3rd | 1 | Smyth, Miss Julia | NaN | NaN | NaN | NaN | NaN | NaN | female |
| 731 | 3rd | 0 | Connors, Mr Patrick | NaN | Queenstown | NaN | NaN | NaN | (171) | male |
| 118 | 1st | 1 | Goldenberg, Mr Samuel L. | 49.0 | Cherbourg | Paris, France / New York, NY | NaN | NaN | 5 | male |

## Производим обработку датасета:

*По своей сути, данные делятся на бинарные, качественные и количественные: например, в нашем случае, качественными данными являются: номер каюты, пол пассажира и его имя, а количественным будет его возраст. Для обучения модели нам не понадобится имя пассажира, так как оно никак не влияет на шансы на выживание. Мы будем использовать колонки "класс", "пол" и "возраст", однако:*

1. У нас не хватает данных по возрасту, поэтому для пассажиров, у которых не указан возраст, укажем его как медианный для этой выборки
2. Столбец "пол" не является бинарным
3. Столбец "класс" является строчным

*.median( )*

*Возвращает медианное значение.*

```
dataset['age'].fillna(dataset['age'].median(), inplace = True)
dataset['age'].head()
```

```
row.names
999     29.0
180     65.0
557     59.0
175     46.0
1233    29.0
Name: age, dtype: float64
```

```
pd.options.mode.chained_assignment = None

data_inputs = dataset[["pclass", "age", "sex"]]
data_inputs["pclass"].replace("3rd", 3, inplace = True)
data_inputs["pclass"].replace("2nd", 2, inplace = True)
data_inputs["pclass"].replace("1st", 1, inplace = True)

data_inputs["sex"] = np.where(data_inputs["sex"] == "female", 0, 1)
data_inputs.head()
```

| row.names | pclass | age | sex |
|---|---|---|---|
| 999 | 3 | 29.0 | 0 |
| 180 | 1 | 65.0 | 1 |
| 557 | 2 | 59.0 | 1 |
| 175 | 1 | 46.0 | 1 |
| 1233 | 3 | 29.0 | 1 |

*Теперь мы можем приступать к работе с моделью:*

## Использование с sklearn:

***Обозначим как ожидаемый исход параметр, выжил ли пассажир:***

```
: expected_output = dataset[["survived"]]
```

***Опишем и обучим модель:***

```
pd.options.mode.chained_assignment = None

inputs_train, inputs_test, expected_output_train, expected_output_test = model_selection.train_test_split(data_inputs \
                                                                        , expected_output, test_size = 0.33, random_state = 42)
rf = ensemble.RandomForestClassifier (n_estimators=100)
rf.fit(inputs_train, expected_output_train.values.ravel())
accuracy = rf.score(inputs_test, expected_output_test)
print("Accuracy = {}%".format(accuracy * 100))
```

```
Accuracy = 79.60526315789474%
```

Точность модели – около 78 процентов

***Запишем модель в файл:***

```
joblib.dump(rf, "titanic_model1", compress=9)
```

```
['titanic_model1']
```

***Опишем четырех случайных пассажиров:***

1. 56-летняя женщина из 1-ого класса
2. 31-летняя женщина из 2-ого класса
3. 29-летний мужчина из 2-ого класса
4. 18-летний мужчина из 3-ого класса

***И попробуем предсказать их шансы на выживание:***

```
: data = pd.DataFrame({'pclass':[1, 2, 2, 3],
        'age':[56, 31, 29, 18],
        'sex':[0,0,1,1]})
rf = joblib.load("titanic_model1")
pred = rf.predict(data)
print(pred)
```

```
[1 1 0 0]
```

*Исходя из модели, выживут только первые два пассажира.*

## Теперь возьмем другой массив данных:

*Этот датасет содержит данные о небольшой экосистеме в южной Аризоне за последние 35 лет. Он является частью большего проекта по изучению влияния различных видов грызунов и насекомых на жизнь растений*

- plots.csv : список участков исследования с их ID и кратким описанием
- species.csv : список с двухсимвольным кодом вида и информации о нем
- surveys.csv : полный список наблюдений над видами на участках

```
species = pd.read_csv('./data/2_ecosystem/species.csv',index_col=[0])
species.head()
```

| species_id | genus | species | taxa |
|---|---|---|---|
| AB | Amphispiza | bilineata | Bird |
| AH | Ammospermophilus | harrisi | Rodent-not censused |
| AS | Ammodramus | savannarum | Bird |
| BA | Baiomys | taylori | Rodent |
| CB | Campylorhynchus | brunneicapillus | Bird |

```
plots = pd.read_csv('./data/2_ecosystem/plots.csv',index_col=[0])
plots.head()
```

| plot_id | plot_type |
|---|---|
| 1 | Spectab exclosure |
| 2 | Control |
| 3 | Long-term Krat Exclosure |
| 4 | Control |
| 5 | Rodent Exclosure |

*.sample( )*

*Возвращает набор случайных данных из массива:*

```
surveys = pd.read_csv('./data/2_ecosystem/surveys.csv',index_col=[0])
surveys.sample(10)
```

| record_id | month | day | year | plot | species | sex | wgt |
|---|---|---|---|---|---|---|---|
| **30037** | 11 | 7 | 1999 | 13 | PP | M | 12.0 |
| **2380** | 1 | 16 | 1980 | 15 | PF | M | 7.0 |
| **32877** | 10 | 13 | 2001 | 21 | SS | NaN | NaN |
| **6036** | 6 | 28 | 1982 | 2 | NL | F | 120.0 |
| **5723** | 4 | 28 | 1982 | 12 | RM | M | 11.0 |
| **22547** | 8 | 26 | 1995 | 19 | PF | M | 8.0 |
| **3114** | 6 | 23 | 1980 | 14 | DM | M | NaN |
| **9597** | 9 | 30 | 1984 | 5 | DM | M | 25.0 |
| **28993** | 12 | 23 | 1998 | 11 | DM | M | 47.0 |
| **11952** | 10 | 5 | 1986 | 13 | DM | M | 51.0 |

***Как мы видим, данные очень неоднородные, например не везде указан вид наблюдаемого животного, его пол или вес. Также на разные колонки разбиты даты:***

*.to_datetime( )*

*Возвращает объект типа datetime64:*

```
surveys['date'] = pd.to_datetime(pd.DataFrame({'year': surveys['year'],
                                'month': surveys['month'],
                                'day': surveys['day']}), errors='coerce')
surveys = surveys.drop(['year','month','day'], 1)
surveys.sample(5)
```

| record_id | plot | species | sex | wgt | date |
|---|---|---|---|---|---|
| **16730** | 4 | DM | F | 48.0 | 1989-11-05 |
| **33518** | 11 | DO | M | 55.0 | 2002-02-10 |
| **22432** | 11 | DM | M | 47.0 | 1995-06-28 |
| **28811** | 12 | DO | F | 46.0 | 1998-11-21 |
| **5820** | 4 | DS | F | 98.0 | 1982-04-29 |

*.dropna( )*

*Удаляет строки с хотя бы одним пропуском:*

```
surveys_clean = surveys.dropna()
surveys_clean.sample(10)
```

| record_id | plot | species | sex | wgt | date |
|---|---|---|---|---|---|
| **22209** | 4 | DM | M | 30.0 | 1995-04-02 |
| **22160** | 4 | DM | F | 40.0 | 1995-03-05 |
| **9759** | 4 | DM | M | 35.0 | 1984-12-31 |
| **13489** | 24 | OL | F | 24.0 | 1987-10-24 |
| **35013** | 18 | PB | F | 29.0 | 2002-11-09 |
| **2254** | 22 | DS | F | 143.0 | 1979-11-18 |
| **35005** | 18 | PB | F | 24.0 | 2002-11-09 |
| **6056** | 15 | OT | F | 17.0 | 1982-06-28 |
| **34796** | 17 | NL | M | 200.0 | 2002-10-05 |
| **15970** | 22 | DM | F | 53.0 | 1989-04-02 |

*.fillna( )*

*Заменяет пропуски на какие-то данные:*

```
surveys_zero = surveys.fillna(0)
surveys_zero.sample(10)
```

| record_id | plot | species | sex | wgt | date |
|---|---|---|---|---|---|
| **1058** | 12 | DM | F | 40.0 | 1978-07-07 00:00:00 |
| **24987** | 24 | PL | F | 27.0 | 1997-02-08 00:00:00 |
| **34946** | 14 | OT | M | 19.0 | 2002-10-06 00:00:00 |
| **33885** | 9 | SS | 0 | 0.0 | 2002-04-17 00:00:00 |
| **5014** | 4 | DM | M | 51.0 | 1981-11-23 00:00:00 |
| **3999** | 9 | DS | F | 137.0 | 1981-03-09 00:00:00 |
| **27308** | 7 | PM | M | 22.0 | 1997-12-28 00:00:00 |
| **43** | 21 | DM | M | 0.0 | 1977-07-18 00:00:00 |
| **4453** | 5 | DM | F | 41.0 | 1981-05-04 00:00:00 |
| **28244** | 8 | PP | M | 17.0 | 1998-07-19 00:00:00 |

***Так как видов у нас много, можем подсчитать сколько раз встречается тот или иной вид:***

```
surveys['species'].value_counts().head()
```

```
DM    10596
PP     3123
DO     3027
PB     2891
RM     2609
Name: species, dtype: int64
```

***Либо просто вывести их:***

*.unique( )*

*Возвращает объект, содержащий уникальные данные:*

```
surveys['species'].unique()
```

```
array(['NL', 'DM', 'PF', 'PE', 'DS', 'PP', 'SH', 'OT', 'DO', 'OX', 'SS',
       'OL', 'RM', nan, 'SA', 'PM', 'AH', 'DX', 'AB', 'CB', 'CM', 'CQ',
       'RF', 'PC', 'PG', 'PH', 'PU', 'CV', 'UR', 'UP', 'ZL', 'UL', 'CS',
       'SC', 'BA', 'SF', 'RO', 'AS', 'SO', 'PI', 'ST', 'CU', 'SU', 'RX',
       'PB', 'PL', 'PX', 'CT', 'US'], dtype=object)
```

*.nunique( )*

*Возвращает их количество:*

```
surveys['species'].nunique()
```

```
48
```

***Pandas позволяет составлять сводные таблицы по двум или нескольким параметрам:***

*.crosstab( )*

*Возвращает сводную таблицу:*

```
pd.crosstab(surveys['sex'],surveys['plot'])
```

| plot | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | ... | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| sex | | | | | | | | | | | | | | | | | | | | | |
| F | 848 | 970 | 893 | 872 | 527 | 737 | 335 | 840 | 853 | 141 | ... | 481 | 222 | 892 | 755 | 522 | 650 | 611 | 671 | 165 | 486 |
| M | 1095 | 1144 | 840 | 1030 | 586 | 764 | 307 | 983 | 1004 | 142 | ... | 409 | 273 | 1053 | 626 | 581 | 613 | 451 | 673 | 207 | 485 |
| P | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Z | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

***Заметим, что помимо привычных M/F, в таблице присутствуют еще три строки***

```
species.loc[surveys[surveys['sex'] == 'P'].species.to_string()[-2:] , :]
```

```
genus       Chaetodipus
species    penicillatus
taxa             Rodent
Name: PP, dtype: object
```

```
species.loc[surveys[surveys['sex'] == 'Z'].species.to_string()[-2:] , :]
```

```
genus       Chaetodipus
species    penicillatus
taxa             Rodent
Name: PP, dtype: object
```

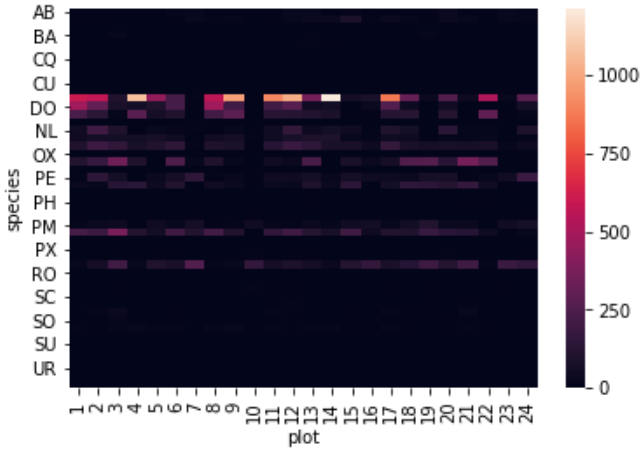
```
species.loc[surveys[surveys['sex'] == 'R'].species.to_string()[-2:] , :]
```

```
genus         Ammospermophilus
species                harrisi
taxa       Rodent-not censused
Name: AH, dtype: object
```

По имени вида существа, мы найдем, что Chaetodipus penicillatus является сумчатым *(pocket)*, а значит P — это тот, кто вынашивает, а Z - тот, кто не вынашивает детеныша в своей сумке. А Ammospermophilus harrisi напротив, практически не имеет внешних половых признаков, а значит ученые просто не смогли определить пол животного.

***Составив тепловую карту (библиотека seaborn) виды/участки, мы заметим, что есть виды, которые встречаются крайне редко на всех участках, а также участки, на которых количество животных заметно ниже, нежели на других:***

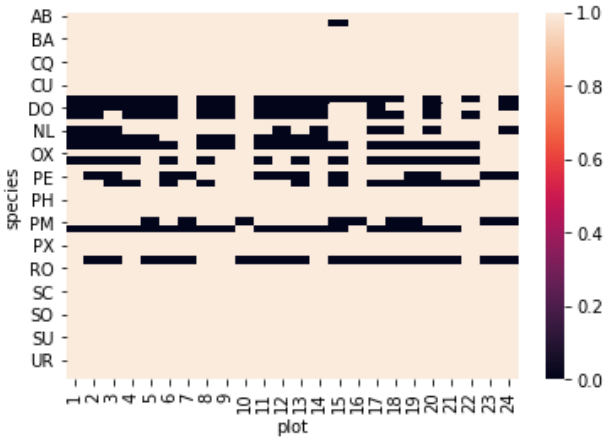
```
seaborn.heatmap(pd.crosstab(surveys['species'],surveys['plot']));
```




***Можем поставить условие, и показать какие виды встречаются меньше 50 раз:***

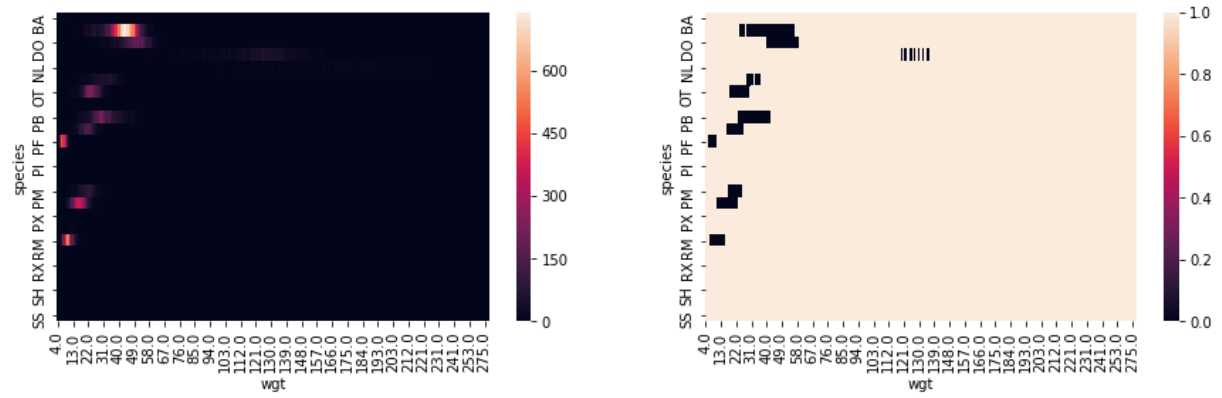
*Т.к. результат будет в формате true/false, белый цвет покажет, что значение истинное, а черный, соответственно, что ложное*

```
seaborn.heatmap(pd.crosstab(surveys['species'],surveys['plot']) < 50);
```



***Проверяя весь датасет на вес существа, увидим, что подавляющая масса зарегистрированных существ весит меньше 50 грамм:***

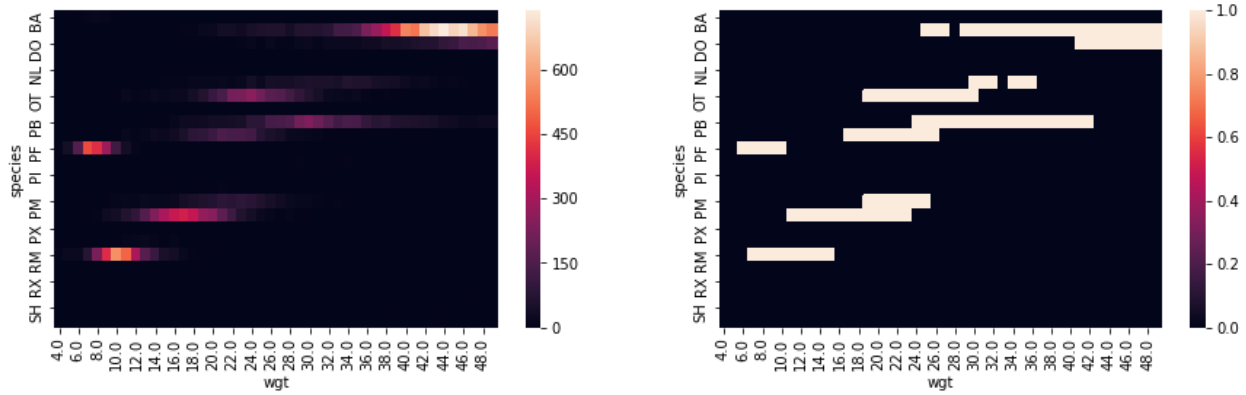
```
fig, ax = plt.subplots(1,2,figsize=(15, 4))
seaborn.heatmap(pd.crosstab(surveys['species'],surveys['wgt']), ax = ax[0]);
seaborn.heatmap(pd.crosstab(surveys['species'],surveys['wgt'])<50, ax = ax[1]);
```




***Мы можем создать для них отдельный датасет и далее работать с ним:***

```
smallCreatures = surveys[surveys['wgt']< 50]

fig, ax = plt.subplots(1,2,figsize=(15, 4))
seaborn.heatmap(pd.crosstab(smallCreatures['species'],smallCreatures['wgt']), ax = ax[0]);
seaborn.heatmap(pd.crosstab(smallCreatures['species'],smallCreatures['wgt'])>50, ax = ax[1]);
```



***Построим ящик с усами для веса самого часто наблюдаемого существа***

```
surveys['species'].value_counts().head(3)
```

```
DM    10596
PP     3123
DO     3027
Name: species, dtype: int64
```

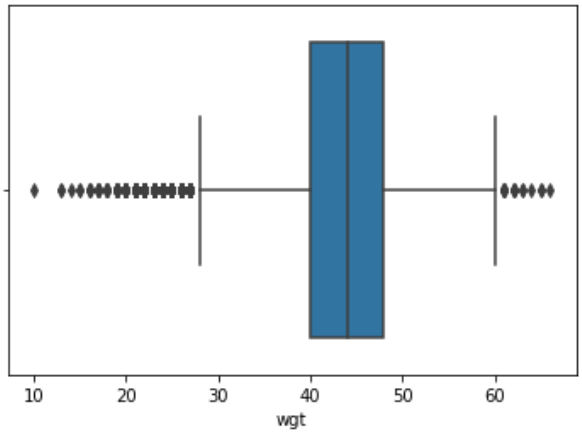
*Мы видим, что такое вид этого существа промаркирован как DM, можем вывести информацию о нем:*

```
species.loc[surveys[surveys['species'] == 'DM'].species.to_string()[-2:] , :]
```

```
genus      Dipodomys
species     merriami
taxa           Rodent
Name: DM, dtype: object
```

```
seaborn.boxplot(surveys[surveys['species'] == 'DM']['wgt'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x263164dbac8>
```




*Мы видим, что медианный вес находится в районе ~43 грамм, а границы ящика примерно на 40 и 47 граммах. Значит примерно четверть замеров показала, что вес меньше, чем 40, и еще четверть больше 47. Усы находятся на 1,5 интерквантильных расстояния (ширины ящика) от медианы, то, что выбивается за эти усы, можем считать выбросами, слишком малыми значениями или слишком большими*

## Теперь возьмем третий массив данных:

*Этот датасет содержит данные о книгах, находящихся в Британской Библиотеке; он отличается тем, что данные в нем неоднородны и перед их использованием, их нужно “почистить”*

```
librarySet = pd.read_csv('./data/3_library/BL-Flickr-Images-Book.csv', index_col=[0])
librarySet.sample(5)
```

| Identifier | Edition Statement | Place of Publication | Date of Publication | Publisher | Title | Author | Contributors | Corporate Author | Corporate Contributors | Former owner | Engraver | Issuance type | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3814731 | NaN | London | 1744 | J. Watts; B. Dod | Mahomet the Impostor. A tragedy. [In verse. Ad... | Voltaire | HOADLY, John - LL.D., Dramatic Writer\|MILLER, ... | NaN | NaN | NaN | NaN | monographic | http: |
| 2420207 | NaN | Hartford [Conn.] | 1847 | NaN | Geography of the State of New York; with stati... | MATHER, J. H. - and BROCKETT (L. P.) | BROCKETT, Linus Pierpont. | NaN | NaN | NaN | NaN | monographic | http: |
| 3076549 | NaN | London | 1877 | NaN | The New Republic: or, Culture, Faith, and Phil... | NaN | Mallock, W. H. (William Hurrell) | NaN | NaN | NaN | NaN | monographic | http: |
| 3919856 | NaN | London | 1697 | Printed for Sam. Briscoe ... and sold by Richa... | Rule a Wife, and Have a Wife. A comedy, etc. [... | NaN | FLETCHER, John - Dramatist | NaN | NaN | NaN | NaN | monographic | http: |
| 499209 | NaN | New York | 1883 | H. Holt & Co. | Lyrical and Dramatic Poems selected from the w... | BROWNING, Robert - the Poet | MASON, Edward T. | NaN | NaN | NaN | NaN | monographic | http: |

***Видим, что многие столбцы содержат пустые, либо неинформативные***

```
librarySet['Corporate Author'].nunique()
```

```
0
```

```
librarySet['Shelfmarks'].unique()
```

```
array(['British Library HMNTS 12641.b.30.',
       'British Library HMNTS 12626.cc.2.',
       'British Library HMNTS 12625.dd.1.', ...,
       'British Library HMNTS|British Library HMNTS 192.e.3-10.|British Library HMNTS G.3246-53.|British Library HMNTS 578.k.3
8.|British Library HMNTS 1303.m.5-9.|British Library HMNTS 579.k.2.(2.)',
       'British Library HMNTS|British Library HMNTS 10353.e.8.',
       'British Library HMNTS|British Library HMNTS 796.i.18-21.|British Library HMNTS G.13643-50.'],
      dtype=object)
```

***Удалим колонки с неважными для анализа данными:***

```
to_drop = ['Edition Statement', 'Corporate Author', 'Corporate Contributors', 'Former owner', 'Engraver' \
           , 'Contributors', 'Issuance type', 'Shelfmarks']

librarySet.drop(to_drop, inplace=True, axis=1)

librarySet.head()
```

| Identifier | Place of Publication | Date of Publication | Publisher | Title | Author | Flickr URL |
|---|---|---|---|---|---|---|
| 206 | London | 1879 [1878] | S. Tinsley & Co. | Walter Forbes. [A novel.] By A. A | A. A. | http://www.flickr.com/photos/britishlibrary/ta... |
| 216 | London; Virtue & Yorston | 1868 | Virtue & Co. | All for Greed. [A novel. The dedication signed... | A., A. A. | http://www.flickr.com/photos/britishlibrary/ta... |
| 218 | London | 1869 | Bradbury, Evans & Co. | Love the Avenger. By the author of "All for Gr... | A., A. A. | http://www.flickr.com/photos/britishlibrary/ta... |
| 472 | London | 1851 | James Darling | Welsh Sketches, chiefly ecclesiastical, to the... | A., E. S. | http://www.flickr.com/photos/britishlibrary/ta... |
| 480 | London | 1857 | Wertheim & Macintosh | [The World in which I live, and my place in it... | A., E. S. | http://www.flickr.com/photos/britishlibrary/ta... |

***Как видим, даты тоже местами некорректны:***

Приведем даты к четырехзначному числовому формату *(в данный момент, у дат тип object)* и вычислим какая доля из них не имеет даты вовсе:

```
librarySet.loc[1905:, 'Date of Publication'].head(10)
```

```
Identifier
1905           1888
1929    1839, 38-54
2836           1897
2854           1865
2956        1860-63
2957           1873
3017           1866
3131           1899
4598           1814
4884           1820
Name: Date of Publication, dtype: object
```

*.to_numeric( )*

*Проверяет на то, является ли объект числом:*

*.isnull( )*

*Проверяет на то, является ли объект нулевым:*

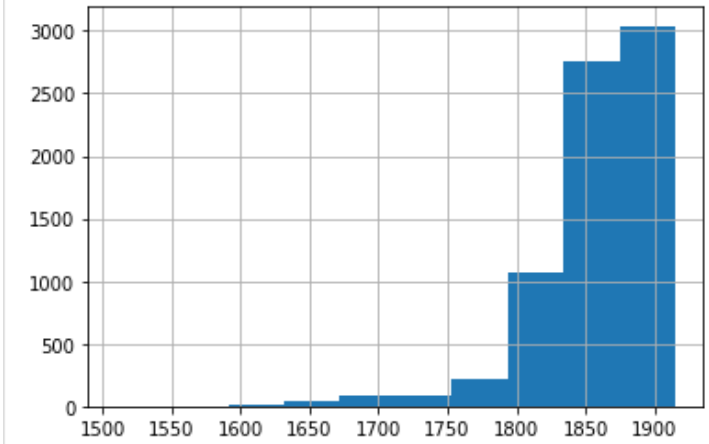
```
extr = librarySet['Date of Publication'].str.extract(r'^(\d{4})', expand=False) # Ищет первые 4 цифры
librarySet['Date of Publication'] = pd.to_numeric(extr, downcast='signed')

librarySet['Date of Publication'].isnull().sum() / len(librarySet)
```

```
0.11717147339205986
```

***Каждая десятая книга не имеет даты. Построим график распределения:***

```
librarySet['Date of Publication'].hist();
```




***Так как датасет очень неоднородный, например, места тут зачастую дублируются на разных языках, его нужно чистить от повторяющихся или некачественных записей***

```
librarySet['Place of Publication'].value_counts().tail(10)
```

```
Copenhagen                                        1
Ἀθηνησιν                                          1
pp. xiv. William Nicol: London, 1847              1
Augusta                                           1
Franz Duncker                                     1
Annapolis                                         1
London; Peter Hill                                1
Cuxhaven                                          1
pp. viii. 154. Wilson & McCormick: Glasgow, 1882  1
Gütersloh                                         1
Name: Place of Publication, dtype: int64
```

***Покажем на примере городов Лондон и Оксфорд***

```
pub = librarySet['Place of Publication']

london = pub.str.contains('London')
london[:5]

oxford = pub.str.contains('Oxford')

librarySet['Place of Publication'] = np.where(london, 'London',
                                     np.where(oxford, 'Oxford',
                                              pub.str.replace('-', ' ')))

librarySet['Place of Publication'].value_counts().head(10)
```

```
London            4219
Paris              479
Edinburgh          208
New York           193
Leipzig            119
Philadelphia        89
Berlin              70
Boston [Mass.]      52
Oxford              49
Dublin              48
Name: Place of Publication, dtype: int64
```

## Вывод

Библиотека Pandas - очень мощная библиотека для обработки больших массивов данных, их подготовки и непосредственно анализа и моделирования. Пакет прежде всего предназначен для очистки и первичной оценки данных по общим показателям. Статистическим пакетом он в полном смысле не является, однако наборы данных типов DataFrame и Series применяются в качестве входных в большинстве модулей анализа данных и машинного обучения (SciPy, Scikit-Learn и других).