# Лабораторная работа (Изучение библиотек обработки данных.)

# Часть 1

## Создание окружения

```
import tensorflow as tf
tf.test.gpu_device_name()
```

```
!pip install -U -q PyDrive

from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# 1. Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

```
!pip install -U pandasql
```



```
file_list = drive.ListFile({'q': "'1FV3XZb9X25mBGBhYOsmg5eyeuJKSnNLp' in parer
for file1 in file_list:
  print('title: %s, id: %s' % (file1['title'], file1['id']))
```

```
train_downloaded = drive.CreateFile({'id': '1Se6oWEujEmZtmnJvcDu9NGKXfG7_DX-v'
train_downloaded.GetContentFile('winter.csv')
test_downloaded = drive.CreateFile({'id': '117EvqN7ZQtIfntEWtdz42OjFnoNdnslR'}
test_downloaded.GetContentFile('summer.csv')
train_downloaded = drive.CreateFile({'id': '1CEYqFeTOOPPVFIYuZZ9otX7_FWP0FE-e'
train_downloaded.GetContentFile('dictionary.csv')
test_downloaded = drive.CreateFile({'id': '11N8YVg6fMtyhlH2D8M796Tkq8y--BmS7'}
test_downloaded.GetContentFile('adult.data.csv')
```

```
!pip install -q matplotlib-venn
```

```
!apt-get -qq install -y libfluidsynth1
```

```
# To determine which version you're using:
!pip show tensorflow

# For the current version:
!pip install --upgrade tensorflow

# For a specific version:
!pip install tensorflow==1.2

# For the latest nightly build:
!pip install tf-nightly
```

```
# https://pypi.python.org/pypi/libarchive
```

This file was updated remotely or in another tab. To force a save, overwriting the last update, select Save from the File menu

```
Selecting previously unselected package libarchive-dev:amd64.
(Reading database ... 131309 files and directories currently installed.)
Preparing to unpack .../libarchive-dev_3.2.2-3.1ubuntu0.3_amd64.deb ...
Unpacking libarchive-dev:amd64 (3.2.2-3.1ubuntu0.3) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Setting up libarchive-dev:amd64 (3.2.2-3.1ubuntu0.3) ...
  Building wheel for libarchive (setup.py) ... done
```

```
# https://pypi.python.org/pypi/pydot
!apt-get -qq install -y graphviz && pip install -q pydot
import pydot
```

```
!apt-get -qq install python-cartopy python3-cartopy
```

```
import cartopy
```

```
Selecting previously unselected package python-pkg-resources.
(Reading database ... 131365 files and directories currently installed.)
Preparing to unpack .../0-python-pkg-resources_39.0.1-2_all.deb ...
Unpacking python-pkg-resources (39.0.1-2) ...
Selecting previously unselected package python-pyshp.
Preparing to unpack .../1-python-pyshp_1.2.12+ds-1_all.deb ...
Unpacking python-pyshp (1.2.12+ds-1) ...
Selecting previously unselected package python-shapely.
Preparing to unpack .../2-python-shapely_1.6.4-1_amd64.deb ...
Unpacking python-shapely (1.6.4-1) ...
Selecting previously unselected package python-six.
Preparing to unpack .../3-python-six_1.11.0-2_all.deb ...
Unpacking python-six (1.11.0-2) ...
Selecting previously unselected package python-cartopy:amd64.
Preparing to unpack .../4-python-cartopy_0.14.2+dfsg1-2build3_amd64.deb ...
Unpacking python-cartopy:amd64 (0.14.2+dfsg1-2build3) ...
Selecting previously unselected package python3-pkg-resources.
Preparing to unpack .../5-python3-pkg-resources_39.0.1-2_all.deb ...
Unpacking python3-pkg-resources (39.0.1-2) ...
Selecting previously unselected package python3-pyshp.
Preparing to unpack .../6-python3-pyshp_1.2.12+ds-1_all.deb ...
Unpacking python3-pyshp (1.2.12+ds-1) ...
Selecting previously unselected package python3-shapely.
Preparing to unpack .../7-python3-shapely_1.6.4-1_amd64.deb ...
Unpacking python3-shapely (1.6.4-1) ...
Selecting previously unselected package python3-six.
Preparing to unpack .../8-python3-six_1.11.0-2_all.deb ...
Unpacking python3-six (1.11.0-2) ...
Selecting previously unselected package python3-cartopy:amd64.
Preparing to unpack .../9-python3-cartopy_0.14.2+dfsg1-2build3_amd64.deb ..
Unpacking python3-cartopy:amd64 (0.14.2+dfsg1-2build3) ...
Setting up python-shapely (1.6.4-1) ...
Setting up python-pyshp (1.2.12+ds-1) ...
Setting up python3-six (1.11.0-2) ...
Setting up python3-shapely (1.6.4-1) ...
Setting up python3-pyshp (1.2.12+ds-1) ...
Setting up python3-pkg-resources (39.0.1-2) ...
Setting up python-pkg-resources (39.0.1-2) ...
Setting up python-six (1.11.0-2) ...
Setting up python3-cartopy:amd64 (0.14.2+dfsg1-2build3) ...
Setting up python-cartopy:amd64 (0.14.2+dfsg1-2build3) ...
```



```
import numpy as np
import pandas as pd
```

```
data = pd.read_csv('adult.data.csv')
```

## Работа с данными

```
data.head()
```

**1. Сколько мужчин и женщин представлены в этом наборе данных?**

```
data['sex'].value_counts()
```

**2. Каков средний возраст женщин?**

```
data.loc[data['sex'] == 'Female', 'age'].mean()
```

**3. Какова доля немецких граждан?**

```
float((data['native-country'] == 'Germany').sum()) / data.shape[0]
```

**4-5. Каковы среднее значение и стандартное отклонение возраста тех, кто получает более 50K в год (функция зарплаты) и тех, кто получает менее 50K в год?**



```
print("The average age of the rich: {0} +- {1} years, poor - {2} +- {3} years.
    round(ages1.mean()), round(ages1.std(), 1),
    round(ages2.mean()), round(ages2.std(), 1)))
```

**6. Правда ли, что люди, получающие более 50 тысяч, имеют хотя бы среднее школьное образование?**

```
data.loc[data['salary'] == '>50K', 'education'].unique() # No
```

**7. Отображение статистики возраста для каждой расы (функция расы) и каждого пола. Используйте groupby () и describe (). Найти максимальный возраст мужчин Амер-Индо-эскимосской расы.**

```
for (race, sex), sub_df in data.groupby(['race', 'sex']):
    print("Race: {0}, sex: {1}".format(race, sex))
    print(sub_df['age'].describe())
```

**8. Среди кого больше доля тех, кто много зарабатывает (>50 тыс.): среди женатых или одиноких мужчин (характеристика семейного положения)? Считайте женатыми тех, кто имеет семейное положение, начиная с женатого, остальные считаются холостяками.**

```
data.loc[(data['sex'] == 'Male') &
     (data['marital-status'].isin(['Never-married',
                                   'Separated',
                                   'Divorced',
                                   'Widowed'])), 'salary'].value_counts()
```

```
data.loc[(data['sex'] == 'Male') &
     (data['marital-status'].str.startswith('Married')), 'salary'].value_count
```

```
data['marital-status'].value_counts()
```

**9. Каково максимальное количество часов, которое человек работает в неделю (функция часов в неделю)? Сколько человек работает такое количество часов и каков процент тех, кто много зарабатывает среди них?**

```
max_load = data['hours-per-week'].max()
print("Max time - {0} hours./week.".format(max_load))

num_workaholics = data[data['hours-per-week'] == max_load].shape[0]
print("Total number of such hard workers {0}".format(num_workaholics))

rich_share = float(data[(data['hours-per-week'] == max_load)
                 & (data['salary'] == '>50K')].shape[0]) / num_workaholics
print("Percentage of rich among them {0}%".format(int(100 * rich_share)))
```

**10. Подсчитайте среднее время работы (часы-в-неделю) тех, кто зарабатывает мало и много (зарплата) для каждой страны (родной страны)**



```
for (country, salary), sub_df in data.groupby(['native-country', 'salary']):
    print(country, salary, round(sub_df['hours-per-week'].mean(), 2))
```

**Элегантный метод:**

```
pd.crosstab(data['native-country'], data['salary'],
            values=data['hours-per-week'], aggfunc=np.mean).T
```

# Часть 2

## Загружаем данные

```
# импортируем Pandas и Numpy
import pandas as pd
import numpy as np
```

```
dictionary = pd.read_csv('dictionary.csv')
dictionary.head()
```



```
summer = pd.read_csv('summer.csv')
summer.head()
```

```
winter = pd.read_csv('winter.csv')
winter.head()
```

## Работа с данными

```
dictionary.head()
```

```
winter.head()
```

### Соединение таблиц

```
def connection_pandas(dictionary,summer):
        result = pd.merge(dictionary, summer, left_on = 'Code', right_on = 'Co
        return result

connection_pandas(dictionary, summer).head()
```

```
import pandasql as ps
pysql = lambda a: ps.sqldf(a, globals())
def connection_pandasql(dictionary,summer):
    query = "select * from dictionary,summer where dictionary.Code = summer.Co
    join_result = pysql(query)
    return join_result
abc = connection_pandasql(dictionary, summer)
connection_pandasql(dictionary, summer).head()
```



### Сравнение времени выполнения запросов

```
import time
```

```
class Profiler(object):
    def __enter__(self):
        self._startTime = time.time()

    def __exit__(self, type, value, traceback):
        print("Elapsed time: {:.3f} sec".format(time.time() - self._startTime)

with Profiler() as p:
    connection_pandas(dictionary, summer)
```

```
with Profiler() as p:
    connection_pandas(dictionary, winter)
```

```
with Profiler() as p:
    connection_pandasql(dictionary, summer)
```

```
with Profiler() as p:
    connection_pandasql(dictionary, winter)
```

**Вывод: соединение с помощью pandas работает в 30 быстрее, чем pandasql**

**Агрегирование: произвольный запрос на группировку набора данных с использованием функций агрегирования**

```
def aggregation_pandas(dictionary,summer):
    result = pd.merge(dictionary, summer, left_on = 'Code', right_on = 'Countr
    final_0 = result[result['Year'] == 2012]
    final = final_0[final_0['Medal'] == 'Gold'].groupby("Country_x").agg({
        "Medal": "count",
        'Discipline' : 'nunique',
        'Gender' : 'nunique',
    })
    return final

aggregation_pandas(dictionary, summer).head(10)
```

```
def aggregation_pandasql(summer):
    query = '''
    SELECT Country, count(Medal), count(DISTINCT Discipline), count(DISTINCT (
    WHERE Medal == 'Gold' and Year == 2012 and Country != 'None'
    GROUP BY Country
    '''
    return ps.sqldf(query,locals())

aggregation_pandasql(summer).head(10)
```

| | Country | count(Medal) | count(DISTINCT Discipline) | count(DISTINCT Gender) |
|---|---|---|---|---|
| **0** | ALG | 1 | 1 | 1 |
| **1** | ARG | 1 | 1 | 1 |
| **2** | AUS | 19 | 5 | 2 |
| **3** | AZE | 2 | 1 | 1 |
| **4** | BAH | 4 | 1 | 1 |
| **5** | BLR | 3 | 2 | 2 |
| **6** | BRA | 14 | 3 | 2 |
| **7** | CAN | 1 | 1 | 1 |
| **8** | CHN | 56 | 13 | 2 |
| **9** | COL | 1 | 1 | 1 |

**Сравнение времени выполнения запросов агрегирования**

```
import seaborn
import matplotlib.pyplot as plt
with Profiler() as p:
    aggregation_pandas(dictionary,summer)
```

```
Elapsed time: 0.025 sec
```



```
    aggregation_pandasql(summer)
```

**Вывод: pandas работает значительно быстрее, чем pandasql (в 10 раз)**