

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение  
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

«Методы искусственного интеллекта»

Отчёт по лабораторной работе №3

Вариант №7

Выполнил:

студент группы ИСТбд-42

Егоров Иван

Проверил:

доцент кафедры ИВК, к.т.н.

Шишкин В.В.

Ульяновск  
2022

## Задание 1.

“Генерация данных”.

Создать симулированный набор данных и записать его на диск в виде csv файла со следующими параметрами:

- количество строк не менее 1000 (задается случайным образом);
- структура набора:
  - табельный номер;
  - Фамилия И.О.;
  - пол;
  - год рождения;
  - год начала работы в компании;
  - подразделение;
  - должность;
  - оклад;
  - количество выполненных проектов

Результат.

Generation process

```
number , full_name , gender , birth_date , start_date , division , position , salary , completed_projects ,
1 , Маврина Р.К. , Жен , 2.1.1995 , 6.12.2013 , Отдел поддержки ИТ , Специалист , 26924 , 38 ,
2 , Райкин С.К. , Муж , 8.9.1987 , 19.12.2016 , Отдел контроля качества и процессов , Специалист , 100506 , 24 ,
3 , Щукин Г.К. , Муж , 26.12.1994 , 20.6.2022 , Отдел разработки ПО , Middle разработчик , 68244 , 7 ,
4 , Павлов Ж.Б. , Муж , 27.10.1994 , 9.8.2014 , Отдел поддержки ИТ , Специалист , 27356 , 18 ,
5 , Павлова У.З. , Жен , 8.5.1990 , 10.1.2016 , Отдел развития ИТ , Руководитель отдела развития ИТ , 76308 , 51 ,
6 , Кабаков Г.М. , Муж , 14.12.1993 , 26.1.2012 , Отдел разработки ПО , Middle разработчик , 50172 , 2 ,
7 , Данилина Г.И. , Жен , 27.11.1986 , 9.8.2013 , Отдел информационной безопасности , Начинаящий специалист , 55460 , 17 ,
8 , Хабалова Э.Л. , Жен , 17.9.1996 , 9.11.2017 , Отдел поддержки ИТ , Специалист , 28422 , 22 ,
9 , Царёв Б.К. , Муж , 2.12.1989 , 25.2.2012 , Отдел поддержки ИТ , Руководитель отдела поддержки ИТ , 32868 , 45 ,
10 , Данилина У.А. , Жен , 27.2.2003 , 26.12.2021 , Отдел информационной безопасности , Специалист , 164000 , 22 ,
```

## Задание 2.

“Получение статистических характеристик при помощи библиотеки numpy”.

Прочитать сгенерированный набор данных в виде списков и получить с помощью программирования и методов библиотеки numpy для разных по типу признаков столбцов (не менее 3) основные статистические характеристики (например для порядкового типа: минимум, максимум, среднее, дисперсия, стандартное отклонение, медиана, мода).

Результат.

Statistical characteristics

```
For column number: min=1 ; max=1999 ; ave=1000.0 ; disp=333000.0 ; std=577.0615218501404 ; median=1000.0 ; mode=1
For column salary: min=10047 ; max=299190 ; ave=89180.19009504752 ; disp=4033217756.9813724 ; std=63507.61967655041 ; median=71600.0 ; mode=30472
For column projects: min=1 ; max=60 ; ave=21.15057528764382 ; disp=235.83875779810867 ; std=15.35704261236872 ; median=18.0 ; mode=18
```

## Задание 3.

“Получение статистических характеристик при помощи библиотеки pandas”.

Прочитать сгенерированный набор данных в виде датафрейма и получить с помощью методов библиотеки pandas для тех же столбцов те же статистические характеристики. Продемонстрировать применение не менее 3 методов библиотеки pandas.

Результат.

```
Statistical characteristics
```

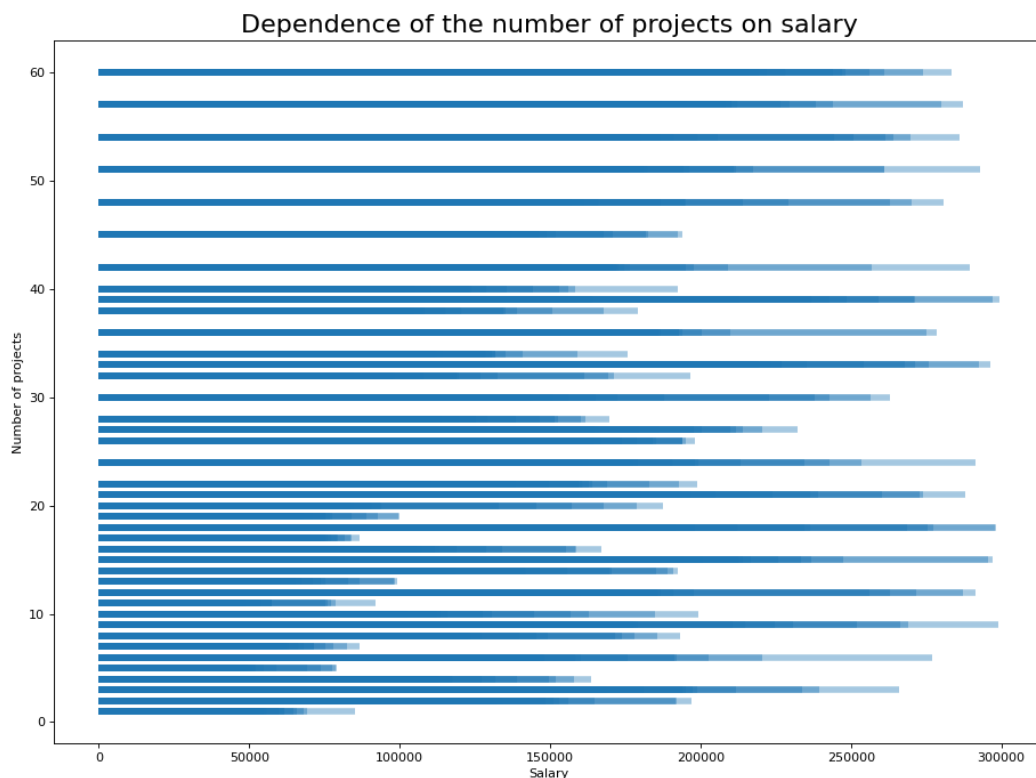
```
For column number: min=1 ; max=1999 ; ave=1000.0 ; disp=333166.666666667 ; std=577.2059135756205 ; median=1000.0 ; mode=1  
For column salary: min=10047 ; max=299190 ; ave=89180.19009504752 ; disp=4035236384.487369 ; std=63523.51048617645 ; median=71600.0 ; mode=30472  
For column projects: min=1 ; max=60 ; ave=21.15057528764382 ; disp=235.95679521442403 ; std=15.36088523537703 ; median=18.0 ; mode=18
```

#### Задание 4.

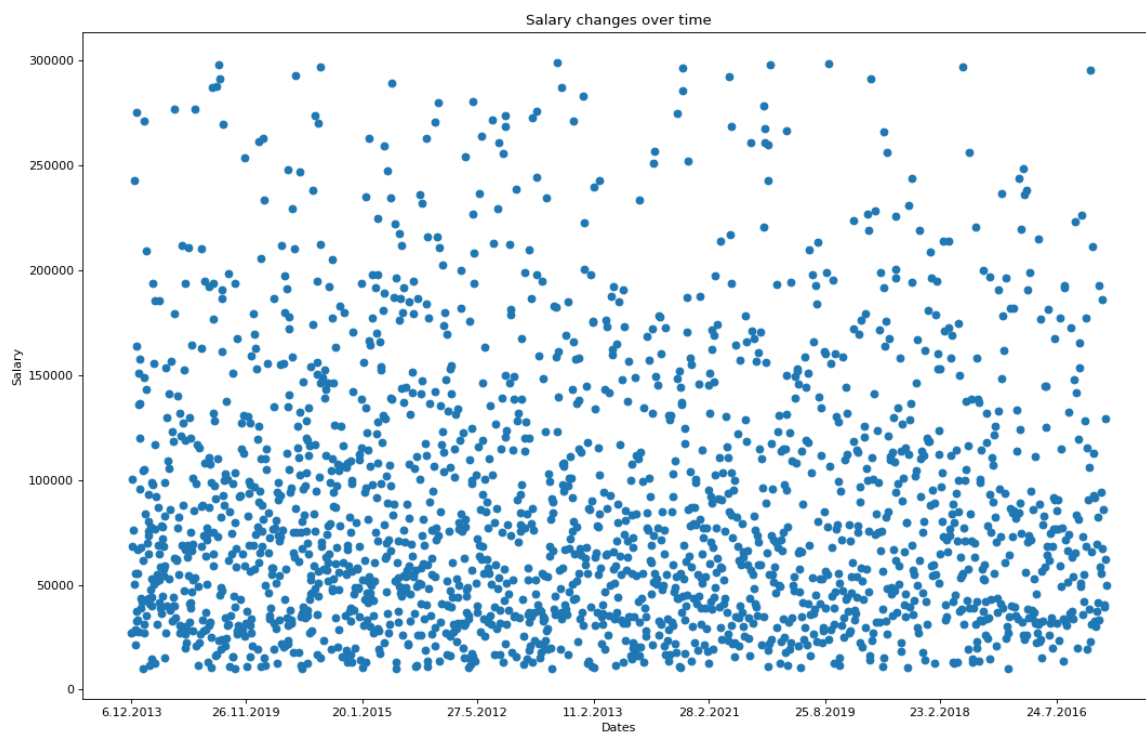
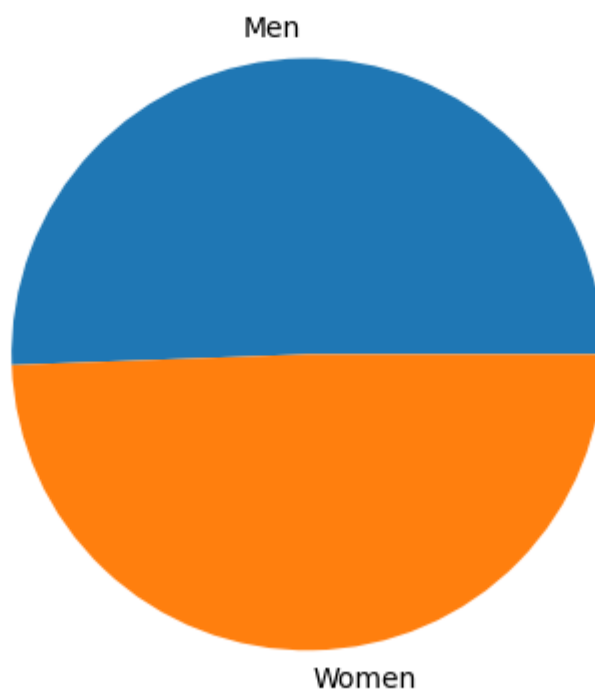
“Построение графиков”.

Построить не менее 3 разнотипных графиков.

Результат.



Distribution of men and women within the firm



### Задание 5.

“Оценка библиотек numpy и pandas”.

Оценить возможности библиотек csv, numpy, pandas в форме отчета по лабораторной работе.

Результат.

CSV.

Формат CSV (Comma Separated Values) является наиболее распространенным и удобным форматом для работы с данными в виде таблиц. Формат является достаточно универсальным и позволяет хранить разнотипные данные. Модуль CSV позволяет работать с CSV файлами не задумываясь о их структуре и т.д.

NumPy.

Библиотека NumPy позволяет использовать различные математические вычисления. Библиотека является достаточно быстрой т.к. базируется на коде написанном на языках C/C++/Fortran. Из положительных сторон можно отметить: использование высокоуровневых функций при работе с библиотекой; наличие разнообразных математических средств ; высокая скорость работы.

Pandas.

Библиотека Pandas позволяет осуществлять анализ данных. Библиотека является достаточно быстрой т.к. базируется на библиотеке NumPy. Из положительных сторон можно отметить: использование высокоуровневых функций при работе с библиотекой; наличие разнообразных средств для анализа данных; высокая скорость работы.

Код.

```
import csv
import numpy as np
from numpy.random import choice
from numpy import genfromtxt
import datetime as dt
import os
import pandas as ps
import matplotlib.pyplot as plt
import matplotlib.ticker as mticker
import matplotlib.dates as mdates
```

```
def graph():
```

```
    MyDataFrame = ps.read_csv('data.csv')
    numbers = MyDataFrame["number"]
    num_min = numbers.min()
    num_max = numbers.max()
    num_ave = numbers.mean()
```

```

num_disp = numbers.var()
num_std = numbers.std()
num_median = numbers.median()
num_mode = mode(numbers)
salary = MyDataFrame["salary"]
sal_min = salary.min()
sal_max = salary.max()
sal_ave = salary.mean()
sal_disp = salary.var()
sal_std = salary.std()
sal_median = salary.median()
sal_mode = mode(salary)
projects = MyDataFrame["completed_projects"]
pjt_min = projects.min()
pjt_max = projects.max()
pjt_ave = projects.mean()
pjt_disp = projects.var()
pjt_std = projects.std()
pjt_median = projects.median()
pjt_mode = mode(projects)

plt.figure(figsize=(14, 10), dpi=80)
plt.hlines(y=projects, xmin=0, xmax=salary, color='C0', alpha=0.4, linewidth=5)
plt.gca().set(ylabel='Number of projects', xlabel='Salary')
plt.title('Dependence of the number of projects on salary', fontdict={'size': 20})
plt.show()

```

```

data=[MyDataFrame["gender"].value_counts()["Муж"],MyDataFrame["gender"].value_counts()["Жен"]]
plt.pie(data, labels=["Men","Women"])
plt.title("Distribution of men and women within the firm")
plt.ylabel("")
plt.show()

```

```

data=MyDataFrame
myLocator = mticker.MultipleLocator(4)
plt.figure(figsize=(16, 10), dpi=80)
plt.plot_date(data["start_date"],data["salary"])
plt.gca().xaxis.set_major_locator(mdates.AutoDateLocator())
plt.ylabel('Salary')
plt.xlabel('Dates')
plt.title("Salary changes over time")
plt.show()

```

```
def mode(values):
```

```
    dict={ }
    for elem in values:
        if elem in dict:
            dict[elem]+=1
        else:
            dict[elem]=1
    v = list(dict.values())
    k = list(dict.keys())

    return k[v.index(max(v))]
```

```
def generate():
```

```
    MyData = [["number", "full_name", "gender", "birth_date", "start_date",
"division", "position", "salary",
                "completed_projects"]]
    Gender = ["Муж", "Жен"]
    Surnames = ["Антипов", "Бабаев", "Вавилов", "Галкин", "Данилин",
"Евсюткин", "Жеглов", "Задорнов", "Ивачев",
                "Кабаков", "Лабутин",
                "Маврин", "Назаров", "Овсеев", "Павлов", "Райкин", "Савочкин",
"Табаков", "Уваров", "Фандеев",
                "Хабалов", "Царёв", "Чадов",
                "Шаляпин", "Щукин", "Эвентов", "Юров", "Ягодин"]
    Initials = ["А", "Б", "В", "Г", "Д", "Ж", "З", "И", "К", "Л", "М", "Н", "Р", "С",
"Т", "У", "Ф", "Э", "Ю", "Я"]
    Divisions = ["Отдел информационной безопасности", "Отдел разработки
ПО", "Отдел контроля качества и процессов",
                "Отдел развития ИТ", "Отдел поддержки ИТ"]
    Positions = ["Руководитель отдела информационной безопасности",
"Специалист", "Начинающий специалист"],
                ["Руководитель отдела разработки ПО", "Senior разработчик",
"Middle разработчик"],
                ["Руководитель отдела контроля качества и процессов",
"Специалист", "Работник"],
                ["Руководитель отдела развития ИТ", "Специалист", "Работник"],
                ["Руководитель отдела поддержки ИТ", "Специалист", "Работник"]]

    for i in range(1, 2000):
        np.random.seed(i)
        num = i
        full_name = ""
        gend = ""
```

```

birth = ""
start = ""
div = ""
pos = ""
salary = 0
completed_projects = 0

gender = np.random.randint(0, 2)
gend = Gender[gender]
if (gender != 0):
    full_name = Surnames[np.random.randint(0, 27)] + "a" +
Initials[np.random.randint(0, 19)] + "." + Initials[
    np.random.randint(0, 19)] + "."
else:
    full_name = Surnames[np.random.randint(0, 27)] + " " +
Initials[np.random.randint(0, 19)] + "." + Initials[
    np.random.randint(0, 19)] + "."
current_date = dt.date.today()
year = current_date.year - np.random.randint(0, 11)
month = np.random.randint(1, 13)
day = np.random.randint(1, 29)
start = str(day) + "." + str(month) + "." + str(year)
byear = year - np.random.randint(18, 31)
bmonth = np.random.randint(1, 13)
bday = np.random.randint(1, 29)
birth = str(bday) + "." + str(bmonth) + "." + str(byear)
divis = np.random.randint(0, 5)
div = Divisions[divis]
posit = choice([0, 1, 2], 1, [0.1, 0.3, 0.6])[0]
pos = Positions[divis][posit]
salary = np.random.randint(10000, 20001) * (5 - divis) * (3 - posit)
completed_projects = np.random.randint(1, 21) * (3 - posit)

MyData.append([num, full_name, gend, birth, start, div, pos, salary,
completed_projects])

for per in MyData:
    for param in per:
        print(param, end=" , ")
    print("")

with open("data.csv", mode="w", encoding='utf-8') as w_file:
    file_writer = csv.writer(w_file, delimiter="," , lineterminator="\r")
    for per in MyData:
        file_writer.writerow(per)

```



```

def np_stat():

    MyData = []

    with open('data.csv', mode="r", encoding='utf-8') as f:
        reader = csv.reader(f)
        for row in reader:
            data=row

    MyData.append([data[0],data[1],data[2],data[3],data[4],data[5],data[6],data[7],data
[8]])

    for per in MyData:
        for param in per:
            print(param, end=" , ")
        print("")

    MyData=np.array(MyData)
    numbers=np.array(MyData[:,0])
    numbers=np.delete(numbers, 0)
    numbers=[int(item) for item in numbers]
    num_min=np.min(numbers)
    num_max=np.max(numbers)
    num_ave= np.average(numbers)
    num_disp=np.var(numbers)
    num_std=np.std(numbers)
    num_median=np.median(numbers)
    num_mode=mode(numbers)
    salary=np.array(MyData[:,7])
    salary=np.delete(salary,0)
    salary = [int(item) for item in salary]
    sal_min=np.min(salary)
    sal_max=np.max(salary)
    sal_ave=np.average(salary)
    sal_disp=np.var(salary)
    sal_std=np.std(salary)
    sal_median = np.median(salary)
    sal_mode=mode(salary)
    projects=np.array(MyData[:,8])
    projects = np.delete(projects, 0)
    projects = [int(item) for item in projects]
    pjt_min = np.min(projects)
    pjt_max = np.max(projects)
    pjt_ave = np.average(projects)

```

```

pjt_disp = np.var(projects)
pjt_std = np.std(projects)
pjt_median = np.median(projects)
pjt_mode = mode(projects)
print(numbers)

print("")
print("Statistical characteristics")
print("")
print("For column number: min="+str(num_min)+" ; max="+str(num_max)+" ;
ave="+str(num_ave)+" ; disp="+str(num_disp)+" ; std="+str(num_std)+" ;
median="+str(num_median)+" ; mode="+str(num_mode))
print("For column salary: min=" + str(sal_min) + " ; max=" + str(sal_max) + " ;
ave=" + str(sal_ave) + " ; disp=" + str(sal_disp) + " ; std=" + str(sal_std) + " ;
median=" + str(sal_median) + " ; mode=" + str(sal_mode))
print("For column projects: min=" + str(pjt_min) + " ; max=" + str(pjt_max) + " ;
ave=" + str(pjt_ave) + " ; disp=" + str(pjt_disp) + " ; std=" + str(pjt_std) + " ;
median=" + str(pjt_median) + " ; mode=" + str(pjt_mode))

```

```

def ps_stat():

```

```

    MyDataFrame = ps.read_csv('data.csv')
    numbers=MyDataFrame["number"]
    num_min = numbers.min()
    num_max = numbers.max()
    num_ave = numbers.mean()
    num_disp = numbers.var()
    num_std = numbers.std()
    num_median = numbers.median()
    num_mode = mode(numbers)
    salary = MyDataFrame["salary"]
    sal_min = salary.min()
    sal_max = salary.max()
    sal_ave = salary.mean()
    sal_disp = salary.var()
    sal_std = salary.std()
    sal_median = salary.median()
    sal_mode = mode(salary)
    projects = MyDataFrame["completed_projects"]
    pjt_min = projects.min()
    pjt_max = projects.max()
    pjt_ave = projects.mean()
    pjt_disp = projects.var()
    pjt_std = projects.std()
    pjt_median = projects.median()

```

```

pjt_mode = mode(projects)
print(MyDataFrame.to_string())

print("")
print("Statistical characteristics")
print("")
print("For column number: min=" + str(num_min) + " ; max=" + str(num_max)
+ " ; ave=" + str(num_ave) + " ; disp=" + str(num_disp) + " ; std=" + str(num_std)
+ " ; median=" + str(num_median) + " ; mode=" + str(num_mode))
print("For column salary: min=" + str(sal_min) + " ; max=" + str(sal_max) + " ;
ave=" + str(sal_ave) + " ; disp=" + str(sal_disp) + " ; std=" + str(sal_std) + " ;
median=" + str(sal_median) + " ; mode=" + str(sal_mode))
print("For column projects: min=" + str(pjt_min) + " ; max=" + str(pjt_max) + " ;
ave=" + str(pjt_ave) + " ; disp=" + str(pjt_disp) + " ; std=" + str(pjt_std) + " ;
median=" + str(pjt_median) + " ; mode=" + str(pjt_mode))

if __name__ == '__main__':

    print("Generation process")
    print("")

    generate()

    print("")
    print("CSV file created (Press Enter)")
    input()

    print("Numpy statistics")
    print("")

    np_stat()

    print("")
    print("CSV file analyzed (Press Enter)")
    input()

    print("Pandas statistics")
    print("")

    ps_stat()

    print("")
    print("CSV file analyzed (Press Enter)")
    input()

```

```
print("Graphics")  
print("")
```

```
graph()
```

```
print("")  
print("CSV file removed")  
os.remove("data.csv")
```