МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

ИФТЭБ
*ИНСТИТУТ ФИНАНСОВЫХ ТЕХНОЛОГИЙ И ЭКОНОМИЧЕСКОЙ БЕЗОПАСНОСТИ*

КАФЕДРА 75 «ФИНАНСОВЫЙ МОНИТОРИНГ»

# Отчет по лабораторным работам №4, №5, №6
по курсу «Специальные технологии баз данных»

Выполнила
студентка группы С20-702
Нуритдинходжаева А.А.

Преподаватель: Манаенкова Т.А.

Москва 2023

**Лабораторная работа №4**
**Вариант 1**
**Задание 1**

Зарегистрируйтесь на сайте https://www.kaggle.com/datasets и загрузите с него набор статистических данных, посвящённый опросам людей https://www.kaggle.com/freecodecamp/2016-new-coder-survey-/version/1

1. На основе загруженного CSV-файла создайте Pandas DataFrame, подобрав
правильные типы данных столбцов.

2. Создайте новый Pandas DataFrame, выбрав только переменные EmploymentField, EmploymentStatus, Gender, JobPref, JobWherePref, MaritalStatus, Income.

3. Удалите все наблюдения, содержащие либо значения поля пол (Gender), отличные от male или female, либо значения NA (нет ответа) в каких-либо из полей.

4. Исследуйте связи между парами переменных (используйте только наблюдения, где эти поля заполнены):

   a. Gender, JobPref;

   b. Gender, JobWherePref;

   c. JobWherePref, MaritalStatus;

   d. EmploymentField, JobWherePref;

   e. EmploymentStatus, JobWherePref.

Выполняя исследование, не используйте процедуру ANOVA. Для каждой пары постройте таблицу сопряжённости, таблицу ожидаемых значений. Обоснованно выберите один из методов: хи-квадрат Пирсона, хи-квадрат Пирсона с поправкой Йейтса, точный критерий Фишера (обычный или на основе приближения МонтеКарло), точный критерий Фримана-Холтона (обычный или на основе приближения Монте-Карло).

5. Для каждой пары интерпретируйте результаты.

6. Замените переменную Income на три уровня дохода: низкий, средний, высокий.

7. Исследуйте связи между парой переменных Gender, Income (в новом формате) аналогично заданию 4. Интерпретируйте результаты.

```
import pandas as pd
import scipy.stats as sst

df = pd.read_csv('2016-Data.csv', delimiter = ',', parse_dates = ['Part1EndTime', 'Part1StartTime', 'Part2EndTime', 'Part2StartTime'],
                 dtype = {'CodeEventOther' : str, 'JobRoleInterestOther' : str})
pd.to_numeric(df['Age'], errors = 'coerce')
pd.to_numeric(df['Income'], errors = 'coerce')
pd.set_option('display.max_columns', 2000) #для того, чтобы ввыводило все столбцы
pd.set_option('display.width', 20000) #максимальная ширина, чтобы ничего не переносилось
print(df)
"""
import pandas as pd
from sklearn.preprocessing import StandardScaler

numeric_columns = df.select_dtypes(include=[np.number]).columns.tolist()

scaler = StandardScaler()
df[numeric_columns] = scaler.fit_transform(df[numeric_columns])
print(df)"""

#2
df1 = df[['EmploymentField', 'EmploymentStatus', 'Gender', 'JobPref', 'JobWherePref', 'MaritalStatus', 'Income']]
print(df1)

#3
df2 = df1.dropna()
df2 = df2[((df2['Gender'] == 'male') | (df2['Gender'] == 'female'))].dropna()
#pd.set_option('display.max_rows', None)
print(df2)

missing_values = df2.isnull().any()

print("Пропущенные значения в каждом столбце:")
print(missing_values)
```

```
all_fields_filled = not missing_values.any()
print(f"Все ли поля заполнены в DataFrame? {all_fields_filled}")

#4
#a
#Gender, JobPref
print("\n", "a", "\n")
sopr = pd.crosstab(df2.Gender, df2.JobPref, margins = True)
print("ТАБЛИЦА СОПРЯЖЕННОСТИ:", '\n', '\n', sopr, '\n')

sorp_exp = pd.crosstab(df2.Gender, df2.JobPref, margins = False)
exp = sst.contingency.expected_freq(sorp_exp)
print("ОЖИДАЕМЫЕ ЗНАЧЕНИЯ:", '\n', '\n', exp, '\n')

#хи2 статистика Пирсона
print(sst.chi2_contingency(sopr, correction = False))

#b
#Gender, JobWherePref
print("\n", "b", "\n")
sopr = pd.crosstab(df2.Gender, df2.JobWherePref, margins = True)
print("ТАБЛИЦА СОПРЯЖЕННОСТИ:", '\n', '\n', sopr, '\n')

sorp_exp = pd.crosstab(df2.Gender, df2.JobWherePref, margins = False)
exp = sst.contingency.expected_freq(sorp_exp)
print("ОЖИДАЕМЫЕ ЗНАЧЕНИЯ:", '\n', '\n', exp, '\n')

#хи2 статистика
print(sst.chi2_contingency(sopr, correction = False))

#c
#JobWherePref, MaritalStatus
print("\n", "c", "\n")
sopr = pd.crosstab(df2.JobWherePref, df2.MaritalStatus, margins = True)
print("ТАБЛИЦА СОПРЯЖЕННОСТИ:", '\n', '\n', sopr, '\n')

sorp_exp = pd.crosstab(df2.JobWherePref, df2.MaritalStatus, margins = False)
exp = sst.contingency.expected_freq(sorp_exp)
print("ОЖИДАЕМЫЕ ЗНАЧЕНИЯ:", '\n', '\n', exp, '\n')

#d
#EmploymentField, JobWherePref
print("\n", "d", "\n")
sopr = pd.crosstab(df2.EmploymentField, df2.JobWherePref, margins = True)
```

```
print("ТАБЛИЦА СОПРЯЖЕННОСТИ:", '\n', '\n', sopr, '\n')

sorp_exp = pd.crosstab(df2.EmploymentField, df2.JobWherePref, margins =
False)
exp = sst.contingency.expected_freq(sorp_exp)
print("ОЖИДАЕМЫЕ ЗНАЧЕНИЯ:", '\n', '\n', exp, '\n')

#e
#EmploymentStatus, JobWherePref
print("\n", "e", "\n")
sopr = pd.crosstab(df2.EmploymentStatus, df2.JobWherePref, margins = True)
print("ТАБЛИЦА СОПРЯЖЕННОСТИ:", '\n', '\n', sopr, '\n')

sorp_exp = pd.crosstab(df2.EmploymentStatus, df2.JobWherePref, margins =
False)
exp = sst.contingency.expected_freq(sorp_exp)
print("ОЖИДАЕМЫЕ ЗНАЧЕНИЯ:", '\n', '\n', exp, '\n')

#хи2 статистика
print(sst.chi2_contingency(sopr, correction = False))

#6
income_bins = [0, 50000, 80000, float('inf')]
income_labels = ['Низкий', 'Средний', 'Высокий']
df2['Income_Level'] = pd.cut(df2['Income'], bins=income_bins,
labels=income_labels, right=False)
print(df2)

df2 = df1.dropna()
df2 = df2[((df2['Gender'] == 'male') | (df2['Gender'] == 'female'))].dropna()
#pd.set_option('display.max_rows', None)
print(df2)

#7
sop = pd.crosstab(df2.Gender, df2.Income_Level, margins = True)
print("ТАБЛИЦА СОПРЯЖЕННОСТИ:", '\n', '\n', sopr, '\n')

sop_exp = pd.crosstab(df2.Gender, df2.Income_Level, margins = False)
exp = sst.contingency.expected_freq(sop_exp)
print("ОЖИДАЕМЫЕ ЗНАЧЕНИЯ:", '\n', '\n', exp, '\n')

#хи2 статистика
print(sst.chi2_contingency(sop, correction = False))
```

**Лабораторная работа №5**
**Вариант 1**
**Задание 1**

Зарегистрируйтесь на сайте https://www.kaggle.com/datasets и загрузите с него набор

статистических данных, посвящённый опросам людей

https://www.kaggle.com/freecodecamp/2016-new-coder-survey-/version/1

1. На основе загруженного CSV-файла создайте Pandas DataFrame, подобрав правильные типы данных столбцов.
2. Создайте новый Pandas DataFrame, выбрав только переменные CityPopulation EmploymentStatus Gender HasDebt JobPref JobWherePref MaritalStatus Income SchoolDegree.
3. Удалите все наблюдения, содержащие либо значения поля пол (Gender), отличные от male или female, либо значения NA (нет ответа) в каких-либо из полей.
4. С помощью однофакторного дисперсионного анализа проверьте, как доход зависит от SchoolDegree. При этом проверьте:

   a. Нормальность распределения дохода (методы Жака (Харке)-Бера, Шапиро-Уилка, Андерсона-Дарлинга, Колмогорова-Смирнова):

      i. Если нормальность не выполняется, выполните лог-трансформацию дохода и проверьте заново.

      ii. Если нормальность не выполняется, ограничьте выборку 100 первыми записями.

   b. Отсутствие автокорреляции (тест Дарбина — Уотсона);

   c. Гомоскедастичность (Omnibus Test);

   d. Отсутствие мультиколлинеарности (Cond. Number).
5. Дайте интерпретацию результатам.
6. Проанализируйте уровни с помощью теста Тьюки.

7. С помощью многофакторного дисперсионного анализа проверьте, как доход зависит от остальных переменных, включите в проверку комбинацию Gender и MaritalStatus. При этом проверьте:

a. Нормальность распределения дохода (методы Жака (Харке)-Бера, Шапиро-Уилка, Андерсона-Дарлинга, Колмогорова-Смирнова):

i. Если нормальность не выполняется, выполните лог-трансформацию дохода и проверьте заново.

ii. Если нормальность не выполняется, попробуйте применить какие либо-методы, описанные в [1] (стр. 27) (Просто можно о них знать, какие существуют).

iii. Если нормальность не выполняется, ограничьте выборку 100 первыми записями.

b. Отсутствие автокорреляции (тест Дарбина — Уотсона);

c. Гомоскедастичность (Omnibus Test);

d. Отсутствие мультиколлинеарности (Cond. Number).

8. Дайте интерпретацию результатам.

```
import pandas as pd
import scipy.stats as sst
import numpy as np
import statsmodels.api as sm
from statsmodels.formula.api import ols
import matplotlib.pyplot as plt
import seaborn as sns

#1
df = pd.read_csv('2016-FCC-Data.csv', delimiter = ',', parse_dates =
['Part1EndTime', 'Part1StartTime', 'Part2EndTime', 'Part2StartTime'],
                 dtype = {'CodeEventOther' : str, 'JobRoleInterestOther' : str})
#df = pd.read_csv('2016-FCC-New-Coders-Survey-Data.csv', delimiter = ',',
parse_dates = ['Part1EndTime', 'Part1StartTime', 'Part2EndTime',
'Part2StartTime'],
  #               dtype = {'CodeEventOther' : str, 'JobRoleInterestOther' : str})
pd.to_numeric(df['Income'], errors = 'coerce')

pd.set_option('display.max_columns', 2000)
```

```
pd.set_option('display.width', 20000)
print(df)

#2
df1 = df[['CityPopulation', 'EmploymentStatus', 'Gender', 'HasDebt', 'JobPref',
'JobWherePref', 'MaritalStatus', 'Income', 'SchoolDegree']]
print(df1)

#3
df1 = df1.dropna().sample(100, random_state=1234567)
df1 = df1[((df1['Gender'] == 'male') | (df1['Gender'] == 'female'))]
print(df1)

#4
  print(df1['SchoolDegree'].unique())

  g1 = df1[df1['SchoolDegree'] == "master's degree (non-professional)"]['Income']
  g2 = df1[df1['SchoolDegree'] == 'high school diploma or equivalent
(GED)']['Income']
  g3 = df1[df1['SchoolDegree'] == 'some college credit, no degree']['Income']
  g4 = df1[df1['SchoolDegree'] == "bachelor's degree"]['Income']
  g5 = df1[df1['SchoolDegree'] == 'professional degree (MBA, MD, JD,
etc.)']['Income']
  g6 = df1[df1['SchoolDegree'] == 'trade, technical, or vocational training'
]['Income']
  g7 = df1[df1['SchoolDegree'] == "associate's degree"]['Income']
  g8 = df1[df1['SchoolDegree'] == 'Ph.D.']['Income']
  g9 = df1[df1['SchoolDegree'] == 'some high school']['Income']
  g10 = df1[df1['SchoolDegree'] == 'no high school (secondary school)']['Income']

  print(sst.f_oneway(g1, g2, g3, g4, g5, g6, g7, g8, g9, g10), '\n')

#a
#Жака-Бера
print('Жака-Бера', '\n', '\n', sst.jarque_bera(df1['Income']), '\n') #отвергается

#Шапиро-Уилка
print('Шапиро-Уилка', '\n', '\n', sst.shapiro(df1['Income']), '\n') #отвергается

#Андерсона-Дарлинга
print('Андерсона-Дарлинга', '\n', '\n', sst.anderson(list(df1['Income'])),
'\n')#отвергаетс

#Колмогорова-Смирнова
```

```
print('Колмогорова-Смирнова', '\n', '\n', sst.kstest(list(df1['Income']), 'norm'), '\n')
#отвергается

# распределение не нормальное 0,05 > 2,44e-28

model = ols('Income ~ SchoolDegree', df1).fit()
print(model.summary())

#лог-трансформация
df1['Income'] = df1[['Income']].applymap(lambda x: np.log(x + 1))
model = ols('Income ~ SchoolDegree', df1).fit()
print(model.summary())

# зависимость лог-транформированного среднего дохода от образования
есть, 6,05e-06<0.05

# нормальность есть 0,54>0.05

#автокорреляция
df2 = pd.get_dummies(df1, columns=['SchoolDegree'], drop_first=True)
x = df2[['SchoolDegree_associate\'s degree','SchoolDegree_bachelor\'s
degree','SchoolDegree_high school diploma or equivalent
(GED)','SchoolDegree_master\'s degree (non-professional)','SchoolDegree_no high
school (secondary school)','SchoolDegree_professional degree (MBA, MD, JD,
etc.)','SchoolDegree_some college credit, no degree','SchoolDegree_some high
school','SchoolDegree_trade, technical, or vocational training']]
y = df2['Income']
x_ =sm.add_constant(x)
model = sm.OLS(y,x_).fit()
print(sm.stats.durbin_watson(model.resid))

#c
#гомоскедостичность
g1 = df1[df1['SchoolDegree'] == "master's degree (non-professional)"]['Income']
g2 = df1[df1['SchoolDegree'] == 'high school diploma or equivalent
(GED)']['Income']
g3 = df1[df1['SchoolDegree'] == 'some college credit, no degree']['Income']
g4 = df1[df1['SchoolDegree'] == "bachelor's degree"]['Income']
g5 = df1[df1['SchoolDegree'] == 'professional degree (MBA, MD, JD,
etc.)']['Income']
g6 = df1[df1['SchoolDegree'] == 'trade, technical, or vocational training'
]['Income']
g7 = df1[df1['SchoolDegree'] == "associate's degree"]['Income']
g8 = df1[df1['SchoolDegree'] == 'Ph.D.']['Income']
g9 = df1[df1['SchoolDegree'] == 'some high school']['Income']
```

```
g10 = df1[df1['SchoolDegree'] == 'no high school (secondary school)']['Income']

print(sst.levene(g1, g2, g3, g4, g5, g6, g7, g8, g9, g10))

g1 = df1[df1['SchoolDegree'] == "master's degree (non-professional)"]
g2 = df1[df1['SchoolDegree'] == 'high school diploma or equivalent (GED)']
g3 = df1[df1['SchoolDegree'] == 'some college credit, no degree']
g4 = df1[df1['SchoolDegree'] == "bachelor's degree"]
g5 = df1[df1['SchoolDegree'] == 'professional degree (MBA, MD, JD, etc.)']
g6 = df1[df1['SchoolDegree'] == 'trade, technical, or vocational training' ]
g7 = df1[df1['SchoolDegree'] == "associate's degree"]
g8 = df1[df1['SchoolDegree'] == 'Ph.D.']
g9 = df1[df1['SchoolDegree'] == 'some high school']
g10 = df1[df1['SchoolDegree'] == 'no high school (secondary school)']
plt.ylim(7, 15)
plt.boxplot((g1['Income'], g2['Income'], g3['Income'], g4['Income'], g5['Income'],
g6['Income'], g7['Income'], g8['Income'],
            g9['Income'], g10['Income']), labels = ["master's degree (non-
professional)", 'high school diploma or equivalent (GED)',
                                        'some college credit, no degree', "bachelor's
degree",
                                        'professional degree (MBA, MD, JD, etc.)',
                                        'trade, technical, or vocational training',
"associate's degree", 'Ph.D.',
                                        'some high school', 'no high school (secondary
school)'])
plt.show()

import seaborn as sns
import matplotlib.pyplot as plt

sns.distplot(g1['Income'], label="master's degree (non-professional)")
sns.distplot(g2['Income'], label='high school diploma or equivalent (GED)')
sns.distplot(g3['Income'], label='some college credit, no degree')
sns.distplot(g4['Income'], label="bachelor's degree")
sns.distplot(g5['Income'], label='professional degree (MBA, MD, JD, etc.)')
sns.distplot(g6['Income'], label='trade, technical, or vocational training')
sns.distplot(g7['Income'], label="associate's degree")
sns.distplot(g8['Income'], label='Ph.D.')
sns.distplot(g9['Income'], label='some high school')
sns.distplot(g10['Income'], label='no high school (secondary school)')

plt.xlim(5, 15)
#plt.legend()
plt.show()
```

```
#критерий Тьюки
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.stats.multicomp import MultiComparison

mc= MultiComparison(df1['Income'],df1['SchoolDegree'])
mc_results=mc.tukeyhsd()
print(mc_results)

#многофакторный анализ
model = ols('Income ~ C(Gender)*C(MaritalStatus)', df1).fit()
print(model.summary())

an_1 = sm.stats.anova_lm(model, typ = 1)
an_2 = sm.stats.anova_lm(model, typ = 2)
an_3 = sm.stats.anova_lm(model, typ = 3)
print(an_1)
print(an_2)
print(an_3)

model = ols('Income ~ C(Gender)+C(MaritalStatus)', df1).fit()
print(model.summary())

an_1 = sm.stats.anova_lm(model, typ = 1)
an_2 = sm.stats.anova_lm(model, typ = 2)
an_3 = sm.stats.anova_lm(model, typ = 3)
print(an_1)
print(an_2)
print(an_3)

mc= MultiComparison(df1['Income'],df1['Gender'])
mc_results=mc.tukeyhsd()
print(mc_results)

mc= MultiComparison(df1['Income'],df1['MaritalStatus'])
mc_results=mc.tukeyhsd()
print(mc_results)
```

**Лабораторная работа №6**
**Вариант 1**
**Задание**

1. Загрузите с указанного в вашем варианте адреса набор данных.

2. Импортируйте загруженные данные в набор данных в Pandas DataFrame.

3. Постройте 90%, 95%, 99% предективные эллипсы для всех возможных пар полей, кроме поля, содержащего классификацию наблюдений.

4. Выполните стандартизацию всех полей, кроме поля, содержащего классификацию наблюдений.

5. Выполните расчёт главных компонент для загруженного набора данных, используя все числовые поля, кроме поля, содержащего классификацию наблюдений.

6. Определите главные компоненты, охватывающие 95% изменчивости данных и формулы для их вычисления на основе исходных столбцов.

7. Определите главные компоненты согласно методу Кайзера и формулы для их вычисления на основе исходных столбцов.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math
from sklearn import preprocessing
from sklearn.decomposition import PCA
from matplotlib.patches import Ellipse
import matplotlib.transforms as transforms

#2
f = open('winequality-red.dat', 'r')
skip_rows = 0
COLUMNS = []
for line in f:
  if line[0] == '@':
    string = line.split(' ')
    if string[0] == '@attribute':
      COLUMNS.append(string[1])
    skip_rows += 1
COLUMNS = list(COLUMNS)
f.close()

df = pd.read_table('winequality-red.dat', skiprows=skip_rows, sep=',', names=COLUMNS)
```

```
print(df)

#3
def confidence_ellipse(x,y,ax,p_value, facecolor = 'none', **kwargs):
  if x.size != y.size:
    raise ValueError("x и y должны быть одного размера")

  cov = np.cov(x,y)
  pearson = cov[0,1]/np.sqrt(cov[0,0]*cov[1,1])
  ell_radius_x = np.sqrt(1+pearson)
  ell_radius_y = np.sqrt(1-pearson)
  ellipse = Ellipse((0,0),
                    width = ell_radius_x*2,
                    height = ell_radius_y*2,
                    facecolor = facecolor,
                    **kwargs)

  if p_value >0 and p_value <1:
    n_std = math.sqrt(-2*math.log(p_value))
  else:
    print ("Error")

  scale_x = np.sqrt(cov[0,0]) * n_std
  mean_x = np.mean(x)

  scale_y = np.sqrt(cov[1, 1]) * n_std
  mean_y = np.mean(y)

  transf = transforms.Affine2D() \
      .rotate_deg(45) \
      .scale(scale_x,scale_y) \
      .translate(mean_x,mean_y)
  ellipse.set_transform(transf + ax.transData)
  return ax.add_patch(ellipse)


fig, ax = plt.subplots(figsize =(8,8))
confidence_ellipse(df['FixedAcidity'], df['VolatileAcidity'], ax, 0.1, edgecolor =
'green')
confidence_ellipse(df['FixedAcidity'], df['VolatileAcidity'], ax, 0.05, edgecolor =
'red')
confidence_ellipse(df['FixedAcidity'], df['VolatileAcidity'], ax, 0.01, edgecolor =
'blue')
ax.scatter(df['FixedAcidity'], df['VolatileAcidity'], s = 10)
plt.show()
```

```
fig, ax = plt.subplots(figsize =(8,8))
confidence_ellipse(df['FixedAcidity'], df['CitricAcid'], ax, 0.1, edgecolor = 'green')
confidence_ellipse(df['FixedAcidity'], df['CitricAcid'], ax, 0.05, edgecolor = 'red')
confidence_ellipse(df['FixedAcidity'], df['CitricAcid'], ax, 0.01, edgecolor = 'blue')
ax.scatter(df['FixedAcidity'], df['CitricAcid'], s = 10)
plt.show()

#4
count_range1 = preprocessing.StandardScaler()
x = count_range1.fit_transform(df[['FixedAcidity']])
df['Normal FixedAcidity']= x[0:]

count_range2 = preprocessing.StandardScaler()
x = count_range2.fit_transform(df[['VolatileAcidity']])
df['Normal VolatileAcidity']= x[0:]

count_range3 = preprocessing.StandardScaler()
x = count_range3.fit_transform(df[['CitricAcid']])
df['Normal CitricAcid']= x[0:]

count_range4 = preprocessing.StandardScaler()
x = count_range4.fit_transform(df[['ResidualSugar']])
df['Normal ResidualSugar']= x[0:]

count_range5 = preprocessing.StandardScaler()
x = count_range5.fit_transform(df[['Chlorides']])
df['Normal Chlorides']= x[0:]

count_range6 = preprocessing.StandardScaler()
x = count_range6.fit_transform(df[['FreeSulfurDioxide']])
df['Normal FreeSulfurDioxide']= x[0:]

count_range7 = preprocessing.StandardScaler()
x = count_range7.fit_transform(df[['TotalSulfurDioxide']])
df['Normal TotalSulfurDioxide']= x[0:]

count_range8 = preprocessing.StandardScaler()
x = count_range8.fit_transform(df[['Density']])
df['Normal Density']= x[0:]

count_range9 = preprocessing.StandardScaler()
x = count_range9.fit_transform(df[['PH']])
df['Normal PH']= x[0:]
```

```
count_range10 = preprocessing.StandardScaler()
x = count_range10.fit_transform(df[['Sulphates']])
df['Normal Sulphates']= x[0:]

count_range11 = preprocessing.StandardScaler()
x = count_range11.fit_transform(df[['Alcohol']])
df['Normal Alcohol']= x[0:]

df

#5
pca = PCA(n_components= 11)
df1 = pca.fit_transform(df[['Normal FixedAcidity', 'Normal VolatileAcidity','Normal CitricAcid',
'Normal ResidualSugar', 'Normal Chlorides', 'Normal FreeSulfurDioxide','Normal TotalSulfurDioxide', 'Normal Density',
'Normal PH', 'Normal Sulphates', 'Normal Alcohol']])

cov = pca.get_covariance()
print('Матрица ковариации', '\n', cov, '\n')

[en, ev] = np.linalg.eig(cov)
print('Матрица преобразования', '\n',ev)

#6
explained_variance = pca.explained_variance_ # Собсивенные числа
explained_variance_ratio = pca.explained_variance_ratio_ # доля важности каждой новойкомпоненты на исходную модель
print (explained_variance)
print (explained_variance_ratio)

#7
# Согласно Кайзеру, мы берем все ГК которые больше 1, тем самым для снижения признаков пр-ва нужно взять первые 4 ГК
```