МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ
до комп'ютерного практикуму на Python
Частина 1

«Описова статистика та статистична діагностика»

з курсу «Математична статистика»
для студентів спеціальностей
113 «Прикладна математика»,
122 «Комп’ютерні науки»

Затверджено
редакційно-видавничою
радою університету,
протокол № від 25.06.2020 р.

Харків
НТУ «ХПІ»
2020

Методичні вказівки до виконання індивідуальних завдань комп'ютерного практикуму на Python з курсу «Математична статистика» для студентів технічних дисциплін / уклад. О.О. Ларін, О. І. Суханова – Харків : НТУ «ХПІ». – 48 с.

Укладач О.О. Ларін, О. І. Суханова

Рецензент О.О. Водка

Кафедра динаміки та міцності машин

Вступ

Сучасна підготовка студентів з математичної статистики потребує викладання матеріалу багатьох питань, в тому числі, питання описової статистики та статистичної діагностики.

Лабораторний практикум проводиться на мові Python. Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня. (Найчастіше вживане прочитання — «Па́йтон», запозичено назву з британського шоу Монті Пайтон [Wikipedia]).

Пропонується використовувати пакет бібліотек та компілятор з дистрибутиву **Anaconda** та проводити розробку в середовищі **PyCharm**.

Anaconda – дистрибутив для мов програмування Python та R з відкритим кодом для обробки даних великого об'єму, побудови аналітичних прогнозів і наукових обчислень [Wikipedia].

PyCharm — інтегроване середовище розробки для мови програмування Python. Надає засоби для аналізу коду, графічний зневаджувач, інструмент для запуску юніт-тестів і підтримує веб-розробку на Django. PyCharm розроблена чеською компанією JetBrains на основі IntelliJ IDEA [Wikipedia].

Для виконання лабораторного практикуму необхідно завантажити та встановити на комп’ютері: **Anaconda 5.0.0 for Windows Installer (Python 3.6 version)** (https://www.anaconda.com/download/) та **PyCharm Community Edition** (https://www.jetbrains.com/pycharm/).

Лабораторна робота №1

Знайомство з мовою Python та середовищам PyCharm.

Побудова апроксимацій типових сигналів рядами Фур'є.

Для створення першої програми створіть новий проект.

(MainMenu->File->NewProject)

Додайте новий файл

(MainMenu->File->New->PythonFile або у дереві проекту правою кнопкою миші New->PythonFile)

У редакційному вікні введіть наступний код:

```
print('Hello world!')
```

Для компіляції та виконання програми: **MainMenu-Run->Run**

Перша компіляція програми може мати певну затримку у часі, що пов'язана з початковою індексацією файлів середовищем PyCharm. Слід зачекати одо 5 хвилин після першого запуску середовища до початку повноцінної роботи.

Результат роботи програми можна бачити у новому вікні, що з'явиться в нижній частині програми.

Наступним кроком лабораторної роботи є створення програми, яка будує графік функції. Для реалізації цієї задачі потрібно додати бібліотеки з математичними функціями (**math**) та засобами побудови графіків (**matplotlib.pyplot**). Крім того знадобиться використання масивів, операції з якими доступні в рамках використання бібліотеки чисельних методів **numpy**. Для додавання у програму функцій з бібліотеки **math** необхідно використати команду: *import math* Далі в коді програми можна посилатись на функції математичної бібліотеки. Приклад програми розрахунку синуса:

```
import math

a=math.sin(3.1415)
b=math.cos(math.pi)
print(a)
print(b)
```

Іншим шляхом завантаження функцій зовнішніх бібліотек є імпорт всіх імен з модуля командою *from math import* * У цьому випадку можливе безпосереднє використання функції модуля у програмі без згадування його імені.

```
from math import *

a=sin(3.1415)
b=cos(pi)
print(a)
print(b)
```

Проте можливе перекривання імен з різних бібліотек, що може призвести до появи помилок у коді. Тому краще використовувати функцію імпорту або використовувати її перейменування на більш коротке та зручне ім’я. Наприклад для бібліотеки numpy часто використовують скорочене ім’я **np**, а для бібліотеки **matplotlib.pyplot** – **plt**. Для цього використовується код типу: **import numpy as np**

Для побудови графіку функцій створюємо два масиви x та y, які спочатку заповнюються нулями, а потім в циклі отримають необхідні розрахункові значення.

Слід звернути увагу, що у мові Python тема операторних дужок (як {…} в C++ або Begin …end в Pascal). Для формування тіла циклу або умови тощо використовується форматування тексту коду, тобто тілом циклу є код який слідує за оператором та має відступ від краю на чотири пробіла.

Функція plot має синтаксис ідентичний з прийнятим у MatLab: Перший та другий параметр є масиви, що задають абсцису та ординату графіка, а далі у одинарних лапках йде маркер для лінії та її колір. Наприклад: plot(x,y,'--r') зображує графік червоною пунктирною лінією, а код plot(x,y,'-b') – суцільною синьою.

```
from math import *
import numpy as np
import matplotlib.pyplot as plt

N=int(100)

x=np.zeros(N,float)
y=np.zeros(N,float)

x0=0
xn=10
dx=(xn-x0)/N

x[0]=x0
for n in range(1,N):
    x[n]=x[n-1]+dx
    y[n]=x[n]**2*sin(x[n])

plt.plot(x,y,'--r')
plt.grid()
plt.show()
```

Результат роботи програми з’явиться у окремому вікні

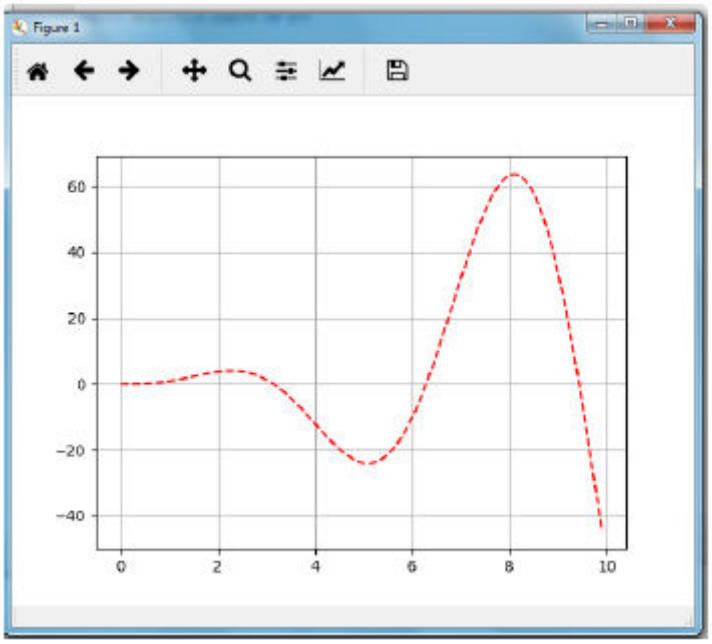

Рисунок 1 – Результат побудови графіка функції $x^2\mathrm{Sin}(x)$

Наступною частиною лабораторної роботи є побудова апроксимації рядом Фур'є прямокутного імпульсу, що задається виразом:

$$y(t)=\begin{cases}-\frac{1}{2},-\pi\le t\le 0\\ \frac{1}{2},0\le t\le \pi\end{cases}$$

Графік цієї функції представлено на рис. 2, а нижче код для його побудови

```
x0=-pi
xn=pi
dx=(xn-x0)/N
x[0]=x0

for n in range(1,N):
    x[n]=x[n-1]+dx

    if ((x[n]>-pi) and (x[n]<0)):
        y[n]=-1/2
    elif ((x[n]>0) and (x[n]<pi)):
        y[n]=1/2

plt.plot(x,y,'--r')
```

```
plt.grid()
plt.show()
```

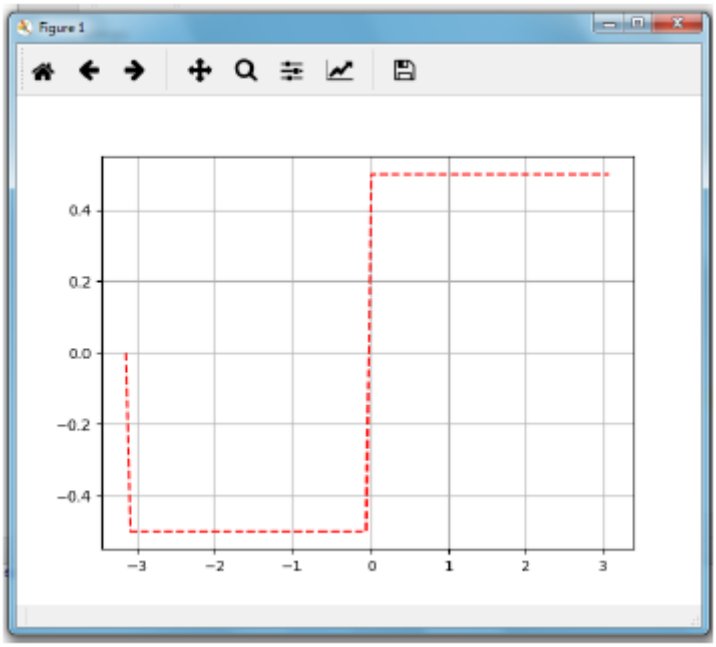

Рисунок 2 – Функція прямокутного імпульсу

Оскільки дана функція є нечьотною, то її ряд Фур'є є рядом синусів, коефіцієнти якого визначаються за формулою:

$$y(t)=\sum_{k=1}^{\infty} b_k \sin(kt), \quad b_k=\frac{1}{\pi}\int_{-\pi}^{\pi} y(t)\sin(kt)dt=\frac{1}{\pi}\int_{0}^{\pi}\sin(kt)dt$$

$$b_k=\frac{1}{\pi}\frac{1+(-1)^{k+1}}{k}=\begin{cases}\frac{2}{\pi k}, k-нечьотні\\ 0, k-чьотні\end{cases}$$

Нижче представлено код побудови апроксимації функції прямокутного імпульсу рядом Фур'є та результат роботи програми на рис.3

```
Nf=10
b=np.zeros(Nf,float)
Fb=np.zeros(Nf,float)

# to find Fourie coeff
for k in range(1,Nf):
    if k % 2 == 0: # even
        b[k]=0
    else: # odd
```

```
        b[k]=2/pi/k
for n in range(1,N):
    x[n]=x[n-1]+dx

    z[n] = b[1] * sin(x[n])
    for k in range(2,Nf):
        z[n]=z[n]+b[k]*sin(k*x[n])

plt.plot(x,z,'-b')
plt.plot(x,y,'--r')
plt.grid()
plt.show()
```

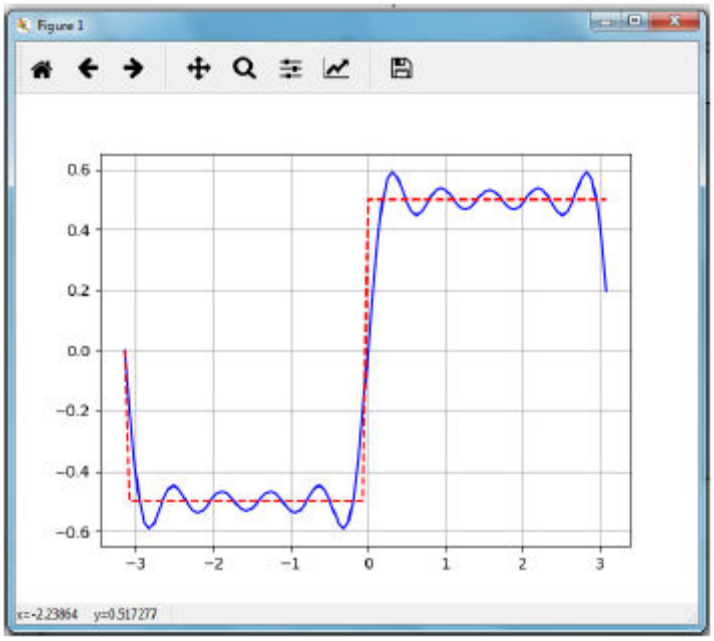

Рисунок 3 – Апроксимації прямокутного імпульсу рядом Фур’є

Завдання для самостійного виконання:

Визначте скільки яленів ряду Фур’є необхідно для якісної апроксимації функції прямокутного імпульсу.

Побудуйте зведений графік, що демонструє збіжність ряду Фур’є до шуканої функції.

Проведіть апроксимації за межі періоду (-π,π).

Проведіть аналогічний аналіз для функції сигналу у вигляді «піли», що на проміжку (-π,π) має вираз:

$$y(t) = t, -\pi \le t \le \pi$$

Лабораторна робота №2

Вирішення задачі діагностики методом Байеса

Створіть новий проект.

(MainMenu->File->NewProject)

Додайте новий файл

(MainMenu->File->New->PythonFile або у дереві проекту правою кнопкою миші **New->PythonFile)**

Створюємо просту програму для визначення діагнозу підшипників відповідно до наступних умов:

З 1000 обстежених підшипників передньої підвіски автомобілів 900 підшипників виробили ресурс в справному стані і 100-в несправному. Всі підшипники були обстежені за такими ознаками:

- загальний рівень вібрації;

- температура;

- забруднення мастила.

У 70% справних підшипників загальний рівень вібрації лежав в діапазоні від 0,25 до 0,5 g, у 20% справних підшипників - від 0,5 до 0,75 g і у 10% -> 0,75g. У 80% справних підшипників температура лежала в діапазоні 50-70 град, у 10% - в діапазоні 70-90 град. І у 10% -> 90 град. У 90% справних підшипників забруднення мастила було в межах норми. У 80% несправних підшипників спостерігалася вібрація> 0,75 g, у 15% несправних підшипників вібрація в діапазоні 0,5-0,75g. У 85% несправних підшипників температура була> 90 град, у 8% несправних підшипників - в діапазоні 70-90 град. У 70% несправних підшипників забруднення мастила було вище норми.

Визначити: Імовірність справного стану підшипника при спостереженні вібрації в діапазоні 0,25-0,5g, температури - 50-70 град, забруднення мастила в межах норми.

```
from math import *
import numpy as np
import matplotlib.pyplot as plt

N=1000
D1=900 # исправный подшипник
D2=100 # неисправный подшипник

# K1 - вибрация
# K11 - вибрация в диапазоне 0,25-0,5
# K12 - вибрация в диапазоне 0,5-0,75
# K13 - вибрация в диапазоне >0,75

PrK11_D1=0.7
PrK12_D1=0.2
PrK13_D1=0.1

PrK11_D2=0.05
PrK12_D2=0.15
PrK13_D2=0.8

# K2 - температура
# K21 - температура в диапазоне 50-70 C
# K22 - температура в диапазоне 70-90 C
# K23 - температура в диапазоне >90 C

PrK21_D1=0.8
PrK22_D1=0.1
PrK23_D1=0.1

PrK21_D2=0.07
PrK22_D2=0.08
PrK23_D2=0.85

# K3 - загрязнение смазки
# K31 - в пределах нормы
# K32 – повышенное

PrK31_D1=0.9
PrK32_D1=0.1

PrK31_D2=0.3
PrK32_D2=0.7

# K - k11*k22*k31
print('При наблюдении: ')
print(' вибрации в диапазоне 0.25-0,5 (признак k11);')
print(' температуре в диапазоне 50-70 (признак k22);')
print(' загрязнение смазки в пределах нормы (признак k31).')
```

```
PrK_D1=PrK11_D1*PrK22_D1*PrK31_D1
PrK_D2=PrK11_D2*PrK22_D2*PrK31_D2

PrD1=D1/N
PrD2=D2/N

PrD1_K=PrD1*PrK_D1/(PrD1*PrK_D1+PrD2*PrK_D2)
PrD2_K=PrD2*PrK_D2/(PrD1*PrK_D1+PrD2*PrK_D2)

print('Вероятность исправного состояния (диагноз D1) составляет:')
print(PrD1_K)
print('Вероятность неисправного состояния (диагноз D2) составляет: ')
print(PrD2_K)
```

Самостійно відповідно до Вашого варіанту:

1. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні 0,5-0,75g, температури - 50-70 ^{o}C, забруднення мастила в нормі.

2. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні > 0,75g, температури - 50-70 ^{o}C, забруднення мастила в нормі.

3. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні 0,25-0,5g, температури - 70-90 ^{o}C, забруднення мастила в нормі.

4. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні 0,5-0,75g, температури - 70-90 ^{o}C, забруднення мастила в нормі.

5. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні > 0,75g, температури - 70-90 ^{o}C, забруднення мастила в нормі.

6. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні 0,25-0,5g, температури -> 90 ^{o}C, забруднення мастила в нормі.

7. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні 0,5-0,75g, температури -> 90 ^{o}C, забруднення мастила в нормі.

8. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні > 0,75g, температури -> 90 ^{o}C, забруднення мастила в межах норми.

9. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні 0,25-0,5g, температури - 50-70 ^{o}C, забруднення мастила вище норми.

10. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні > 0,75g, температури - 50-70 ^{o}C., Забруднення мастила вище норми.

11. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні 0,25-0,5g, температури - 70-90 ^{o}C, забруднення мастила вище норми.

12. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні 0,5-0,75g, температури - 70-90 ^{o}C. Забруднення мастила вище норми.

13. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні - > 0,75g, температури - 70-90 ^{o}C, забруднення мастила вище норми.

14. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні 0,25-0,5g, температури - 70-90 ^{o}C, забруднення мастила вище норми.

15. Імовірність справного стану підшипника при спостереженні вібрації в діапазоні 0,5-0,75g, температури - 70-90 ^{o}C, забруднення мастила вище норми.

Додамо зміни у програму з метою підвищення рівня її можливої автоматизації

```
PrKD=np.zeros((4,4,3))
PrD=np.zeros(4)
PrK_D=np.zeros(4)
PrD_K=np.zeros(4)

PrKD[1,1,1]=PrK11_D1
PrKD[1,2,1]=PrK12_D1
PrKD[1,3,1]=PrK13_D1

PrKD[1,1,2]=PrK11_D2
PrKD[1,2,2]=PrK12_D2
PrKD[1,3,2]=PrK13_D2

PrKD[2,1,1]=PrK21_D1
PrKD[2,2,1]=PrK22_D1
PrKD[2,3,1]=PrK23_D1

PrKD[2,1,2]=PrK21_D2
PrKD[2,2,2]=PrK22_D2
PrKD[2,3,2]=PrK23_D2
PrKD[3,1,1]=PrK31_D1
```

```
PrKD[3,2,1]=PrK32_D1
PrKD[3,1,2]=PrK31_D2
PrKD[3,2,2]=PrK32_D2

D=[0,900,100]
K=[0,1,3,2]

for i in range(1,np.size(D)):
    PrD[i] = D[i] / N
    PrK_D[i]=PrKD[1,K[1],i]
    for j in range(2,np.size(K)):
        PrK_D[i]=PrK_D[i]*PrKD[j,K[j],i]

PrK=PrD[1]*PrK_D[1]
for i in range(2,np.size(D)):
    PrK=PrK+PrD[i]*PrK_D[i]

for i in range(1,np.size(D)):
    PrD_K[i]=PrD[i]*PrK_D[i]/PrK

print('----------')
print('Вероятность исправного состояния (диагноз D1) составляет:')
print(round(PrD_K[1]*100,2),'%')

X=[1,2]
Y=[round(PrD_K[1]*100,2),round(PrD_K[2]*100,2)]

plt.bar(X,Y)
plt.text(X[0],Y[0]/2,'Исправное состояние',ha='center')
plt.text(X[0],Y[0]/3,Y[0],ha='center')
plt.text(X[1],Y[1]/2,'Неисправное состояние',ha='center')
plt.text(X[1],Y[1]/3,Y[1],ha='center')
plt.xlabel('Диагнозы')
plt.ylabel('Вероятность диагноза')
plt.show()
```

Самостійно додайте зміни у програму так аби вхідними даними були виключно кількісні показники спостереження, а не апріорні ймовірності, які мають розраховуватись із даних відомих з спостереження.

Після чого по варіантах розв’яжіть наступні задачі.

Уточнити апріорні ймовірності появи справного і несправного станів, а також умовні ймовірності ознак, якщо в результаті обстеження 1001 підшипника встановлено, що у нього:

1. було справний стан і спостерігалися: вібрація 0,25-0,5g, температура - 50-70 град, забруднення мастила в нормі.

2. було справний стан і спостерігалися: вібрація 0,5-0,75g, температура - 50-70 град, забруднення мастила в межах норми.

3. було справний стан і спостерігалися: вібрація> 0,75g, температура - 50-70 град, забруднення мастила в межах норми.

4. було справний стан і спостерігалися: вібрація 0,25-0,5g, температура - 70-90 град, забруднення мастила в межах норми.

5. було справний стан і спостерігалися: вібрація 0,5-0,75g, температура - 70-90 град, забруднення мастила в межах норми.

6. було справний стан і спостерігалися: вібрація -> 0,75g, температура - 70-90 град, забруднення мастила в межах норми.

7. було справний стан і спостерігалися: вібрація 0,25-0,5g, температура -> 90 град, забруднення мастила в межах норми.

8. було справний стан і спостерігалися: вібрація 0,5-0,75g, температура -> 90 град., забруднення мастила в межах норми.

9. було справний стан і спостерігалися: вібрація> 0,75g, температура -> 90 град., забруднення мастила в межах норми.

10. було справний стан і спостерігалися: вібрація 0,25-0,5g, температура - 50-70 град, забруднення мастила вище норми.

11. було справний стан і спостерігалися: вібрація 0,5-0,75g,температура - 50-70 град, забруднення мастила вище норми.

12. було справний стан і спостерігалися: вібрація> 0,75g, температура - 50-70 град, забруднення мастила вище норми.

13. було справний стан і спостерігалися: вібрація 0,25-0,5g, температура - 70-90 град, забруднення мастила вище норми.

14. було, справний стан і спостерігалися: вібрація 0,5-0,75g, температура - 70-90 град, забруднення мастила вище норми.

15. було справний стан і спостерігалися: вібрація -> 0,75g, температура - 70-90 град, забруднення мастила вище норми.

Лабораторна робота №3

Вирішення задачі медичної діагностики методом Байеса

В результаті спостереження за історіями хвороби хворих з трьома захворюваннями печінки:

- камені жовчних проток;
- аскаридоз жовчних проток;
- паренхіматозний гепатит;

були отримані статистичні дані, які збережені у табличному вигляді.

Для диференціальної діагностики використовують 6 ознак:

- вік (до 20 років, від 20 до 40 років, 40-60 років, понад 60 років);
- блювота (є, відсутня);
- жовтушність (склер (очей), шкіри і слизових, відсутня);
- напади болю в правому підреберї (є, немає);
- збільшення печінки (збільшена, не збільшена);
- апетит (є, відсутній).

Статистичні дані можуть бути завантажені за посиланням відповідно до варіантів:

https://sites.google.com/site/oleksiylarin/given-courses

Необхідно відповідно до свого варіанту завантажити 3 файли із даними щодо симптомів у хворим на різні діагнози:

Ascariasis_var i.csv

Hepatitis_var i.csv

Stones_var i.csv

Необхідно також завантажити файл **Example.csv** для виконання прикладу роботи програми, як подана нижче.

Для виконання лабораторної роботи необхідно:

Створіть новий проект.

(MainMenu->File->NewProject)

Додайте новий файл

(MainMenu->File->New->PythonFile або у дереві проекту правою кнопкою миші **New->PythonFile)**

Нижче наведено приклад коду. Який демонструє можливості та особливості мови Python для того аби провести завантаження та почати обробку даних із текстових файлів.

```
import numpy as np

Sympthoms=np.zeros((7,5))

# Open the file with read only permit
text_file=open("Example.csv", "r")
lin=text_file.readlines()
text_file.close()

D1Size=len(lin)
Diagnose=D1Size-1

print(Diagnose)

for i in range(1,D1Size):
    print(lin[i])
    data=lin[i].split(';')

    Age=int(data[0])
    if Age<20:
        Sympthoms[1,1] += 1
    elif (Age>=20) and (Age<40):
        Sympthoms[1,2] += 1
    elif (Age>=40) and (Age<60):
        Sympthoms[1,3] += 1
    else:
        Sympthoms[1,4] += 1

    if data[2]=='eye':
        Sympthoms[3,1] += 1
    elif data[2]=='skin':
        Sympthoms[3,2] += 1
    else:
        Sympthoms[3,3] += 1

PrKD=Sympthoms/Diagnose

print(PrKD)
```

Необхідно самостійно написати програму, яка по кожному діагнозу проведе завантаження та обробку статистичних даних, також провести діагностику хворого метод Байеса відповідно до варіантів.

1) Хворий у віці 47 років, скарги на блювоту, відсутність апетиту. При огляді виявлено збільшення печінки, жовтушність шкіри і слизових. Нападів болю в правому підребер'ї немає.

2) Хворий у віці 19 років, скарги на блювоту, відсутність апетиту, і на напади болю в правому підребер'ї. При огляді виявлено збільшення печінки, жовтушність шкіри і слизових.

3) Хворий у віці 28 років, скарги на блювоту, відсутність апетиту, напади болю в правому підребер'ї. При огляді - збільшення печінки, жовтяниці немає.

4) Хворий у віці 56 років скаржиться на напади болю в правому підребер'ї і на відсутність апетиту. Блювоти немає. При огляді виявлено збільшення печінки. Жовтяниці немає.

5) Хворий у віці 49 років, скарги на блювоту, відсутність апетиту і на напади болю в правому підребер'ї. При огляді виявлено збільшення печінки, жовтушність шкіри і слизових.

6) Хворий у віці 22 року, скарги на блювоту і відсутність апетиту. При огляді виявлено жовтяничність склер, печінка не збільшена, нападів болю в правому підребер'ї немає.

7) Хворий у віці 66 років, при огляді виявлено збільшення печінки і жовтушність шкіри і слизових. Блювоти і нападів болю в правому підребер'ї немає, на апетит не скаржиться.

8) Хворий у віці 36 років, скарга на відсутність апетиту. При огляді виявлено збільшення печінки. Жовтяниці немає, блювоти і нападів болю в правому підребер'ї теж немає.

9) Хворий у віці 25 років, скарги на блювоту. Апетит є. При огляді, виявлено збільшення печінки і жовтушність склер. Нападів болю в правому підребер'ї немає.

10) Хворий у віці 37 років, скарги на блювоту. При огляді виявлено збільшення печінки і жовтушність склер. Апетит є, болів у правому підребер'ї немає.

Додатково: самостійно створіть систему візуалізації даних використовуючи бібліотеку Matplotlib.

Лабораторна робота №3а

Створення графічного інтерфейсу для системи, яка проводить автоматизовану медичну діагностику

Створіть новий проект.

(**MainMenu->File->NewProject**)

Додайте новий файл

(**MainMenu->File->New->PythonFile** або у дереві проекту правою кнопкою миші **New->PythonFile)**

Одним з інструментів створення графічних інтерфейсів є бібліотека **tkinter.** Дана бібліотека дозволяє створювати графічні вікна та розміщувати на них інтерактивні елементи вводу та виводу інформації. Бібліотека є кросплатформеною та може використовуватись в більшості сучасних операційних систем (Windows, Linux, Mac OS X тощо).

Бібліотека складається із віджетів, деякі найбільш поширені наведено в таблиці нижче. Для цих віджетів запропоновано безліч загальних та/або специфічних методів їх настроювання та відображення.

Назва віджету	Опис	Основні методи настройки	Опис методів настройки
Toplevel	Загальне вікно	**title**	заголовок вікна
		iconify / deiconify	згорнути / разгорнути вікно
		withdraw	"сховати" вікно (зробити невидимим). Для того, аби повернути це вікно deiconify
		resizable	дозвіл на зміну розмірів вікна
		geometry	встановлює геометрію вікна у форматі ШИРИНАхВИСОТА+x+y наприклад: geometry("600x400+40+80"
		transient	робить вікно залежним від іншого

Button	Кнопка	**text**	який текст буде відображений на кнопці (в прикладі - ок)
		width, height	відповідно, ширина і довжина кнопки
		bg	колір кнопки (скорочено від background, в прикладі колір - чорний)
		fg	колір тексту на кнопці (скорочено від foreground, в прикладі колір - червоний)
		font	шрифт і його розмір
Label	Напис на вікні		ті ж, що і кнопка
Entry	Поле вводу тексту	**borderwidth**	ширина бордюру елемента
		bd	скорочення від borderwidth
		width	задає довжину елемента в знакомісцях
		show	задає відображений символ
Text	Вікно виводу багато строкового тексту	**insert**	додають, видаляють або витягають текст. Перший аргумент – місце вставки у вигляді 'x.y', де x - це рядок, а y - стовпець
		delete	
		get	
Listbox	Спадаючий список		
Frame	Гуртуюча рамка		
Checkbutton	Кнопка для вибору / відзначення		
Radiobutton	Позначка для вибору одного з елементів		
Scrollbar	Навігація по вікну		

Окрім представлених вище методів необхідно відмітити наявність методів пакування або розміщення / **pack()**; **place()**; **grid()** /, які є доступними для всіх віджетів та дозволяють їх розміщувати у вікні або на фреймі.

Нижче наведено приклад коду, який демонструє можливості та особливості бібліотеки **tkinter** мови Python та дозволяє створити найпростіший віконний додаток, що має кнопку та поля введення/виводу даних.

```
from tkinter import *

root = Tk()

root.title("GUI Example")
root.geometry('800x600+100+50')
edit1 = Entry(width=50)

button1 = Button(text="Run", width=20)
label1=Label()

def strToSortlist(event):
    s = edit1.get()
    label1['text'] = s

button1.bind('<Button-1>', strToSortlist)

edit1.place(x=10,y=10)
button1.place(x=10,y=30)
label1.place(x=100,y=70,width=30)

root.mainloop()
```

Завдання: створить власний додаток, який дозволяє розраховувати площу 2х різних геометричних фігур.

Бібліотека **tkinter** дозволяє також організувати процедуру імпортування та включення візуальних об’єктів, які створюються за допомогою інших бібліотек. Наприклад корисною є можливість включення у вікно додатку елементів виводу графіків функцій, які створюються за допомогою MatplotLib.

Нижче наведено приклад коду, який демонструє можливості щодо імпорту елементів бібліотеки MatplotLib на вікно. У наведеному прикладі виводиться графік y=A*sin(2*Pi*t). Наведений код передбачає, що амплітуда

синусу A на початку роботи програми дорівнює 2. Але може бути змінена введенням у відповідне вікно. Графік перебудовується при натисканні на кнопку.

```
from tkinter import *
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg, NavigationToolbar2Tk)

# Implement the default Matplotlib key bindings.
from matplotlib.backend_bases import key_press_handler
from matplotlib.figure import Figure
import matplotlib.pyplot as plt

import numpy as np

root = Tk()
root.title("GUI Example")
root.geometry('800x800+100+50')
edit1 = Entry(width=30)
edit1.insert(1,'2')
button1 = Button(text="Run", width=20)
label1=Label()

a = float(edit1.get())

plt.ion()
fig = Figure(figsize=(5, 4), dpi=100)
t = np.arange(0, 3, .01)
fig.add_subplot(111).plot(t, a * np.sin(2 * np.pi * t))

edit1.place(x=10,y=10)
button1.place(x=10,y=30)
label1.place(x=100,y=70,width=30)

canvas = FigureCanvasTkAgg(fig, root) # A tk.DrawingArea.
canvas.draw()
canvas.get_tk_widget().place(x=200,y=10,width=600, height=400)

def strToSortlist(event):
    s = edit1.get()
    a= float(s)
    fig.clear()
    fig.add_subplot(111).plot(t, a * np.sin(2 * np.pi * t))
    canvas.draw()

button1.bind('<Button-1>', strToSortlist)

root.mainloop()
```

Завдання: Оновіть даний додаток, таким чином, щоб існувала можливість додавати декілька графіків на осі, вводити діапазон зміни аргументу, а також параметри функції.

Основне завдання по лабораторній роботі

Необхідно побудувати додаток із графічним інтерфейсом користувача, який дозволяє завантажувати (обирати на комп’ютері) дані щодо історії хвороби пацієнтів, вводити інформацію стосовно поточних симптомів пацієнту та розраховувати на основі методу Байєса ймовірності його діагнозів та візуалізувати отримані дані графічно.

Лабораторна робота №4
Підбір виду розподілу

Дана робота присвячена обробці масиву даних, побудові гістограми та вибору відповідного закону розподілу.

Статистичною гіпотезою називається будь-яке припущення про закон розподілу генеральної сукупності або його параметрах. Ми будемо оцінювати статистичні гіпотези, пов'язані з законом, функцією або щільністю розподілу.

Часто доводиться вирішувати питання про те, чи є розподіл генеральної сукупності X, з якої взята вибірка, нормальним.

Статистична гіпотеза про нормальний розподіл генеральної сукупності X, називається основною.

Критерієм згоди називається критерій перевірки правильності підбору теоретичного розподілу за вибіркою.

Перед застосуванням критеріїв згоди потрібно підібрати по вибірці передбачуваний теоретичний розподіл, а потім вже перевіряти правильність підбору. Тому спочатку розглянемо підбір теоретичного розподілу і його параметрів, а потім – два найбільш часто використовуваних критеріїв згоди.

Зазвичай спочатку підбирається вид розподілу: структура формули для функції, закону або щільності розподілу. Далі знаходяться його параметри, тобто числа, що входять у вираз для функції, закону або щільності розподілу. Результатом такого підбору буде деяка статистична гіпотеза, яку надалі перевіряють.

Закони розподілу відрізняються один від одного аналітичними виразами для функції розподілу $F(x)$, щільності розподілу $f(x)$ (для безперервних величин) або закону розподілу $p(x_k)$ (для дискретних величин). Для того, щоб по вибірці підібрати правильний закон, потрібно побудувати графік вибіркової функції розподілу $F^*(x)$, щільності розподілу $f^*(x)$ або закону розподілу $p^*(x_k)$ і подивитися, який розподіл краще

підходить. Якщо підходять кілька розподілів, беремо їх усі, а за критеріями згоди далі перевіряємо, яке краще підходить.

Статистичні дані можуть бути завантажені за посиланням відповідно до варіантів:

https://sites.google.com/site/oleksiylarin/given-courses

Необхідно відповідно до свого варіанту завантажити файл із даними: **ZNO_Var_i.csv**

Необхідно також завантажити файл **Example.csv** для виконання прикладу роботи програми, як подана нижче.

Для виконання лабораторної роботи необхідно:

Створіть новий проект.
(**MainMenu->File->NewProject**)

Додайте новий файл
(**MainMenu->File->New->PythonFile** або у дереві проекту правою кнопкою миші **New->PythonFile)**

Нижче наведено приклад коду, який демонструє можливості та особливості мови Python для того аби провести завантаження та почати обробку даних із текстових файлів.

```
from math import *

import numpy as np
import matplotlib.pyplot as plt

# Open the file with read only permit
text_file=open("Example.csv", "r")
lin=text_file.readlines()
text_file.close()

N=len(lin)-1
Age=np.zeros(N)
MarkMath= np.zeros(N)

print(N)

for i in range(1,N):
    data=lin[i].split(';')
    Age[i]=int(data[1])
    MarkMath[i]= int(data[5])

Age[0]= Age[1]
MarkMath[0]= MarkMath[1]
```

```
Age=sorted(Age)
Age_max=Age[N-1]
Age_min=Age[1]

print('The age of students is between (min='+str(Age_min)+'; max='str(Age_max)+').')
```

Знайдемо математичне очікування (функція **mean** бібліотеки **numpy**), дисперсію (функція **var** бібліотеки **numpy**), середньоквадратичне відхилення (функція **std** бібліотеки **numpy**), неусунуту асиметрію (функція **skew**) і неусунутий ексцес (функція **kurtosis**). Виведемо усі ці значення на екран.

```
from scipy.stats import *

MeanGradeOnMath=np.mean(MarkMath)
stdGradeOnMath=np.std(MarkMath)
MedianGradeOnMath=np.median(MarkMath)
LPercentileGradeOnMath=np.percentile(MarkMath,5)
RPercentileGradeOnMath=np.percentile(MarkMath,95)

print('Average students grade on math is  '+str(MeanGradeOnMath))
print('with standard deviation   '+str(stdGradeOnMath))
print('the median of distribution is    '+str(MedianGradeOnMath))
print('the 5% persentile of distribution is    '+str(LPercentileGradeOnMath))
print('the 95% persentile of distribution is    '+str(RPercentileGradeOnMath))

l = plt.plot(Age, MarkMath, 'ro')
plt.setp(l, markersize=10)

MeanArr=np.zeros(N)+MeanGradeOnMath
MedArr=np.zeros(N)+MedianGradeOnMath
stdArr=np.zeros(N)+stdGradeOnMath
LPArr=np.zeros(N)+LPercentileGradeOnMath
RPArr=np.zeros(N)+RPercentileGradeOnMath


plt.plot(Age,MeanArr,'r-',lw=3)
plt.plot(Age,MedArr,'b-',lw=1)

plt.plot(Age,LPArr,'g:',lw=1)
plt.plot(Age,RPArr,'g:',lw=1)

plt.plot(Age,MeanArr+stdArr,'b:',lw=1)
plt.plot(Age,MeanArr-stdArr,'b:',lw=1)
plt.show()
```

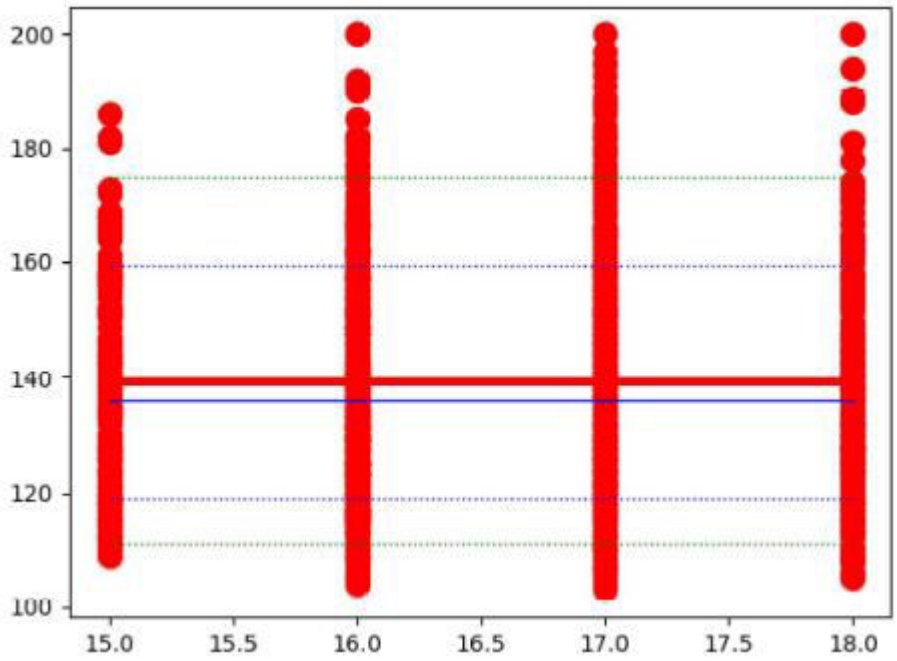

Рисунок 1 – Розподіл результатів ЗНО з математики по віку студентів

Стовпчаста діаграма числа влучень в кожну ділянку називається гістограмою. Гістограма в Python будується за допомогою тієї ж функції **histogram із бібліотеки** , яка підраховує кількість повторень кожного значення. Їй потрібно передати в якості параметрів вектор даних і, при необхідності, кількість інтервалів або їх центри. Задаємо кількість інтервалів. Знайдемо ширину кожного інтервалу (вона позначена в програмі ідентифікатором d). будемо припускати розподіл безперервним, тому побудуємо гістограму. Вона відображена на рисунку 2.

```
MarkMath=sorted(MarkMath)
Math_max = MarkMath[N-1]
Math_min = MarkMath[1]

k = round(N**0.5)

d = (Math_max-Math_min)/k
dell = (Math_max-Math_min)/20
xl = Age_min-dell
xr = Math_max+dell

histogr, b = np.histogram(MarkMath,k, density=True)
```

```
plt.hist(MarkMath, bins=b, density=True)
plt.show()
```

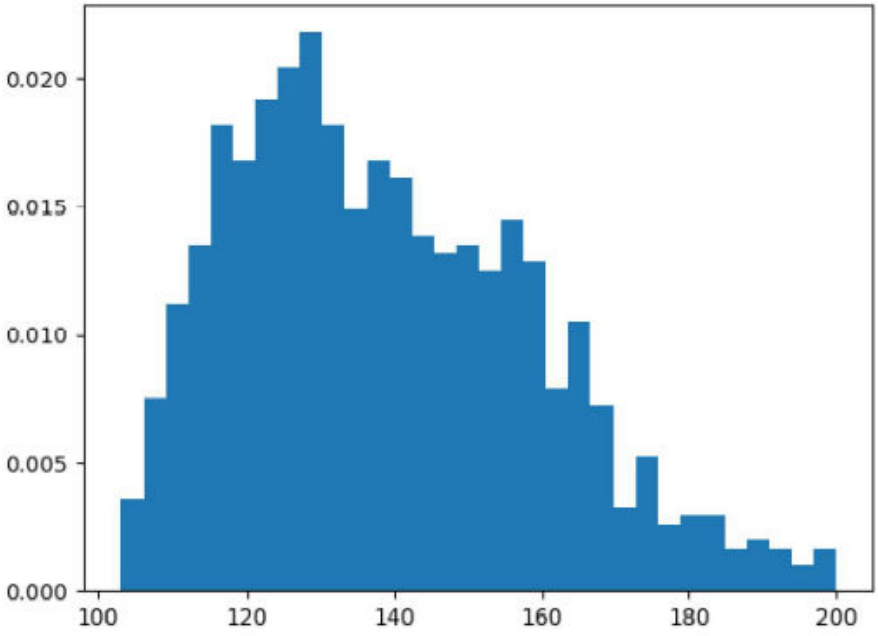

Рисунок 2 – Гістограма розподілення

Тепер – основне правило підбору теоретичного розподілу: його вид підбирається виходячи з виду гістограми (для безперервного розподілу) або виду вибіркового закону розподілу (для дискретних величин). Ми дивимося на побудований графік і порівнюємо його з графіками щільності (законів) розподілу. На що схоже, те і вибираємо.

Підібрати теоретичний розподіл – це значить вибрати функцію (закон, щільність) розподілу. Вид (аналітичний вираз) для них ми вже вибрали. Але в це аналітичний вираз входять деякі числові параметри, які також потрібно підібрати.

Апроксимуємо гістограму нормальним законом. Використаємо для цього метод моментів. Параметрами цього закону є математичне очікування (Mx) та середньоквадратичне відхилення (Sx):

$$y = \frac{1}{Sx \cdot \sqrt{2\pi}} \cdot e^{-0.5 \cdot (\frac{1}{Sx \cdot (x - Mx)})^2}$$

Виведемо на екран отриману криву.

```
xx = np.linspace(Math_min, Math_max, 100)
Mx=MeanGradeOnMath
Sx=stdGradeOnMath
y = ((1 / (np.sqrt(2 * np.pi) * Sx)) * np.exp(-0.5 * (1 / Sx * (xx - Mx))**2))

plt.plot(xx ,y, '-r')
plt.show()
```

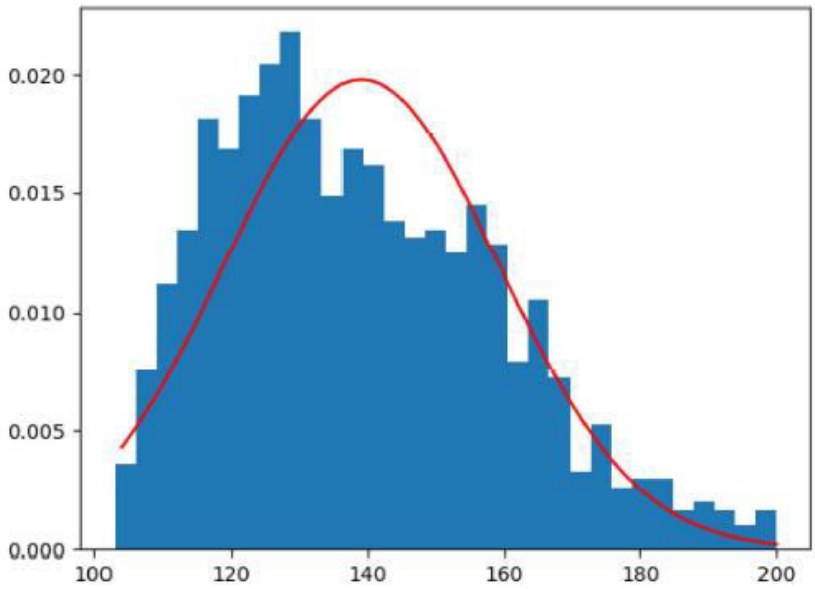

Рисунок 3 – Гістограма розподілення та спроба її апроксимації законом Гауса

Такий розподіл не підходить. Спробуємо гамма-розподіл.

$$y = \frac{1}{G(a) \cdot p^a \cdot (x - dd)^{(a-1)}} \cdot e^{-\frac{x-dd}{p}}$$

Тут використовуються інші параметри. За допомогою методу максимальної правдоподібності формується помилка апроксимації теоретичної кривої дискретного масиву значень гістограми. Далі параметри закону знаходяться з мінімуму цієї помилки. Для цього в Python є готова функція **fit**, яка є стандартним методом для більшості законів, що присутні в бібліотеці **statsmodels.distributions**. Наприклад, для вибору гамма-розподілу, у якого є 3 параметри, код Python буде виглядати наступним чином:

```
import scipy.stats as stat
import scipy.special as spec

a, dd, p = stat.gamma.fit(MarkMath)

yy = 1/spec.gamma(a)/p**a*(xx-dd)**(a-1)*np.exp(-(xx-dd)/p)
plt.plot(xx, yy, '-g')
plt.show()
```

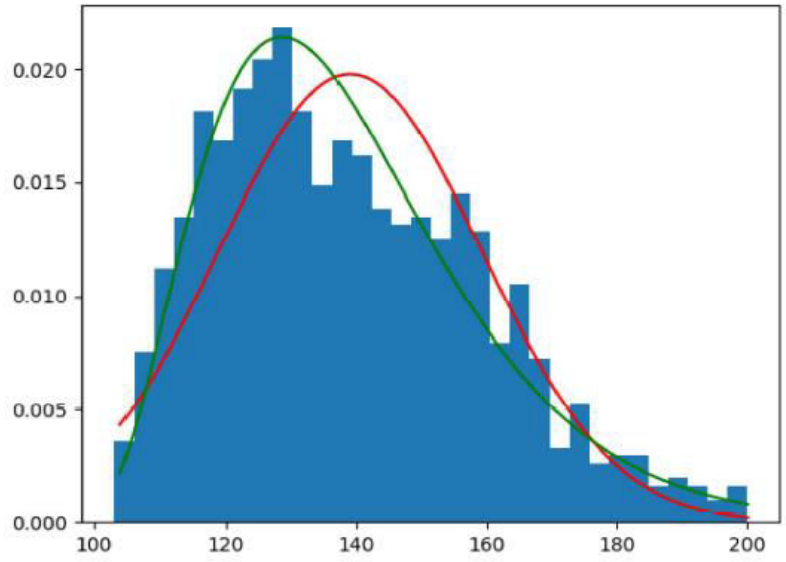

Рисунок 4 – Гістограма розподілення та спроба її апроксимації законом Гауса та гама-розподілом

На мові програмування Python можливо задавати закон автоматично за допомогою команди **gamma.pdf** з використанням стандартних параметрів (*a, loc, scale*). Виведемо на екран отриману криву та порівняймо її з попередньою.

```
yyy = gamma.pdf(xx, a=a, loc=dd, scale=p)
plt.plot(xx, yyy, '--c')
plt.show
```

Критерій згоди Пірсона.

У цьому критерії використовується аналог евклідової метрики: мірою відмінності є зважена сума квадратів різниць між теоретичними і емпіричними числами влучень в інтервали. Таким чином, для застосування цього критерію потрібно розбити весь діапазон зміни даних на інтервали. Ці інтервали можуть бути довільними, не обов'язково однакової довжини. Потрібно лише, щоб теоретичне число попадань в кожен з них було менше 5.

Якщо ця умова не виконується і теоретичне число попадань в будь-якої інтервал менше 5, потрібно приєднати його до сусіднього або провести розбивку на інтервали по-іншому. Після розбивки на інтервали підраховуємо емпіричні та теоретичні числа влучень в них. Карл Пірсон досліджував випадкову величину, реалізацією якої є значення χ2 і показав, що, якщо все $np_j \geq 5$, то, незалежно від розподілу генеральної сукупності X, ця величина має приблизно χ2 – розподіл Пірсона з k – m ступенями свободи, де m – число обмежень, яка дорівнює кількості параметрів обраного розподілу плюс 1:

$$\chi^2 = \sum_{j=1}^{k} \frac{(n_j - n p_j)^2}{n p_j}.$$

Позначимо емпіричні числа влучень через n_j, де j – номер інтервалу. Кожне n_jпотрібно порівняти з теоретичним числом попадань в цей інтервал, що дорівнює np_j, де n – загальне число дослідів, а p_j – теоретична ймовірність попадання величини X в j-й інтервал:

$$p_j = \int_{a_j}^{b_j} f(x)\,dx = F(b_j) - F(a_j)$$

Тут a_j, b_j – кордони j-го інтервалу, f(x) – теоретична щільність розподілу, F(x) – теоретична функція розподілу.

Таким чином, для того, щоб знайти помилку χ2 потрібно знайти теоретичні частоти потрапляння в інтервал – тобто p_j – слід скористатися функціями розподілення такого теоретичного закону, що використовується для апроксимації. У Python є готові функції для обчислення функції розподілу в заданій точці **cdf**. Вони є методами відповідних законів. Таким чином, код за визначенням теоретичних частот попадання p_j наступний:

```
exp_freqG=np.zeros(k)
exp_freqN=np.zeros(k)
exp_freqR=np.zeros(k)

for i in range(0,k):
   exp_freqG[i]=gamma.cdf(xmin+(i+1)*d, a=a, loc=dd, scale=p)-gamma.cdf(xmin+i*d, a=a, loc=dd,
scale=p)
   exp_freqN[i] = norm.cdf(xmin + (i + 1) * d, Mx,Sx) - norm.cdf(xmin + i * d,  Mx,Sx)
```

Емпіричні числа влучень через n_j визначимо з гістограми, в Python функція **histogram** фактично визначає ці значення, але відносно ширини інтервалу. Фактично це нам дозволяє знайти n_j наступним кодом:

```
obs_freq=(histogr*d)
```

Знайдемо χ2-статистику, критичні значення для всіх передбачуваних розподілів, і виберемо той розподіл, для якого нерівність виконується найкращим чином.

Щоб визначити параметр χ2 квадрат у Python є функція **chisquare**, яка повертає параметр з заданою ймовірністю. Якщо за розрахунковими даними значення ймовірності виявиться дуже малою величиною, наприклад *0,01*, то відмінності між досліджуваними рядами потрібно вважати істотними, тобто гіпотеза не приймається. Якщо ж імовірність виявиться не малою, розбіжності вважаються випадковими і гіпотеза приймається.

Код який визначає ці параметри наступний:

```
chiG,pG=chisquare(obs_freq,exp_freqG)
print('Chi squred test for Gamma distribution')
print(chiG,pG)
chiN,pN=chisquare(obs_freq,exp_freqN)
print('Chi squred test for Hauss distribution')
print(chiN,pN)
chiR,pR=chisquare(obs_freq,exp_freqR)
print('Chi squred test for Rayleigh distribution')
print(chiR,pR)
plt.show()
```

Лабораторна робота №4а

Прийняття статистичних рішень

Дана робота присвячена обробці масиву даних, побудові гістограми, вибору відповідного закону розподілу та оцінці риску.

Нехай проводиться діагностика захворювання covid-19 шляхом оцінки температури (параметр x). Завдання полягає у виборі значення x_0 параметра x таким чином, що при $x > x_0$ слід поставити діагноз, що людина хвора covid-19, а при $x < x_0$, що здорова. Так як стан системи характеризується одним параметром, то система має одномірний простір ознак. Домовимося вважати: D_1 – здорова людина (немає вірусної хвороби) і D_2 – хвора на вірус людина. Тоді вказане правило рішення полягає в наступному:

$$\text{при } x < x_0 \; x \in D_1, \text{ при } x > x_0 \; x \in D_2 \qquad (1)$$

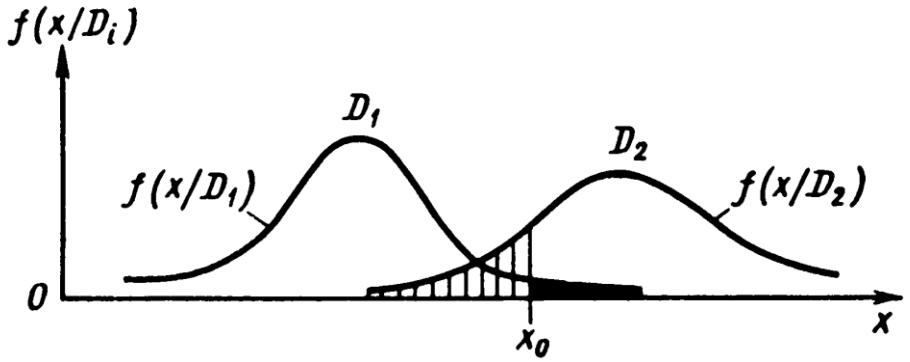

Рисунок 1 – Статистичний розподіл щільності ймовірності діагностичного параметру x для здорового D_1 та хворого D_2 станів

У залежності від ряду факторів, розподіл x для хворих і здорових людей показано на рисунку 1. Істотно, що області здорового і хворого станів перетинаються і тому принципово не може бути вибране значення x_0, при якому правило (1) не давало б помилкових рішень. Завдання полягає в тому,

щоб вибір x_0 був у деякому сенсі оптимальним, наприклад давав найменше число помилкових рішень.

Розглянемо спочатку можливі помилки при прийнятті рішення.

Помилкова тривога та пропуск цілі. Помилковою тривогою називається випадок, коли приймається рішення про наявність захворювання, але в дійсності людина перебуває в здоровому стані (замість D_1 приймається D_2). Пропуск цілі - прийняття рішення про здоровий стан, тоді як людина хвора (замість D_2 приймається D_1).

Статистичні дані можуть бути завантажені за посиланням відповідно до варіантів:
https://sites.google.com/site/oleksiylarin/given-courses/statistics/statistics_l4a

Для виконання прикладу лабораторної роботи необхідно завантажити файл **Example.csv** та здійснити наступний код:

Створіть новий проект.
(**MainMenu->File->NewProject**)

Додайте новий файл
(**MainMenu->File->New->PythonFile** або у дереві проекту правою кнопкою миші **New->PythonFile)**

Нижче наведено приклад коду, який демонструє можливості та особливості мови Python для того аби провести завантаження та почати обробку даних із текстових файлів. У файлі знаходиться така інформація: ім’я, вік, температура здорового стану та температура хворого стану людини.

```
from math import *
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stat

# Open the file with read only permit
text_file = open("Example.csv", "r")
```

```
lin = text_file.readlines()
text_file.close()

D1Size=len(lin)-1
DataTemp=np.zeros((D1Size,3))
Age=np.zeros((D1Size,1))

for i in range(1,D1Size):
   data=lin[i].split(';')
   Age[i]=int(data[1])
   print(data)
   DataTemp[i,1] = float(data[2])
   DataTemp[i,2] = float(data[3])
```

Виберемо температури здорового стану людей Знайдемо математичне очікування (функція **mean** бібліотеки **numpy**), середньоквадратичне відхилення (функція **std** бібліотеки **numpy**), мінімальне та максимальне значення температури (функції **min** та **max** відповідно бібліотеки **numpy**) . Виведемо ці значення на екран.

```
DataTempN=sorted(DataTemp[:,1])
DataTempN[0]=DataTempN[1]

MeanTempN=np.mean(DataTempN)
stdTempN=np.std(DataTempN)
TN_min=min(DataTempN)
TN_max=max(DataTempN)

print('Average temperature of healthy person h is  '+str(MeanTempN))
print('with standard deviation   '+str(stdTempN))
print('All the data on normal temperature is between  '+str(TN_min)+' and  '+ str(TN_max))
```

Можна бачити, що середнє значення температури тіла здорого пацієнта складає 36,6 ^{o}C разом із тим існує певна варіація цього значення, тобто є люди з меншою температурою, є і з більшою, навіть вище за 37,5 ^{o}C.

Самостійно знайдіть та виведіть ці значення для температури хворого стану людей.

Стовпчаста діаграма числа влучень в кожну ділянку називається гістограмою. Гістограма в Python будується за допомогою функції **histogram**,

яка підраховує кількість повторень кожного значення. Побудуємо гістограму для температур здорового стану людей. Вона відображена на рисунку 1.

```
histT,bT=np.histogram(DataTempN,8,density=True)
plt.hist(DataTempN,bins=bT,density=True)
plt.show()
```

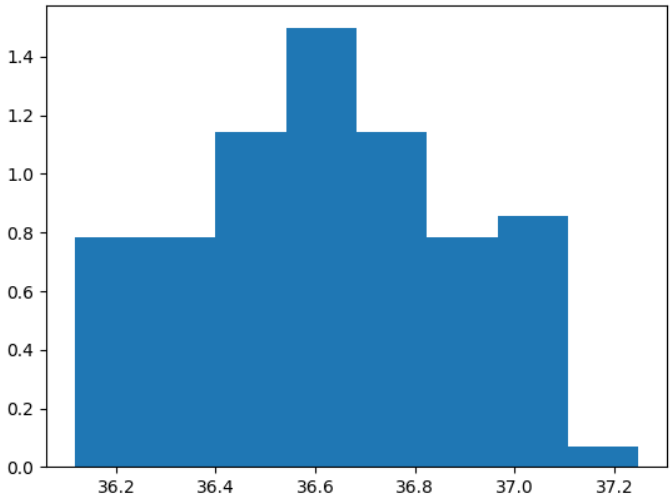

Рисунок 1 – Гістограма розподілення

Самостійно побудуйте гістограму для температур хворого стану людей та виведіть ці дві діаграми на одному графіку.

Тепер підбираємо теоретичний розподіл: його вид підбирається виходячи з виду гістограми (для безперервного розподілу) або виду вибіркового закону розподілу (для дискретних величин). Ми дивимося на побудований графік і порівнюємо його з графіками щільності (законів) розподілу. На що схоже, те і вибираємо. Підібрати теоретичний розподіл – це значить вибрати функцію (закон, щільність) розподілу. Вид (аналітичний вираз) для них ми вже вибрали. Але в це аналітичний вираз входять деякі числові параметри, які також потрібно підібрати. Апроксимуємо гістограму нормальним законом (рисунок 2).

```
mTN,sTN = stat.norm.fit(DataTempN)
mTF,sTF = stat.norm.fit(DataTempF)

xx=np.linspace(0.99*TN_min,1.1*TF_max,100)
yTN=stat.norm.pdf(xx,loc=mTN,scale=sTN)
yTF=stat.norm.pdf(xx,loc=mTF,scale=sTF)

plt.plot(xx,yTN,'-b')
plt.plot(xx,yTF,'-r')
plt.show()
```

Для свого варіанту потрібно зробити апроксимацію. Скоріш за все такий закон не підходитиме. Тому самостійно підберіть закон розподілу та апроксимуйте гістограми. Поясніть, чому саме цей закон було обрано.

На основі статистичних даних по симптому (підвищена температура тіла) у пацієнтів із захворюваннями на вірусну інфекцію можна робити висновок щодо граничного значення температури тіла при якомі можна вважати, що пацієнт є хворим. Задаємо це значення, як 37 ^{o}C. Значення x_0 (TN_Edge), та позначимо його на нашому графіку (рисунок 2).

```
TN_Edge=37.0
xxEdge=[TN_Edge,TN_Edge]
yyEdge=[0,0.8]

plt.plot(xxEdge,yyEdge,'-k')
plt.show()
```

Прийняття рішення на основі даного симптому:

температура тіла менша за 37^{o}C – не хвора людина;

температура тіла вища менша за 37^{o}C – не хвора людина.

Така оцінка може давати помилки у прийнятті рішень, бо можуть бути випадки що людина має більшу за 37^{o}C температуру тіла але не має вірусної хвороби, та навпаки.

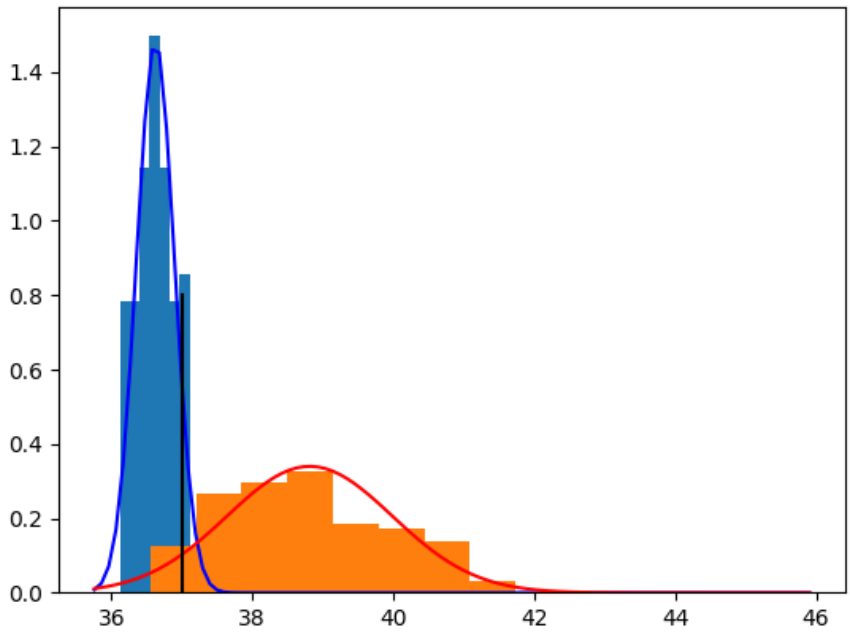

Рисунок 2 – Апроксимація нормальним законом та x_0

Позначимо H_{ij} $(i, j = 1, 2)$ можливі рішення за правилом (1) (перший нижній індекс відповідає індексу прийнятого діагнозу, другий – індексу дійсного стану). Тоді H_{11} та H_{22} – правильні рішення, H_{12} – помилкова тривога, H_{21} – пропуск дефекту.

Розглянемо ймовірність помилкової тривоги (1-го типу) при використанні правила (1). Площа під кривою щільності ймовірності здорового стану, що відповідає $x > x_0$, висловлює умовну ймовірність ситуації $x > x_0$ для здорових людей

$$P(x > x_0/D_1) = \int_{x_0}^{\infty} f(x/D_1)\,dx. \tag{2}$$

Імовірність помилкової тривоги дорівнює ймовірності добутку двох подій: наявність здорового стану і значення $x > x_0$. Тоді

$$P(H_{21}) = P(D_1)\,P(x > x_0/D_1) = P_1 \int_{x_0}^{\infty} f(x/D_1)\,dx, \tag{3}$$

де $P_1 = P(D_1)$ – апріорна ймовірність діагнозу D_1 (вважається відомою на підставі попередніх статистичних даних). Подібним чином знаходиться ймовірність пропуску дефекту (2-го типу)

$$P(H_{12}) = P(D_2)\,P(x < x_0/D_2) = P_2 \int_{-\infty}^{x_0} f(x/D_2)\,dx. \tag{4}$$

Знайдемо ймовірність прийняття правильного рішення щодо здорової або хворої людини, а також ймовірність помилок 1-го та 2-го типів за допомогою модулів Python. Для інтегрування використовуємо функцію **trapz**, крім того формуємо масив для інтегрування із заданим діапазоном від 0.9*TN_min – трошки менше ніж найнижче значення з масиву даних (бо застосовувати нескінченність тут не доцільно) та до граничного значення TN_Edge = 37^{o}C. Позначимо цей масив значень на 100 елементів **tN**, що забезпечить якісне інтегрування. Для цього діапазонів інтегрування розраховуємо масиви значень підінтегральної функції **yTN**.

```
tN=np.linspace(0.99*TN_min,TN_Edge,100)
yTN=stat.norm.pdf(tN,loc=mTN,scale=sTN)
Pr11=np.trapz(yTN,tN)

print('Probablity of correct decision for healthy person is '+ str(Pr11))
```

Аналогічно розраховуються всі значення ймовірностей рішень, як дійсних так і помилок.

```
tF=np.linspace(TN_Edge,1.1*TF_max,100)
yTF=stat.norm.pdf(tF,loc=mTN,scale=sTN)
Pr12=np.trapz(yTF,tF)

tN=np.linspace(0.99*TN_min,TN_Edge,100)
yTN=stat.norm.pdf(tN,loc=mTF,scale=sTF)
tF=np.linspace(TN_Edge,1.1*TF_max,100)
yTF=stat.norm.pdf(tF,loc=mTF,scale=sTF)
```

```
Pr21=np.trapz(yTN,tN)
Pr22=np.trapz(yTF,tF)

print('Probablity of correct decision for healthy person is '+ str(Pr11))
print('Probablity of the mistake of 1st type "the healthy person with high temperature" is  '+
str(Pr12))
print('Probablity of correct decision for ill person is  '+ str(Pr22))
print('Probablity of the mistake of 2nd type "the ill person with low temperature" is  '+ str(Pr21))
```

Імовірність прийняття помилкового рішення складається з ймовірностей помилкової тривоги і пропуску дефекту. Якщо приписати "ціни" цих помилок, то отримаємо вираз для середнього ризику

$$R = C_{21}P_1 \int_{x_0}^{\infty} f(x/D_1)\,dx + C_{12}P_2 \int_{-\infty}^{x_0} f(x/D_2)\,dx. \tag{5}$$

Зрозуміло, що ціна помилки має умовне значення, але вона повинна врахувати передбачувані наслідки помилкової тривоги і пропуску дефекту. У загальному випадку середній ризик (очікувана величина втрати) виражається рівністю

В даній лабораторній роботі запропоновано для прикладу, що ймовірність апріорна вірусної хвороби це $10^5/30^6$, що відповідає загально статистичним даним по хворим на короно-вірус в Італії на 30 мільйонну країну 100000 підтверджених випадків.

Для визначення вартості помилки зроблено припущення, що випадок помилки першого роду тобто коли здорова людина вважається хворою буде коштувати 400 грн (витрати на непотрібні аналізи), а у випадку помилки 2го роду, коли хворий буде помилково пропущеним він потребуватиме складного лікування а також заразить багато людей і сумарне лікування коштуватиме 200000 грн (50 контактів та 4000 грн на лікування кожного).

Код що наведено нижче надає можливість розрахувати ризики помилок та відповідну вартість для страхування організації, яка має

співробітників та перевіряє їхньою температуру для того аби визначити можливість його допуску на підприємство.

```
C12=400  #  the cost of the mistake of 1st type
C21=200000 #  the cost of the mistake of 2nd type

P2=1e5/30e6
P1=1-P2
R=C12*P1*Pr12+C21*P2*Pr21
print('The risk is  ' + str(R))
print('The apriory risk for 1st mistake is ' + str(C12*P1))
print('The apriory risk for 2nd mistake is ' + str(C21*P2))
```

Зрозуміло, що занчення у 37°C можливо не є оптимальним з точки зору ризиків. Спробуйте подивитись як змінюється ризик при встановленні інших меж на температуру 36.6 °C або 38 °C.

Метод мінімального ризику. Знайдемо граничне значення x_0 у правилі (1) з умови мінімуму середнього ризику. Диференціюючи (6) по x_0 і прирівнюючи похідну нулю, отримаємо умову екстремуму

$$\frac{dR}{dx_0} = C_{11}P_1 f(x_0/D_1) - C_{21}P_1 f(x_0/D_1) + C_{12}P_2 f(x_0/D_2) - \\ - C_{22}P_2 f(x_0/D_2) = 0 \qquad (7)$$

або

$$\frac{f(x_0/D_1)}{f(x_0/D_2)} = \frac{(C_{12} - C_{22})\,P_2}{(C_{21} - C_{11})\,P_1}. \qquad (8)$$

Ця умова часто визначає два значення x_0, з яких одне відповідає мінімуму, друге – максимуму ризику (рисунок 3)

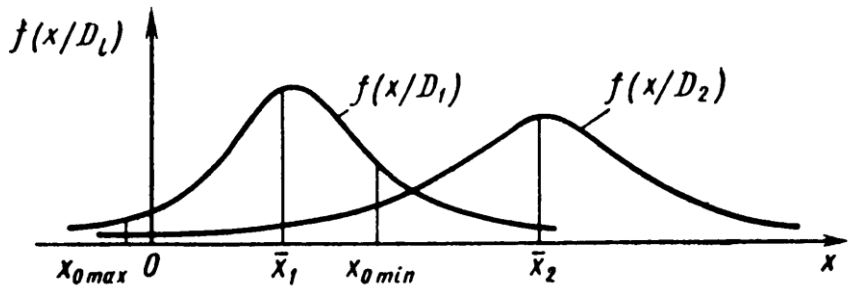

Рисунок 3 – Точки екстремуму середнього ризику помилкових рішень

Для визначення нелінійного рівняння потрібно використати функцію **funk**

```
def func(x0):
    #F1=1/sqrt(2*np.pi*sTN)*exp(-(x0-mTN)**2/2/sTN)
    #F2=1/sqrt(2*np.pi*sTF)*exp(-(x0-mTF)**2/2/sTF)
    y=x0*x0-4
    return y #F1/F2-(C21*P2)/(C12*P1)

from scipy.optimize import fsolve

x00=fsolve(func,-1)

print(mTN,mTF)
print(x00)
```

Самостійно знайти мінімальний ризик у залежності від температури людини.

Необхідно відповідно до свого варіанту завантажити файли із даними щодо відомостей по температурі пацієнтів, які не мають вірусних захворювань **Hospital_1_visitorsTemp.csv** та пацієнтів поліклініки, які мали підтвердження вірусної хвороби **Hospital_1_FluTemp.csv**

N	P1	C12	P2	C21
1	0.1	30	0.05	30
2	0.3	10	0.1	10
3	0.1	20	0.15	20
4	0.1	30	0.2	30
5	0.3	50	0.03	50
6	0.2	10	0.07	10
7	0.09	15	0.12	15
8	0.3	25	0.13	25
9	0.25	15	0.21	15
10	0.15	25	0.075	25
11	0.35	65	0.025	65
12	0.21	40	0.1	40
13	0.1	100	0.05	100
14	0.15	10	0.15	10
15	0.215	35	0.06	35
16	0.11	20	0.23	20
17	0.1	15	0.17	15
18	0.3	80	0.005	80
19	0.09	45	0.22	45
20	0.2	60	0.11	60

ПРОЕКТНА РОБОТА з курсу СТАТИСТИКА

У рамках курсу статистика, заключним елементом є виконання студентами групових комплексних проектів. Виконання проектної роботи та її захист є обов'язковим елементом навчальної дисципліни, результати якої входять до формування підсумкової оцінки.

Вимоги щодо форми проектної роботи та процедури її виконання:

1) Проектна робота виконується командою з 2-4 студентів.

2) Початок виконання роботи на 13-тому тижні і термін її виконання 2 тижня.

3) Форма звітності: стислий анотований звіт та публічний захист із презентацією результатів виконання проекту та демонстрацією програмних засобів, якими було досягнуто ці результати.

4) Атестацію проектних робіт виконує комісія, що складається з лектора дисципліни та викладачів, які здійснюють проведення практичних/лабораторних робіт.

5) Студенти можуть вільно обирати програмно-технічні засоби для виконання даної проектної роботи.

Основні вимоги до проектної роботи по суті:

Робота передбачає здійснення студентами статистичного аналізу даних.

Команда має обрати масив даних, з певної предметної області з об'ємом порядку 1000 значень або більше (в окремих випадках можна застосувати й менші об'єми даних).

Предметна область (тематика) проекту може обиратись командою студентів довільно з використанням відкритих джерел даних (це можуть бути дані щодо екології та її зміни, дані спортивних змагань, відкритих даних по фінансовим, ринковим питанням, медична статистика, інфраструктурне забезпечення, безпека тощо). Наведений перелік є дуже умовним і не має на меті обмежити вибір студентів, але тема має бути погоджена з головним лектором перед початком виконання проекту, так само як і склад команди.

Деякі сайти з публічною інформацією для пошуку статистичних даних (лише загальні рекомендації для натхнення, Ви можете обирати будь-які інформаційні джетала, що не порушують закон або етичні норми):

http://www.ukrstat.gov.ua/

http://data.city.kharkov.ua/

https://www.cdc.gov/nchs/nhanes/index.htm

https://stats.oecd.org/

https://www.nationmaster.com/

https://www.statista.com/

Статистичний аналіз повинен містити:

– описову частину: визначення статистичних ознак (середнє значення, розкид, асиметрію), визначення довірчих інтервалів цих ознак; побудову гістограм(и) і закону(ів) розподілу та обґрунтування його вибору.

– питання діагностики (застосування методів Байєса або середнього ризику) або прогнозу (проведення дисперсійного та регресійного аналізів).

– Питання аналізу зміни результатів діагностики чи прогнозу при здійснення спрямованих заходів на зміну вихідної інформації, шляхом проведення процедури статистичних симуляцій типу Монте-Карло (даний розділ не є обов’язковим).

В будь-якому разі виконання проекту повинно мати узагальнюючі висновки, умовиводи зроблені на основі представленої статистики та можливо рекомендації.

За результатом виконання проекту команда може отримати до 15 балів до підсумкової атестації. Оцінюється проект в цілому, тобто уся команда, але оцінка на відповіді на запитання є персональною для кожного студента члена команди.

При оцінюванні враховується:

Критерій		Якісна оцінка		бали
1	Повнота виконання роботи по суті	Фактично лише описова статистика		1
		Здійснення діагностичного або прогнозного аналізу		3
		Якісний аналіз, умовиводи, статистичне моделювання		5
2	Складність аналізу з математичної точки зору	В межах повторення практичних занять або лабораторної роботи		1
		Застосування комплексу методів, які вивчались		2
3	Складність, ефективність та відповідність алгоритмічних рішень	Фактично відсутність етапу програмування в проекті		0
		Прості програмні рішення		1
		Якісний ефективний код, наявність інтерфейсу, що робить аналіз готовим програмним продуктом		2
4	Якість презентації під час захисту	Погано		1
		Добре		2
		Відмінно (оформлення результатів, слайдів, доповідь)		3
5	Відповіді на запитання під час захисту	Погано		0
		Задовільно		1
		Добре		2
		Відміно		3
Загальна оцінка:			**E**	**3**
			D	**7**
			C	**10**
			B	**13**
			A	**15**

Висновки

За освітньо-професійною програмою підготовки бакалаврів зі спеціальності 113 «Прикладна математика» та 122 «Комп’ютерні науки» на третьому році навчання відповідно до навчального плану вивчається дисципліна «Математична статистика». В рамках даного курсу студентами виконуються лабораторні завдання та проектна робота.

При виконанні лабораторних завдань студент на практиці закріплює знання за дисципліною, що вивчається, демонструє уміння застосовувати їх на під час вирішення окремих задач.

В даних методичних вказівках до виконання до комп'ютерного практикуму на Python надається порядок виконання лабораторних робіт, контрольні питання по даним роботам, опис порядку виконання проектної роботи.

Запропоновано послідовний підхід до викладання навчального матеріалу методичних вказівок до виконання комп'ютерного практикуму на Python з курсу «Математична статистика», який відповідає освітній програмі.

Навчальне видання

ЛАРІН Олексій Олександрович
СУХАНОВА Ольга Ігорівна

ОПИСОВА СТАТИСТИКА ТА СТАТИСТИЧНА ДІАГНОСТИКА

Методичні вказівки до виконання індивідуальних завдань з курсу «Математична статистика» для студентів всіх форм навчання, спеціальності 113 «Прикладна математика» та 122 «Комп'ютерні науки»

Українською мовою

Відповідальний за випуск проф. Львов Г.І.
Роботу до видання рекомендував проф. Бреславський Д.В.

В авторській редакції

План ____ р., поз. ___

Підп. до друку _____. Формат 60×84 1/16. Папір офсетний.
Riso-друк. Гарнітура Times New Roman. Ум. друк. арк._____.
Наклад 50 прим. Зам. № _____. Ціна договірна.
Видавець Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 3657 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2
Виготовлювач_____
_____
_____