 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №5**

з дисципліни «Методи та технології обчислювального інтелекту»

на тему: «Градієнтний спуск»

**Виконала:**

студентка гр. БС-01мп

Харченко Наталя

**Перевірив:**

доц. Федорін І.В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

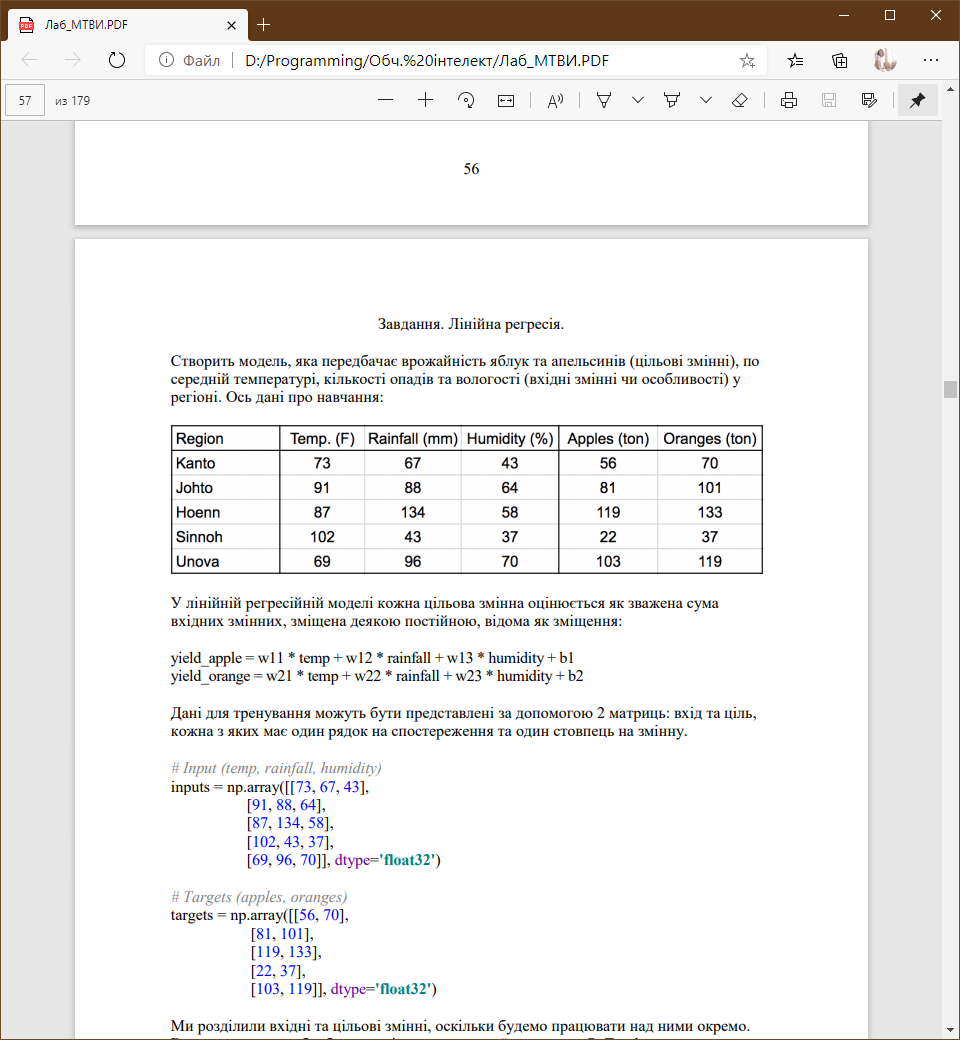
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2020

**Практичне завдання**

Створить модель, яка передбачає врожайність яблук та апельсинів (цільові змінні), по середній температурі, кількості опадів та вологості (вхідні змінні чи особливості) у регіоні. Ось дані про навчання:



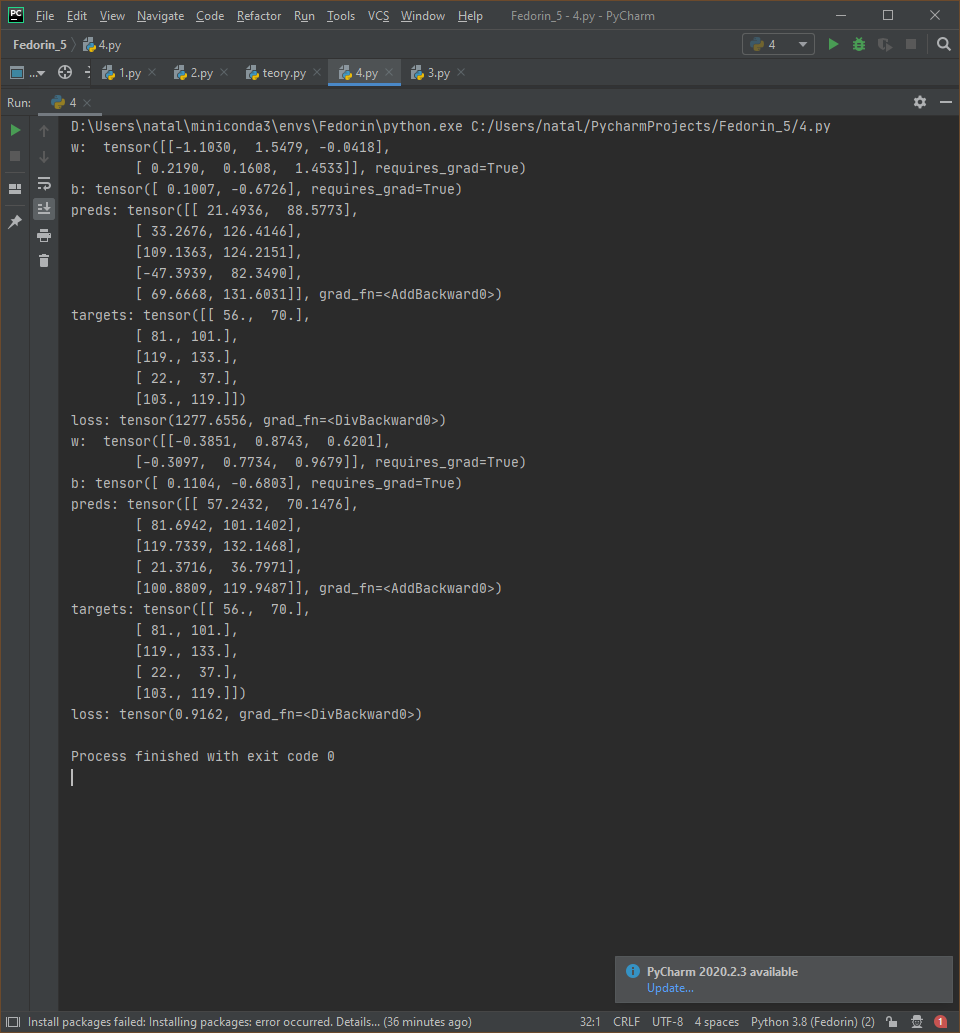
Завдання: натренуйте модель деяку кількість епох. Подивіться як змінюються ваги та похибка (втрати). Знайдіть оптимальну кількість епох для тренування. Порахуйте MSE

**Лістинг програми:**

import torch  
import numpy as np  
  
# Input (temp, rainfall, humidity)  
inputs = np.array([[73, 67, 43],  
 [91, 88, 64],  
 [87, 134, 58],  
 [102, 43, 37],  
 [69, 96, 70]], dtype='float32')  
# Targets (apples, oranges)  
targets = np.array([[56, 70],  
 [81, 101],  
 [119, 133],  
 [22, 37],  
 [103, 119]], dtype='float32')  
  
# Convert inputs and targets to tensors  
inputs = torch.from\_numpy(inputs)  
targets = torch.from\_numpy(targets)  
  
w = torch.randn(2, 3, requires\_grad=True)  
b = torch.randn(2, requires\_grad=True)  
  
  
# MSE loss  
def mse(t1, t2):  
 diff = t1 - t2  
 return torch.sum(diff \* diff) / diff.numel()  
  
  
def model(x):  
 return x @ w.t() + b  
  
print('w: ', w, '\nb:', b)  
# Generate predictions  
preds = model(inputs)  
# Calculate the loss  
loss = mse(preds, targets)  
# Compute gradients  
loss.backward()  
print('preds:', preds)  
print('targets:', targets)  
print('loss:', loss)  
  
for \_ in range(2500):  
 preds = model(inputs)  
 loss = mse(preds, targets)  
 loss.backward()  
 # Adjust weights & reset gradients  
 with torch.no\_grad():  
 w -= w.grad \* 1e-5  
 b -= b.grad \* 1e-5  
 w.grad.zero\_()  
 b.grad.zero\_()  
  
print('w: ', w, '\nb:', b)  
preds = model(inputs)  
loss = mse(preds, targets)  
print('preds:', preds)  
print('targets:', targets)  
print('loss:', loss)

**Результат:**

*До тренування*



*Після тренування*

