**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **ІКНІ**

Кафедра **СШІ**

**ЗВІТ**

До лабораторної роботи №1

**З дисципліни:**

**«Машинне навчання»**

**Виконав:**

ст. гр. КН-310

Шиманський П.С.

**Прийняла:**

Якимишин Х.М.

Львів - 2020

**Тема:** основи роботи з PyTorch

**Мета:** ​засвоїти основні відомості про роботу з фреймворком PyTorch. Навчитись розпізнавати не типові елементи в даних.

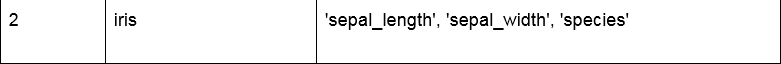
**Завдання:** ​ Порахувати outliers для датасету і колонок згідно варіантів, а саме: 1. Порахувати z-score незалежно для кожної з згаданих цифрових колонок. Для обрахунку використати лише PyTorch.

2. Агрегувати пораховані z-score(наприклад усередненням). Візуалізувати датасет і знайдені нетипові дані(ті, для яких z-score > 3) на scatter plot. Нетипові дані і решту датасету звізуалізувати різними кольорами.

3. Якщо серед згаданих колонок є категоріальні дані, порахувати z-score для кожної з колонок в межах кожної з можливих категорій. Наприклад для iris датасету колонка ‘species’ може набувати значень 'setosa', 'versicolor', 'virginica'. Потрібно порахувати z-score для решти колонок лише для значень які в колонці ‘species’ мають значення ‘setosa’, після цього повторити те саме для значень які мають ‘versicolor’ і завершити обрахунком z-score для значень що мають ‘virginica’ в полі ‘species’

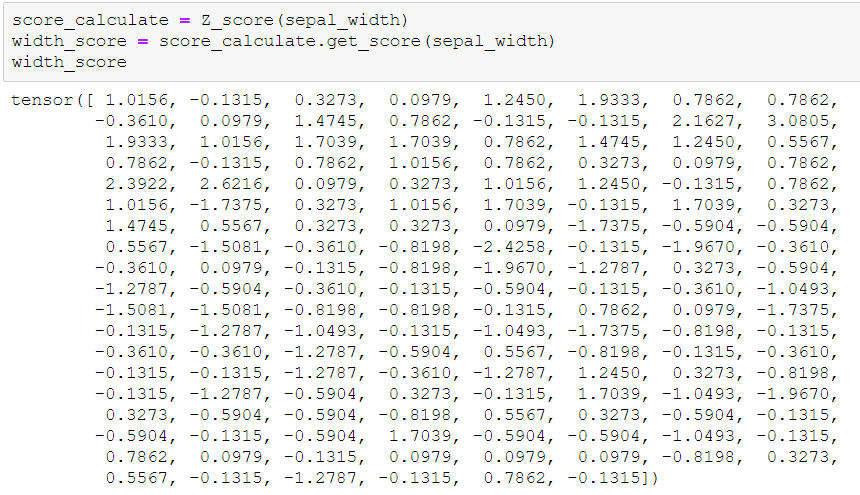
4. Аналогічно до пункту 2, агрегувати отримані значення і визначити нетипові дані.

**Варіант 2**

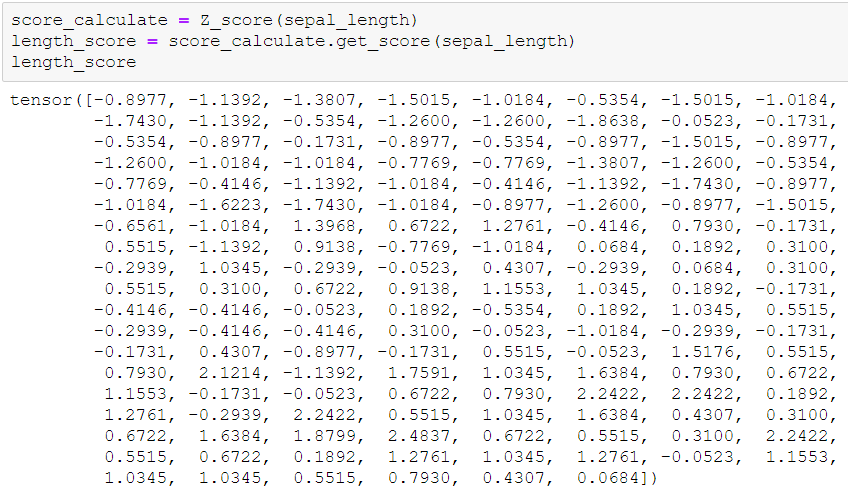
**** ****

**Хід роботи:**

1. Порахував Zscore для колонок sepal\_width, sepal\_length:

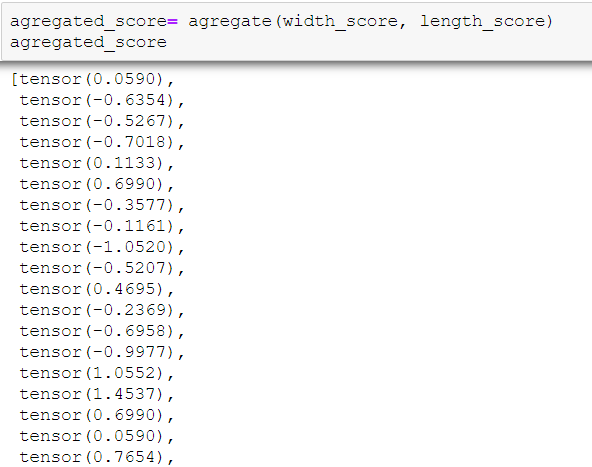


**Рис.1.** Zscore для sepal\_width

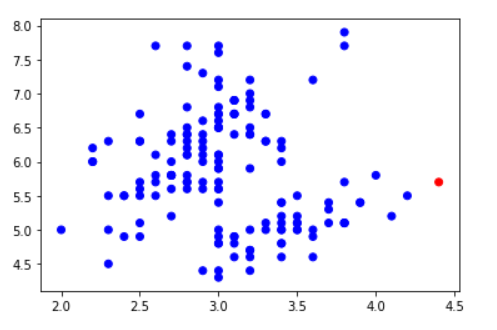


**Рис.2.** Zscore для sepal\_length

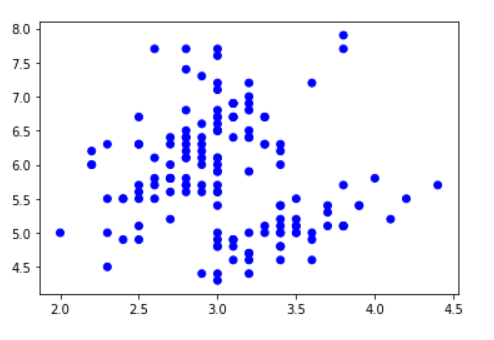
1. Агрегував пораховані дані Zscore:



**Рис.3.** Агреговані дані Zscore

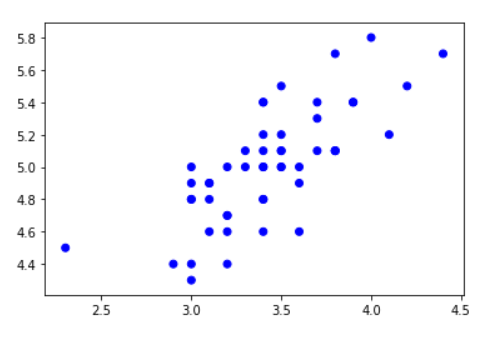


**Рис.4.** Візуалізація не типових заначень та типових sepal width(синій колір – типові, червоний – не типові)

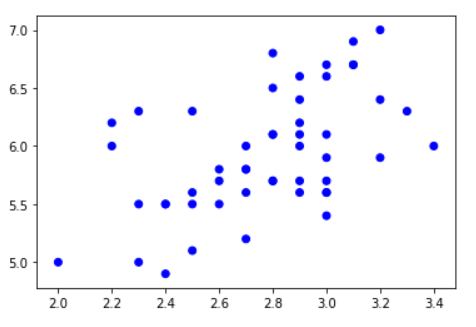


**Рис.5.** Візуалізація агрегованих даних Zscore(синій колір – типові)

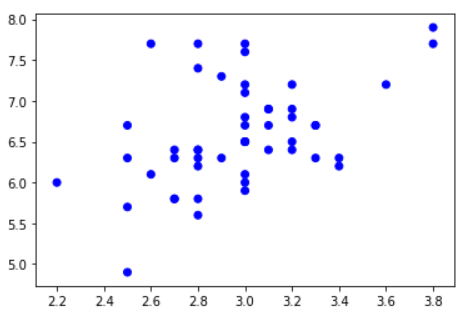
1. Так як у датасету iris є категоріальні дані, то потрібно ще порахувати Zscore для кожної з них:



**Рис.6.** Візуалізація категорії setosa та її Zscore (синій колір – типові для setosa)

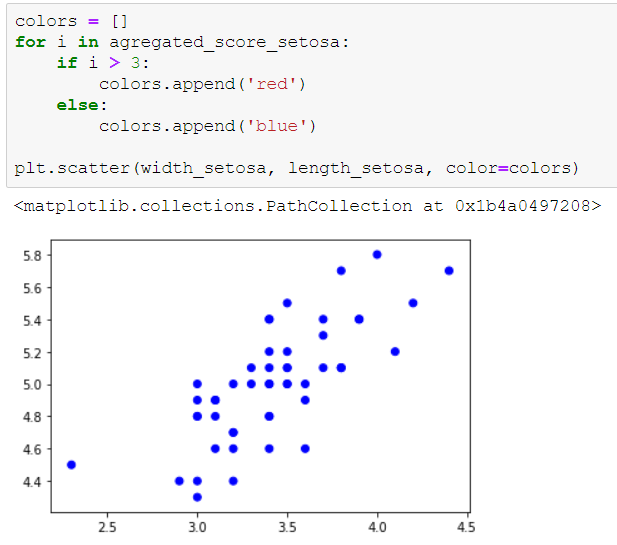


**Рис.7.** Візуалізація versicolor та її Zscore(синій колір – типові для versicolor)

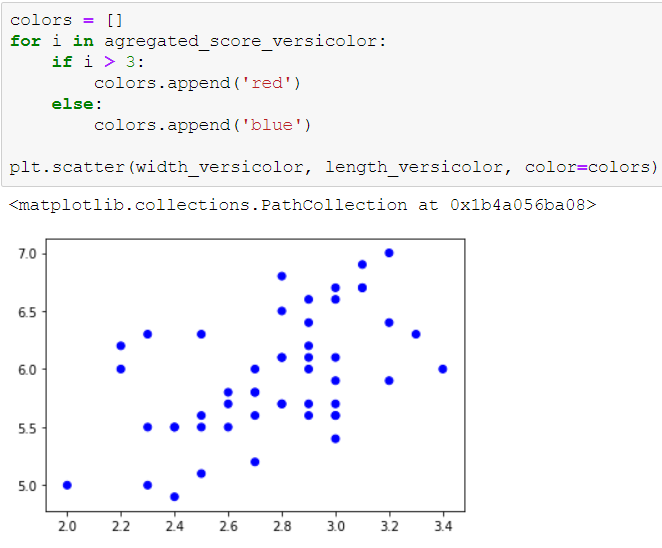


**Рис.8.** Візуалізація virginica та її Zscore (синій колір – типові для virginica)

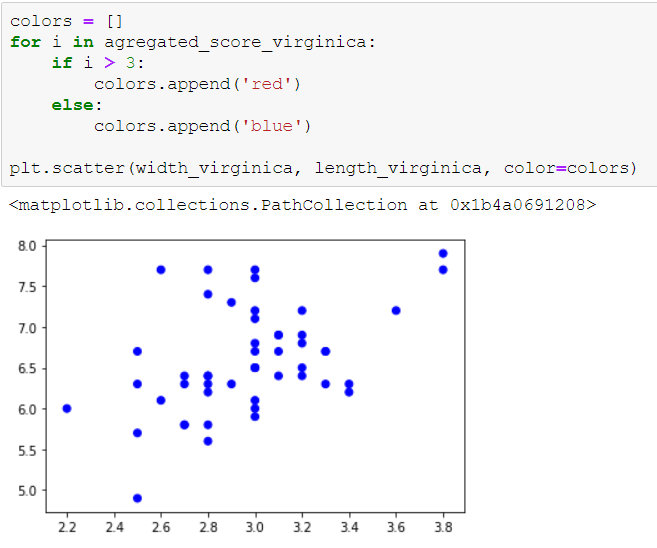
1. Агрегувати категоріальні дані setosa, virginica, versicolor:



**Рис.9.** Агрегував дані setosa (синій колір – агреговані для setosa)



**Рис.10.** Агрегував дані versicolor (синій колір – агреговані для versicolor)



**Рис.11.** Агрегував дані virginica(синій колір – агреговані для virginica)

**Висновок:**

На лабораторній було засвоєно основні відомості про роботу з фреймворком PyTorch, як розпізнавати не типові елементи в даних. Також були агреговані не типові елементи в датасеті iris після чого візуалізовано.

**Код програми:**

import torch

import seaborn as sns

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

iris = sns.load\_dataset("iris")

iris

class Z\_score:

def \_\_init\_\_(self, columns):

self.mean = torch.mean(columns)

self.sigma = torch.std(columns, axis = 0)

def get\_score(self, x):

return(x - self.mean)/self.sigma

def avarage\_score(self, x):

return torch.mean((x - self.mean)/self.sigma)

sepal\_width = torch.tensor(iris['sepal\_width'].values, dtype = torch.float)

sepal\_length = torch.tensor(iris['sepal\_length'].values, dtype = torch.float)

print(sepal\_width)

def agregate(score1, score2):

array = []

for i,j in zip(score1, score2):

array.append((i+j)/2)

return array

score\_calculate = Z\_score(sepal\_width)

width\_score = score\_calculate.get\_score(sepal\_width)

width\_score

score\_calculate = Z\_score(sepal\_length)

length\_score = score\_calculate.get\_score(sepal\_length)

length\_score

agregated\_score= agregate(width\_score, length\_score)

agregated\_score

colors = []

for i,j in zip(length\_score, width\_score):

if i > 3 or j > 3:

colors.append('red')

else:

colors.append('blue')

plt.scatter(sepal\_width, sepal\_length, color=colors)

colors = []

for i in agregated\_score:

if i > 3:

colors.append('red')

else:

colors.append('blue')

plt.scatter(sepal\_width, sepal\_length, color=colors)

setosa = iris.query("species == 'setosa'")

versicolor = iris.query("species == 'versicolor'")

virginica = iris.query("species == 'virginica'")

width\_setosa = torch.tensor(setosa['sepal\_width'].values, dtype=torch.float)

length\_setosa = torch.tensor(setosa['sepal\_length'].values, dtype=torch.float)

score\_calculate = Z\_score(width\_setosa)

width\_setosa\_score = score\_calculate.get\_score(width\_setosa)

score\_calculate = Z\_score(length\_setosa)

length\_setosa\_score = score\_calculate.get\_score(length\_setosa)

agregated\_score\_setosa = agregate(width\_setosa\_score, length\_setosa\_score)

colors = []

for i,j in zip(length\_setosa\_score, width\_setosa\_score):

if i > 3 or j > 3:

colors.append('red')

else:

colors.append('blue')

plt.scatter(width\_setosa, length\_setosa, color=colors)

colors = []

for i in agregated\_score\_setosa:

if i > 3:

colors.append('red')

else:

colors.append('blue')

plt.scatter(width\_setosa, length\_setosa, color=colors)

width\_versicolor = torch.tensor(versicolor['sepal\_width'].values, dtype=torch.float)

length\_versicolor = torch.tensor(versicolor['sepal\_length'].values, dtype=torch.float)

score\_calculate = Z\_score(width\_versicolor)

width\_versicolor\_score = score\_calculate.get\_score(width\_versicolor)

score\_calculate = Z\_score(length\_versicolor)

length\_versicolor\_score = score\_calculate.get\_score(length\_versicolor)

agregated\_score\_versicolor = agregate(width\_versicolor\_score, length\_versicolor\_score)

colors = []

for i,j in zip(length\_versicolor\_score, width\_versicolor\_score):

if i > 3 or j > 3:

colors.append('red')

else:

colors.append('blue')

plt.scatter(width\_versicolor, length\_versicolor, color=colors)

colors = []

for i in agregated\_score\_versicolor:

if i > 3:

colors.append('red')

else:

colors.append('blue')

plt.scatter(width\_versicolor, length\_versicolor, color=colors)

width\_virginica = torch.tensor(virginica['sepal\_width'].values, dtype=torch.float)

length\_virginica = torch.tensor(virginica['sepal\_length'].values, dtype=torch.float)

score\_calculate = Z\_score(width\_virginica)

width\_virginica\_score = score\_calculate.get\_score(width\_virginica)

score\_calculate = Z\_score(length\_virginica)

length\_virginica\_score = score\_calculate.get\_score(length\_virginica)

agregated\_score\_virginica = agregate(width\_virginica\_score, length\_virginica\_score)

colors = []

for i,j in zip(length\_virginica\_score, width\_virginica\_score):

if i > 3 or j > 3:

colors.append('red')

else:

colors.append('blue')

plt.scatter(width\_virginica, length\_virginica, color=colors)

colors = []

for i in agregated\_score\_virginica:

if i > 3:

colors.append('red')

else:

colors.append('blue')

plt.scatter(width\_virginica, length\_virginica, color=colors)