Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра КІТС

Звіт

З лабораторної роботи №6

з дисципліни: «Основи обчислюваного інтелекту»

на тему: «ПОПЕРЕДНЯ ОБРОБКА ДАНИХ»

Варіант №20(10)

Виконали: Перевірив:

ст. групи КІУКІ-20-10 Татарніков А. О.

Череповський М.А.

Харків 2023

6.1 Мета роботи:

Набуття навичок вирішення завдань попередньої обробка даних.

Хід роботи

Відповідно до табл. 1 отримати завдання на лабораторну роботу.

Вихідні дані до лабораторної роботи взяті з UCI-репозиторію, де кожній вибірці даних відповідає папка з відповідною назвою, в якій знаходяться файл з описом предметної області та безпосередньо файл з даними.

Таблиця 1

|  |  |
| --- | --- |
| Варіанту | Назва вибірки даних |
| 10 | ionosphere |

* Виконати опис початкових даних:
* виділити атрибути;

Вхідний набір містить 34 атрибуту, 33 числових атрибута та один бінарний. Перші 33 атрибута називаються “а” та номер атрибута. Останній має назву “class”.

* описати діапазон значень кожного з атрибутів або множину його значень (в залежності від початкових даних);

Перші 33 атрибута містять значення від -1 до 1, а останній може бути лише бінарним значенням.

* описати цільовий атрибут.

Цільовим атрибутом в датасеті Ionosphere є останній стовпчик з назвою "class". Цей атрибут містить бінарні значення "g" та "b", які відповідають можливим результатам радарного випромінювання (g - гудок, b - погано).

Таким чином, мета моделі, побудованої на основі цього датасету, полягає в тому, щоб передбачити, чи буде радарне випромінювання прийнято, чи відкинуто на основі вхідних атрибутів.

Код:

from pandas import pandas  
import statistics  
from sklearn.preprocessing import PowerTransformer  
import numpy as np  
from sklearn.preprocessing import QuantileTransformer  
  
def main():  
 ionosphere = pandas.read\_csv("ionosphere.csv")  
 mathExps = []  
 medians = []  
 standardDeviations = []  
 recurrentSpivednesses = []  
 hmeanes = []  
 mads = []  
 ranges = []  
 despertiones = []  
 maxes = []  
 mines = []  
 centers = []  
 normals = []  
  
 # ionosphere.fillna(ionosphere.mean(), inplace=True)  
 for i in range(1, 34):  
 mathExps.append(mathExpectation(ionosphere[prapare(i)]))  
 medians.append(median(ionosphere[prapare(i)]))  
 standardDeviations.append(standardDeviation(ionosphere[prapare(i)]))  
 recurrentSpivednesses.append(recurrentSpivedness(ionosphere[prapare(i)]))  
 hmeanes.append(attrHmean(ionosphere[prapare(i)]))  
 mads.append(attrMad(ionosphere[prapare(i)]))  
 ranges.append(attrRange(ionosphere[prapare(i)]))  
 despertiones.append( despertion(ionosphere[prapare(i)]))  
 maxes.append(max(ionosphere[prapare(i)]))  
 mines.append(min(ionosphere[prapare(i)]))  
 normals.append(normal(ionosphere[prapare(i)]))  
 centers.append(center(ionosphere[prapare(i)]))  
 print("mathExps: ", mathExps)  
 print("medians: ", medians)  
 print("standardDeviations: ", standardDeviations)  
 print("recurrentSpivednesses: ", recurrentSpivednesses)  
 print("hmeanes: ", hmeanes)  
 print("ranges: ", ranges)  
 print("despertiones: ", despertiones)  
 print("maxes: ", maxes)  
 print("mines: ", mines)  
 print("normals: ", normals)  
 print("centers: ", centers)  
 print("hyperBalls: ", hyperBallEncode(ionosphere))  
 print("hyperCubes: ", hyperCube(ionosphere))  
  
   
def prapare(number):  
 if number < 10:  
 return "a0" + str(number)  
 else:  
 return 'a' + str(number)  
  
  
def mathExpectation(attr):  
 mean = statistics.mean(attr)  
 return mean  
  
  
def median(attr):  
 mean = statistics.median(attr)  
 return mean  
  
  
def attrHmean(attr):  
 return (np.max(attr) + np.min(attr)) / 2  
  
  
  
def standardDeviation(attr):  
 mean = statistics.mean(attr)  
 std\_dev = statistics.stdev(attr) # oбчислення стандартного відхилення від середнього  
 return std\_dev  
  
  
def attrMad(attr):  
 mean = statistics.mean(attr)  
 # обчислення середнього модуля відхилень від середнього  
 return statistics.mean(np.absolute(attr - mean))  
  
  
  
def attrRange(attr):  
 return max(attr) - min(attr)  
  
  
def despertion(attr):  
 return statistics.variance(attr)  
  
def recurrentSpivedness(attr):  
 # compute the cumulative mean and median of the sequence  
 mean\_seq = np.cumsum(attr) / np.arange(1, len(attr) + 1)  
 median\_seq = np.array([np.median(attr[:i + 1]) for i in range(len(attr))])  
  
 # compute the recurrent ratios of mean and median  
 ratios = mean\_seq[:-1] / median\_seq[:-1]  
 return ratios  
  
  
def center(attr):  
 return attr - attr.mean()  
  
def normal(attr):  
 return (attr - attr.min()) / (attr.max() - attr.min())  
  
def hyperBallEncode(attr):  
 qt = QuantileTransformer(output\_distribution='normal')  
 qtData = qt.fit\_transform(attr.iloc[:, :-1])  
 return qtData  
  
  
def hyperCube(attr):  
 pt = PowerTransformer(method='yeo-johnson')  
 ptData = pt.fit\_transform(attr.iloc[:, :-1])  
 return ptData  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Результат виконання коду:

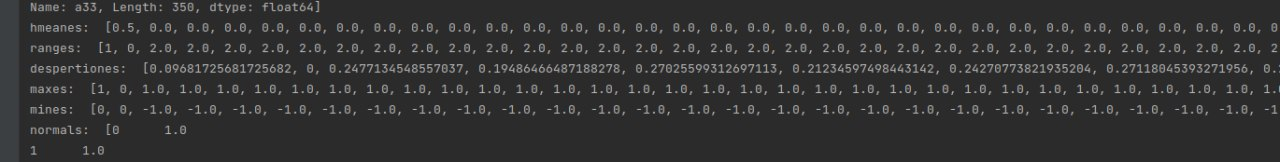


Рис 6.1 - Результат виконання коду(частина перша)

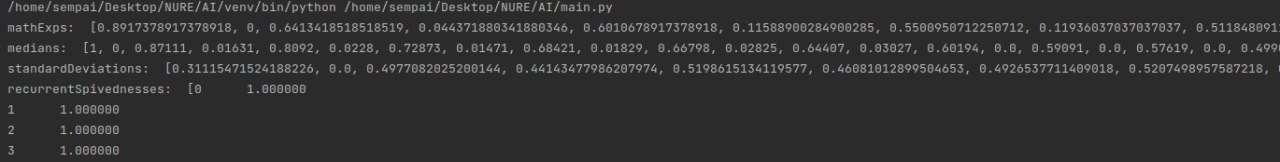


Рис 6.2 - Результат виконання коду(частина друга)

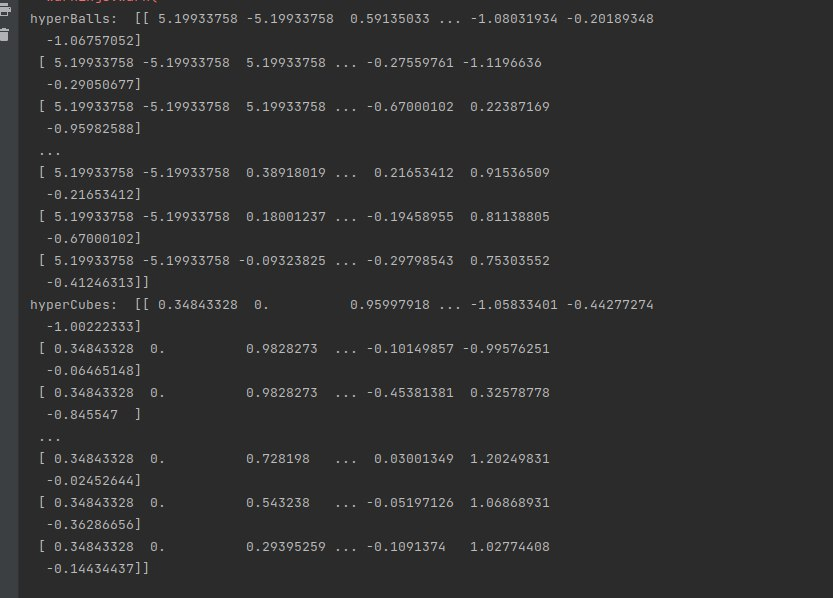


Рис 6.3 - Результат виконання коду(частина третя)

Висновок:

Для виконання цього завдання мною було вивчено та закріплено навички для обробки даних за допомогою Python. Для цього використовував бібліотеку pandas. Також використовував бібліотеки np, sklearn. За допомогою цих пакетів вдалось легко та швидко виконати аналіз вхідних даних. Також sklearn використовував для кодування на гіперкулю та гіперкуб.