Lavina Golani

16

Experiment 6:

**Winner Takes All:**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

file = '/home/lavina/Desktop/pima-indians-diabetes-database/diabetes.csv'

dataFrame = pd.read\_csv(file, names=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'])

dataFrame = dataFrame.dropna()

# print(dataFrame)

X = dataFrame.loc[:, ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]

y = dataFrame.loc[:, ['Outcome']]

data = X

# rows = labels and columns=features

scaler = StandardScaler()

scaler.fit(data)

StandardScaler(copy=True, with\_mean=True, with\_std=True)

X\_train = scaler.transform(data)

weights = np.random.rand(2, 8)

# weights = [[0.46, 0.461, 0.47, 0.44, 0.46, 0.46, 0.44, 0.46], [0.46, 0.462, 0.465, 0.46, 0.46, 0.46, 0.45, 0.45]]

score = [0, 0]

print(weights)

learningRate = 0.0001

groupA = []

groupB = []

print(X\_train)

index =1

for x in X\_train:

#x[1] has all the features& values print(x[1])

score1 = np.dot(x, weights[0])

score2 = np.dot(x, weights[1])

print('score1', score1)

print('score2', score2)

if score1 > score2:

error = 0.5 \* (np.dot(x - weights[0], x - weights[0]))

groupA.append(index)

itr = 0

while error > 0.001 and itr < 5:

weights[0] = weights[0] + learningRate \* (x - weights[0])

error = 0.5 \* (np.dot(x - weights[0], x - weights[0]))

itr += 1

else:

error = 0.5 \* (np.dot(x - weights[1], x - weights[1]))

groupB.append(index)

itr = 0

while error > 0.001 and itr < 5:

weights[1] = weights[1] + learningRate \* (x - weights[1])

error = 0.5 \* (np.dot(x - weights[1], x - weights[1]))

itr += 1

index +=1

print(groupA)

print(groupB)

OUTPUT:

[3, 4, 6, 7, 8, 9, 10, 14, 15, 16, 18, 21, 24, 25, 26, 28, 29, 32, 33, 34, 36, 41, 50, 51, 52, 53, 54, 55, 56, 57, 60, 61, 64, 66, 69, 70, 72, 74, 76, 80, 81, 82, 84, 86, 89, 90, 91, 92, 95, 96, 97, 98, 99, 102, 103, 104, 106, 108, 109, 111, 112, 113, 115, 119, 120, 122, 123, 128, 131, 133, 135, 136, 138, 140, 143, 145, 146, 147, 150, 151, 152, 154, 157, 158, 159, 163, 164, 166, 167, 170, 172, 173, 175, 176, 181, 183, 184, 187, 189, 190, 191, 192, 195, 196, 197, 198, 200, 204, 206, 209, 211, 215, 216, 217, 218, 221, 223, 225, 226, 229, 232, 233, 235, 240, 241, 242, 243, 244, 245, 248, 249, 250, 253, 255, 258, 259, 261, 262, 269, 270, 272, 274, 277, 278, 280, 282, 283, 287, 289, 290, 294, 296, 297, 298, 299, 302, 305, 307, 308, 310, 311, 312, 313, 314, 316, 317, 319, 321, 322, 326, 327, 330, 332, 334, 335, 336, 339, 341, 343, 347, 348, 349, 354, 357, 360, 361, 365, 366, 367, 368, 369, 371, 372, 374, 375, 376, 377, 378, 382, 383, 385, 386, 389, 391, 393, 394, 396, 397, 399, 406, 408, 410, 412, 413, 414, 415, 416, 419, 420, 422, 424, 425, 426, 427, 428, 431, 433, 434, 439, 442, 443, 447, 448, 449, 450, 451, 453, 455, 456, 458, 461, 462, 466, 467, 469, 475, 477, 478, 479, 481, 483, 484, 486, 487, 489, 495, 497, 498, 499, 500, 501, 508, 509, 512, 514, 515, 516, 519, 520, 521, 522, 523, 525, 526, 527, 528, 529, 531, 534, 536, 539, 541, 542, 545, 546, 548, 551, 552, 554, 555, 556, 560, 562, 563, 564, 566, 568, 569, 570, 572, 573, 574, 575, 576, 577, 579, 582, 585, 586, 588, 590, 592, 595, 596, 598, 600, 602, 603, 605, 607, 608, 609, 610, 611, 612, 613, 616, 618, 620, 621, 625, 627, 630, 632, 633, 634, 635, 638, 640, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 663, 666, 668, 669, 670, 672, 679, 680, 681, 686, 687, 688, 689, 695, 696, 697, 698, 699, 701, 705, 707, 708, 710, 711, 714, 716, 719, 722, 727, 729, 730, 731, 732, 734, 737, 739, 742, 743, 746, 749, 753, 754, 761, 763, 766, 768]

[1, 2, 5, 11, 12, 13, 17, 19, 20, 22, 23, 27, 30, 31, 35, 37, 38, 39, 40, 42, 43, 44, 45, 46, 47, 48, 49, 58, 59, 62, 63, 65, 67, 68, 71, 73, 75, 77, 78, 79, 83, 85, 87, 88, 93, 94, 100, 101, 105, 107, 110, 114, 116, 117, 118, 121, 124, 125, 126, 127, 129, 130, 132, 134, 137, 139, 141, 142, 144, 148, 149, 153, 155, 156, 160, 161, 162, 165, 168, 169, 171, 174, 177, 178, 179, 180, 182, 185, 186, 188, 193, 194, 199, 201, 202, 203, 205, 207, 208, 210, 212, 213, 214, 219, 220, 222, 224, 227, 228, 230, 231, 234, 236, 237, 238, 239, 246, 247, 251, 252, 254, 256, 257, 260, 263, 264, 265, 266, 267, 268, 271, 273, 275, 276, 279, 281, 284, 285, 286, 288, 291, 292, 293, 295, 300, 301, 303, 304, 306, 309, 315, 318, 320, 323, 324, 325, 328, 329, 331, 333, 337, 338, 340, 342, 344, 345, 346, 350, 351, 352, 353, 355, 356, 358, 359, 362, 363, 364, 370, 373, 379, 380, 381, 384, 387, 388, 390, 392, 395, 398, 400, 401, 402, 403, 404, 405, 407, 409, 411, 417, 418, 421, 423, 429, 430, 432, 435, 436, 437, 438, 440, 441, 444, 445, 446, 452, 454, 457, 459, 460, 463, 464, 465, 468, 470, 471, 472, 473, 474, 476, 480, 482, 485, 488, 490, 491, 492, 493, 494, 496, 502, 503, 504, 505, 506, 507, 510, 511, 513, 517, 518, 524, 530, 532, 533, 535, 537, 538, 540, 543, 544, 547, 549, 550, 553, 557, 558, 559, 561, 565, 567, 571, 578, 580, 581, 583, 584, 587, 589, 591, 593, 594, 597, 599, 601, 604, 606, 614, 615, 617, 619, 622, 623, 624, 626, 628, 629, 631, 636, 637, 639, 641, 642, 643, 658, 659, 660, 661, 662, 664, 665, 667, 671, 673, 674, 675, 676, 677, 678, 682, 683, 684, 685, 690, 691, 692, 693, 694, 700, 702, 703, 704, 706, 709, 712, 713, 715, 717, 718, 720, 721, 723, 724, 725, 726, 728, 733, 735, 736, 738, 740, 741, 744, 745, 747, 748, 750, 751, 752, 755, 756, 757, 758, 759, 760, 762, 764, 765, 767]

**K-means**

from sklearn.cluster import KMeans

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report,confusion\_matrix

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

file = '/home/lavina/Desktop/pima-indians-diabetes-database/diabetes.csv'

dataFrame = pd.read\_csv(file, names=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'])

X = dataFrame.loc[:, ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]

y = dataFrame.loc[:, ['Outcome']]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30)

kmeans = KMeans(n\_clusters=2, random\_state=0, max\_iter=200).fit(X\_train)

print(kmeans.labels\_)

predictions = kmeans.predict(X\_test)

print(kmeans.cluster\_centers\_)

print(confusion\_matrix(y\_test, predictions))

print(classification\_report(y\_test, predictions))

