**EGE UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**204 DATA STRUCTURES (3+1)**

**2023–2024 FALL SEMESTER**

**PROJECT-1 REPORT**

**(Arrays, Matrices, Methods, Classes, Random Numbers)**

**DELIVERY DATE**

19/11/23

**PREPARED BY**

05220000981, Burak Taşkın

052100000278, İsmail Erkan

İçindekiler

[1) KARAYOLLARI UZAKLIK HESAPLAMALARI 2](#_Toc149748651)

[1.a Uzaklık Matrisi Oluşturma 2](#_Toc149748652)

[1.a.1 Kodlar 2](#_Toc149748653)

[1.a.2 Ekran görüntüleri 2](#_Toc149748654)

[1.b Verilen İlden Belli Bir Uzaklığa Kadar Olan İllerin ve Uzaklıklarının Listelenmesi 2](#_Toc149748655)

[1.b.1 Kodlar 2](#_Toc149748656)

[1.b.2 Ekran görüntüleri 2](#_Toc149748657)

[1.c Türkiye’deki Birbirine En Yakın İki İlin ve En Uzak İki İlin Bulunması 2](#_Toc149748658)

[1.c.1 Kodlar 2](#_Toc149748659)

[1.c.2 Ekran görüntüleri 2](#_Toc149748660)

[1.d Verilen İlden Verilen Mesafe Kullanılarak En fazla Kaç İl Dolaşılabildiğinin Bulunması 2](#_Toc149748661)

[1.d.1 Kodlar 2](#_Toc149748662)

[1.d.2 Algoritma ve Açıklama 2](#_Toc149748663)

[1.d.3 Ekran görüntüleri 2](#_Toc149748664)

[1.e Matris Şeklinde İllerin Adlarıyla Birlikte Ekrana Listelenmesi 2](#_Toc149748665)

[1.e.1 Kodlar 2](#_Toc149748666)

[1.e.2 Ekran görüntüleri 2](#_Toc149748667)

[2) DEVELOPING A PERCEPTRON MODEL and IMPLEMENTATION of a REGRESSION EXAMPLE 3](#_Toc149748668)

[2.a Neuron (Sinir Hücresi) Sınıfı 3](#_Toc149748669)

[2.a.1 Kaynak Kod 3](#_Toc149748670)

[2.a.2 Açıklama 3](#_Toc149748671)

[2.b Eğitim 3](#_Toc149748672)

[2.b.1 Kaynak Kod 3](#_Toc149748673)

[2.b.2 Açıklama 3](#_Toc149748674)

[2.b.3 Ekran Görüntüleri 3](#_Toc149748675)

[2.c Modelin Görmediği Veriden Sınav Sonucu Tahminleme 3](#_Toc149748676)

[2.c.1 Kaynak Kod 3](#_Toc149748677)

[2.c.2 Sonuçlar/Ekran görüntüleri 3](#_Toc149748678)

[2.d Deneyler 3](#_Toc149748679)

[2.d.1 Kaynak Kod 3](#_Toc149748680)

[2.d.2 Sonuçlar 3](#_Toc149748681)

[Öz değerlendirme Tablosu 4](#_Toc149748682)

[1) KARAYOLLARI UZAKLIK HESAPLAMALARI 2](#_Toc149748651)

# 1) KARAYOLLARI UZAKLIK HESAPLAMALARI

//The platform, version, and programming language used

## 1.a Uzaklık Matrisi Oluşturma

### 1.a.1 Kodlar

int[][] jagged\_array = new int[82][];

jagged\_array[0] = new int[] { 0 };

jagged\_array[1] = new int[] { 0, 0 };

jagged\_array[2] = new int[] { 0, 337, 0 };

jagged\_array[3] = new int[] { 0, 578, 915, 0 };

jagged\_array[4] = new int[] { 0, 973, 642, 1320, 0 };

jagged\_array[5] = new int[] { 0, 603, 627, 596, 740, 0 };

jagged\_array[6] = new int[] { 0, 492, 732, 255, 1055, 331, 0 };

jagged\_array[7] = new int[] { 0, 547, 884, 287, 1424, 823, 542, 0 };

jagged\_array[8] = new int[] { 0, 1001, 715, 1224, 364, 679, 960, 1430, 0 };

jagged\_array[9] = new int[] { 0, 880, 1217, 340, 1634, 935, 595, 345, 1564, 0 };

jagged\_array[10] = new int[] { 0, 902, 1239, 324, 1579, 845, 536, 505, 1454, 291, 0 };

jagged\_array[11] = new int[] { 0, 785, 1032, 207, 1361, 626, 314, 472, 1236, 525, 257, 0 };

jagged\_array[12] = new int[] { 0, 631, 347, 1096, 352, 641, 887, 1188, 371, 1398, 1410, 1187, 0 };

jagged\_array[13] = new int[] { 0, 741, 410, 1287, 232, 832, 1078, 1288, 545, 1589, 1601, 1378, 192, 0 };

jagged\_array[14] = new int[] { 0, 688, 920, 417, 1146, 411, 190, 682, 1021, 735, 434, 215, 1075, 1266, 0 };

jagged\_array[15] = new int[] { 0, 648, 985, 166, 1402, 761, 421, 121, 1390, 286, 393, 351, 1166, 1357, 561, 0 };

jagged\_array[16] = new int[] { 0, 856, 1104, 278, 1418, 683, 385, 543, 1293, 441, 152, 95, 1259, 1450, 272, 422, 0 };

jagged\_array[17] = new int[] { 0, 1096, 1383, 517, 1697, 962, 665, 695, 1572, 448, 194, 374, 1538, 1729, 551, 588, 269, 0 };

jagged\_array[18] = new int[] { 0, 587, 763, 398, 986, 247, 131, 685, 881, 737, 666, 447, 918, 1109, 233, 564, 504, 783, 0 };

jagged\_array[19] = new int[] { 0, 582, 697, 504, 831, 91, 240, 732, 720, 844, 785, 563, 762, 953, 349, 670, 634, 914, 155, 0 };

jagged\_array[20] = new int[] { 0, 762, 1099, 222, 1516, 817, 477, 227, 1446, 131, 292, 407, 1280, 1471, 617, 168, 442, 482, 620, 726, 0 };

jagged\_array[21] = new int[] { 0, 537, 206, 1108, 436, 706, 898, 1084, 509, 1409, 1421, 1199, 141, 204, 1086, 1177, 1270, 1550, 929, 773, 1291, 0 };

jagged\_array[22] = new int[] { 0, 1183, 1415, 682, 1641, 906, 685, 922, 1516, 667, 419, 480, 1570, 1761, 495, 826, 428, 232, 727, 844, 709, 1581, 0 };

jagged\_array[23] = new int[] { 0, 494, 283, 959, 492, 538, 749, 1051, 511, 1260, 1272, 1050, 141, 332, 937, 1028, 1121, 1401, 780, 605, 1142, 152, 1432, 0 };

jagged\_array[24] = new int[] { 0, 676, 546, 955, 370, 375, 690, 1059, 376, 1269, 1214, 996, 272, 463, 781, 1037, 1053, 1332, 621, 466, 1151, 407, 1276, 267 };

jagged\_array[25] = new int[] { 0, 807, 521, 1140, 182, 560, 876, 1244, 194, 1454, 1400, 1181, 177, 351, 966, 1222, 1238, 1517, 807, 651, 1336, 315, 1461, 317, 191, 0 };

jagged\_array[26] = new int[] { 0, 690, 952, 141, 1298, 574, 233, 421, 1202, 474, 303, 80, 1107, 1298, 290, 300, 152, 431, 376, 482, 356, 1119, 555, 970, 933, 1118, 0 };

jagged\_array[27] = new int[] { 0, 216, 151, 794, 757, 577, 647, 763, 846, 1096, 1118, 948, 476, 525, 836, 864, 1020, 1299, 678, 613, 978, 321, 1330, 338, 602, 652, 868, 0 };

jagged\_array[28] = new int[] { 0, 720, 705, 867, 538, 322, 603, 1094, 357, 1207, 1098, 879, 536, 710, 664, 1033, 936, 1215, 524, 363, 1089, 674, 1159, 555, 294, 359, 845, 694, 0 };

jagged\_array[29] = new int[] { 0, 789, 686, 1026, 390, 431, 762, 1172, 320, 1366, 1256, 1038, 388, 562, 823, 1150, 1095, 1374, 683, 522, 1248, 526, 1318, 407, 145, 211, 1004, 763, 159, 0 };

jagged\_array[30] = new int[] { 0, 901, 654, 1480, 424, 1164, 1333, 1448, 755, 1781, 1804, 1633, 492, 324, 1521, 1550, 1705, 1984, 1364, 1252, 1663, 483, 2016, 631, 794, 606, 1553, 685, 963, 814, 0 };

jagged\_array[31] = new int[] { 0, 196, 316, 774, 951, 675, 688, 743, 980, 1076, 1098, 981, 610, 720, 884, 844, 1052, 1291, 776, 711, 958, 515, 1379, 472, 748, 786, 886, 194, 791, 861, 880, 0 };

jagged\_array[32] = new int[] { 0, 618, 956, 167, 1372, 723, 383, 131, 1352, 288, 395, 352, 1136, 1327, 562, 30, 423, 590, 526, 632, 170, 1148, 828, 999, 1007, 1192, 301, 834, 995, 1120, 1520, 814, 0 };

jagged\_array[33] = new int[] { 0, 70, 407, 571, 1043, 636, 484, 466, 1071, 810, 895, 778, 701, 811, 681, 574, 849, 1088, 580, 575, 693, 606, 1176, 564, 746, 877, 683, 286, 790, 859, 971, 265, 563, 0 };

jagged\_array[34] = new int[] { 0, 951, 1183, 450, 1409, 674, 453, 715, 1284, 682, 393, 248, 1338, 1529, 263, 594, 244, 316, 495, 612, 650, 1349, 232, 1200, 1044, 1229, 323, 1098, 927, 1086, 1784, 1147, 595, 943, 0 };

jagged\_array[35] = new int[] { 0, 907, 1244, 328, 1648, 924, 584, 448, 1552, 128, 175, 430, 1425, 1616, 607, 389, 325, 327, 726, 833, 234, 1436, 546, 1287, 1283, 1468, 413, 1123, 1195, 1354, 1808, 1102, 391, 899, 566, 0 };

jagged\_array[36] = new int[] { 0, 1012, 726, 1342, 214, 762, 1078, 1447, 203, 1656, 1602, 1383, 382, 447, 1168, 1425, 1440, 1719, 1009, 854, 1538, 520, 1663, 522, 393, 205, 1321, 857, 561, 413, 554, 991, 1395, 1082, 1431, 1671, 0 };

jagged\_array[37] = new int[] { 0, 693, 868, 503, 990, 255, 237, 790, 846, 843, 680, 462, 892, 1083, 247, 669, 518, 798, 106, 193, 725, 959, 741, 791, 625, 810, 482, 784, 489, 648, 1414, 882, 631, 686, 509, 832, 1013, 0 };

jagged\_array[38] = new int[] { 0, 335, 414, 523, 816, 343, 315, 615, 822, 824, 847, 616, 569, 760, 504, 593, 688, 967, 346, 281, 706, 581, 998, 432, 451, 636, 536, 330, 494, 564, 1015, 428, 563, 328, 766, 851, 838, 452, 0 };

jagged\_array[39] = new int[] { 0, 1159, 1391, 659, 1617, 883, 662, 924, 1492, 681, 433, 457, 1547, 1738, 471, 803, 443, 246, 704, 821, 723, 1558, 68, 1409, 1252, 1438, 532, 1307, 1135, 1294, 1992, 1355, 804, 1152, 209, 560, 1640, 718, 975, 0 };

jagged\_array[40] = new int[] { 0, 377, 549, 425, 940, 312, 183, 568, 941, 765, 706, 483, 704, 895, 371, 546, 555, 834, 214, 221, 647, 715, 866, 566, 575, 760, 403, 465, 584, 688, 1150, 562, 516, 369, 634, 753, 962, 319, 133, 842, 0 };

jagged\_array[41] = new int[] { 0, 840, 1072, 340, 1298, 563, 342, 605, 1173, 569, 280, 138, 1227, 1418, 152, 483, 131, 398, 385, 502, 540, 1239, 345, 1090, 933, 1118, 213, 988, 816, 975, 1673, 1036, 485, 833, 113, 453, 1321, 399, 656, 321, 523, 0 };

jagged\_array[42] = new int[] { 0, 357, 695, 226, 1111, 510, 259, 320, 1117, 527, 550, 433, 875, 1066, 455, 296, 504, 743, 354, 419, 410, 887, 908, 738, 746, 932, 337, 573, 781, 859, 1259, 553, 266, 350, 676, 554, 1134, 460, 302, 885, 255, 566, 0 };

jagged\_array[43] = new int[] { 0, 675, 1012, 97, 1377, 652, 312, 361, 1281, 414, 227, 110, 1193, 1384, 321, 240, 182, 421, 455, 561, 296, 1204, 586, 1055, 1012, 1197, 79, 891, 924, 1083, 1576, 871, 242, 668, 354, 335, 1399, 567, 619, 562, 482, 243, 323, 0 };

jagged\_array[44] = new int[] { 0, 397, 186, 861, 585, 460, 652, 953, 605, 1163, 1175, 953, 235, 426, 840, 931, 1024, 1304, 683, 527, 1045, 246, 1335, 97, 361, 411, 872, 241, 538, 501, 725, 375, 901, 466, 1103, 1190, 615, 713, 335, 1312, 469, 993, 641, 958, 0 };

jagged\_array[45] = new int[] { 0, 886, 1223, 307, 1627, 903, 562, 427, 1531, 152, 138, 393, 1403, 1594, 570, 368, 288, 328, 705, 812, 213, 1415, 556, 1266, 1262, 1447, 392, 1102, 1174, 1333, 1787, 1081, 370, 878, 529, 37, 1650, 811, 830, 570, 732, 417, 533, 314, 1169, 0 };

jagged\_array[46] = new int[] { 0, 197, 162, 776, 807, 501, 572, 744, 827, 1077, 1095, 873, 457, 573, 760, 846, 944, 1224, 603, 538, 959, 368, 1255, 319, 575, 633, 792, 78, 618, 688, 764, 176, 816, 267, 1023, 1104, 837, 709, 255, 1232, 389, 913, 555, 872, 222, 1083, 0 };

jagged\_array[47] = new int[] { 0, 542, 294, 1120, 513, 800, 974, 1089, 604, 1422, 1444, 1274, 236, 281, 1162, 1190, 1346, 1625, 1004, 868, 1304, 94, 1657, 247, 501, 410, 1194, 326, 769, 620, 388, 521, 1160, 612, 1424, 1449, 615, 1054, 656, 1633, 791, 1314, 900, 1217, 341, 1428, 404, 0 };

jagged\_array[48] = new int[] { 0, 856, 1193, 348, 1642, 944, 604, 309, 1572, 98, 389, 533, 1406, 1597, 743, 240, 539, 547, 746, 853, 151, 1418, 766, 1269, 1277, 1463, 482, 1072, 1215, 1374, 1757, 1051, 270, 775, 776, 226, 1665, 852, 833, 780, 773, 666, 536, 423, 1172, 251, 1053, 1398, 0 };

jagged\_array[49] = new int[] { 0, 741, 455, 1206, 241, 751, 997, 1298, 464, 1507, 1520, 1297, 111, 81, 1185, 1276, 1369, 1648, 1027, 872, 1390, 249, 1680, 251, 381, 270, 1217, 586, 629, 480, 381, 720, 1246, 811, 1447, 1534, 347, 1001, 679, 1656, 814, 1337, 985, 1302, 344, 1513, 566, 362, 1516, 0 };

jagged\_array[50] = new int[] { 0, 287, 497, 444, 888, 357, 275, 536, 894, 745, 768, 576, 652, 843, 463, 514, 647, 927, 306, 295, 628, 664, 958, 515, 523, 709, 496, 413, 567, 636, 1098, 483, 484, 280, 726, 772, 911, 412, 79, 935, 92, 616, 223, 540, 418, 751, 338, 739, 754, 762, 0 };

jagged\_array[51] = new int[] { 0, 210, 548, 464, 939, 439, 339, 544, 945, 765, 787, 637, 703, 894, 546, 533, 709, 981, 389, 377, 647, 715, 1041, 566, 574, 760, 557, 426, 618, 687, 1111, 405, 504, 202, 809, 792, 962, 494, 130, 1017, 175, 698, 243, 560, 469, 771, 389, 752, 774, 813, 83, 0 };

jagged\_array[52] = new int[] { 0, 707, 731, 823, 583, 278, 558, 1050, 402, 1162, 1053, 835, 581, 755, 620, 988, 891, 1171, 480, 319, 1045, 719, 1115, 600, 339, 404, 801, 681, 45, 204, 1007, 779, 951, 777, 882, 1151, 606, 445, 481, 1091, 540, 772, 737, 880, 564, 1130, 605, 814, 1171, 674, 614, 605, 0 };

jagged\_array[53] = new int[] { 0, 917, 772, 1076, 430, 531, 811, 1303, 148, 1415, 1306, 1088, 428, 602, 873, 1241, 1144, 1424, 733, 572, 1298, 566, 1368, 568, 306, 250, 1054, 891, 209, 172, 854, 989, 1204, 987, 1135, 1404, 344, 698, 692, 1344, 793, 1025, 990, 1133, 662, 1383, 816, 660, 1424, 520, 764, 815, 254, 0 };

jagged\_array[54] = new int[] { 0, 803, 1035, 302, 1261, 526, 305, 567, 1136, 620, 319, 100, 1190, 1381, 115, 446, 157, 436, 347, 464, 502, 1201, 380, 1052, 896, 1081, 176, 950, 779, 938, 1636, 999, 448, 796, 148, 492, 1283, 361, 618, 357, 486, 37, 528, 206, 955, 455, 875, 1277, 629, 1300, 578, 661, 734, 988, 0 };

jagged\_array[55] = new int[] { 0, 754, 756, 676, 730, 131, 411, 903, 548, 1016, 906, 688, 718, 909, 473, 842, 745, 1024, 333, 172, 898, 835, 968, 667, 451, 550, 654, 706, 191, 350, 1154, 804, 804, 746, 736, 1004, 752, 298, 453, 944, 393, 625, 590, 733, 589, 983, 630, 929, 1024, 827, 467, 549, 147, 400, 588, 0 };

jagged\_array[56] = new int[] { 0, 715, 384, 1295, 328, 893, 1085, 1262, 641, 1596, 1608, 1386, 288, 96, 1273, 1364, 1457, 1737, 1116, 960, 1478, 187, 1768, 339, 558, 447, 1306, 499, 806, 658, 291, 694, 1335, 785, 1536, 1623, 542, 1146, 768, 1745, 902, 1426, 1074, 1391, 433, 1602, 547, 235, 1605, 177, 851, 902, 851, 697, 1388, 1022, 0 };

jagged\_array[57] = new int[] { 0, 865, 882, 675, 893, 257, 409, 962, 711, 1015, 861, 642, 862, 1053, 427, 841, 699, 978, 278, 264, 897, 961, 922, 793, 595, 713, 654, 832, 354, 513, 1317, 929, 803, 857, 690, 1004, 915, 182, 545, 899, 491, 580, 632, 748, 715, 983, 756, 1055, 1024, 971, 584, 666, 310, 563, 542, 163, 1148, 0 };

jagged\_array[58] = new int[] { 0, 428, 413, 706, 618, 222, 442, 810, 624, 1020, 987, 764, 480, 671, 630, 788, 836, 1115, 444, 289, 902, 492, 1125, 324, 253, 438, 684, 402, 296, 366, 971, 499, 758, 497, 893, 1034, 640, 475, 202, 1101, 326, 782, 498, 763, 246, 1013, 326, 586, 1029, 590, 275, 326, 307, 494, 745, 351, 679, 477, 0 };

jagged\_array[59] = new int[] { 0, 1082, 1313, 581, 1539, 805, 584, 846, 1414, 617, 369, 379, 1469, 1660, 393, 725, 379, 182, 626, 743, 659, 1480, 148, 1331, 1174, 1360, 454, 1229, 1057, 1216, 1914, 1277, 726, 1074, 131, 496, 1562, 640, 897, 124, 764, 243, 807, 484, 1234, 506, 1154, 1555, 716, 1578, 857, 939, 1013, 1266, 279, 866, 1667, 821, 1023, 0 };

jagged\_array[60] = new int[] { 0, 492, 516, 645, 674, 113, 380, 872, 616, 984, 926, 703, 584, 775, 522, 810, 775, 1054, 335, 180, 866, 595, 1017, 427, 309, 494, 623, 466, 259, 365, 1074, 564, 772, 562, 785, 973, 696, 366, 267, 994, 332, 674, 559, 701, 349, 952, 391, 689, 993, 693, 339, 390, 215, 468, 637, 242, 782, 368, 111, 916, 0 };

jagged\_array[61] = new int[] { 0, 838, 778, 997, 482, 452, 733, 1224, 227, 1337, 1228, 1009, 479, 653, 794, 1163, 1066, 1345, 654, 493, 1219, 617, 1289, 499, 237, 302, 975, 812, 130, 102, 906, 910, 1125, 908, 1057, 1325, 422, 619, 613, 1265, 714, 946, 911, 1054, 592, 1304, 737, 712, 1345, 572, 685, 736, 175, 79, 909, 321, 749, 484, 415, 1187, 389, 0 };

jagged\_array[62] = new int[] { 0, 630, 419, 1078, 418, 498, 813, 1182, 424, 1392, 1338, 1119, 145, 336, 904, 1160, 1176, 1455, 745, 589, 1274, 280, 1399, 140, 128, 239, 1056, 475, 417, 268, 635, 609, 1130, 700, 1167, 1406, 441, 748, 574, 1375, 698, 1056, 869, 1135, 233, 1385, 455, 374, 1400, 254, 646, 697, 462, 430, 1019, 574, 431, 718, 376, 1297, 432, 360, 0 };

jagged\_array[63] = new int[] { 0, 355, 111, 933, 613, 710, 786, 902, 686, 1234, 1257, 1087, 318, 381, 974, 1003, 1158, 1438, 817, 752, 1117, 177, 1469, 318, 573, 492, 1006, 139, 788, 713, 547, 333, 973, 424, 1237, 1261, 697, 923, 468, 1446, 603, 1127, 712, 1030, 269, 1240, 217, 188, 1210, 426, 552, 564, 814, 742, 1089, 839, 355, 965, 496, 1368, 599, 804, 445, 0 };

jagged\_array[64] = new int[] { 0, 690, 1027, 112, 1432, 707, 367, 290, 1336, 272, 224, 249, 1208, 1399, 459, 169, 320, 420, 510, 616, 154, 1219, 642, 1070, 1067, 1252, 217, 906, 979, 1138, 1592, 886, 170, 683, 492, 216, 1454, 615, 635, 656, 537, 381, 338, 138, 973, 195, 888, 1232, 280, 1318, 556, 575, 935, 1188, 344, 788, 1406, 787, 818, 592, 756, 1109, 1190, 1045, 0 };

jagged\_array[65] = new int[] { 0, 900, 569, 1422, 227, 967, 1212, 1447, 558, 1723, 1735, 1513, 327, 159, 1373, 1491, 1584, 1864, 1214, 1058, 1605, 363, 1868, 466, 598, 410, 1433, 684, 766, 618, 197, 879, 1462, 970, 1636, 1750, 357, 1217, 895, 1845, 1029, 1526, 1201, 1518, 560, 1729, 732, 440, 1732, 216, 978, 1029, 811, 657, 1488, 957, 255, 1120, 806, 1767, 901, 709, 470, 540, 1533, 0 };

jagged\_array[66] = new int[] { 0, 490, 613, 479, 841, 199, 214, 681, 828, 818, 760, 537, 700, 891, 402, 644, 609, 888, 245, 108, 700, 712, 897, 544, 476, 661, 457, 529, 471, 589, 1191, 626, 606, 482, 665, 807, 864, 298, 197, 874, 113, 555, 368, 536, 466, 786, 453, 806, 827, 810, 205, 288, 427, 680, 517, 280, 899, 372, 227, 796, 219, 601, 599, 667, 591, 1026, 0 };

jagged\_array[67] = new int[] { 0, 764, 996, 483, 1205, 471, 266, 748, 1058, 801, 500, 281, 1107, 1298, 158, 627, 338, 617, 292, 409, 683, 1163, 561, 1014, 841, 1026, 356, 912, 701, 860, 1630, 960, 629, 757, 329, 673, 1228, 214, 580, 538, 447, 218, 531, 387, 916, 636, 836, 1238, 810, 1217, 540, 622, 657, 910, 181, 510, 1350, 394, 691, 460, 582, 831, 964, 1050, 525, 1433, 479, 0 };

jagged\_array[68] = new int[] { 0, 268, 573, 368, 964, 421, 224, 460, 970, 669, 692, 522, 728, 919, 420, 438, 594, 873, 319, 330, 552, 740, 915, 591, 599, 785, 442, 484, 693, 712, 1169, 464, 408, 261, 683, 696, 987, 425, 155, 891, 108, 572, 147, 464, 494, 675, 414, 810, 678, 838, 76, 116, 648, 840, 535, 501, 927, 597, 351, 813, 415, 761, 722, 623, 480, 1054, 221, 496, 0 };

jagged\_array[69] = new int[] { 0, 801, 674, 1079, 304, 457, 815, 1183, 310, 1393, 1297, 1078, 302, 476, 863, 1161, 1135, 1414, 704, 548, 1275, 440, 1358, 395, 157, 125, 1057, 730, 234, 86, 728, 872, 1131, 870, 1126, 1407, 327, 723, 575, 1335, 699, 1015, 871, 1136, 489, 1386, 699, 535, 1402, 395, 648, 699, 279, 247, 978, 425, 572, 588, 377, 1257, 391, 177, 256, 617, 1191, 532, 600, 923, 724, 0 };

jagged\_array[70] = new int[] { 0, 293, 630, 339, 1112, 621, 370, 376, 1118, 641, 663, 546, 876, 1034, 566, 409, 617, 857, 465, 530, 523, 829, 1022, 739, 747, 933, 451, 508, 791, 860, 1194, 488, 379, 233, 789, 668, 1135, 571, 303, 998, 320, 679, 118, 436, 642, 646, 490, 835, 649, 986, 256, 178, 848, 988, 642, 702, 1008, 743, 499, 920, 563, 909, 870, 647, 451, 1202, 433, 642, 212, 872, 0 };

jagged\_array[71] = new int[] { 0, 483, 658, 338, 985, 260, 74, 562, 889, 678, 619, 397, 813, 1004, 262, 504, 468, 748, 105, 169, 560, 824, 757, 676, 620, 805, 316, 574, 532, 691, 1259, 671, 459, 475, 525, 667, 1007, 210, 242, 733, 109, 414, 250, 395, 578, 645, 498, 900, 687, 923, 201, 284, 488, 741, 377, 341, 1011, 382, 371, 655, 309, 662, 743, 712, 450, 1139, 143, 338, 215, 744, 361, 0 };

jagged\_array[72] = new int[] { 0, 628, 297, 1207, 365, 805, 998, 1175, 563, 1508, 1521, 1298, 195, 133, 1186, 1277, 1370, 1649, 1029, 873, 1391, 99, 1681, 252, 467, 369, 1218, 412, 728, 580, 374, 607, 1247, 698, 1449, 1535, 580, 1059, 680, 1657, 815, 1338, 986, 1304, 346, 1514, 460, 148, 1517, 214, 763, 814, 773, 620, 1301, 934, 87, 1060, 591, 1579, 694, 671, 340, 268, 1319, 292, 811, 1262, 839, 494, 921, 924, 0 };

jagged\_array[73] = new int[] { 0, 712, 464, 1290, 424, 998, 1143, 1258, 737, 1591, 1614, 1444, 384, 192, 1331, 1360, 1515, 1794, 1174, 1066, 1474, 293, 1826, 445, 654, 543, 1363, 496, 902, 753, 192, 690, 1330, 781, 1594, 1618, 638, 1280, 825, 1803, 960, 1483, 1069, 1387, 539, 1597, 574, 198, 1567, 273, 909, 921, 947, 793, 1446, 1128, 99, 1253, 784, 1725, 888, 845, 527, 357, 1402, 351, 1004, 1407, 980, 668, 1004, 1069, 183, 0 };

jagged\_array[74] = new int[] { 0, 784, 1016, 518, 1171, 436, 286, 805, 1026, 858, 589, 370, 1072, 1263, 177, 684, 427, 706, 278, 374, 740, 1140, 650, 972, 806, 991, 446, 932, 669, 828, 1595, 980, 646, 777, 418, 762, 1193, 181, 600, 627, 467, 308, 551, 476, 894, 725, 856, 1235, 866, 1182, 559, 642, 624, 877, 270, 478, 1327, 361, 656, 549, 547, 799, 929, 1070, 630, 1398, 498, 89, 516, 903, 662, 358, 1240, 1427, 0 };

jagged\_array[75] = new int[] { 0, 1037, 751, 1361, 305, 781, 1097, 1465, 117, 1675, 1564, 1345, 407, 484, 1130, 1443, 1402, 1681, 990, 829, 1557, 545, 1625, 546, 412, 230, 1339, 881, 466, 432, 641, 1015, 1413, 1107, 1393, 1689, 89, 955, 857, 1602, 981, 1282, 1153, 1418, 640, 1668, 862, 639, 1684, 403, 930, 981, 511, 257, 1245, 657, 580, 820, 659, 1524, 715, 336, 460, 721, 1473, 444, 882, 1168, 1006, 346, 1154, 1026, 599, 676, 1135, 0 };

jagged\_array[76] = new int[] { 0, 1077, 746, 1464, 144, 884, 1199, 1568, 336, 1777, 1723, 1505, 495, 336, 1290, 1546, 1562, 1841, 1130, 975, 1660, 540, 1785, 635, 514, 326, 1442, 861, 682, 534, 418, 1055, 1516, 1146, 1553, 1792, 135, 1134, 960, 1761, 1084, 1442, 1255, 1521, 729, 1771, 908, 616, 1786, 384, 1032, 1083, 727, 477, 1405, 874, 432, 1037, 762, 1683, 818, 556, 562, 716, 1576, 222, 985, 1349, 1108, 448, 1256, 1129, 469, 527, 1315, 222, 0 };

jagged\_array[77] = new int[] { 0, 905, 1137, 346, 1363, 629, 407, 610, 1238, 507, 218, 124, 1292, 1483, 217, 489, 69, 335, 450, 567, 508, 1304, 410, 1155, 998, 1184, 219, 1053, 881, 1040, 1738, 1101, 491, 898, 178, 391, 1386, 464, 721, 387, 588, 65, 572, 249, 1058, 354, 978, 1379, 605, 1402, 681, 763, 837, 1090, 103, 690, 1491, 645, 847, 309, 739, 1011, 1121, 1192, 387, 1591, 620, 284, 637, 1080, 685, 479, 1403, 1549, 373, 1347, 1507, 0 };

jagged\_array[78] = new int[] { 0, 714, 946, 449, 1106, 371, 216, 735, 958, 788, 567, 349, 1007, 1198, 134, 614, 405, 685, 192, 309, 670, 1075, 629, 907, 741, 926, 427, 862, 601, 760, 1530, 910, 576, 707, 396, 740, 1128, 113, 530, 605, 397, 286, 481, 454, 829, 704, 787, 1169, 797, 1117, 490, 572, 557, 810, 249, 410, 1262, 294, 591, 527, 482, 731, 864, 1001, 560, 1333, 429, 101, 446, 823, 593, 288, 1174, 1358, 86, 1067, 1250, 351, 0 };

jagged\_array[79] = new int[] { 0, 251, 210, 830, 812, 640, 711, 798, 885, 1131, 1154, 1011, 517, 580, 899, 900, 1083, 1362, 742, 677, 1014, 375, 1394, 402, 665, 691, 931, 63, 757, 827, 740, 146, 870, 321, 1162, 1158, 896, 847, 393, 1370, 528, 1051, 609, 926, 305, 1137, 141, 381, 1107, 625, 476, 461, 744, 954, 1014, 769, 554, 895, 465, 1292, 529, 876, 538, 193, 942, 739, 592, 975, 520, 793, 544, 637, 467, 550, 995, 920, 915, 1116, 925, 0 };

jagged\_array[80] = new int[] { 0, 89, 244, 667, 880, 603, 581, 636, 909, 969, 991, 874, 539, 648, 777, 737, 945, 1185, 676, 640, 851, 444, 1272, 401, 677, 715, 779, 123, 720, 790, 809, 127, 707, 159, 1040, 995, 919, 782, 356, 1248, 465, 929, 446, 764, 304, 974, 105, 449, 944, 648, 376, 299, 707, 917, 892, 732, 623, 858, 428, 1170, 492, 839, 537, 262, 779, 807, 555, 853, 357, 801, 381, 571, 535, 619, 873, 944, 984, 994, 803, 159, 0 };

jagged\_array[81] = new int[] { 0, 735, 966, 371, 1192, 458, 237, 636, 1067, 688, 387, 168, 1122, 1313, 47, 514, 225, 504, 279, 396, 571, 1133, 448, 984, 827, 1013, 244, 882, 710, 869, 1567, 930, 516, 727, 216, 560, 1215, 293, 550, 425, 417, 106, 501, 274, 887, 523, 807, 1208, 697, 1231, 510, 592, 666, 919, 68, 519, 1320, 474, 676, 347, 569, 840, 950, 1021, 412, 1420, 449, 113, 466, 910, 613, 308, 1232, 1378, 202, 1177, 1336, 171, 180, 945, 823, 0 };

string[] ilIsimleri = new string[]

{

"İller", "Adana", "Adiyaman", "Afyonkarahisar", "Ağri", "Amasya", "Ankara", "Antalya", "Artvin", "Aydin",

"Balikesir", "Bilecik", "Bingöl", "Bitlis", "Bolu", "Burdur", "Bursa", "Çanakkale", "Çankiri", "Çorum",

"Denizli", "Diyarbakir", "Edirne", "Elaziğ", "Erzincan", "Erzurum", "Eskişehir", "Gaziantep", "Giresun",

"Gümüşhane", "Hakkari", "Hatay", "Isparta", "Mersin", "İstanbul", "İzmir", "Kars", "Kastamonu", "Kayseri",

"Kirklareli", "Kirşehir", "Kocaeli", "Konya", "Kütahya", "Malatya", "Manisa", "Kahramanmaraş", "Mardin",

"Muğla", "Muş", "Nevşehir", "Niğde", "Ordu", "Rize", "Sakarya", "Samsun", "Siirt", "Sinop", "Sivas",

"Tekirdağ", "Tokat", "Trabzon", "Tunceli", "Şanliurfa", "Uşak", "Van", "Yozgat", "Zonguldak", "Aksaray",

"Bayburt", "Karaman", "Kirikkale", "Batman", "Şirnak", "Bartin", "Ardahan", "Iğdir", "Yalova", "Karabük",

"Kilis", "Osmaniye", "Düzce"

};

### 1.a.2 Ekran görüntüleri

Dosya okutma yapmadığım için ekran görüntüsü ve kod aynı

## 1.b Verilen İlden Belli Bir Uzaklığa Kadar Olan İllerin ve Uzaklıklarının Listelenmesi

### 1.b.1 Kodlar

static List<IlBilgisi> IlveUzakliklariBul(string[] ilIsimleri, int[][] uzakliklar, int ilNo, int mesafe)

{

List<IlBilgisi> sonuclar = new List<IlBilgisi>();

for (int i = 0; i < uzakliklar.Length; i++)

{

if (uzakliklar[ilNo][i] <= mesafe && uzakliklar[ilNo][i] != 0)

{

IlBilgisi il = new IlBilgisi

{

Isim = ilIsimleri[i],

Uzaklik = uzakliklar[ilNo][i]

};

sonuclar.Add(il);

}

}

return sonuclar;

}

### 1.b.2 Ekran görüntüleri

Bir il numarasi girin (1-81): 11

Bir mesafe girin: 200

------------------------------------------------------------------------

b.Verilen ilden belli bir uzaklığa kadar olan illerin ve uzaklıklarının listelenmesi:

------------------------------------------------------------------------

Belirtilen mesafeye kadar olan iller:

Bursa - 95 km

Eskişehir - 80 km

Kocaeli - 138 km

Kütahya - 110 km

Sakarya - 100 km

Yalova - 124 km

Düzce - 168 km

Bir il numarasi girin (1-81): 43

Bir mesafe girin: 200

------------------------------------------------------------------------

b.Verilen ilden belli bir uzaklığa kadar olan illerin ve uzaklıklarının listelenmesi:

------------------------------------------------------------------------

Belirtilen mesafeye kadar olan iller:

Afyonkarahisar - 97 km

Bilecik - 110 km

Bursa - 182 km

Eskişehir - 79 km

Uşak - 138 km

Bir il numarasi girin (1-81): 26

Bir mesafe girin: 400

------------------------------------------------------------------------

b.Verilen ilden belli bir uzaklığa kadar olan illerin ve uzaklıklarının listelenmesi:

------------------------------------------------------------------------

Belirtilen mesafeye kadar olan iller:

Afyonkarahisar - 141 km

Ankara - 233 km

Balikesir - 303 km

Bilecik - 80 km

Bolu - 290 km

Burdur - 300 km

Bursa - 152 km

Çankiri - 376 km

Denizli - 356 km

Isparta - 301 km

İstanbul - 323 km

Kocaeli - 213 km

Konya - 337 km

Kütahya - 79 km

Manisa - 392 km

Sakarya - 176 km

Uşak - 217 km

Zonguldak - 356 km

Kirikkale - 316 km

Yalova - 219 km

Düzce - 244 km

## 1.c Türkiye’deki Birbirine En Yakın İki İlin ve En Uzak İki İlin Bulunması

### 1.c.1 Kodlar

static int UzaklikHesapla(int[][] uzakliklar, int il1, int il2)

{

return uzakliklar[il1][il2];

}

// Türkiye'deki en yakin iki il ve en uzak iki ilin bulunmasi

static void EnYakinEnUzakIlleriBul(string[] ilIsimleri, int[][] uzakliklar)

{

int enYakinUzaklik = int.MaxValue;

int enUzakUzaklik = 0;

string enYakinIl1 = "";

string enYakinIl2 = "";

string enUzakIl1 = "";

string enUzakIl2 = "";

for (int il1 = 1; il1 < ilIsimleri.Length; il1++)

{

for (int il2 = il1 + 1; il2 < ilIsimleri.Length; il2++)

{

int uzaklik = UzaklikHesapla(uzakliklar, il1, il2);

if (uzaklik < enYakinUzaklik)

{

enYakinUzaklik = uzaklik;

enYakinIl1 = ilIsimleri[il1];

enYakinIl2 = ilIsimleri[il2];

}

if (uzaklik > enUzakUzaklik)

{

enUzakUzaklik = uzaklik;

enUzakIl1 = ilIsimleri[il1];

enUzakIl2 = ilIsimleri[il2];

}

}

}

//c.Türkiye’deki birbirine en yakın iki ilin ve en uzak iki ilin bulunması:

Console.WriteLine("\n------------------------------------------------------------------------");

Console.WriteLine("\nc.Türkiye’deki birbirine en yakın iki ilin ve en uzak iki ilin bulunması:");

Console.WriteLine("\n------------------------------------------------------------------------");

Console.WriteLine("En yakin iki il: " + enYakinIl1 + " - " + enYakinIl2 + " (" + enYakinUzaklik + " km)");

Console.WriteLine("En uzak iki il: " + enUzakIl1 + " - " + enUzakIl2 + " (" + enUzakUzaklik + " km)");

}

### 1.c.2 Ekran görüntüleri

c.Türkiye'deki birbirine en yakın iki ilin ve en uzak iki ilin bulunması:

------------------------------------------------------------------------

En yakin iki il: Burdur - Isparta (30 km)

En uzak iki il: Edirne - Hakkari (2016 km)

## 1.d Verilen İlden Verilen Mesafe Kullanılarak En fazla Kaç İl Dolaşılabildiğinin Bulunması

### 1.d.1 Kodlar

static void EnFazlaIlDolasim(string[] ilIsimleri, int[][] uzakliklar, int baslangicIlNo, int maksMesafe, List<int> enFazlaIlDolananYol, int toplamUzaklik)

{

enFazlaIlDolananYol.Add(baslangicIlNo);

while (true)

{

int enKucukMesafe = int.MaxValue;

int enKucukMesafeIl = -1;

for (int hedefIlNo = 0; hedefIlNo < ilIsimleri.Length; hedefIlNo++)

{

if (!enFazlaIlDolananYol.Contains(hedefIlNo) && uzakliklar[baslangicIlNo][hedefIlNo] != 0 && uzakliklar[baslangicIlNo][hedefIlNo] < enKucukMesafe)

{

enKucukMesafe = uzakliklar[baslangicIlNo][hedefIlNo];

enKucukMesafeIl = hedefIlNo;

}

}

if (enKucukMesafeIl == -1 || toplamUzaklik + enKucukMesafe > maksMesafe)

{

break; // En kısa mesafe ili bulunamadı veya maksimum mesafe aşıldı

}

toplamUzaklik += enKucukMesafe;

enFazlaIlDolananYol.Add(enKucukMesafeIl);

baslangicIlNo = enKucukMesafeIl;

}

Console.WriteLine("En fazla il dolaşılan mesafe: " + toplamUzaklik);

}

### 1.d.2 Algoritma ve Açıklama

ilIsimleri (string[]): İllerin isimlerini içeren dizi.

uzakliklar (int[][]): İller arasındaki uzaklıkları temsil eden bir matris. uzakliklar[i][j], i'inci il ile j'inci il arasındaki uzaklığı içerir.

baslangicIlNo (int): Başlangıç ilinin indisini belirten bir sayı.

maksMesafe (int): En fazla gidilecek mesafeyi belirten bir sınır.

enFazlaIlDolananYol (List<int>): En fazla il dolaşan yolun indislerini içeren bir liste.

toplamUzaklik (int): Şu ana kadar gidilen toplam mesafeyi tutan bir sayı.

Fonksiyon, başlangıç ilini temsil eden baslangicIlNo ile başlar. Bu il, enFazlaIlDolananYol listesine eklenir.

Sonsuz bir döngü başlar ve her iterasyonda en kısa mesafedeki il seçilir.

Seçilen il, daha önce ziyaret edilmediyse ve başlangıç ilinden bu il arasında bir uzaklık varsa, ve bu uzaklık daha önce bulunan en küçük mesafeden küçükse, bu il yeni en küçük mesafe olarak kaydedilir.

Eğer en küçük mesafe hala int.MaxValue değerindeyse, bu durumda en kısa mesafe ili bulunamadığı anlamına gelir ve döngüden çıkılır.

Aksi takdirde, toplam uzaklık güncellenir, seçilen il enFazlaIlDolananYol listesine eklenir ve başlangıç ilini güncelleyerek bu işlem tekrarlanır.

Sonuç olarak, en fazla il dolaşılan mesafe ve bu rotayı temsil eden indisler Console.WriteLine ile ekrana yazdırılır.

### 1.d.3 Ekran görüntüleri

Bir il numarasi girin (1-81): 11

Bir mesafe girin: 200

En fazla il dolaşılan mesafe: 159

En fazla il dolaşılan yol:

Bilecik -> Eskişehir -> Kütahya

Bir il numarasi girin (1-81): 35

Bir mesafe girin: 590

En fazla il dolaşılan mesafe: 566

En fazla il dolaşılan yol:

İzmir -> Manisa -> Balikesir -> Bursa -> Yalova -> Kocaeli -> Sakarya -> Düzce

Bir il numarasi girin (1-81): 16

Bir mesafe girin: 300

En fazla il dolaşılan mesafe: 286

En fazla il dolaşılan yol:

Bursa -> Yalova -> Kocaeli -> Sakarya -> Düzce -> Bolu

## 1.e Matris Şeklinde İllerin Adlarıyla Birlikte Ekrana Listelenmesi

### 1.e.1 Kodlar

static void RastgeleUzakliklariListele(string[] ilIsimleri, int[][] uzakliklar)

{

Random random = new Random();

Console.WriteLine("\n------------------------------------------------------------------------");

Console.WriteLine("\ne) Rastgele (random) 5 farklı sayı attırarak bu plaka numaralı iller arasındaki uzaklıkları matris şeklinde illerin adlarıyla birlikte ekrana listeleyen metot:");

Console.WriteLine("\n------------------------------------------------------------------------");

int[] rastgeleIller = new int[5];

// 5 farklı il seç

for (int i = 0; i < 5; i++)

{

int randomIl;

do

{

randomIl = random.Next(1, ilIsimleri.Length);

} while (Array.IndexOf(rastgeleIller, randomIl) != -1);

rastgeleIller[i] = randomIl;

}

// Seçilen her iki il için uzaklığı listeleyin

for (int i = 0; i < 5; i++)

{

for (int j = i + 1; j < 5; j++)

{

Console.WriteLine($"{ilIsimleri[rastgeleIller[i]]} - {ilIsimleri[rastgeleIller[j]]}: {uzakliklar[rastgeleIller[i]][rastgeleIller[j]]} km");

}

}

}

### 1.e.2 Ekran görüntüleri

# Bartin - Şirnak: 1427 km

# Bartin - Kirikkale: 358 km

# Bartin - Sinop: 361 km

# Bartin - Erzurum: 991 km

# Şirnak - Kirikkale: 1069 km

# Şirnak - Sinop: 1253 km

# Şirnak - Erzurum: 543 km

# Kirikkale - Sinop: 382 km

# Kirikkale - Erzurum: 805 km

# Sinop - Erzurum: 713 km

Kilis - Van: 739 km

Kilis - Muş: 625 km

Kilis - Diyarbakir: 375 km

Kilis - Batman: 467 km

Van - Muş: 216 km

Van - Diyarbakir: 363 km

Van - Batman: 292 km

Muş - Diyarbakir: 249 km

Muş - Batman: 214 km

Diyarbakir - Batman: 99 km

Eskişehir - Samsun: 654 km

Eskişehir - Bayburt: 1057 km

Eskişehir - Düzce: 244 km

Eskişehir - Kirklareli: 532 km

Samsun - Bayburt: 425 km

Samsun - Düzce: 519 km

Samsun - Kirklareli: 944 km

Bayburt - Düzce: 910 km

Bayburt - Kirklareli: 1335 km

Düzce - Kirklareli: 425 km

# 2) DEVELOPING A PERCEPTRON MODEL and IMPLEMENTATION of a REGRESSION EXAMPLE

//İnteliJ IDEA , 19.0.1 , Java

## 2.a Neuron (Sinir Hücresi) Sınıfı

### 2.a.1 Kaynak Kod

***import*** java.util.Random;  
  
***public class*** Neuron {  
  
 ***private double*** learningRate;  
 ***private double***[] weights;  
 ***public double*** [] [] testData;  
  
 *// Getters and setters.* ***public double***[] getWeights() {  
 ***return*** weights;  
 }  
  
 ***public void*** setWeights(***double***[] weights) {  
 ***this***.weights = weights;  
 }  
  
 ***public double*** getLearningRate() {  
 ***return*** learningRate;  
 }  
  
 ***public void*** setLearningRate(***double*** learningRate) {  
 ***this***.learningRate = learningRate;  
 }  
  
 ***public double***[][] getTestData() {  
 ***return*** testData;  
 }  
  
 ***public void*** setTestData(***double***[][] testData) {  
 ***this***.testData = testData;  
 }  
  
 *// Values of weights are initialized randomly not manuel.  
 // Therefore, it is not included as a parameter in the constructor.* ***public*** Neuron(***double*** learningRate, ***double***[][] testData) {  
 ***this***.testData = testData;  
 ***this***.learningRate = learningRate;  
 weights = ***new double***[2];  
 Random rand = ***new*** Random();  
  
 *// initialize the weights randomly* ***for*** (***int*** i = 0; i < 2; i++) {  
 weights[i] = rand.nextDouble();  
 }  
 }  
  
 *// Calculate output by using weights and inputs list.* ***private double*** calculateOutput(***double***[] inputs) {  
  
 ***double*** sum = 0;  
 ***for*** (***int*** i = 0; i < inputs.length; i++) {  
 sum += inputs[i] \* weights[i];  
 }  
  
 ***return*** sum;  
 }  
  
 *// Train process.* ***private void*** train(***double***[] inputs, ***double*** target) {  
  
 ***double*** output = calculateOutput(inputs);  
 ***double*** result = target - output;  
 ***for*** (***int*** i = 0; i < weights.length; i++) {  
 ***double*** weightUpdate = learningRate \* result \* inputs[i];  
 weights[i] += weightUpdate;  
 }  
 }  
  
 *// Trains the neuron depending on the number of epochs.* ***public void*** epoch(***int*** numEpochs){  
 ***for*** (***int*** epoch = 0; epoch < numEpochs; epoch++) {  
 ***for*** (***double***[] data : testData) {  
 ***double*** inputs[] = {(***double***) data[0] / 10, (***double***) data[1] / 15 };  
 ***double*** target = (***double***) data[2] / 100;  
 train(inputs, target);  
 }  
 }  
 System.out.println();  
  
 }  
 *// Calculate mse value.* ***public double*** calculateMse(){  
 ***double*** mse = 0;  
 ***for*** (***double***[] data : testData) {  
 ***double*** inputs[] = {(***double***) data[0] / 10, (***double***) data[1] / 15 };  
 ***double*** target = (***double***) data[2] / 100;  
 ***double*** prediction = calculateOutput(inputs);  
 mse += Math.pow(target - prediction, 2);  
 }  
 mse /= testData.length;  
  
 ***return*** mse;  
 }  
  
 ***public void*** showOutputs(){  
 System.out.println(" Input Values - Target Value - Predicted Value");  
 ***for*** (***double***[] data : testData) {  
 ***double*** inputs[] = {(***double***) data[0] / 10, (***double***) data[1] / 15 };  
 ***double*** target = (***double***) data[2] / 100;  
 ***double*** prediction = calculateOutput(inputs);  
 System.out.printf(" %4.2f ", inputs[0] );  
 System.out.printf(" %4.2f ", inputs[1]);  
 System.out.printf(" %4.2f ", target);  
 System.out.printf(" %4.2f ", prediction);  
 System.out.println();  
 }  
 }  
  
  
}  
 */// The end of the neuron class.*

***public class*** Main {  
 ***public static void*** main(String[] args) {  
  
 ***double***[][] testData = {  
 {7.6, 11, 77},  
 {8, 10, 70},  
 {6.6, 8, 55},  
 {8.4, 10, 78},  
 {8.8, 12, 95},  
 {7.2, 10, 67},  
 {8.1, 11, 80},  
 {9.5, 9, 87},  
 {7.3, 9, 60},  
 {8.9, 11, 88},  
 {7.5, 11, 72},  
 {7.6, 9, 58},  
 {7.9, 10, 70},  
 {8, 10, 76},  
 {7.2, 9, 58},  
 {8.8, 10, 81},  
 {7.6, 11, 74},  
 {7.5, 10, 67},  
 {9, 10, 82},  
 {7.7, 9, 62},  
 {8.1, 11, 82}  
  
 };  
 *// New input value* ***double*** testData2[][] = {  
 {9.2, 9.3, 83},  
 {7.4, 9.1, 66},  
 {8.1, 12, 88},  
 {7.6, 8, 62},  
 {7.3, 9, 57},  
 {8.1, 10, 71}  
 };  
  
 Neuron n1 = ***new*** Neuron(0.05,testData);  
 n1.epoch(10);  
 System.out.println("Mse:" + n1.calculateMse());;  
 n1.showOutputs();  
  
 *// The predictions are calculated with updated weights of first neuron .* System.out.println(" New inputs: ");  
 n1.setTestData(testData2);  
 n1.showOutputs();  
  
 System.out.println();  
 ***int***[] epoklar = {10, 50, 100};  
 ***double***[] lambdalar = {0.01, 0.025, 0.05};  
 ***double***[][] mseSonuclari = ***new double***[epoklar.length][lambdalar.length];  
  
 *// Code that writes the 3\*3 table using the lists above.* ***for***(***int*** count = 0; count < 3; count++ ){  
 ***for*** (***int*** i = 0; i < epoklar.length; i++) {  
 ***for*** (***int*** j = 0; j < lambdalar.length; j++) {  
 Neuron n2 = ***new*** Neuron(lambdalar[j],testData);  
 ***int*** epok = epoklar[i];  
 ***double*** lambda = lambdalar[j];  
 n2.epoch(epoklar[i]);  
 mseSonuclari[i][j] = n2.calculateMse();  
 }  
 }  
  
 ***for*** (***int*** i = 0; i < epoklar.length; i++) {  
 ***for*** (***int*** j = 0; j < lambdalar.length; j++) {  
 System.out.println("Epok: " + epoklar[i] + ", λ: "  
 + lambdalar[j] + " - MSE: " + mseSonuclari[i][j]);  
 }  
 }  
 } *// end of the for loop.* }  
}

### 

### 2.a.2 Açıklama

## Kullanılan veri yapıları

Tek boyutlu array

İki boyutlu array

## 2.b Eğitim

### 2.b.1 Kaynak Kod

***private double*** calculateOutput(***double***[] inputs) {  
  
 ***double*** sum = 0;  
 ***for*** (***int*** i = 0; i < inputs.length; i++) {  
 sum += inputs[i] \* weights[i];  
 }  
  
 ***return*** sum;  
}  
  
*// Train process.****private void*** train(***double***[] inputs, ***double*** target) {  
  
 ***double*** output = calculateOutput(inputs);  
 ***double*** result = target - output;  
 ***for*** (***int*** i = 0; i < weights.length; i++) {  
 ***double*** weightUpdate = learningRate \* result \* inputs[i];  
 weights[i] += weightUpdate;  
 }  
}  
  
*// Trains the neuron depending on the number of epochs.****public void*** epoch(***int*** numEpochs){  
 ***for*** (***int*** epoch = 0; epoch < numEpochs; epoch++) {  
 ***for*** (***double***[] data : testData) {  
 ***double*** inputs[] = {(***double***) data[0] / 10, (***double***) data[1] / 15 };  
 ***double*** target = (***double***) data[2] / 100;  
 train(inputs, target);  
 }  
 }  
 System.out.println();  
  
}

### 2.b.2 Açıklama

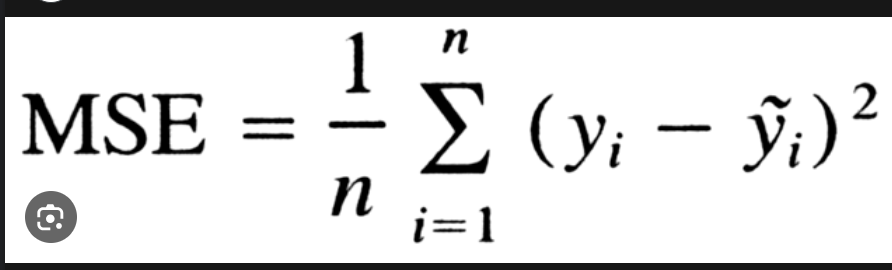
//Eğitimde kullanılan verilerden modelinizin hatasını 10 devir için MSE (Mean Square Error) cinsinden hesaplayıp yazınız. MSE’nin ne amaçla kullanıldığını ve formülünü araştırıp, kaynağıyla birlikte kısaca anlatımını gerçekleştiriniz.

Mean Squared Error (MSE), regresyon modellerinin performansını değerlendirmek amacıyla kullanılan bir ölçüdür. Regresyon modelleri, bağımlı ve bağımsız değişkenler arasındaki ilişkiyi modellemek için kullanılır. MSE, bu modellerin gerçek değerlerle ne kadar uyumlu olduğunu ölçer ve bu uyumu nicel bir değerle ifade eder.

MSE'nin amacı, modelin tahmin ettiği değerlerin gerçek değerlere ne kadar yakın veya uzak olduğunu ölçmektir. Bu ölçüm, her bir veri noktasının tahmini ile gerçek değer arasındaki farkın karesini alarak yapılır. Ardından, bu kare hataların ortalaması alınarak genel bir performans ölçüsü elde edilir.

MSE'nin avantajları şunlardır:

1. **Kolay Hesaplanabilirlik:** MSE, tahmin hatalarının karelerinin ortalamasını alarak basit bir matematiksel formüle sahiptir, bu da hesaplamayı kolaylaştırır.
2. **Duyarlılık:** Hata karelerinin alınması, büyük hataların küçük hatalara göre daha fazla ağırlık taşımasını sağlar. Bu, modelin büyük hatalar yapması durumunda bu hataların daha belirgin bir şekilde ortaya çıkmasını sağlar.
3. **Genel Performans Değerlendirmesi:** MSE, modelin genel performansını tek bir sayı ile ifade eder. Bu sayı ne kadar küçükse, modelin tahminleri o kadar iyi olarak kabul edilir.

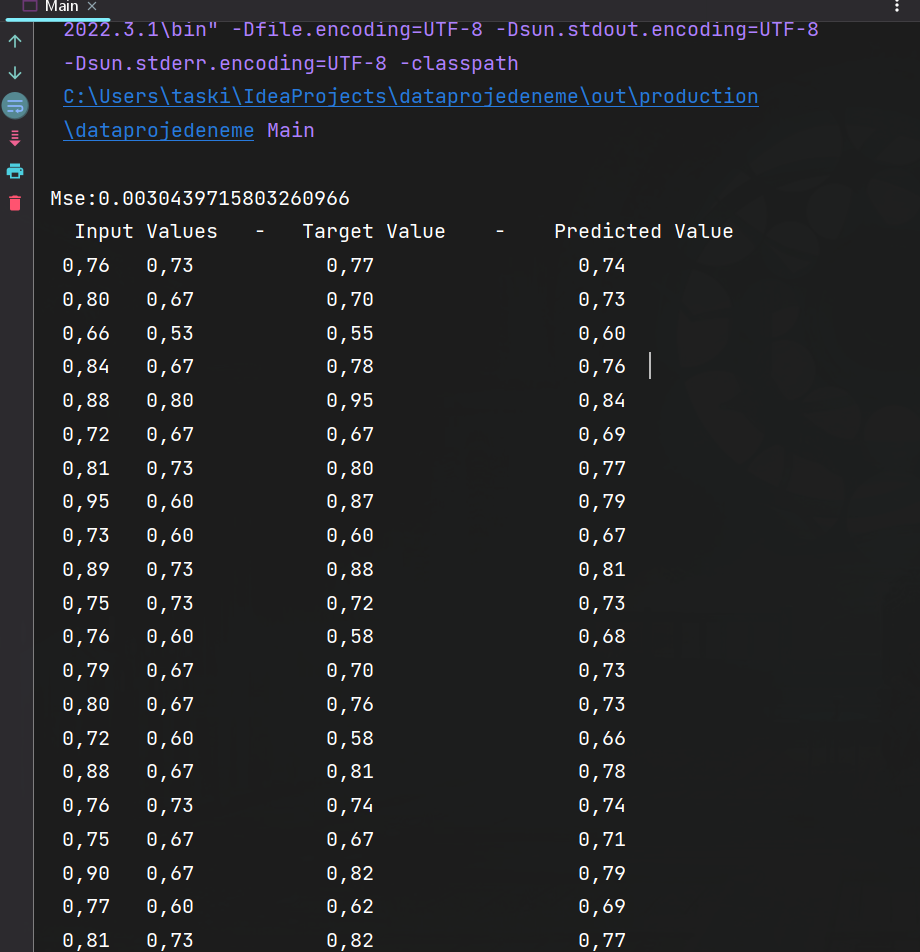


*n*, veri noktalarının sayısını temsil eder.

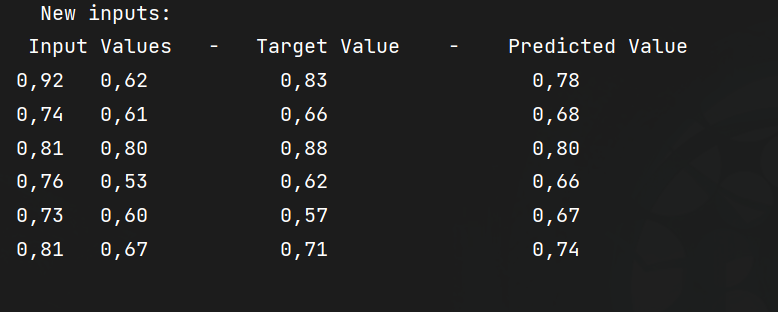
*yi*​, gerçek değerleri temsil eder.

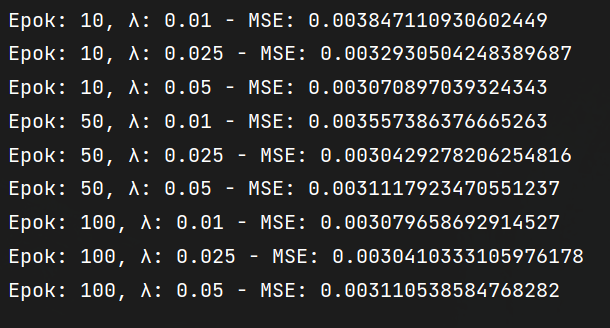
y^​*i*​, model tarafından yapılan tahminleri temsil eder.

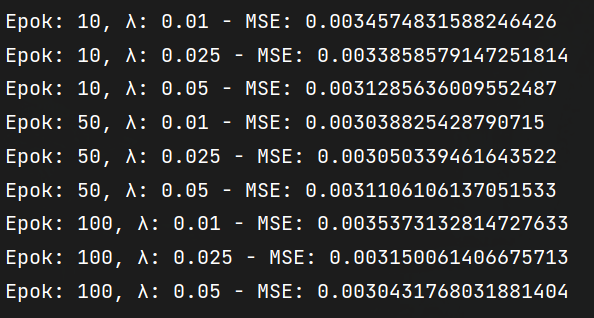
### 2.b.3 Ekran Görüntüleri



//Konsol çıktısına ait ekran görüntülerini buraya ekleyiniz







metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 2.c Modelin Görmediği Veriden Sınav Sonucu Tahminleme

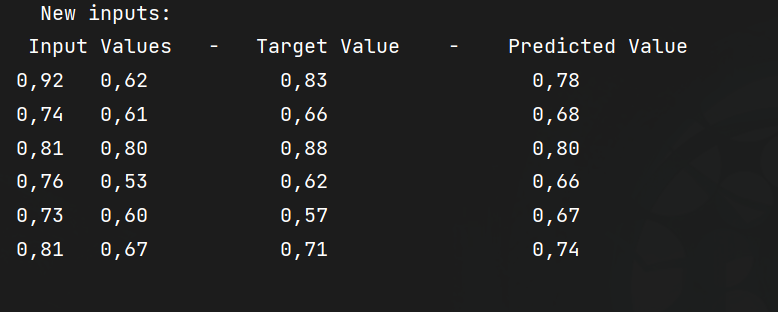
### 2.c.1 Kaynak Kod

***public void*** showOutputs(){  
 System.out.println(" Input Values - Target Value - Predicted Value");  
 ***for*** (***double***[] data : testData) {  
 ***double*** inputs[] = {(***double***) data[0] / 10, (***double***) data[1] / 15 };  
 ***double*** target = (***double***) data[2] / 100;  
 ***double*** prediction = calculateOutput(inputs);  
 System.out.printf(" %4.2f ", inputs[0] );  
 System.out.printf(" %4.2f ", inputs[1]);  
 System.out.printf(" %4.2f ", target);  
 System.out.printf(" %4.2f ", prediction);  
 System.out.println();  
 }  
}

// İlgili kaynak kodu buraya ekleyiniz.

### 2.c.2 Sonuçlar/Ekran görüntüleri

// Eğitim verileri dışında farklı 5 tane girdi verisi oluşturulması ve modelin bu veriler üzerinde yaptığı tahminleme sonuçlarının ekran görüntüsünü ekleyiniz.



## 2.d Deneyler

### 2.d.1 Kaynak Kod

// İlgili kaynak kodu buraya ekleyiniz.

### 2.d.2 Sonuçlar

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Deney 1 | 10 Epok | | 50 Epok | 100 Epok | | λ = 0.01 | 0.0038471109 | | 0..0035573 | 0.00307965 | | λ = 0.025 | | 0.00329305 | 0.003042927 | 0.0030410 | | λ = 0.05 | 0.00307089 | | 0.003111792 | 0.0031105 | | |  |  |  |  | | --- | --- | --- | --- | | Deney 2 | 10 Epok | 50 Epok | 100 Epok | | λ = 0.01 | 0.00345748 | 0.0030388 | 0.003537 | | λ = 0.025 | 0.00338585 | 0.0030503 | 0.0031500 | | λ = 0.05 | 0.0031285 | 0.0031106 | 0.0030431768031881404 | |

En aşağıya yazılmıştır.

|  |  |  |  |
| --- | --- | --- | --- |
| Deney 3 | 10 Epok | 50 Epok | 100 Epok |
| λ = 0.01 | 0.00624823 | 0.00308824 | 0.0031014 |
| λ = 0.025 | 0.0034690 | 0.0032581 | 0.003074032 |
| λ = 0.05 | 0.00383863 | 0.00333441 | 0.00304313 |

# Öz değerlendirme Tablosu

|  |  |  |  |
| --- | --- | --- | --- |
| **Proje 1 Maddeleri** | **Not** | **Tahmini Not** | **Açıklama** |
| 1.a | 10 | 10 | Yapıldı |
| 1.b | 5 | 5 | Yapıldı |
| 1.c | 5 | 5 | Yapıldı |
| 1.d | 15 | 15 | Yapıldı |
| 1.e | 5 | 5 | Yapıldı |
| 2.a | 10 | 10 | Yapıldı |
| 2.b | 15 | 15 | Yapıldı |
| 2.c | 5 | 5 | Yapıldı |
| 2.d | 10 | 10 | Yapıldı |
| Rapor | 10 | 10 | Yapıldı |
| Öz değerlendirme Tablosu | 10 | 10 | Yapıldı |
| **Toplam** | **100** | **100** |  |

**Açıklama kısmında yapıldı, yapılmadı bilgisi ve hangi maddelerin nasıl yapıldığı veya neden yapılamadığı kısaca yazılmalıdır.**

**Not: Raporu teslim edilmeyen projeler değerlendirmeye alınmayacaktır.**

MSE sonuçlarının birbirine yakın olması, modelin farklı deneme koşullarında benzer derecede başarılı olduğunu gösterebilir. MSE, regresyon modelinin tahminlerinin gerçek değerlere ne kadar yakın olduğunu ölçen bir metrik olduğu için, bu durum genellikle olumlu bir işarettir.

Eğer farklı deneme koşullarında elde edilen MSE değerleri birbirine yakınsa, bu şu anlamları taşıyabilir:

1. **Modelin Kararlılığı:** Model, farklı veri alt kümeleri veya parametre ayarları üzerinde tutarlı bir performans sergiliyor olabilir. Bu, modelin genel olarak istikrarlı bir şekilde çalıştığını gösterir.
2. **Genelleme Yeteneği:** Model, yeni ve görmediği veri noktalarına karşı genelleme yeteneğine sahip olabilir. Yani, eğitim verileri dışındaki veri noktaları üzerinde de benzer derecede iyi tahminler yapabilir.