**SAMPLE CODINGS**

import pandas as pd

df = pd.read\_csv("WA\_Fn-UseC\_-HR-Employee-Attrition.csv")

df

Age Attrition BusinessTravel DailyRate Department DistanceFromHome Education EducationField EmployeeCount EmployeeNumber ... RelationshipSatisfaction StandardHours StockOptionLevel TotalWorkingYears TrainingTimesLastYear WorkLifeBalance YearsAtCompany YearsInCurrentRole YearsSinceLastPromotion YearsWithCurrManager

0 41 Yes Travel\_Rarely 1102 Sales 1 2 Life Sciences 1 1 ... 1 80 0 8 0 1 6 4 0 5

1 49 No Travel\_Frequently 279 Research & Development 8 1 Life Sciences 1 2 ... 4 80 1 10 3 3 10 7 1 7

2 37 Yes Travel\_Rarely 1373 Research & Development 2 2 Other 1 4 ... 2 80 0 7 3 3 0 0 0 0

3 33 No Travel\_Frequently 1392 Research & Development 3 4 Life Sciences 1 5 ... 3 80 0 8 3 3 8 7 3 0

4 27 No Travel\_Rarely 591 Research & Development 2 1 Medical 1 7 ... 4 80 1 6 3 3 2 2 2 2

... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...

1465 36 No Travel\_Frequently 884 Research & Development 23 2 Medical 1 2061 ... 3 80 1 17 3 3 5 2 0 3

1466 39 No Travel\_Rarely 613 Research & Development 6 1 Medical 1 2062 ... 1 80 1 9 5 3 7 7 1 7

1467 27 No Travel\_Rarely 155 Research & Development 4 3 Life Sciences 1 2064 ... 2 80 1 6 0 3 6 2 0 3

1468 49 No Travel\_Frequently 1023 Sales 2 3 Medical 1 2065 ... 4 80 0 17 3 2 9 6 0 8

1469 34 No Travel\_Rarely 628 Research & Development 8 3 Medical 1 2068 ... 1 80 0 6 3 4 4 3 1 2

1470 rows × 35 columns

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1470 entries, 0 to 1469

Data columns (total 35 columns):

# Column Non-Null Count Dtype

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

0 Age 1470 non-null int64

1 Attrition 1470 non-null object

2 BusinessTravel 1470 non-null object

3 DailyRate 1470 non-null int64

4 Department 1470 non-null object

5 DistanceFromHome 1470 non-null int64

6 Education 1470 non-null int64

7 EducationField 1470 non-null object

8 EmployeeCount 1470 non-null int64

9 EmployeeNumber 1470 non-null int64

10 EnvironmentSatisfaction 1470 non-null int64

11 Gender 1470 non-null object

12 HourlyRate 1470 non-null int64

13 JobInvolvement 1470 non-null int64

14 JobLevel 1470 non-null int64

15 JobRole 1470 non-null object

16 JobSatisfaction 1470 non-null int64

17 MaritalStatus 1470 non-null object

18 MonthlyIncome 1470 non-null int64

19 MonthlyRate 1470 non-null int64

20 NumCompaniesWorked 1470 non-null int64

21 Over18 1470 non-null object

22 OverTime 1470 non-null object

23 PercentSalaryHike 1470 non-null int64

24 PerformanceRating 1470 non-null int64

25 RelationshipSatisfaction 1470 non-null int64

26 StandardHours 1470 non-null int64

27 StockOptionLevel 1470 non-null int64

28 TotalWorkingYears 1470 non-null int64

29 TrainingTimesLastYear 1470 non-null int64

30 WorkLifeBalance 1470 non-null int64

31 YearsAtCompany 1470 non-null int64

32 YearsInCurrentRole 1470 non-null int64

33 YearsSinceLastPromotion 1470 non-null int64

34 YearsWithCurrManager 1470 non-null int64

dtypes: int64(26), object(9)

memory usage: 402.1+ KB

df.dtypes

Age int64

Attrition object

BusinessTravel object

DailyRate int64

Department object

DistanceFromHome int64

Education int64

EducationField object

EmployeeCount int64

EmployeeNumber int64

EnvironmentSatisfaction int64

Gender object

HourlyRate int64

JobInvolvement int64

JobLevel int64

JobRole object

JobSatisfaction int64

MaritalStatus object

MonthlyIncome int64

MonthlyRate int64

NumCompaniesWorked int64

Over18 object

OverTime object

PercentSalaryHike int64

PerformanceRating int64

RelationshipSatisfaction int64

StandardHours int64

StockOptionLevel int64

TotalWorkingYears int64

TrainingTimesLastYear int64

WorkLifeBalance int64

YearsAtCompany int64

YearsInCurrentRole int64

YearsSinceLastPromotion int64

YearsWithCurrManager int64

dtype: object

df.isnull().sum()

Age 0

Attrition 0

BusinessTravel 0

DailyRate 0

Department 0

DistanceFromHome 0

Education 0

EducationField 0

EmployeeCount 0

EmployeeNumber 0

EnvironmentSatisfaction 0

Gender 0

HourlyRate 0

JobInvolvement 0

JobLevel 0

JobRole 0

JobSatisfaction 0

MaritalStatus 0

MonthlyIncome 0

MonthlyRate 0

NumCompaniesWorked 0

Over18 0

OverTime 0

PercentSalaryHike 0

PerformanceRating 0

RelationshipSatisfaction 0

StandardHours 0

StockOptionLevel 0

TotalWorkingYears 0

TrainingTimesLastYear 0

WorkLifeBalance 0

YearsAtCompany 0

YearsInCurrentRole 0

YearsSinceLastPromotion 0

YearsWithCurrManager 0

dtype: int64

df.isnull().value\_counts()

Age Attrition BusinessTravel DailyRate Department DistanceFromHome Education EducationField EmployeeCount EmployeeNumber EnvironmentSatisfaction Gender HourlyRate JobInvolvement JobLevel JobRole JobSatisfaction MaritalStatus MonthlyIncome MonthlyRate NumCompaniesWorked Over18 OverTime PercentSalaryHike PerformanceRating RelationshipSatisfaction StandardHours StockOptionLevel TotalWorkingYears TrainingTimesLastYear WorkLifeBalance YearsAtCompany YearsInCurrentRole YearsSinceLastPromotion YearsWithCurrManager

False False False False False False False False False False False False False False False False False False False False False False False False False False False False False False False False False False False 1470

Name: count, dtype: int64

df

Age Attrition BusinessTravel DailyRate Department DistanceFromHome Education EducationField EmployeeCount EmployeeNumber ... RelationshipSatisfaction StandardHours StockOptionLevel TotalWorkingYears TrainingTimesLastYear WorkLifeBalance YearsAtCompany YearsInCurrentRole YearsSinceLastPromotion YearsWithCurrManager

0 41 Yes Travel\_Rarely 1102 Sales 1 2 Life Sciences 1 1 ... 1 80 0 8 0 1 6 4 0 5

1 49 No Travel\_Frequently 279 Research & Development 8 1 Life Sciences 1 2 ... 4 80 1 10 3 3 10 7 1 7

2 37 Yes Travel\_Rarely 1373 Research & Development 2 2 Other 1 4 ... 2 80 0 7 3 3 0 0 0 0

3 33 No Travel\_Frequently 1392 Research & Development 3 4 Life Sciences 1 5 ... 3 80 0 8 3 3 8 7 3 0

4 27 No Travel\_Rarely 591 Research & Development 2 1 Medical 1 7 ... 4 80 1 6 3 3 2 2 2 2

... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...

1465 36 No Travel\_Frequently 884 Research & Development 23 2 Medical 1 2061 ... 3 80 1 17 3 3 5 2 0 3

1466 39 No Travel\_Rarely 613 Research & Development 6 1 Medical 1 2062 ... 1 80 1 9 5 3 7 7 1 7

1467 27 No Travel\_Rarely 155 Research & Development 4 3 Life Sciences 1 2064 ... 2 80 1 6 0 3 6 2 0 3

1468 49 No Travel\_Frequently 1023 Sales 2 3 Medical 1 2065 ... 4 80 0 17 3 2 9 6 0 8

1469 34 No Travel\_Rarely 628 Research & Development 8 3 Medical 1 2068 ... 1 80 0 6 3 4 4 3 1 2

1470 rows × 35 columns

df['Age'].unique()

array([41, 49, 37, 33, 27, 32, 59, 30, 38, 36, 35, 29, 31, 34, 28, 22, 53,

24, 21, 42, 44, 46, 39, 43, 50, 26, 48, 55, 45, 56, 23, 51, 40, 54,

58, 20, 25, 19, 57, 52, 47, 18, 60], dtype=int64)

import seaborn as sns

import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = (18,8)

sns.countplot(x='Age', data=df)

<Axes: xlabel='Age', ylabel='count'>

df['Attrition'].unique()

array(['Yes', 'No'], dtype=object)

plt.rcParams["figure.figsize"] = (10,5)

sns.countplot(x = 'Attrition', data =df)

<Axes: xlabel='Attrition', ylabel='count'>

df['BusinessTravel'].unique()

array(['Travel\_Rarely', 'Travel\_Frequently', 'Non-Travel'], dtype=object)

sns.countplot(x = 'BusinessTravel', data =df)

<Axes: xlabel='BusinessTravel', ylabel='count'>

df['DailyRate'].unique()

array([1102, 279, 1373, 1392, 591, 1005, 1324, 1358, 216, 1299, 809,

153, 670, 1346, 103, 1389, 334, 1123, 1219, 371, 673, 1218,

419, 391, 699, 1282, 1125, 691, 477, 705, 924, 1459, 125,

895, 813, 1273, 869, 890, 852, 1141, 464, 1240, 1357, 994,

721, 1360, 1065, 408, 1211, 1229, 626, 1434, 1488, 1097, 1443,

515, 853, 1142, 655, 1115, 427, 653, 989, 1435, 1223, 836,

1195, 1339, 664, 318, 1225, 1328, 1082, 548, 132, 746, 776,

193, 397, 945, 1214, 111, 573, 1153, 1400, 541, 432, 288,

669, 530, 632, 1334, 638, 1093, 1217, 1353, 120, 682, 489,

807, 827, 871, 665, 1040, 1420, 240, 1280, 534, 1456, 658,

142, 1127, 1031, 1189, 1354, 1467, 922, 394, 1312, 750, 441,

684, 249, 841, 147, 528, 594, 470, 957, 542, 802, 1355,

1150, 1329, 959, 1033, 1316, 364, 438, 689, 201, 1427, 857,

933, 1181, 1395, 662, 1436, 194, 967, 1496, 1169, 1145, 630,

303, 1256, 440, 1450, 1452, 465, 702, 1157, 602, 1480, 1268,

713, 134, 526, 1380, 140, 629, 1356, 328, 1084, 931, 692,

1069, 313, 894, 556, 1344, 290, 138, 926, 1261, 472, 1002,

878, 905, 1180, 121, 1136, 635, 1151, 644, 1045, 829, 1242,

1469, 896, 992, 1052, 1147, 1396, 663, 119, 979, 319, 1413,

944, 1323, 532, 818, 854, 1034, 771, 1401, 1431, 976, 1411,

1300, 252, 1327, 832, 1017, 1199, 504, 505, 916, 1247, 685,

269, 1416, 833, 307, 1311, 128, 488, 529, 1210, 1463, 675,

1385, 1403, 452, 666, 1158, 228, 996, 728, 1315, 322, 1479,

797, 1070, 442, 496, 1372, 920, 688, 1449, 1117, 636, 506,

444, 950, 889, 555, 230, 1232, 566, 1302, 812, 1476, 218,

1132, 1105, 906, 849, 390, 106, 1249, 192, 553, 117, 185,

1091, 723, 1220, 588, 1377, 1018, 1275, 798, 672, 1162, 508,

1482, 559, 210, 928, 1001, 549, 1124, 738, 570, 1130, 1192,

343, 144, 1296, 1309, 483, 810, 544, 1062, 1319, 641, 1332,

756, 845, 593, 1171, 350, 921, 1144, 143, 1046, 575, 156,

1283, 755, 304, 1178, 329, 1362, 1371, 202, 253, 164, 1107,

759, 1305, 982, 821, 1381, 480, 1473, 891, 1063, 645, 1490,

317, 422, 1485, 1368, 1448, 296, 1398, 1349, 986, 1099, 1116,

1499, 983, 1009, 1303, 1274, 1277, 587, 413, 1276, 988, 1474,

163, 267, 619, 302, 443, 828, 561, 426, 232, 1306, 1094,

509, 775, 195, 258, 471, 799, 956, 535, 1495, 446, 1245,

703, 823, 1246, 622, 1287, 448, 254, 1365, 538, 525, 558,

782, 362, 1236, 1112, 204, 1343, 604, 1216, 646, 160, 238,

1397, 306, 991, 482, 1176, 913, 1076, 727, 885, 243, 806,

817, 1410, 1207, 1442, 693, 929, 562, 608, 580, 970, 1179,

294, 314, 316, 654, 168, 381, 217, 501, 650, 141, 804,

975, 1090, 346, 430, 268, 167, 621, 527, 883, 954, 310,

719, 725, 715, 657, 1146, 182, 376, 571, 384, 791, 1111,

1243, 1092, 1325, 805, 213, 118, 676, 1252, 286, 1258, 932,

1041, 859, 720, 946, 1184, 436, 589, 760, 887, 1318, 625,

180, 586, 1012, 661, 930, 342, 1230, 1271, 1278, 607, 130,

300, 583, 1418, 1269, 379, 395, 1265, 1222, 341, 868, 1231,

102, 881, 1383, 1075, 374, 1086, 781, 177, 500, 1425, 1454,

617, 1085, 995, 1122, 618, 546, 462, 1198, 1272, 154, 1137,

1188, 188, 1333, 867, 263, 938, 129, 616, 498, 1404, 1053,

289, 1376, 231, 152, 882, 903, 1379, 335, 722, 461, 974,

1126, 840, 1134, 248, 955, 939, 1391, 1206, 287, 1441, 109,

1066, 277, 466, 1055, 265, 135, 247, 1035, 266, 145, 1038,

1234, 1109, 1089, 788, 124, 660, 1186, 1464, 796, 415, 769,

1003, 1366, 330, 1492, 1204, 309, 1330, 469, 697, 1262, 1050,

770, 406, 203, 1308, 984, 439, 793, 1451, 1182, 174, 490,

718, 433, 773, 603, 874, 367, 199, 481, 647, 1384, 902,

819, 862, 1457, 977, 942, 1402, 1421, 1361, 917, 200, 150,

179, 696, 116, 363, 107, 1465, 458, 1212, 1103, 966, 1010,

326, 1098, 969, 1167, 694, 1320, 536, 373, 599, 251, 131,

237, 1429, 648, 735, 531, 429, 968, 879, 640, 412, 848,

360, 1138, 325, 1322, 299, 1030, 634, 524, 256, 1060, 935,

495, 282, 206, 943, 523, 507, 601, 855, 1291, 1405, 1369,

999, 1202, 285, 404, 736, 1498, 1200, 1439, 499, 205, 683,

1462, 949, 652, 332, 1475, 337, 971, 1174, 667, 560, 172,

383, 1255, 359, 401, 377, 592, 1445, 1221, 866, 981, 447,

1326, 748, 990, 405, 115, 790, 830, 1193, 1423, 467, 271,

410, 1083, 516, 224, 136, 1029, 333, 1440, 674, 1342, 898,

824, 492, 598, 740, 888, 1288, 104, 1108, 479, 1351, 474,

437, 884, 1370, 264, 1059, 563, 457, 1313, 241, 1015, 336,

1387, 170, 208, 671, 711, 737, 1470, 365, 763, 567, 486,

772, 301, 311, 584, 880, 392, 148, 708, 1259, 786, 370,

678, 146, 581, 918, 1238, 585, 741, 552, 369, 717, 543,

964, 792, 611, 176, 897, 600, 1054, 428, 181, 211, 1079,

590, 305, 953, 478, 1375, 244, 511, 1294, 196, 734, 1239,

1253, 1128, 1336, 234, 766, 261, 1194, 431, 572, 1422, 1297,

574, 355, 207, 706, 280, 726, 414, 352, 1224, 459, 1254,

1131, 835, 1172, 1266, 783, 219, 1213, 1096, 1251, 1394, 605,

1064, 1337, 937, 157, 754, 1168, 155, 1444, 189, 911, 1321,

1154, 557, 642, 801, 161, 1382, 1037, 105, 582, 704, 345,

1120, 1378, 468, 613, 1023, 628], dtype=int64)

sns.distplot(df['DailyRate'])

df['DailyRate'].skew()

C:\Users\user\AppData\Local\Temp\ipykernel\_17032\667014191.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(df['DailyRate'])

-0.003518568352325854

df['Department'].unique()

array(['Sales', 'Research & Development', 'Human Resources'], dtype=object)

sns.countplot(x = 'Department', data =df)

<Axes: xlabel='Department', ylabel='count'>

sns.countplot(x='Department', hue = 'Attrition', data = df)

<Axes: xlabel='Department', ylabel='count'>

df['DistanceFromHome'].unique()

array([ 1, 8, 2, 3, 24, 23, 27, 16, 15, 26, 19, 21, 5, 11, 9, 7, 6,

10, 4, 25, 12, 18, 29, 22, 14, 20, 28, 17, 13], dtype=int64)

df['Education'].unique()

array([2, 1, 4, 3, 5], dtype=int64)

sns.countplot(x='EducationField', hue = 'Attrition', data = df)

<Axes: xlabel='EducationField', ylabel='count'>

sns.scatterplot(x='Department', y= 'Education', hue = 'Attrition', data = df)

<Axes: xlabel='Department', ylabel='Education'>

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1470 entries, 0 to 1469

Data columns (total 35 columns):

# Column Non-Null Count Dtype

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

0 Age 1470 non-null int64

1 Attrition 1470 non-null object

2 BusinessTravel 1470 non-null object

3 DailyRate 1470 non-null int64

4 Department 1470 non-null object

5 DistanceFromHome 1470 non-null int64

6 Education 1470 non-null int64

7 EducationField 1470 non-null object

8 EmployeeCount 1470 non-null int64

9 EmployeeNumber 1470 non-null int64

10 EnvironmentSatisfaction 1470 non-null int64

11 Gender 1470 non-null object

12 HourlyRate 1470 non-null int64

13 JobInvolvement 1470 non-null int64

14 JobLevel 1470 non-null int64

15 JobRole 1470 non-null object

16 JobSatisfaction 1470 non-null int64

17 MaritalStatus 1470 non-null object

18 MonthlyIncome 1470 non-null int64

19 MonthlyRate 1470 non-null int64

20 NumCompaniesWorked 1470 non-null int64

21 Over18 1470 non-null object

22 OverTime 1470 non-null object

23 PercentSalaryHike 1470 non-null int64

24 PerformanceRating 1470 non-null int64

25 RelationshipSatisfaction 1470 non-null int64

26 StandardHours 1470 non-null int64

27 StockOptionLevel 1470 non-null int64

28 TotalWorkingYears 1470 non-null int64

29 TrainingTimesLastYear 1470 non-null int64

30 WorkLifeBalance 1470 non-null int64

31 YearsAtCompany 1470 non-null int64

32 YearsInCurrentRole 1470 non-null int64

33 YearsSinceLastPromotion 1470 non-null int64

34 YearsWithCurrManager 1470 non-null int64

dtypes: int64(26), object(9)

memory usage: 402.1+ KB

df['Gender'].unique()

array(['Female', 'Male'], dtype=object)

sns.countplot(x='Gender',data=df)

<Axes: xlabel='Gender', ylabel='count'>

df['MonthlyIncome'].unique()

array([5993, 5130, 2090, ..., 9991, 5390, 4404], dtype=int64)

sns.distplot(df['MonthlyIncome'])

df['MonthlyIncome'].skew()

C:\Users\user\AppData\Local\Temp\ipykernel\_17032\2916418802.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with

similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

sns.distplot(df['MonthlyIncome'])

1.3698166808390662

sns.scatterplot(x='MonthlyIncome', y= 'Gender', hue = 'Attrition', data = df)

<Axes: xlabel='MonthlyIncome', ylabel='Gender'>

df.loc[df['BusinessTravel']=='Travel\_Rarely', 'BusinessTravel'] = 0

df.loc[df['BusinessTravel']=='Travel\_Frequently', 'BusinessTravel'] = 1

df.loc[df['BusinessTravel']=='Non-Travel', 'BusinessTravel'] = 2

df.loc[df['Department']=='Sales', 'Department'] = 0

df.loc[df['Department']=='Research & Development', 'Department'] = 1

df.loc[df['Department']=='Human Resources', 'Department'] = 2

df.loc[df['Gender']=='Female', 'Gender'] = 0

df.loc[df['Gender']=='Male', 'Gender'] = 1

df['EducationField'].unique()

array(['Life Sciences', 'Other', 'Medical', 'Marketing',

'Technical Degree', 'Human Resources'], dtype=object)

df.loc[df['EducationField']=='Life Sciences', 'EducationField'] = 0

df.loc[df['EducationField']=='Other', 'EducationField'] = 1

df.loc[df['EducationField']=='Medical', 'EducationField'] = 2

df.loc[df['EducationField']=='Marketing', 'EducationField'] = 3

df.loc[df['EducationField']=='Technical Degree', 'EducationField'] = 4

df.loc[df['EducationField']=='Human Resources', 'EducationField'] = 5

df['JobRole'].unique()

array(['Sales Executive', 'Research Scientist', 'Laboratory Technician',

'Manufacturing Director', 'Healthcare Representative', 'Manager',

'Sales Representative', 'Research Director', 'Human Resources'],

dtype=object)

df.loc[df['JobRole']=='Sales Executive', 'JobRole'] = 0

df.loc[df['JobRole']=='Research Scientist', 'JobRole'] = 1

df.loc[df['JobRole']=='Laboratory Technician', 'JobRole'] = 2

df.loc[df['JobRole']=='Manufacturing Director', 'JobRole'] = 3

df.loc[df['JobRole']=='Healthcare Representative', 'JobRole'] = 4

df.loc[df['JobRole']=='Manager', 'JobRole'] = 5

df.loc[df['JobRole']=='Sales Representative', 'JobRole'] = 6

df.loc[df['JobRole']=='Research Director', 'JobRole'] = 7

df.loc[df['JobRole']=='Human Resources', 'JobRole'] = 8

df['MaritalStatus'].unique()

array(['Single', 'Married', 'Divorced'], dtype=object)

df.loc[df['MaritalStatus']=='Single', 'MaritalStatus'] = 0

df.loc[df['MaritalStatus']=='Married', 'MaritalStatus'] = 1

df.loc[df['MaritalStatus']=='Divorced', 'MaritalStatus'] = 2

df['OverTime'].unique()

array(['Yes', 'No'], dtype=object)

df.loc[df['OverTime']=='Yes', 'OverTime'] = 0

df.loc[df['OverTime']=='No', 'OverTime'] = 1

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1470 entries, 0 to 1469

Data columns (total 35 columns):

# Column Non-Null Count Dtype

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

0 Age 1470 non-null int64

1 Attrition 1470 non-null object

2 BusinessTravel 1470 non-null object

3 DailyRate 1470 non-null int64

4 Department 1470 non-null object

5 DistanceFromHome 1470 non-null int64

6 Education 1470 non-null int64

7 EducationField 1470 non-null object

8 EmployeeCount 1470 non-null int64

9 EmployeeNumber 1470 non-null int64

10 EnvironmentSatisfaction 1470 non-null int64

11 Gender 1470 non-null object

12 HourlyRate 1470 non-null int64

13 JobInvolvement 1470 non-null int64

14 JobLevel 1470 non-null int64

15 JobRole 1470 non-null object

16 JobSatisfaction 1470 non-null int64

17 MaritalStatus 1470 non-null object

18 MonthlyIncome 1470 non-null int64

19 MonthlyRate 1470 non-null int64

20 NumCompaniesWorked 1470 non-null int64

21 Over18 1470 non-null object

22 OverTime 1470 non-null object

23 PercentSalaryHike 1470 non-null int64

24 PerformanceRating 1470 non-null int64

25 RelationshipSatisfaction 1470 non-null int64

26 StandardHours 1470 non-null int64

27 StockOptionLevel 1470 non-null int64

28 TotalWorkingYears 1470 non-null int64

29 TrainingTimesLastYear 1470 non-null int64

30 WorkLifeBalance 1470 non-null int64

31 YearsAtCompany 1470 non-null int64

32 YearsInCurrentRole 1470 non-null int64

33 YearsSinceLastPromotion 1470 non-null int64

34 YearsWithCurrManager 1470 non-null int64

dtypes: int64(26), object(9)

memory usage: 402.1+ KB

x = df[['Age','Department','DistanceFromHome','EducationField','EmployeeNumber','Gender','JobRole','JobSatisfaction','MaritalStatus','MonthlyIncome','NumCompaniesWorked','OverTime','PercentSalaryHike','PerformanceRating','StandardHours','TotalWorkingYears','WorkLifeBalance','YearsAtCompany','YearsInCurrentRole','YearsSinceLastPromotion','YearsWithCurrManager']]

x.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1470 entries, 0 to 1469

Data columns (total 21 columns):

# Column Non-Null Count Dtype

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

0 Age 1470 non-null int64

1 Department 1470 non-null object

2 DistanceFromHome 1470 non-null int64

3 EducationField 1470 non-null object

4 EmployeeNumber 1470 non-null int64

5 Gender 1470 non-null object

6 JobRole 1470 non-null object

7 JobSatisfaction 1470 non-null int64

8 MaritalStatus 1470 non-null object

9 MonthlyIncome 1470 non-null int64

10 NumCompaniesWorked 1470 non-null int64

11 OverTime 1470 non-null object

12 PercentSalaryHike 1470 non-null int64

13 PerformanceRating 1470 non-null int64

14 StandardHours 1470 non-null int64

15 TotalWorkingYears 1470 non-null int64

16 WorkLifeBalance 1470 non-null int64

17 YearsAtCompany 1470 non-null int64

18 YearsInCurrentRole 1470 non-null int64

19 YearsSinceLastPromotion 1470 non-null int64

20 YearsWithCurrManager 1470 non-null int64

dtypes: int64(15), object(6)

memory usage: 241.3+ KB

y = df['Attrition']

# pip install imblearn

from imblearn.over\_sampling import RandomOverSampler

ro=RandomOverSampler()

x\_data,y\_data=ro.fit\_resample(x,y)

from collections import Counter

print("Actual Data:",Counter(y))

Actual Data: Counter({'No': 1233, 'Yes': 237})

print("Artificial Data:",Counter(y\_data))

Artificial Data: Counter({'Yes': 1233, 'No': 1233})

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

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x\_data,y\_data,test\_size=0.2,random\_state=10)

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

from sklearn.preprocessing import RobustScaler

from sklearn.ensemble import RandomForestClassifier

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

rfc = RandomForestClassifier(n\_estimators=60, random\_state=23)

rfc.fit(x\_train,y\_train)

RandomForestClassifier?i

RandomForestClassifier(n\_estimators=60, random\_state=23)

rfc.score(x\_train,y\_train)

1.0

from sklearn.metrics import accuracy\_score

y\_pred = rfc.predict(x\_test )

accuracy\_score(y\_pred,y\_test)

0.9817813765182186

import sklearn.metrics

print(sklearn.metrics.classification\_report(y\_test, y\_pred))

precision recall f1-score support

No 1.00 0.96 0.98 254

Yes 0.96 1.00 0.98 240

accuracy 0.98 494

macro avg 0.98 0.98 0.98 494

weighted avg 0.98 0.98 0.98 494

y\_pred = rfc.predict(x\_test )

y\_true=y\_test

from sklearn.metrics import confusion\_matrix

cm=confusion\_matrix(y\_true,y\_pred)

cm

array([[245, 9],

[ 0, 240]], dtype=int64)

import seaborn as sns

import matplotlib.pyplot as plt

f, ax=plt.subplots(figsize=(5,5))

sns.heatmap(cm,annot=True,linewidths=0.5,linecolor="red",fmt=".0f",ax=ax)

plt.xlabel("y\_pred")

plt.ylabel("y\_true")

plt.show()

import pickle

pickle.dump(rfc,open('job\_random.pkl','wb'))

job\_random = pickle.load(open('job\_random.pkl','rb'))

from sklearn.ensemble import BaggingClassifier

Model=BaggingClassifier()

Model.fit(x\_train, y\_train)

BaggingClassifier?i

BaggingClassifier()

Model.score(x\_train,y\_train)

0.9984787018255578

from sklearn.metrics import accuracy\_score

y\_pred = Model.predict(x\_test )

accuracy\_score(y\_pred,y\_test)

0.951417004048583

print(sklearn.metrics.classification\_report(y\_test, y\_pred))

precision recall f1-score support

No 0.96 0.94 0.95 254

Yes 0.94 0.96 0.95 240

accuracy 0.95 494

macro avg 0.95 0.95 0.95 494

weighted avg 0.95 0.95 0.95 494

y\_pred = Model.predict(x\_test )

y\_true=y\_test

from sklearn.metrics import confusion\_matrix

cm=confusion\_matrix(y\_true,y\_pred)

cm

array([[239, 15],

[ 9, 231]], dtype=int64)

import seaborn as sns

import matplotlib.pyplot as plt

f, ax=plt.subplots(figsize=(5,5))

sns.heatmap(cm,annot=True,linewidths=0.5,linecolor="red",fmt=".0f",ax=ax)

plt.xlabel("y\_pred")

plt.ylabel("y\_true")

plt.show()

import pickle

pickle.dump(Model,open('job\_bagging.pkl','wb'))

bagging = pickle.load(open('job\_bagging.pkl','rb'))