*# This Python 3 environment comes with many helpful analytics libraries installed*

*# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python*

*# For example, here's several helpful packages to load*

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

*# Input data files are available in the read-only "../input/" directory*

*# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory*

import os

for dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

*# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"*

*# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session*

/kaggle/input/employee-attrition/WA\_Fn-UseC\_-HR-Employee-Attrition.csv

In [2]:

import matplotlib.pyplot as plt

import seaborn as sns

import statsmodels.api as sm

import warnings

warnings.filterwarnings('ignore')

In [3]:

data\_path = "/kaggle/input/employee-attrition/WA\_Fn-UseC\_-HR-Employee-Attrition.csv"

data = pd.read\_csv(data\_path)

In [4]:

data.head()

Out[4]:

|  | 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 |

5 rows × 35 columns

Data Cleaning

In [5]:

data.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

In [6]:

*#checking data qaulity*

for col **in** data.columns:

print(f"**{**col**}** number of unique values: **{**data[col].nunique()**}**, data type = **{**data[col].dtype**}**")

print("=="\*20)

print(data[col].unique())

print("=="\*30)

Age number of unique values: 43, data type = int64

========================================

[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]

============================================================

Attrition number of unique values: 2, data type = object

========================================

['Yes' 'No']

============================================================

BusinessTravel number of unique values: 3, data type = object

========================================

['Travel\_Rarely' 'Travel\_Frequently' 'Non-Travel']

============================================================

DailyRate number of unique values: 886, data type = int64

========================================

[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]

============================================================

Department number of unique values: 3, data type = object

========================================

['Sales' 'Research & Development' 'Human Resources']

============================================================

DistanceFromHome number of unique values: 29, data type = int64

========================================

[ 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]

============================================================

Education number of unique values: 5, data type = int64

========================================

[2 1 4 3 5]

============================================================

EducationField number of unique values: 6, data type = object

========================================

['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical Degree'

'Human Resources']

============================================================

EmployeeCount number of unique values: 1, data type = int64

========================================

[1]

============================================================

EmployeeNumber number of unique values: 1470, data type = int64

========================================

[ 1 2 4 ... 2064 2065 2068]

============================================================

EnvironmentSatisfaction number of unique values: 4, data type = int64

========================================

[2 3 4 1]

============================================================

Gender number of unique values: 2, data type = object

========================================

['Female' 'Male']

============================================================

HourlyRate number of unique values: 71, data type = int64

========================================

[ 94 61 92 56 40 79 81 67 44 84 49 31 93 50 51 80 96 78

45 82 53 83 58 72 48 42 41 86 97 75 33 37 73 98 36 47

71 30 43 99 59 95 57 76 87 66 55 32 52 70 62 64 63 60

100 46 39 77 35 91 54 34 90 65 88 85 89 68 69 74 38]

============================================================

JobInvolvement number of unique values: 4, data type = int64

========================================

[3 2 4 1]

============================================================

JobLevel number of unique values: 5, data type = int64

========================================

[2 1 3 4 5]

============================================================

JobRole number of unique values: 9, data type = object

========================================

['Sales Executive' 'Research Scientist' 'Laboratory Technician'

'Manufacturing Director' 'Healthcare Representative' 'Manager'

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

============================================================

JobSatisfaction number of unique values: 4, data type = int64

========================================

[4 2 3 1]

============================================================

MaritalStatus number of unique values: 3, data type = object

========================================

['Single' 'Married' 'Divorced']

============================================================

MonthlyIncome number of unique values: 1349, data type = int64

========================================

[5993 5130 2090 ... 9991 5390 4404]

============================================================

MonthlyRate number of unique values: 1427, data type = int64

========================================

[19479 24907 2396 ... 5174 13243 10228]

============================================================

NumCompaniesWorked number of unique values: 10, data type = int64

========================================

[8 1 6 9 0 4 5 2 7 3]

============================================================

Over18 number of unique values: 1, data type = object

========================================

['Y']

============================================================

OverTime number of unique values: 2, data type = object

========================================

['Yes' 'No']

============================================================

PercentSalaryHike number of unique values: 15, data type = int64

========================================

[11 23 15 12 13 20 22 21 17 14 16 18 19 24 25]

============================================================

PerformanceRating number of unique values: 2, data type = int64

========================================

[3 4]

============================================================

RelationshipSatisfaction number of unique values: 4, data type = int64

========================================

[1 4 2 3]

============================================================

StandardHours number of unique values: 1, data type = int64

========================================

[80]

============================================================

StockOptionLevel number of unique values: 4, data type = int64

========================================

[0 1 3 2]

============================================================

TotalWorkingYears number of unique values: 40, data type = int64

========================================

[ 8 10 7 6 12 1 17 5 3 31 13 0 26 24 22 9 19 2 23 14 15 4 29 28

21 25 20 11 16 37 38 30 40 18 36 34 32 33 35 27]

============================================================

TrainingTimesLastYear number of unique values: 7, data type = int64

========================================

[0 3 2 5 1 4 6]

============================================================

WorkLifeBalance number of unique values: 4, data type = int64

========================================

[1 3 2 4]

============================================================

YearsAtCompany number of unique values: 37, data type = int64

========================================

[ 6 10 0 8 2 7 1 9 5 4 25 3 12 14 22 15 27 21 17 11 13 37 16 20

40 24 33 19 36 18 29 31 32 34 26 30 23]

============================================================

YearsInCurrentRole number of unique values: 19, data type = int64

========================================

[ 4 7 0 2 5 9 8 3 6 13 1 15 14 16 11 10 12 18 17]

============================================================

YearsSinceLastPromotion number of unique values: 16, data type = int64

========================================

[ 0 1 3 2 7 4 8 6 5 15 9 13 12 10 11 14]

============================================================

YearsWithCurrManager number of unique values: 18, data type = int64

========================================

[ 5 7 0 2 6 8 3 11 17 1 4 12 9 10 15 13 16 14]

============================================================

* There are some columns with only 1 unique value. Varience is low.
* We will use varience threshold to 0 and check the columns
* Data quality looks ok. No futher cleaning is required.
* There are some numerical columns which are categorical in nature like JobLevel, RelationshipSatisfaction, PerformanceRating.
* EmployeeNumber is an identification number. We will remove the same

In [7]:

data.drop('EmployeeNumber',axis=1,inplace=True) *#droping EmployeeNumber*

In [8]:

data.describe()

Out[8]:

|  | Age | DailyRate | DistanceFromHome | Education | EmployeeCount | EnvironmentSatisfaction | HourlyRate | JobInvolvement | JobLevel | JobSatisfaction | ... | RelationshipSatisfaction | StandardHours | StockOptionLevel | TotalWorkingYears | TrainingTimesLastYear | WorkLifeBalance | YearsAtCompany | YearsInCurrentRole | YearsSinceLastPromotion | YearsWithCurrManager |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| count | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.0 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | ... | 1470.000000 | 1470.0 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 |
| mean | 36.923810 | 802.485714 | 9.192517 | 2.912925 | 1.0 | 2.721769 | 65.891156 | 2.729932 | 2.063946 | 2.728571 | ... | 2.712245 | 80.0 | 0.793878 | 11.279592 | 2.799320 | 2.761224 | 7.008163 | 4.229252 | 2.187755 | 4.123129 |
| std | 9.135373 | 403.509100 | 8.106864 | 1.024165 | 0.0 | 1.093082 | 20.329428 | 0.711561 | 1.106940 | 1.102846 | ... | 1.081209 | 0.0 | 0.852077 | 7.780782 | 1.289271 | 0.706476 | 6.126525 | 3.623137 | 3.222430 | 3.568136 |
| min | 18.000000 | 102.000000 | 1.000000 | 1.000000 | 1.0 | 1.000000 | 30.000000 | 1.000000 | 1.000000 | 1.000000 | ... | 1.000000 | 80.0 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 30.000000 | 465.000000 | 2.000000 | 2.000000 | 1.0 | 2.000000 | 48.000000 | 2.000000 | 1.000000 | 2.000000 | ... | 2.000000 | 80.0 | 0.000000 | 6.000000 | 2.000000 | 2.000000 | 3.000000 | 2.000000 | 0.000000 | 2.000000 |
| 50% | 36.000000 | 802.000000 | 7.000000 | 3.000000 | 1.0 | 3.000000 | 66.000000 | 3.000000 | 2.000000 | 3.000000 | ... | 3.000000 | 80.0 | 1.000000 | 10.000000 | 3.000000 | 3.000000 | 5.000000 | 3.000000 | 1.000000 | 3.000000 |
| 75% | 43.000000 | 1157.000000 | 14.000000 | 4.000000 | 1.0 | 4.000000 | 83.750000 | 3.000000 | 3.000000 | 4.000000 | ... | 4.000000 | 80.0 | 1.000000 | 15.000000 | 3.000000 | 3.000000 | 9.000000 | 7.000000 | 3.000000 | 7.000000 |
| max | 60.000000 | 1499.000000 | 29.000000 | 5.000000 | 1.0 | 4.000000 | 100.000000 | 4.000000 | 5.000000 | 4.000000 | ... | 4.000000 | 80.0 | 3.000000 | 40.000000 | 6.000000 | 4.000000 | 40.000000 | 18.000000 | 15.000000 | 17.000000 |

8 rows × 25 columns

In [9]:

*#dependent and independent segregation*

X = data.drop('Attrition', axis=1)

Y = data['Attrition']

In [10]:

*#encoding dependent variable*

Y = Y.map(lambda x: 1 if x == 'Yes' else 0)

X.columns = [col.replace(' ','\_') for col **in** X.columns]

*#encoding categorical columns*

cat\_col = [col for col **in** X.columns if data[col].dtype == np.object\_]

X = pd.get\_dummies(X, drop\_first= True, columns= cat\_col,dtype=int)

In [11]:

Y.value\_counts(normalize= True)

Out[11]:

Attrition

0 0.838776

1 0.161224

Name: proportion, dtype: float64

* Data is imbalance.

In [12]:

print(f"Shape of X post Encoding: **{**X.shape[1]**}**")

Shape of X post Encoding: 46

In [13]:

*#checking zero variance columns*

from sklearn.feature\_selection import VarianceThreshold

vt = VarianceThreshold(threshold = 0)

vt.fit(X)

zero\_car\_cals = X.columns[~vt.get\_support()]

print(f"Columns with zero varience: **{**zero\_car\_cals**}**")

Columns with zero varience: Index(['EmployeeCount', 'StandardHours'], dtype='object')

In [14]:

*#removing the zero vaiance columns*

X = X.drop(zero\_car\_cals,axis = 1)

In [15]:

linkcode

print(f"Shape of X post zero var column removal: **{**X.shape[1]**}**")

dict\_col ={col : X[col].nunique() for col **in** X.columns}

dict\_col = dict(sorted(dict\_col.items(), key=lambda x:x[1]))

dict\_co

*#creating number columns and categorical columns*

threshold = 5

num\_col = [col for col **in** X.columns if X[col].nunique() >threshold]

cat\_col = [col for col **in** X.columns if X[col].nunique() <=threshold]

*# This Python 3 environment comes with many helpful analytics libraries installed*

*# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python*

*# For example, here's several helpful packages to load*

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

*# Input data files are available in the read-only "../input/" directory*

*# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory*

import os

for dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

*# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"*

*# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session*

/kaggle/input/employee-attrition/WA\_Fn-UseC\_-HR-Employee-Attrition.csv

In [2]:

import matplotlib.pyplot as plt

import seaborn as sns

import statsmodels.api as sm

import warnings

warnings.filterwarnings('ignore')

In [3]:

data\_path = "/kaggle/input/employee-attrition/WA\_Fn-UseC\_-HR-Employee-Attrition.csv"

data = pd.read\_csv(data\_path)

In [4]:

data.head()

Out[4]:

|  | 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 |

5 rows × 35 columns

Data Cleaning

In [5]:

data.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

In [6]:

*#checking data qaulity*

for col **in** data.columns:

print(f"**{**col**}** number of unique values: **{**data[col].nunique()**}**, data type = **{**data[col].dtype**}**")

print("=="\*20)

print(data[col].unique())

print("=="\*30)

Age number of unique values: 43, data type = int64

========================================

[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]

============================================================

Attrition number of unique values: 2, data type = object

========================================

['Yes' 'No']

============================================================

BusinessTravel number of unique values: 3, data type = object

========================================

['Travel\_Rarely' 'Travel\_Frequently' 'Non-Travel']

============================================================

DailyRate number of unique values: 886, data type = int64

========================================

[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]

============================================================

Department number of unique values: 3, data type = object

========================================

['Sales' 'Research & Development' 'Human Resources']

============================================================

DistanceFromHome number of unique values: 29, data type = int64

========================================

[ 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]

============================================================

Education number of unique values: 5, data type = int64

========================================

[2 1 4 3 5]

============================================================

EducationField number of unique values: 6, data type = object

========================================

['Life Sciences' 'Other' 'Medical' 'Marketing' 'Technical Degree'

'Human Resources']

============================================================

EmployeeCount number of unique values: 1, data type = int64

========================================

[1]

============================================================

EmployeeNumber number of unique values: 1470, data type = int64

========================================

[ 1 2 4 ... 2064 2065 2068]

============================================================

EnvironmentSatisfaction number of unique values: 4, data type = int64

========================================

[2 3 4 1]

============================================================

Gender number of unique values: 2, data type = object

========================================

['Female' 'Male']

============================================================

HourlyRate number of unique values: 71, data type = int64

========================================

[ 94 61 92 56 40 79 81 67 44 84 49 31 93 50 51 80 96 78

45 82 53 83 58 72 48 42 41 86 97 75 33 37 73 98 36 47

71 30 43 99 59 95 57 76 87 66 55 32 52 70 62 64 63 60

100 46 39 77 35 91 54 34 90 65 88 85 89 68 69 74 38]

============================================================

JobInvolvement number of unique values: 4, data type = int64

========================================

[3 2 4 1]

============================================================

JobLevel number of unique values: 5, data type = int64

========================================

[2 1 3 4 5]

============================================================

JobRole number of unique values: 9, data type = object

========================================

['Sales Executive' 'Research Scientist' 'Laboratory Technician'

'Manufacturing Director' 'Healthcare Representative' 'Manager'

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

============================================================

JobSatisfaction number of unique values: 4, data type = int64

========================================

[4 2 3 1]

============================================================

MaritalStatus number of unique values: 3, data type = object

========================================

['Single' 'Married' 'Divorced']

============================================================

MonthlyIncome number of unique values: 1349, data type = int64

========================================

[5993 5130 2090 ... 9991 5390 4404]

============================================================

MonthlyRate number of unique values: 1427, data type = int64

========================================

[19479 24907 2396 ... 5174 13243 10228]

============================================================

NumCompaniesWorked number of unique values: 10, data type = int64

========================================

[8 1 6 9 0 4 5 2 7 3]

============================================================

Over18 number of unique values: 1, data type = object

========================================

['Y']

============================================================

OverTime number of unique values: 2, data type = object

========================================

['Yes' 'No']

============================================================

PercentSalaryHike number of unique values: 15, data type = int64

========================================

[11 23 15 12 13 20 22 21 17 14 16 18 19 24 25]

============================================================

PerformanceRating number of unique values: 2, data type = int64

========================================

[3 4]

============================================================

RelationshipSatisfaction number of unique values: 4, data type = int64

========================================

[1 4 2 3]

============================================================

StandardHours number of unique values: 1, data type = int64

========================================

[80]

============================================================

StockOptionLevel number of unique values: 4, data type = int64

========================================

[0 1 3 2]

============================================================

TotalWorkingYears number of unique values: 40, data type = int64

========================================

[ 8 10 7 6 12 1 17 5 3 31 13 0 26 24 22 9 19 2 23 14 15 4 29 28

21 25 20 11 16 37 38 30 40 18 36 34 32 33 35 27]

============================================================

TrainingTimesLastYear number of unique values: 7, data type = int64

========================================

[0 3 2 5 1 4 6]

============================================================

WorkLifeBalance number of unique values: 4, data type = int64

========================================

[1 3 2 4]

============================================================

YearsAtCompany number of unique values: 37, data type = int64

========================================

[ 6 10 0 8 2 7 1 9 5 4 25 3 12 14 22 15 27 21 17 11 13 37 16 20

40 24 33 19 36 18 29 31 32 34 26 30 23]

============================================================

YearsInCurrentRole number of unique values: 19, data type = int64

========================================

[ 4 7 0 2 5 9 8 3 6 13 1 15 14 16 11 10 12 18 17]

============================================================

YearsSinceLastPromotion number of unique values: 16, data type = int64

========================================

[ 0 1 3 2 7 4 8 6 5 15 9 13 12 10 11 14]

============================================================

YearsWithCurrManager number of unique values: 18, data type = int64

========================================

[ 5 7 0 2 6 8 3 11 17 1 4 12 9 10 15 13 16 14]

============================================================

* There are some columns with only 1 unique value. Varience is low.
* We will use varience threshold to 0 and check the columns
* Data quality looks ok. No futher cleaning is required.
* There are some numerical columns which are categorical in nature like JobLevel, RelationshipSatisfaction, PerformanceRating.
* EmployeeNumber is an identification number. We will remove the same

In [7]:

data.drop('EmployeeNumber',axis=1,inplace=True) *#droping EmployeeNumber*

In [8]:

data.describe()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| count | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.0 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | ... | 1470.000000 | 1470.0 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 |
| mean | 36.923810 | 802.485714 | 9.192517 | 2.912925 | 1.0 | 2.721769 | 65.891156 | 2.729932 | 2.063946 | 2.728571 | ... | 2.712245 | 80.0 | 0.793878 | 11.279592 | 2.799320 | 2.761224 | 7.008163 | 4.229252 | 2.187755 | 4.123129 |
| std | 9.135373 | 403.509100 | 8.106864 | 1.024165 | 0.0 | 1.093082 | 20.329428 | 0.711561 | 1.106940 | 1.102846 | ... | 1.081209 | 0.0 | 0.852077 | 7.780782 | 1.289271 | 0.706476 | 6.126525 | 3.623137 | 3.222430 | 3.568136 |
| min | 18.000000 | 102.000000 | 1.000000 | 1.000000 | 1.0 | 1.000000 | 30.000000 | 1.000000 | 1.000000 | 1.000000 | ... | 1.000000 | 80.0 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 30.000000 | 465.000000 | 2.000000 | 2.000000 | 1.0 | 2.000000 | 48.000000 | 2.000000 | 1.000000 | 2.000000 | ... | 2.000000 | 80.0 | 0.000000 | 6.000000 | 2.000000 | 2.000000 | 3.000000 | 2.000000 | 0.000000 | 2.000000 |
| 50% | 36.000000 | 802.000000 | 7.000000 | 3.000000 | 1.0 | 3.000000 | 66.000000 | 3.000000 | 2.000000 | 3.000000 | ... | 3.000000 | 80.0 | 1.000000 | 10.000000 | 3.000000 | 3.000000 | 5.000000 | 3.000000 | 1.000000 | 3.000000 |
| 75% | 43.000000 | 1157.000000 | 14.000000 | 4.000000 | 1.0 | 4.000000 | 83.750000 | 3.000000 | 3.000000 | 4.000000 | ... | 4.000000 | 80.0 | 1.000000 | 15.000000 | 3.000000 | 3.000000 | 9.000000 | 7.000000 | 3.000000 | 7.000000 |
| max | 60.000000 | 1499.000000 | 29.000000 | 5.000000 | 1.0 | 4.000000 | 100.000000 | 4.000000 | 5.000000 | 4.000000 | ... | 4.000000 | 80.0 | 3.000000 | 40.000000 | 6.000000 | 4.000000 | 40.000000 | 18.000000 | 15.000000 | 17.000000 |

8 rows × 25 columns

In [9]:

*#dependent and independent segregation*

X = data.drop('Attrition', axis=1)

Y = data['Attrition']

In [10]:

*#encoding dependent variable*

Y = Y.map(lambda x: 1 if x == 'Yes' else 0)

X.columns = [col.replace(' ','\_') for col **in** X.columns]

*#encoding categorical columns*

cat\_col = [col for col **in** X.columns if data[col].dtype == np.object\_]

X = pd.get\_dummies(X, drop\_first= True, columns= cat\_col,dtype=int)

In [11]:

Y.value\_counts(normalize= True)

Out[11]:

Attrition

0 0.838776

1 0.161224

Name: proportion, dtype: float64

* Data is imbalance.

In [12]:

print(f"Shape of X post Encoding: **{**X.shape[1]**}**")

Shape of X post Encoding: 46

In [13]:

*#checking zero variance columns*

from sklearn.feature\_selection import VarianceThreshold

vt = VarianceThreshold(threshold = 0)

vt.fit(X)

zero\_car\_cals = X.columns[~vt.get\_support()]

print(f"Columns with zero varience: **{**zero\_car\_cals**}**")

Columns with zero varience: Index(['EmployeeCount', 'StandardHours'], dtype='object')

In [14]:

*#removing the zero vaiance columns*

X = X.drop(zero\_car\_cals,axis = 1)

In [15]:

print(f"Shape of X post zero var column removal: **{**X.shape[1]**}**")

Shape of X post zero var column removal: 44

checking normalcy of the number variables

In [16]:

*#checking the for threshold value to difircate the num\_col and cat\_col*

dict\_col ={col : X[col].nunique() for col **in** X.columns}

dict\_col = dict(sorted(dict\_col.items(), key=lambda x:x[1]))

dict\_col

Out[16]:

{'PerformanceRating': 2,

'BusinessTravel\_Travel\_Frequently': 2,

'BusinessTravel\_Travel\_Rarely': 2,

'Department\_Research & Development': 2,

'Department\_Sales': 2,

'EducationField\_Life Sciences': 2,

'EducationField\_Marketing': 2,

'EducationField\_Medical': 2,

'EducationField\_Other': 2,

'EducationField\_Technical Degree': 2,

'Gender\_Male': 2,

'JobRole\_Human Resources': 2,

'JobRole\_Laboratory Technician': 2,

'JobRole\_Manager': 2,

'JobRole\_Manufacturing Director': 2,

'JobRole\_Research Director': 2,

'JobRole\_Research Scientist': 2,

'JobRole\_Sales Executive': 2,

'JobRole\_Sales Representative': 2,

'MaritalStatus\_Married': 2,

'MaritalStatus\_Single': 2,

'OverTime\_Yes': 2,

'EnvironmentSatisfaction': 4,

'JobInvolvement': 4,

'JobSatisfaction': 4,

'RelationshipSatisfaction': 4,

'StockOptionLevel': 4,

'WorkLifeBalance': 4,

'Education': 5,

'JobLevel': 5,

'TrainingTimesLastYear': 7,

'NumCompaniesWorked': 10,

'PercentSalaryHike': 15,

'YearsSinceLastPromotion': 16,

'YearsWithCurrManager': 18,

'YearsInCurrentRole': 19,

'DistanceFromHome': 29,

'YearsAtCompany': 37,

'TotalWorkingYears': 40,

'Age': 43,

'HourlyRate': 71,

'DailyRate': 886,

'MonthlyIncome': 1349,

'MonthlyRate': 1427}

In [17]:

*#creating number columns and categorical columns*

threshold = 5

num\_col = [col for col **in** X.columns if X[col].nunique() >threshold]

cat\_col = [col for col **in** X.columns if X[col].nunique() <=threshold]

In [18]:

*#kolmogorov-Smirnov Test*

*#H0 = data is normally distributed*

*#H1 = data is not normally distributed*

alpha = 0.05

not\_normal\_col = {}

normal\_col = {}

from scipy.stats import kstest

for col **in** num\_col:

result = kstest(X[col], 'norm')

pvalue = result[1]

if pvalue < alpha:

not\_normal\_col[col] = pvalue

else:

normal\_col[col] = pvalue

print("Normally distributed columns : ", normal\_col.keys())

print("Not Noemally distributed columns : ", not\_normal\_col.keys())

Normally distributed columns : dict\_keys([])

Not Noemally distributed columns : dict\_keys(['Age', 'DailyRate', 'DistanceFromHome', 'HourlyRate', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'PercentSalaryHike', 'TotalWorkingYears', 'TrainingTimesLastYear', 'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion', 'YearsWithCurrManager'])

In [19]:

linkcode

*# Q-Q plot*

from scipy.stats import probplot

plt.figure(figsize=(15,15))

for i **in** range(len(num\_col)):

plt.subplot(len(num\_col)//2+1,2,i+1)

probplot(X[num\_col[i]],dist='norm',plot= plt)

plt.title(f"**{**num\_col[i]**}** Q-Q Distribution")

plt.ylabel('Value')

plt.xlabel('quantiles')

plt.grid(True)

plt.subplots\_adjust(hspace=0.04, wspace=0.04)

plt.tight\_layout()

plt.show()

*#H1 = data is not normally distributed*

alpha = 0.05

not\_normal\_col = {}

normal\_col = {}

from scipy.stats import kstest

for col **in** num\_col:

result = shapiro(X[col])

pvalue = result[1]

if pvalue < alpha:

not\_normal\_col[col] = pvalue

else:

normal\_col[col] = pvalue

print("Normally distributed columns : ", normal\_col.keys())

print("Not Noemally distributed columns : ", not\_normal\_col.keys())

*#VIF*

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

X\_new = sm.add\_constant(X)

*# VIF dataframe*

vif\_data = pd.DataFrame()

vif\_data["feature"] = X\_new.columns

*# calculating VIF for each feature*

vif\_data["VIF"] = [variance\_inflation\_factor(X\_new.values, i)

for i **in** range(len(X\_new.columns))]

vif\_data = vif\_data.loc[1:,:] *#removing the constant column from the dataframe*

*#corelation matrix*

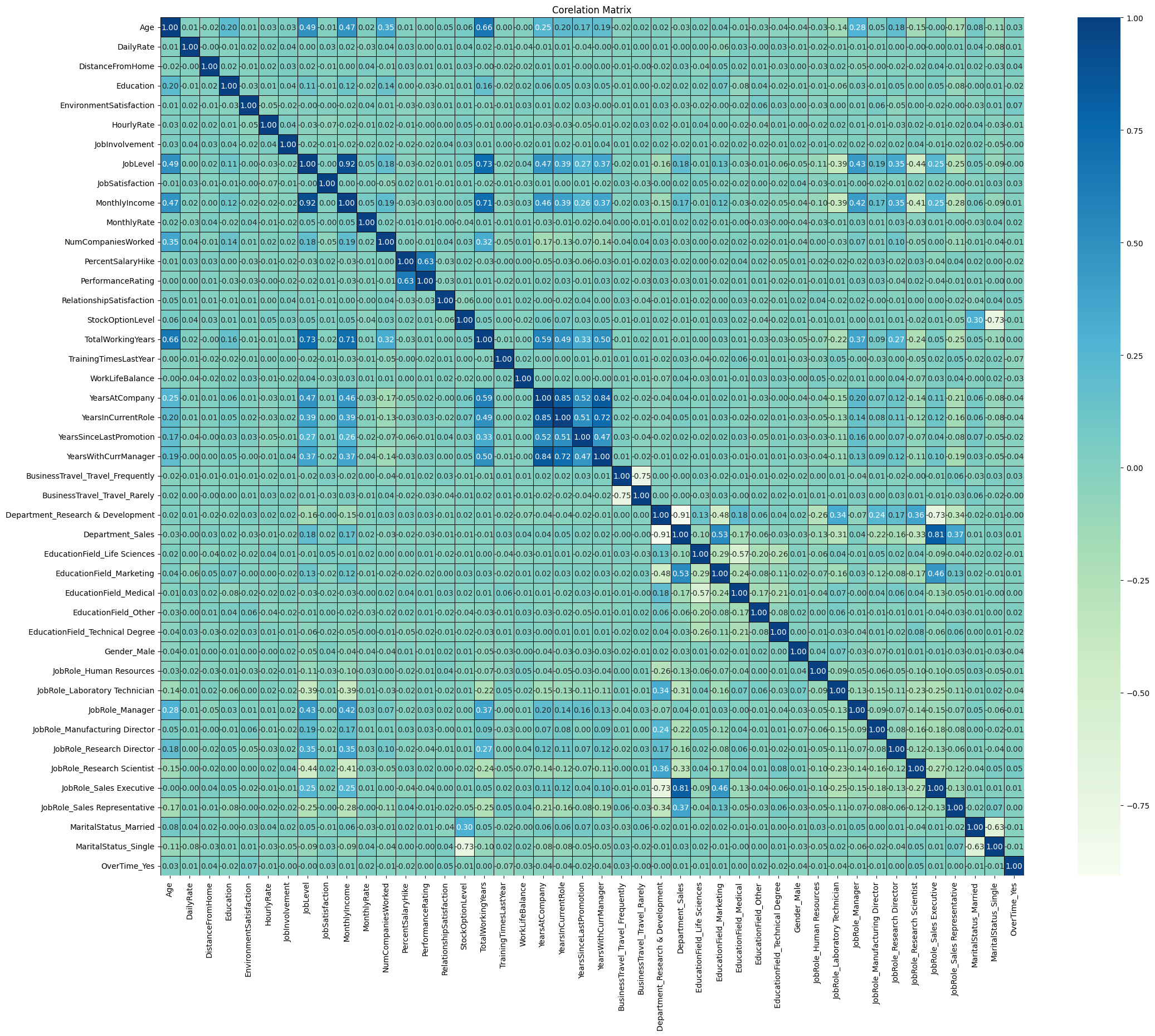
cor\_mat = X.corr(method='spearman') *# as we do not know the distribution of variable.*

plt.figure(figsize=(25,20))

sns.heatmap(cor\_mat,annot= True, fmt = '.2f', cmap='GnBu', linecolor='black', linewidths=0.5)

plt.title("Corelation Matrix")

plt.show()



*# Create a correlation matrix with target variable*

corr\_with\_target = X.corrwith(Y, method='spearman') *#chosing spearman as there is no normal variable.*

*# Sort features by correlation with target variable*

corr\_with\_target = corr\_with\_target.sort\_values(ascending=False)

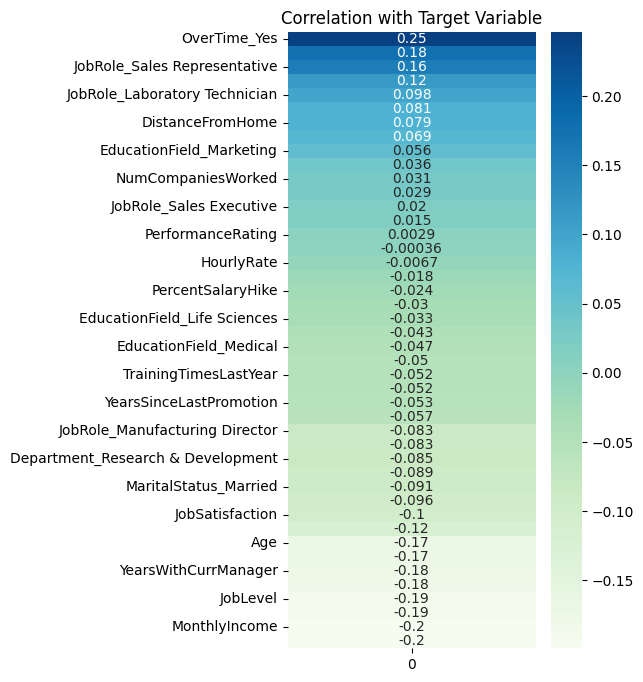
*# Plot the heatmap*

plt.figure(figsize=(4, 8))

sns.heatmap(corr\_with\_target.to\_frame(), cmap='GnBu', annot=True)

plt.title('Correlation with Target Variable')

plt.show()



dt = DecisionTreeClassifier(random\_state=42)

rf = RandomForestClassifier(random\_state=42)

ada = AdaBoostClassifier(random\_state=42)

gb = GradientBoostingClassifier(random\_state=42)

xgb = XGBClassifier(random\_state=42)

lgb = LGBMClassifier(random\_state=42)

models = [dt,rf,ada,gb,xgb,lgb]

class ModelBuilding:

def \_\_init\_\_(self, x\_train, x\_test, y\_train, y\_test, models, smote=False) -> None:

self.x\_train = x\_train

self.x\_test = x\_test

self.y\_train = y\_train

self.y\_test = y\_test

self.models = models

self.smote = smote

def dataBalaencing(self):

if self.smote:

sm = SMOTE()

self.x\_train, self.y\_train = sm.fit\_resample(self.x\_train, self.y\_train, sampling\_strategy = 'minority')

def fitModel(self):

trained\_models = []

metrics = pd.DataFrame(columns=['train\_accuracy', 'test\_accuracy', 'train\_f1', 'test\_f1'])

for model **in** self.models:

model\_building = ModelBuilding(X\_train, X\_test, y\_train, y\_test,models,smote=False)

trained\_models, metrics = model\_building.fitModel()

X\_new = X[final\_selected\_cols]

cor\_mat = X\_new.corr(method='spearman')

plt.figure(figsize=(25,20))

sns.heatmap(cor\_mat,annot= True, fmt = '.2f', cmap='GnBu', linecolor='black', linewidths=0.5)

plt.title("Corelation Matrix")

plt.show()

X\_new.drop(['JobLevel','YearsAtCompany'], axis =1, inplace= True)

final\_selected\_cols.remove('JobLevel')

final\_selected\_cols.remove('YearsAtCompany')

print(len(final\_selected\_cols))

print(X\_new.shape)

scaler = StandardScaler()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_new,Y, random\_state=42, train\_size=0.8, stratify= Y)

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.fit\_transform(X\_test)

X\_train = pd.DataFrame(X\_train\_scaled, columns= X\_train.columns)

X\_test = pd.DataFrame(X\_test\_scaled, columns= X\_test.columns)

dt = DecisionTreeClassifier()

rf = RandomForestClassifier()

ada = AdaBoostClassifier()

gb = GradientBoostingClassifier()

xgb = XGBClassifier()

lgb = LGBMClassifier()

models = [dt,rf,ada,gb,xgb,lgb]

xgb1 = XGBClassifier(

random\_state=42,

learning\_rate=0.01,

max\_depth=5,

n\_estimators=500,

scale\_pos\_weight=2.5,

max\_delta\_step=0.5,

min\_child\_weight=10,

gamma=4,

subsample=0.6,

reg\_lambda = 0.23,

class\_weight='balanced'

)

xgb = XGBClassifier()

models = [xgb,xgb1]

model\_building1 = ModelBuilding(models=models,

x\_train=X\_train,

y\_train= y\_train,

x\_test= X\_test,

y\_test= y\_test,smote= True)

trained\_modelss2, metrics = model\_building1.fitModel()

metrics

model\_building1.report(trained\_models=trained\_modelss2)

from sklearn.decomposition import PCA

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

pca = PCA()

X\_pca = pca.fit(X\_scaled)

plt.figure(figsize=(10,5))

plt.plot(np.cumsum(X\_pca.explained\_variance\_ratio\_))

plt.xlabel("principal components")

plt.ylabel("explained variance ratio or Amount of Info")