Data Encoding

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Read the dataset
df = pd.read_csv(r'F:\Technocolabs\WA_Fn-UseC_-HR-Employee-Attrition.csv')
```

In [3]:
```
df
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7
...	...	...	...	...	...	...	...	...	...	...
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	2061
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	2062
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	2064
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	2065
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	1	2068

1470 rows × 35 columns

In [4]:
```
# Display first few rows
print(df.head())
```

```
   Age Attrition     BusinessTravel  DailyRate              Department  \
0   41       Yes      Travel_Rarely       1102                   Sales
1   49        No  Travel_Frequently        279  Research & Development
2   37       Yes      Travel_Rarely       1373  Research & Development
3   33        No  Travel_Frequently       1392  Research & Development
4   27        No      Travel_Rarely        591  Research & Development

   DistanceFromHome  Education EducationField  EmployeeCount  EmployeeNumber  \
0                 1          2  Life Sciences              1               1
1                 8          1  Life Sciences              1               2
2                 2          2          Other              1               4
3                 3          4  Life Sciences              1               5
4                 2          1        Medical              1               7

   ...  RelationshipSatisfaction StandardHours  StockOptionLevel  \
0  ...                         1            80                 0
1  ...                         4            80                 1
2  ...                         2            80                 0
3  ...                         3            80                 0
4  ...                         4            80                 1

   TotalWorkingYears  TrainingTimesLastYear WorkLifeBalance  YearsAtCompany  \
0                  8                      0               1               6
1                 10                      3               3              10
2                  7                      3               3               0
3                  8                      3               3               8
4                  6                      3               3               2

  YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager
0                  4                        0                     5
1                  7                        1                     7
2                  0                        0                     0
3                  7                        3                     0
4                  2                        2                     2

[5 rows x 35 columns]
```

In [7]:
```
print(df.columns)
```

```
Index(['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'],
      dtype='object')
```

In [5]:
```
# Summary statistics
print(df.describe())
```

```
               Age    DailyRate  DistanceFromHome    Education  EmployeeCount  \
count  1470.000000  1470.000000       1470.000000  1470.000000         1470.0
mean     36.923810   802.485714          9.192517     2.912925            1.0
std       9.135373   403.509100          8.106864     1.024165            0.0
min      18.000000   102.000000          1.000000     1.000000            1.0
25%      30.000000   465.000000          2.000000     2.000000            1.0
50%      36.000000   802.000000          7.000000     3.000000            1.0
75%      43.000000  1157.000000         14.000000     4.000000            1.0
max      60.000000  1499.000000         29.000000     5.000000            1.0

       EmployeeNumber  EnvironmentSatisfaction   HourlyRate  JobInvolvement  \
count     1470.000000              1470.000000  1470.000000     1470.000000
mean      1024.865306                 2.721769    65.891156        2.729932
std        602.024335                 1.093082    20.329428        0.711561
min          1.000000                 1.000000    30.000000        1.000000
25%        491.250000                 2.000000    48.000000        2.000000
50%       1020.500000                 3.000000    66.000000        3.000000
75%       1555.750000                 4.000000    83.750000        3.000000
max       2068.000000                 4.000000   100.000000        4.000000

          JobLevel  ...  RelationshipSatisfaction  StandardHours  \
count  1470.000000  ...               1470.000000         1470.0
mean      2.063946  ...                  2.712245           80.0
std       1.106940  ...                  1.081209            0.0
min       1.000000  ...                  1.000000           80.0
25%       1.000000  ...                  2.000000           80.0
50%       2.000000  ...                  3.000000           80.0
75%       3.000000  ...                  4.000000           80.0
max       5.000000  ...                  4.000000           80.0

       StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
count       1470.000000        1470.000000            1470.000000
mean           0.793878          11.279592               2.799320
std            0.852077           7.780782               1.289271
min            0.000000           0.000000               0.000000
25%            0.000000           6.000000               2.000000
50%            1.000000          10.000000               3.000000
75%            1.000000          15.000000               3.000000
max            3.000000          40.000000               6.000000

       WorkLifeBalance  YearsAtCompany  YearsInCurrentRole  \
count      1470.000000     1470.000000         1470.000000
mean          2.761224        7.008163            4.229252
std           0.706476        6.126525            3.623137
min           1.000000        0.000000            0.000000
25%           2.000000        3.000000            2.000000
50%           3.000000        5.000000            3.000000
75%           3.000000        9.000000            7.000000
max           4.000000       40.000000           18.000000

       YearsSinceLastPromotion  YearsWithCurrManager
count              1470.000000           1470.000000
mean                  2.187755              4.123129
std                   3.222430              3.568136
min                   0.000000              0.000000
25%                   0.000000              2.000000
50%                   1.000000              3.000000
75%                   3.000000              7.000000
max                  15.000000             17.000000

[8 rows x 26 columns]
```

In [8]:
```
if 'categorical_column' in df.columns:
    # One-hot encoding
    encoded_df = pd.get_dummies(df, columns=['categorical_column'])
    print(encoded_df.head())
else:
    print("The column 'categorical_column' does not exist in the DataFrame. Please provide the correct column name."
```

```
The column 'categorical_column' does not exist in the DataFrame. Please provide the correct column name.
```

In [19]:
```
# One-hot encoding
encoded_df = pd.get_dummies(df, columns=['Department'])
```

In [20]:
```
# Label encoding
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['encoded_column'] = label_encoder.fit_transform(df['Department'])
```

In [22]:
```
# Specify the list of categorical columns you want to one-hot encode
categorical_columns = ['BusinessTravel', 'Department', 'EducationField', 'Gender', 'JobRole', 'MaritalStatus', 'Over

# One-hot encoding
encoded_df = pd.get_dummies(df, columns=categorical_columns)
print(encoded_df.head())
```

```
   Age Attrition  DailyRate  DistanceFromHome  Education  EmployeeCount  \
0   41       Yes       1102                 1          2              1
1   49        No        279                 8          1              1
2   37       Yes       1373                 2          2              1
3   33        No       1392                 3          4              1
4   27        No        591                 2          1              1

   EmployeeNumber  EnvironmentSatisfaction  HourlyRate  JobInvolvement  ...  \
0               1                        2          94               3  ...
1               2                        3          61               2  ...
2               4                        4          92               2  ...
3               5                        4          56               3  ...
4               7                        1          40               3  ...

   JobRole_Research Director  JobRole_Research Scientist  \
0                          0                           0
1                          0                           1
2                          0                           0
3                          0                           1
4                          0                           0

   JobRole_Sales Executive  JobRole_Sales Representative  \
0                        1                             0
1                        0                             0
2                        0                             0
3                        0                             0
4                        0                             0

   MaritalStatus_Divorced  MaritalStatus_Married  MaritalStatus_Single  \
0                       0                      0                     1
1                       0                      1                     0
2                       0                      0                     1
3                       0                      1                     0
4                       0                      1                     0

   Over18_Y  OverTime_No  OverTime_Yes
0         1            0             1
1         1            1             0
2         1            0             1
3         1            0             1
4         1            1             0

[5 rows x 57 columns]
```

In [23]:
```
from sklearn.preprocessing import LabelEncoder

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Encode a specific column
df['Attrition_encoded'] = label_encoder.fit_transform(df['Attrition'])
print(df[['Attrition', 'Attrition_encoded']].head())
```

```
  Attrition  Attrition_encoded
0       Yes                  1
1        No                  0
2       Yes                  1
3        No                  0
4        No                  0
```

In [24]:
```
from sklearn.preprocessing import MinMaxScaler

# Initialize MinMaxScaler
scaler = MinMaxScaler()

# Scale numerical features
numerical_columns = ['Age', 'DailyRate', 'DistanceFromHome', 'HourlyRate', 'MonthlyIncome', 'PercentSalaryHike', 'To
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
```

In [25]:
```
from sklearn.model_selection import train_test_split

# Split data into features (X) and target variable (y)
X = df.drop(columns=['Attrition'])
y = df['Attrition']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [ ]:
```

```