```
!pip install pandas numpy matplotlib seaborn
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (2.0.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

```python
import pandas as pd

# Load the dataset with the correct file name
df = pd.read_csv('greendestination.csv')

# Display the first few rows
print("First 5 rows of the dataset:")
print(df.head())

# Check data types and missing values
print("\nDataset Info:")
print(df.info())

# Basic statistics
print("\nBasic Statistics:")
print(df.describe())
```

```
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]
```

```python
# Check for duplicates
print("Number of duplicate rows:", df.duplicated().sum())

# Drop unnecessary columns (e.g., EmployeeCount, Over18, StandardHours are constant)
df = df.drop(columns=['EmployeeCount', 'Over18', 'StandardHours'])

# Convert categorical variables to appropriate types
df['Attrition'] = df['Attrition'].map({'Yes': 1, 'No': 0})  # Convert to binary (1 for Yes, 0 for No)
```

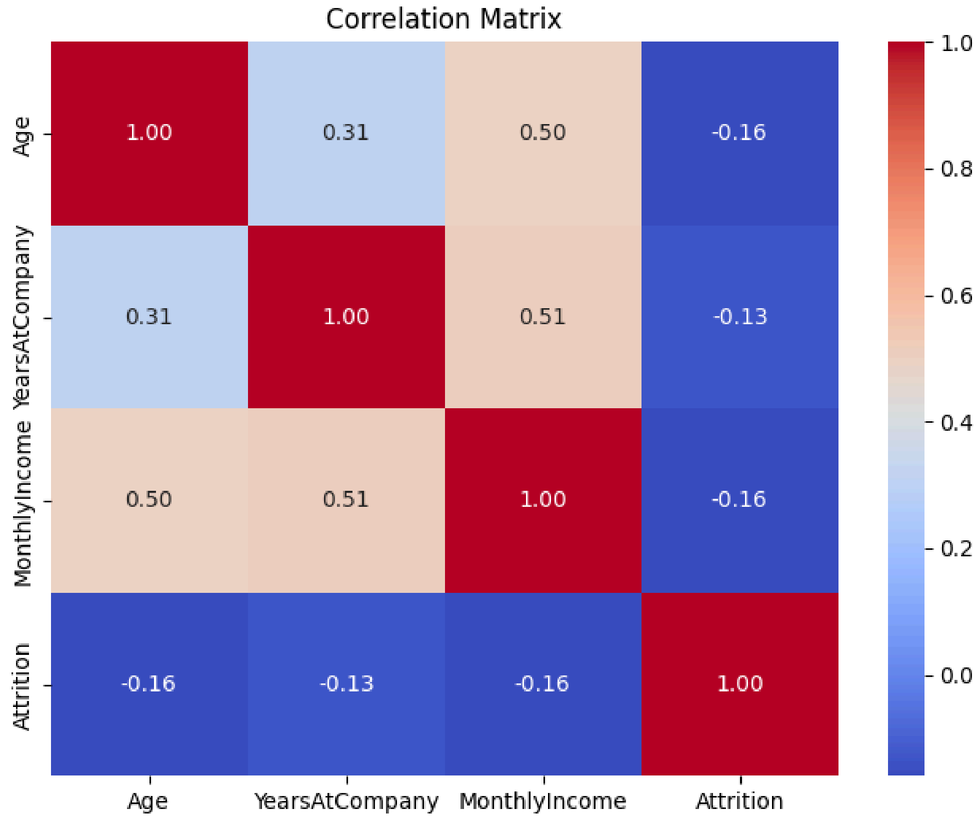
```
Number of duplicate rows: 0
```

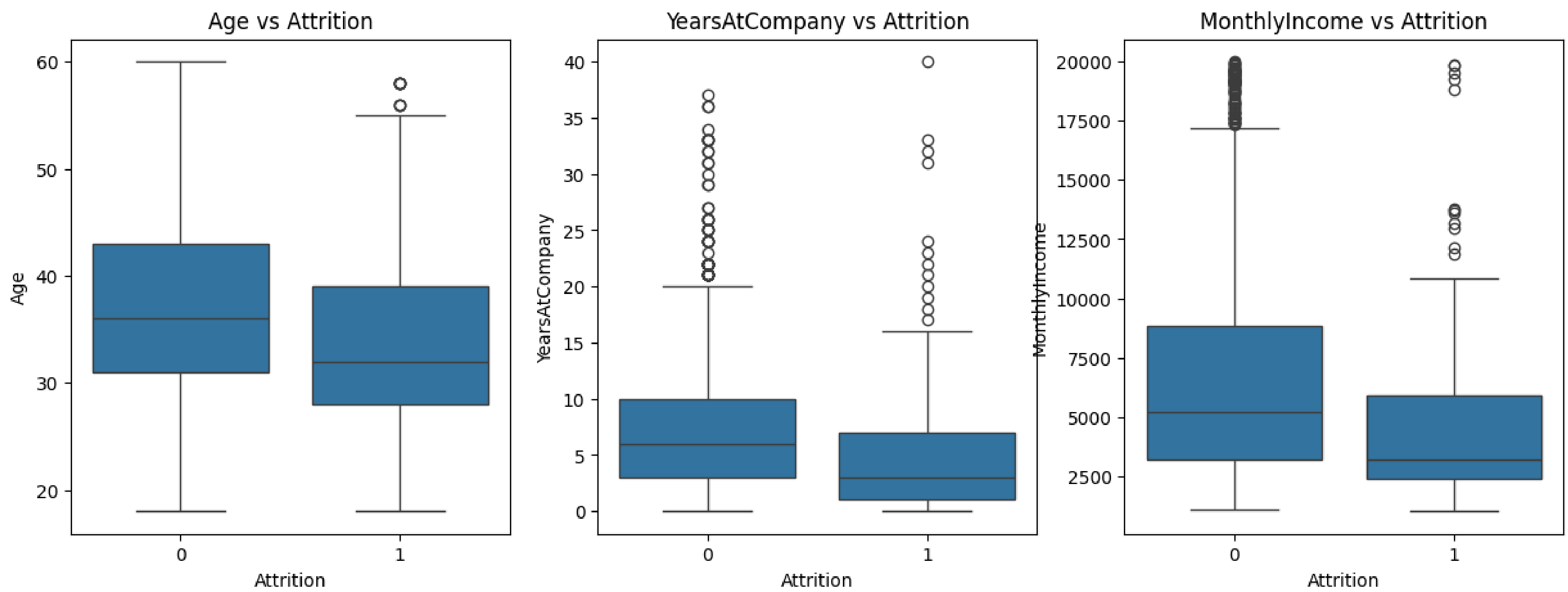
```
# Calculate attrition rate
total_employees = len(df)
employees_left = df['Attrition'].sum()
attrition_rate = (employees_left / total_employees) * 100

print(f"Total Employees: {total_employees}")
print(f"Employees Who Left: {employees_left}")
print(f"Attrition Rate: {attrition_rate:.2f}%")
```

```
Total Employees: 1470
Employees Who Left: 237
Attrition Rate: 16.12%
```

```
# Step 5: Analyze Factors Influencing Attrition (Improved)
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler

# Correlation matrix for numeric variables (unchanged)
numeric_cols = ['Age', 'YearsAtCompany', 'MonthlyIncome', 'Attrition']
correlation = df[numeric_cols].corr()
print("\nCorrelation Matrix:")
print(correlation)

# Visualize correlations (unchanged)
plt.figure(figsize=(8, 6))
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()

# Boxplots to visualize distributions (unchanged)
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
sns.boxplot(x='Attrition', y='Age', data=df)
plt.title('Age vs Attrition')

plt.subplot(1, 3, 2)
sns.boxplot(x='Attrition', y='YearsAtCompany', data=df)
plt.title('YearsAtCompany vs Attrition')

plt.subplot(1, 3, 3)
sns.boxplot(x='Attrition', y='MonthlyIncome', data=df)
plt.title('MonthlyIncome vs Attrition')
plt.show()

# Logistic Regression with scaling and class weighting
X = df[['Age', 'YearsAtCompany', 'MonthlyIncome']]
y = df['Attrition']

# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

# Train the model with class weighting
model = LogisticRegression(class_weight='balanced', random_state=42)
model.fit(X_train, y_train)

# Print coefficients
print("\nLogistic Regression Coefficients (after scaling):")
for feature, coef in zip(X.columns, model.coef_[0]):
    print(f"{feature}: {coef:.4f}")

# Model performance
y_pred = model.predict(X_test)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Correlation Matrix:
                    Age  YearsAtCompany  MonthlyIncome  Attrition
Age            1.000000        0.311309       0.497855  -0.159205
YearsAtCompany 0.311309        1.000000       0.514285  -0.134392
MonthlyIncome  0.497855        0.514285       1.000000  -0.159840
Attrition     -0.159205       -0.134392      -0.159840   1.000000
```

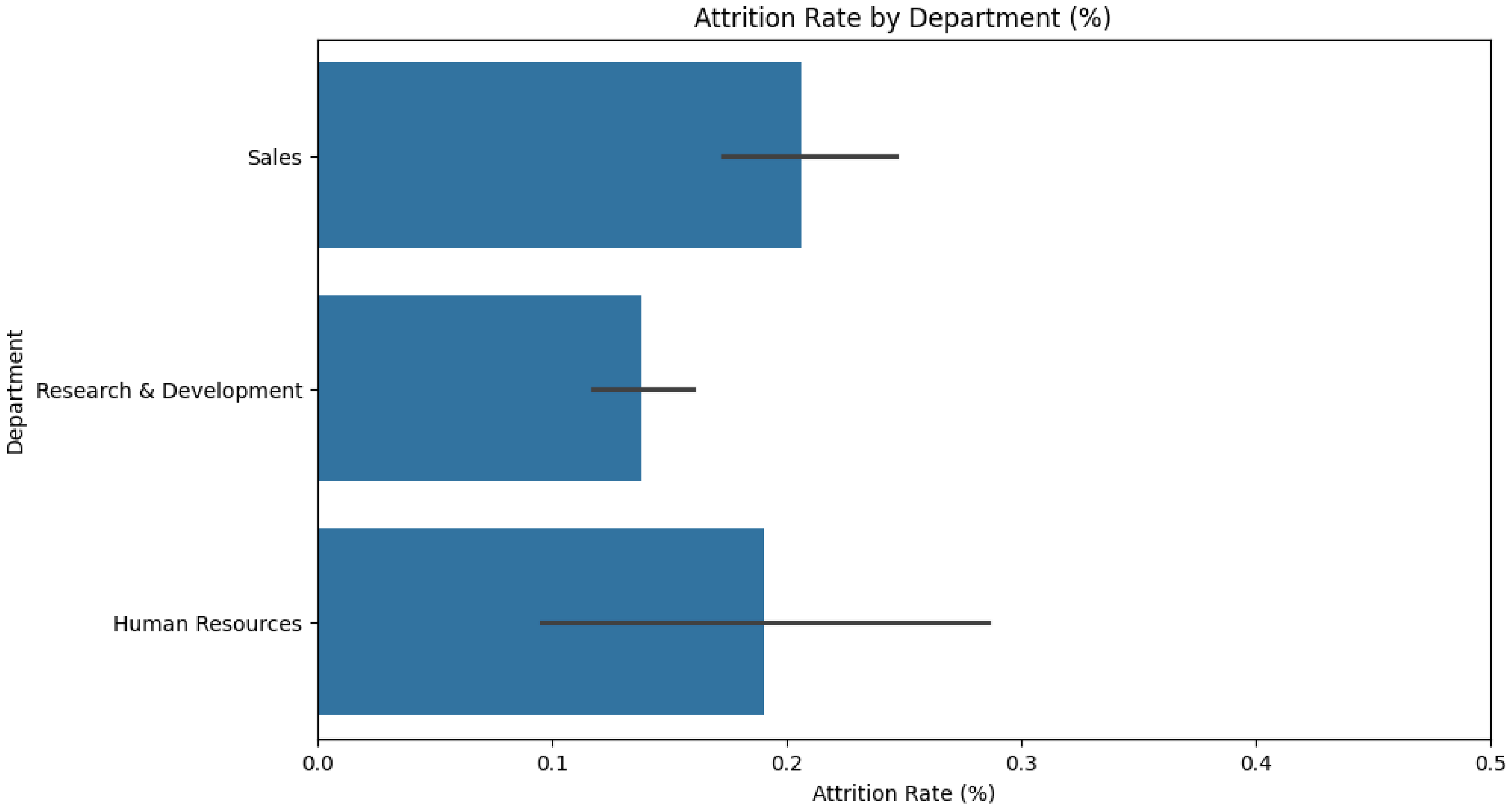




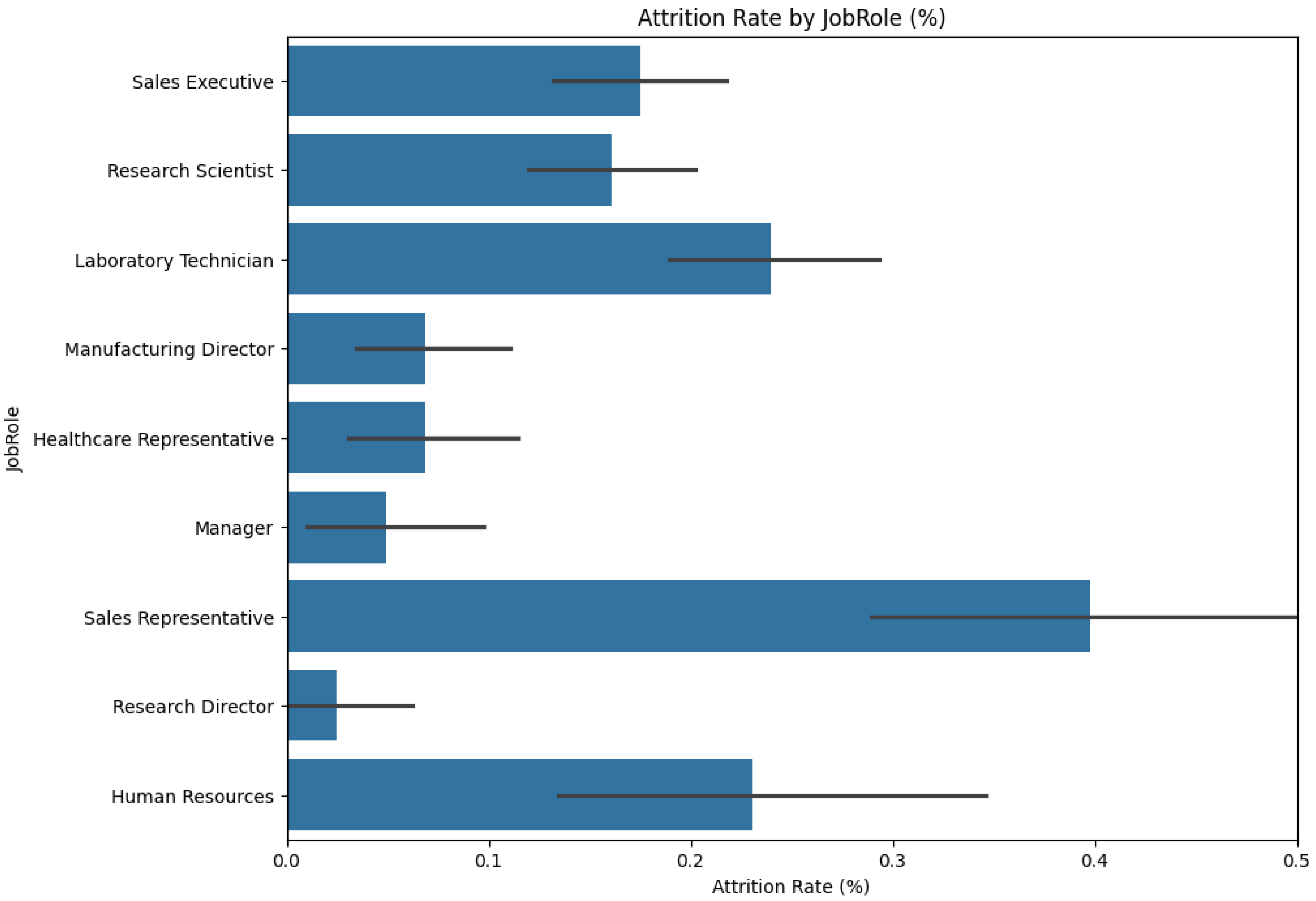

```
Logistic Regression Coefficients (after scaling):
Age: -0.2691
YearsAtCompany: -0.2021
MonthlyIncome: -0.2967

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.58      0.70       380
           1       0.19      0.62      0.29        61

    accuracy                           0.58       441
   macro avg       0.55      0.60      0.50       441
weighted avg       0.81      0.58      0.65       441
```

```
# Analyze Attrition by Department and JobRole
# Calculate attrition rate by Department
print("\nAttrition Rate by Department:")
department_attrition = df.groupby('Department')['Attrition'].mean().sort_values(ascending=False) * 100
print(department_attrition)

# Visualize attrition by Department
plt.figure(figsize=(10, 6))
sns.barplot(x='Attrition', y='Department', data=df)
plt.title('Attrition Rate by Department (%)')
plt.xlabel('Attrition Rate (%)')
plt.xlim(0, 0.5)  # Since Attrition is 0/1, the mean will be between 0 and 1; adjust for better visualization
plt.show()

# Calculate attrition rate by JobRole
print("\nAttrition Rate by JobRole:")
jobrole_attrition = df.groupby('JobRole')['Attrition'].mean().sort_values(ascending=False) * 100
print(jobrole_attrition)

# Visualize attrition by JobRole
plt.figure(figsize=(10, 8))
sns.barplot(x='Attrition', y='JobRole', data=df)
plt.title('Attrition Rate by JobRole (%)')
plt.xlabel('Attrition Rate (%)')
plt.xlim(0, 0.5)  # Adjust for better visualization
plt.show()
```

```
Attrition Rate by Department:
Department
Sales                     20.627803
Human Resources           19.047619
Research & Development    13.839750
Name: Attrition, dtype: float64
```



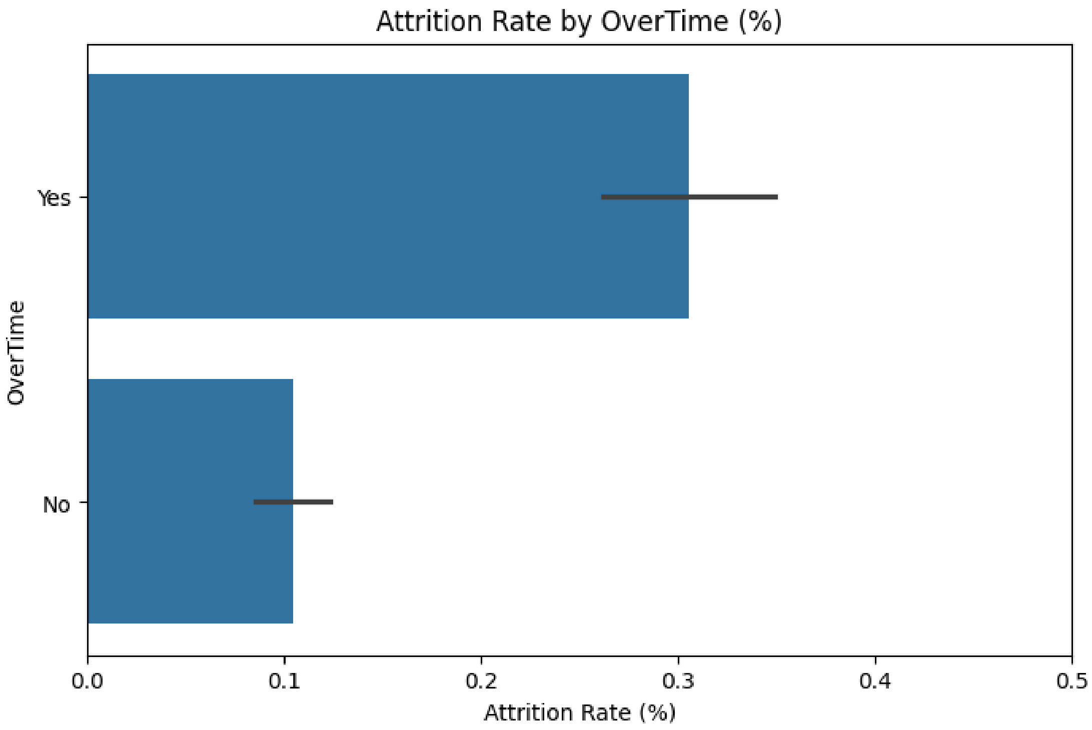

```
Attrition Rate by JobRole:
JobRole
Sales Representative         39.759036
Laboratory Technician        23.938224
Human Resources              23.076923
Sales Executive              17.484663
Research Scientist           16.095890
Manufacturing Director        6.896552
Healthcare Representative     6.870229
Manager                       4.901961
Research Director             2.500000
Name: Attrition, dtype: float64
```



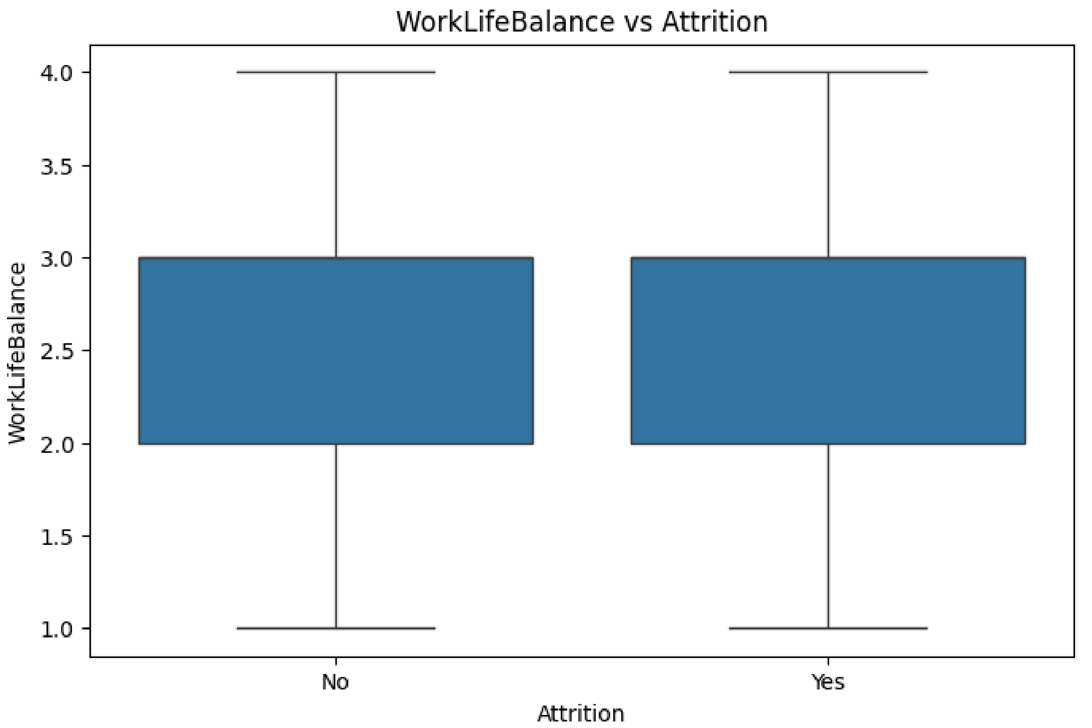

```
# Calculate attrition rate by OverTime
print("\nAttrition Rate by OverTime:")
overtime_attrition = df.groupby('OverTime')['Attrition'].mean().sort_values(ascending=False) * 100
print(overtime_attrition)

# Visualize attrition by OverTime
plt.figure(figsize=(8, 5))
sns.barplot(x='Attrition', y='OverTime', data=df)
plt.title('Attrition Rate by OverTime (%)')
plt.xlabel('Attrition Rate (%)')
plt.xlim(0, 0.5)  # Since Attrition is 0/1, the mean will be between 0 and 1
plt.show()

# Analyze WorkLifeBalance by Attrition
print("\nAverage WorkLifeBalance by Attrition:")
```

```
print('\nAverage WorkLifeBalance by Attrition:')
worklifebalance_attrition = df.groupby('Attrition')['WorkLifeBalance'].mean()
print(worklifebalance_attrition)

# Visualize WorkLifeBalance vs Attrition
plt.figure(figsize=(8, 5))
sns.boxplot(x='Attrition', y='WorkLifeBalance', data=df)
plt.title('WorkLifeBalance vs Attrition')
plt.xticks([0, 1], ['No', 'Yes'])  # Label Attrition as No/Yes for clarity
plt.show()
```

```
Attrition Rate by OverTime:
OverTime
Yes    30.528846
No     10.436433
Name: Attrition, dtype: float64
```




```
Average WorkLifeBalance by Attrition:
Attrition
0    2.781022
1    2.658228
Name: WorkLifeBalance, dtype: float64
```




```
# Add a column for Attrition as a string for Tableau
df['Attrition_Str'] = df['Attrition'].map({1: 'Yes', 0: 'No'})
df.to_csv('processed_employee_data.csv', index=False)
```