In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

In [15]:
```
df=pd.read_csv("C:\\Users\\Admin\\Desktop\\emp.csv")
```

In [16]:
```
df.head()
```

Out[16]:

| | Age | Attrition | BusinessTravel | DailyRate | Department | DistanceFromHome |
|---|---|---|---|---|---|---|
| **0** | 41 | Yes | Travel_Rarely | 1102 | Sales | 1 |
| **1** | 49 | No | Travel_Frequently | 279 | Research & Development | 8 |
| **2** | 37 | Yes | Travel_Rarely | 1373 | Research & Development | 2 |
| **3** | 33 | No | Travel_Frequently | 1392 | Research & Development | 3 |
| **4** | 27 | No | Travel_Rarely | 591 | Research & Development | 2 |

5 rows × 35 columns

In [17]:
```
df.isna().sum()
```

Out[17]:
```
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
```

In [7]:
```
import pandas as pd

print(df.head())
print(df.info())
print(df.isnull().sum())
```

```
Age               0
Attrition         0
BusinessTravel    0
DailyRate         0
Department        0
dtype: int64
<class 'pandas.core.series.Series'>
Index: 35 entries, Age to YearsWithCurrManager
Series name: None
Non-Null Count  Dtype
--------------  -----
35 non-null     int64
dtypes: int64(1)
memory usage: 1.6+ KB
None
0
```

In [18]:
```
df['Attrition'] = df['Attrition'].map({'Yes': 1, 'No': 0})
```

In [19]:
```
df
```

Out[19]:

| | Age | Attrition | BusinessTravel | DailyRate | Department | DistanceFromHom |
|---|---|---|---|---|---|---|
| **0** | 41 | 1 | Travel_Rarely | 1102 | Sales | |
| **1** | 49 | 0 | Travel_Frequently | 279 | Research & Development | |
| **2** | 37 | 1 | Travel_Rarely | 1373 | Research & Development | |
| **3** | 33 | 0 | Travel_Frequently | 1392 | Research & Development | |
| **4** | 27 | 0 | Travel_Rarely | 591 | Research & Development | |
| **...** | ... | ... | ... | ... | ... | |
| **1465** | 36 | 0 | Travel_Frequently | 884 | Research & Development | 2 |
| **1466** | 39 | 0 | Travel_Rarely | 613 | Research & Development | |
| **1467** | 27 | 0 | Travel_Rarely | 155 | Research & Development | |
| **1468** | 49 | 0 | Travel_Frequently | 1023 | Sales | |
| **1469** | 34 | 0 | Travel_Rarely | 628 | Research & Development | |

1470 rows × 35 columns

In [20]:
```
df['Attrition']
```

Out[20]:
```
0       1
1       0
2       1
3       0
4       0
       ..
1465    0
1466    0
1467    0
1468    0
1469    0
Name: Attrition, Length: 1470, dtype: int64
```

In [21]:
```
df = df.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], a
```

In [22]:
```
df
```

Out[22]:

| | Age | Attrition | BusinessTravel | DailyRate | Department | DistanceFromHom |
|---|---|---|---|---|---|---|
| **0** | 41 | 1 | Travel_Rarely | 1102 | Sales | |
| **1** | 49 | 0 | Travel_Frequently | 279 | Research & Development | |
| **2** | 37 | 1 | Travel_Rarely | 1373 | Research & Development | |
| **3** | 33 | 0 | Travel_Frequently | 1392 | Research & Development | |
| **4** | 27 | 0 | Travel_Rarely | 591 | Research & Development | |
| **...** | ... | ... | ... | ... | ... | |
| **1465** | 36 | 0 | Travel_Frequently | 884 | Research & Development | 2 |
| **1466** | 39 | 0 | Travel_Rarely | 613 | Research & Development | |
| **1467** | 27 | 0 | Travel_Rarely | 155 | Research & Development | |
| **1468** | 49 | 0 | Travel_Frequently | 1023 | Sales | |
| **1469** | 34 | 0 | Travel_Rarely | 628 | Research & Development | |

1470 rows × 31 columns

In [23]:
```
le = LabelEncoder()

for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = le.fit_transform(df[col])
```

In [24]:
```
df
```

Out[24]:

| | Age | Attrition | BusinessTravel | DailyRate | Department | DistanceFromHom |
|---|---|---|---|---|---|---|
| **0** | 41 | 1 | 2 | 1102 | 2 | |
| **1** | 49 | 0 | 1 | 279 | 1 | |
| **2** | 37 | 1 | 2 | 1373 | 1 | |
| **3** | 33 | 0 | 1 | 1392 | 1 | |
| **4** | 27 | 0 | 2 | 591 | 1 | |
| **...** | ... | ... | ... | ... | ... | . |
| **1465** | 36 | 0 | 1 | 884 | 1 | 2 |
| **1466** | 39 | 0 | 2 | 613 | 1 | |
| **1467** | 27 | 0 | 2 | 155 | 1 | |
| **1468** | 49 | 0 | 1 | 1023 | 2 | |
| **1469** | 34 | 0 | 2 | 628 | 1 | |

1470 rows × 31 columns

In [25]:
```
X = df.drop('Attrition', axis=1)
y = df['Attrition']
```

In [26]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc
```

In [27]:
```
sc = StandardScaler()
X_train_scaled = sc.fit_transform(X_train)
X_test_scaled = sc.transform(X_test)
```

In [28]:
```
log_model = LogisticRegression()
log_model.fit(X_train_scaled, y_train)
log_pred = log_model.predict(X_test_scaled)
```

In [29]:
```
tree_model = DecisionTreeClassifier()
tree_model.fit(X_train, y_train)
tree_pred = tree_model.predict(X_test)
```

In [33]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
```

In [34]:
```
print("Logistic Regression Accuracy:", accuracy_score(y_test, log_pred))
print(confusion_matrix(y_test, log_pred))
print(classification_report(y_test, log_pred))
```

```
Logistic Regression Accuracy: 0.8945578231292517
[[249   6]
 [ 25  14]]
              precision    recall  f1-score   support

           0       0.91      0.98      0.94       255
           1       0.70      0.36      0.47        39

    accuracy                           0.89       294
   macro avg       0.80      0.67      0.71       294
weighted avg       0.88      0.89      0.88       294
```

In [35]:
```
print("Decision Tree Accuracy:", accuracy_score(y_test, tree_pred))
print(confusion_matrix(y_test, tree_pred))
print(classification_report(y_test, tree_pred))
```

```
Decision Tree Accuracy: 0.7755102040816326
[[219  36]
 [ 30   9]]
              precision    recall  f1-score   support

           0       0.88      0.86      0.87       255
           1       0.20      0.23      0.21        39

    accuracy                           0.78       294
   macro avg       0.54      0.54      0.54       294
weighted avg       0.79      0.78      0.78       294
```

In [37]:
```
log_acc = accuracy_score(y_test, log_pred)
tree_acc = accuracy_score(y_test, tree_pred)
```

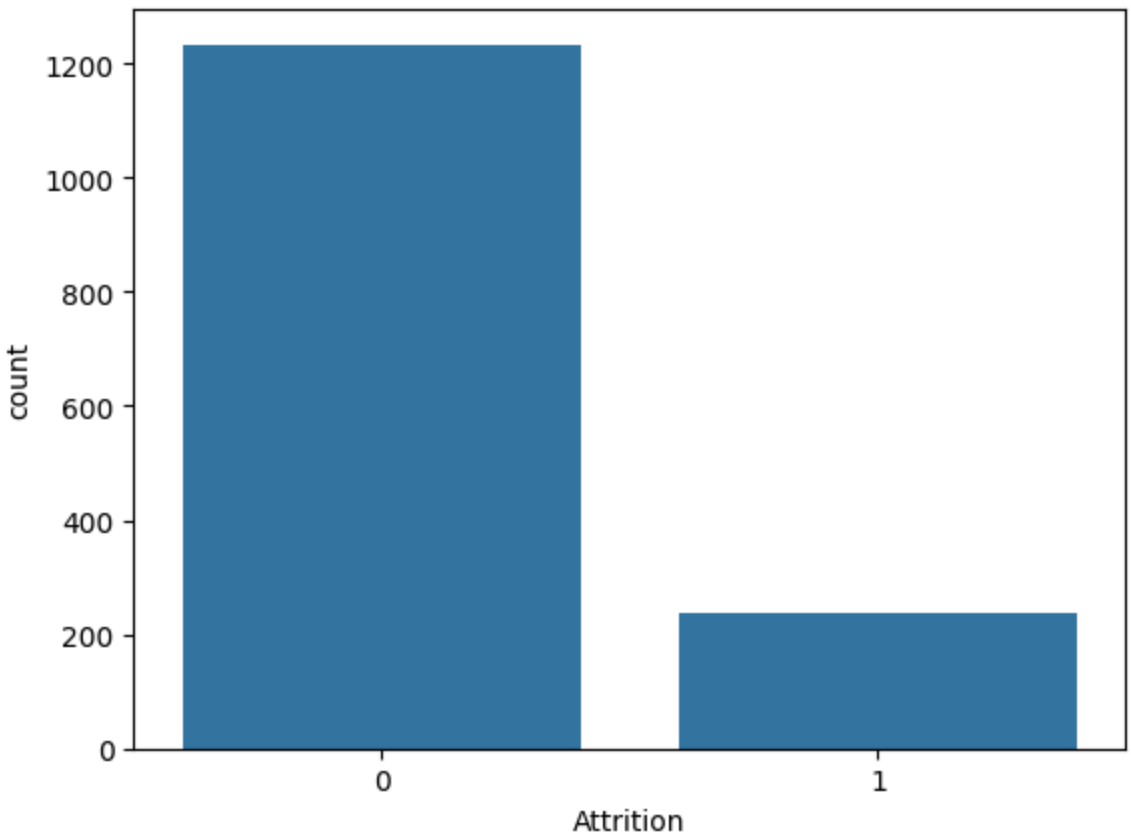
In [38]:
```
log_acc
```

Out[38]:
```
0.8945578231292517
```

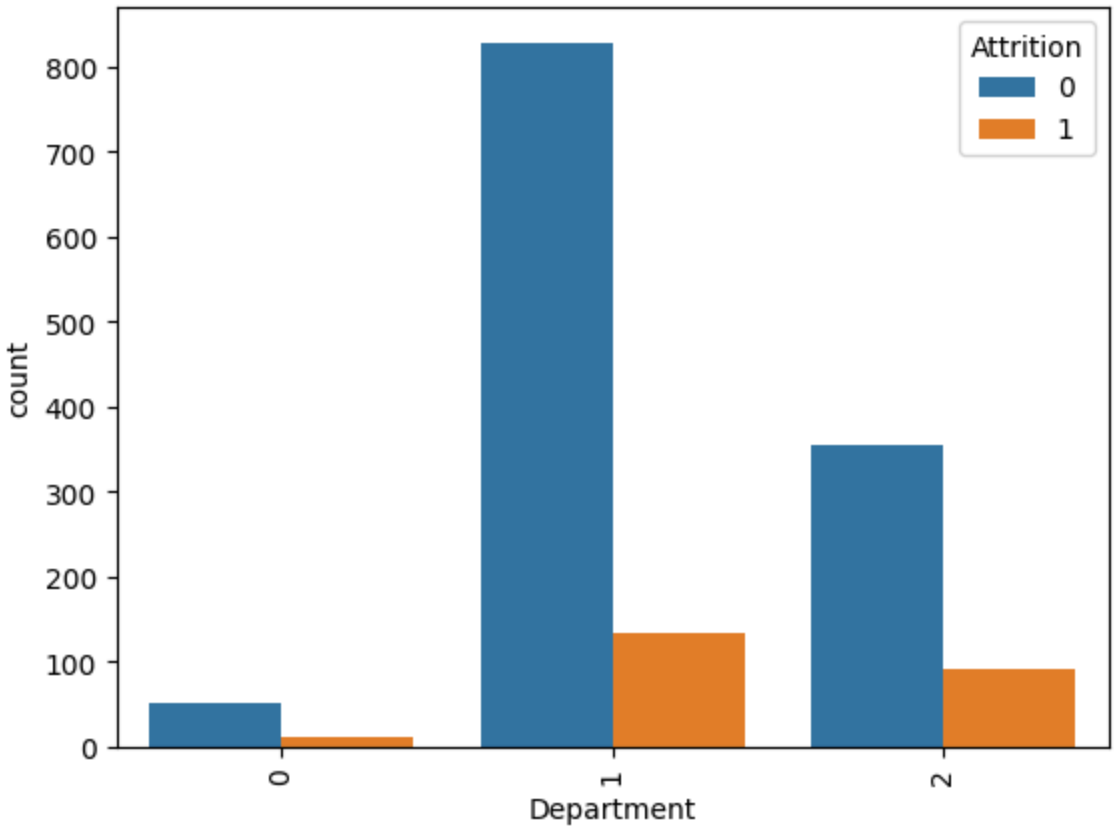
In [39]:
```
tree_acc
```

Out[39]:
```
0.7755102040816326
```

In [40]:
```
sns.countplot(x=df['Attrition'])
plt.show()
```




In [41]:
```
sns.countplot(x=df['Department'], hue=df['Attrition'])
plt.xticks(rotation=90)
plt.show()
```


Attrition
0
1
800
700
600
500
400
300
200
100
0
count
0
1
2
Department