In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import shap
df = pd.read_csv("HR.csv")
df.head()
```

Out[1]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfaction	StandardHours	StockOptionLevel	TotalWorkingYears
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	1	80	0	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	4	80	1	1
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	2	80	0	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	3	80	0	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	4	80	1	

5 rows × 35 columns

In [2]:
```
df.isnull().sum()
df['Attrition'].value_counts()
```

Out[2]:
```
Attrition
No     1233
Yes     237
Name: count, dtype: int64
```

In [3]:
```
df.dropna(inplace=True)
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
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
```

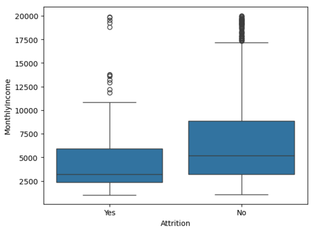
In [5]:
```
df.groupby("Department")["Attrition"].value_counts(normalize=True).unstack()
```

Out[5]:

Attrition	No	Yes
Department		
Human Resources	0.809524	0.190476
Research & Development	0.861602	0.138398
Sales	0.793722	0.206278

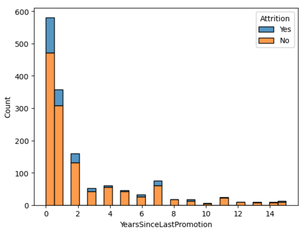
In [6]:
```
sns.boxplot(x="Attrition", y="MonthlyIncome", data=df)
```

Out[6]:
```
<Axes: xlabel='Attrition', ylabel='MonthlyIncome'>
```



In [7]:
```
sns.histplot(data=df, x="YearsSinceLastPromotion", hue="Attrition", multiple="stack")
```

Out[7]:
```
<Axes: xlabel='YearsSinceLastPromotion', ylabel='Count'>
```



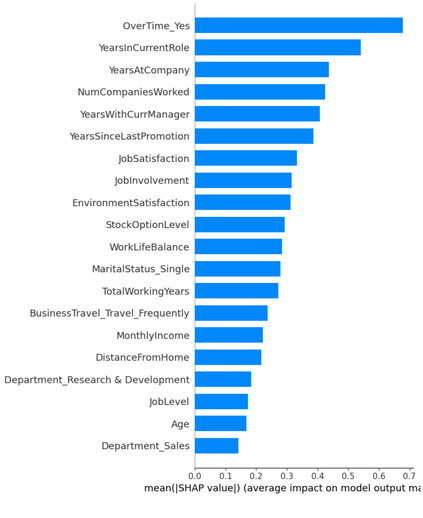
In [8]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Separate features and target
X = df.drop("Attrition", axis=1)
y = df["Attrition"].map({"Yes": 1, "No": 0})   # convert target to 0/1

# One-hot encode categorical features
X = pd.get_dummies(X, drop_first=True)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Train Logistic Regression
lr = LogisticRegression(max_iter=5000, solver="lbfgs")
lr.fit(X_train, y_train)

# Predictions
y_pred = lr.predict(X_test)

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)
```

```
Confusion Matrix:
 [[247   8]
 [ 23  16]]

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.97      0.94       255
           1       0.67      0.41      0.51        39

    accuracy                           0.89       294
   macro avg       0.79      0.69      0.72       294
weighted avg       0.88      0.89      0.88       294

Model Accuracy: 0.8945578231292517
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [9]:
```
# Ensure DataFrames
X_train_df = pd.DataFrame(X_train, columns=X.columns)
X_test_df = pd.DataFrame(X_test, columns=X.columns)

# Use SHAP Explainer (works with linear models)
explainer = shap.Explainer(lr, X_train_df)

# Get SHAP values
shap_values = explainer(X_test_df)

# ✅ Force conversion into 2D numpy array
if hasattr(shap_values, "values"):   # Explanation object
    shap_array = shap_values.values
elif isinstance(shap_values, list):  # list of arrays
    shap_array = np.array(shap_values)
else:                                # fallback
    shap_array = np.atleast_2d(shap_values)

print("SHAP shape:", shap_array.shape)

# Plot summary (bar to be safe)
shap.summary_plot(shap_array, X_test_df, feature_names=X.columns, plot_type="bar")
```

```
SHAP shape: (294, 47)
```



In [10]:
```
df["Attrition_Prob"] = lr.predict_proba(X)[:,1]
df.to_csv("hr_attrition_results.csv", index=False)
```

In [ ]:
```

```