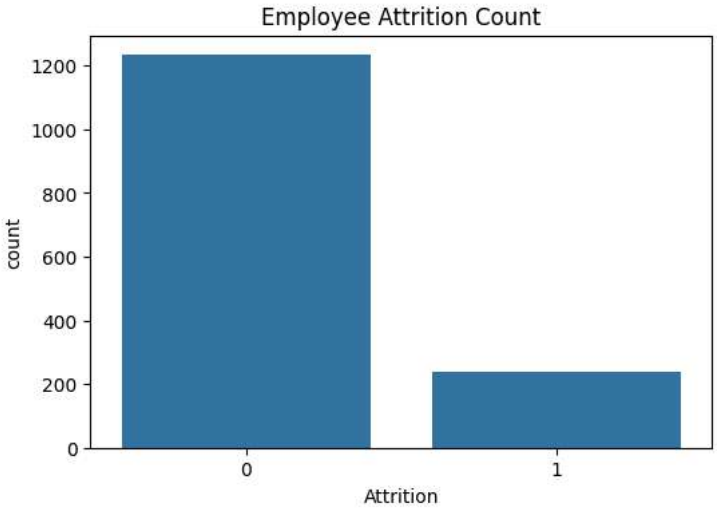
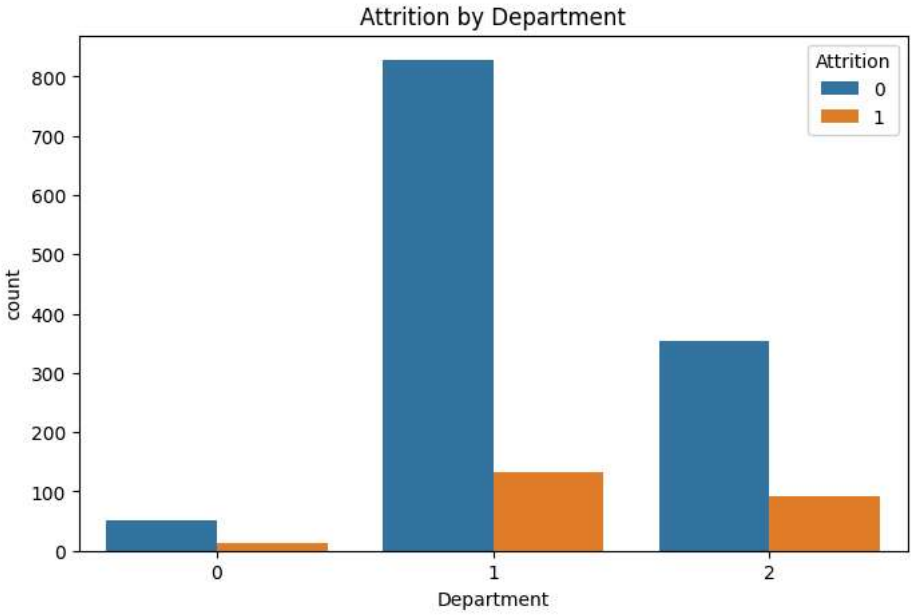
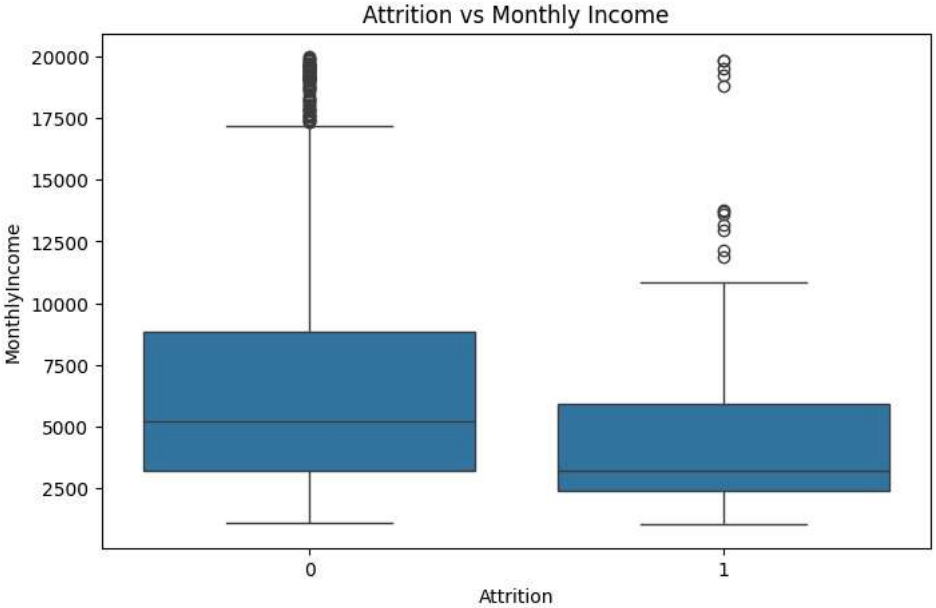
```
# HR Analytics - Predict Employee Attrition
# ----------------------------------------
# Full Python Project Skeleton (Ready to Run)

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import shap

# 1. Load Data
df = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")

# 2. Basic Cleaning
df = df.drop(columns=["EmployeeCount", "EmployeeNumber", "Over18", "StandardHours"])  # redundant cols
target = "Attrition"

# Encode target
df[target] = df[target].map({"Yes": 1, "No": 0})

# Encode categorical variables
categorical_cols = df.select_dtypes(include=["object"]).columns
le = LabelEncoder()
for col in categorical_cols:
    df[col] = le.fit_transform(df[col])

# 3. EDA (Exploratory Data Analysis)
plt.figure(figsize=(6,4))
sns.countplot(data=df, x="Attrition")
plt.title("Employee Attrition Count")
plt.show()

plt.figure(figsize=(8,5))
sns.countplot(data=df, x="Department", hue="Attrition")
plt.title("Attrition by Department")
plt.show()

plt.figure(figsize=(8,5))
sns.boxplot(data=df, x="Attrition", y="MonthlyIncome")
plt.title("Attrition vs Monthly Income")
plt.show()

# 4. Split Data
X = df.drop(columns=[target])
y = df[target]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# 5. Model 1 - Logistic Regression
log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)
y_pred_log = log_model.predict(X_test)

print("🔹 Logistic Regression Results")
print("Accuracy:", accuracy_score(y_test, y_pred_log))
print(classification_report(y_test, y_pred_log))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_log))

# 6. Model 2 - Decision Tree
tree_model = DecisionTreeClassifier(max_depth=5, random_state=42)
tree_model.fit(X_train, y_train)
y_pred_tree = tree_model.predict(X_test)

print("\n🔹 Decision Tree Results")
print("Accuracy:", accuracy_score(y_test, y_pred_tree))
print(classification_report(y_test, y_pred_tree))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_tree))

# 7. Feature Importance (Decision Tree)
feat_importances = pd.Series(tree_model.feature_importances_, index=X.columns)
```

```
feat_importances.nlargest(10).plot(kind="barh", figsize=(8,5))
plt.title("Top 10 Features Driving Attrition (Decision Tree)")
plt.show()

# 8. Explainability with SHAP (Logistic Regression)
explainer = shap.LinearExplainer(log_model, X_train, feature_perturbation="interventional")
shap_values = explainer.shap_values(X_test)

shap.summary_plot(shap_values, X_test, plot_type="bar", max_display=10)
```




```
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:465: 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(
```