```
import io, os, textwrap, warnings
warnings.filterwarnings("ignore")
```

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier, plot_tree
```

```
from google.colab import files
up = files.upload()
```

Choose Files WA_Fn-Us…trition (1).csv

- **WA_Fn-UseC_-HR-Employee-Attrition (1).csv**(text/csv) - 227977 bytes, last modified: 8/22/2025 - 100% done

```
Saving WA_Fn-UseC_-HR-Employee-Attrition (1).csv to WA_Fn-UseC_-HR-Employee-Attrition (1).csv
```

```
df = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition (1).csv")
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCou
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	

5 rows × 35 columns

```
print(df.shape)
print(df.columns.tolist())

df.info()
df.describe(include="all").T.head(15)

print("\nMissing values:")
print(df.isna().sum())
```

```
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB

Missing values:
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
```

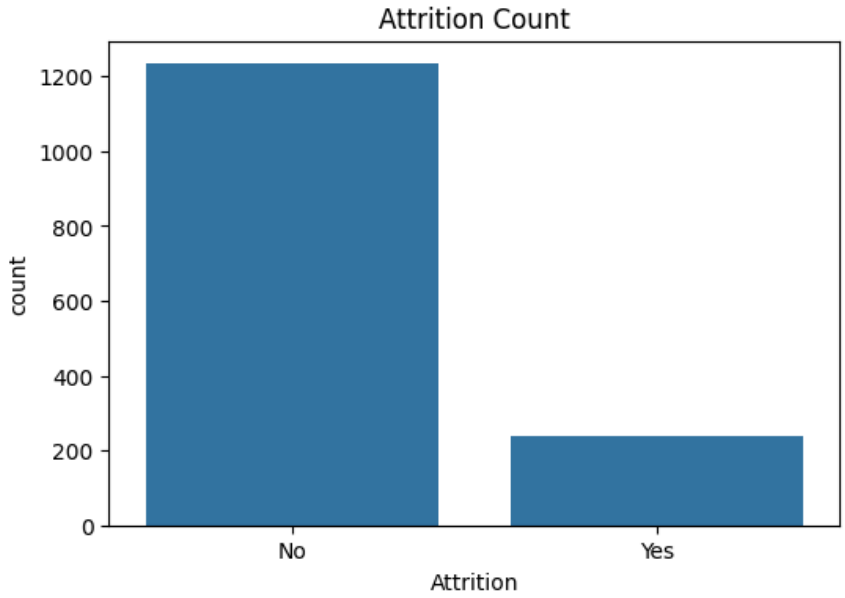
```
for col in df.select_dtypes(include="object").columns:
    df[col] = df[col].astype("category")

df.dtypes.head(15)
```

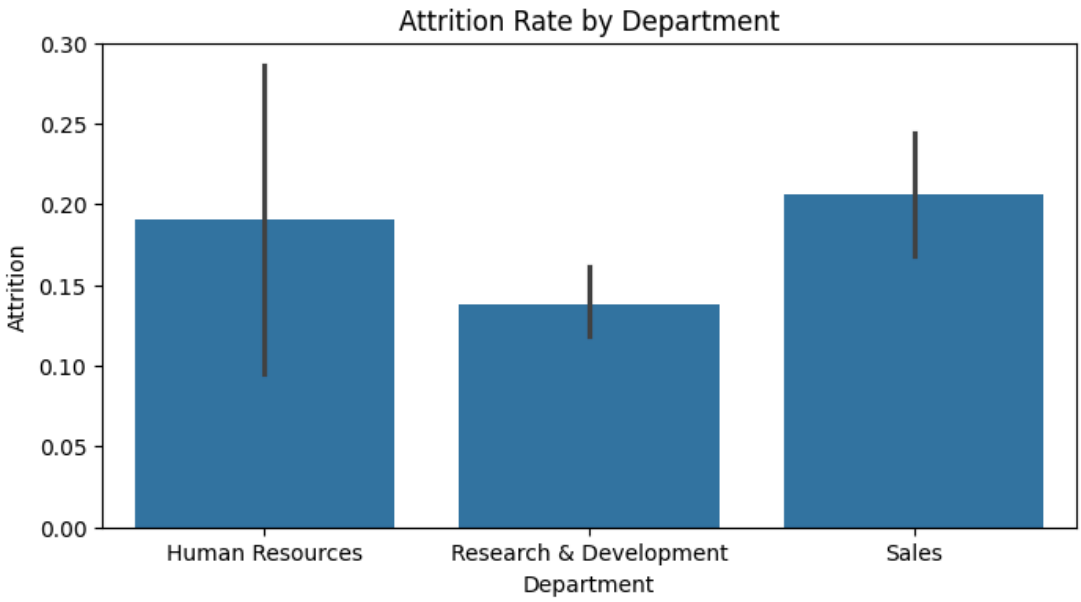
	0
Age	int64
Attrition	category
BusinessTravel	category
DailyRate	int64
Department	category
DistanceFromHome	int64
Education	int64
EducationField	category
EmployeeCount	int64
EmployeeNumber	int64
EnvironmentSatisfaction	int64
Gender	category
HourlyRate	int64
JobInvolvement	int64
JobLevel	int64

dtype: object

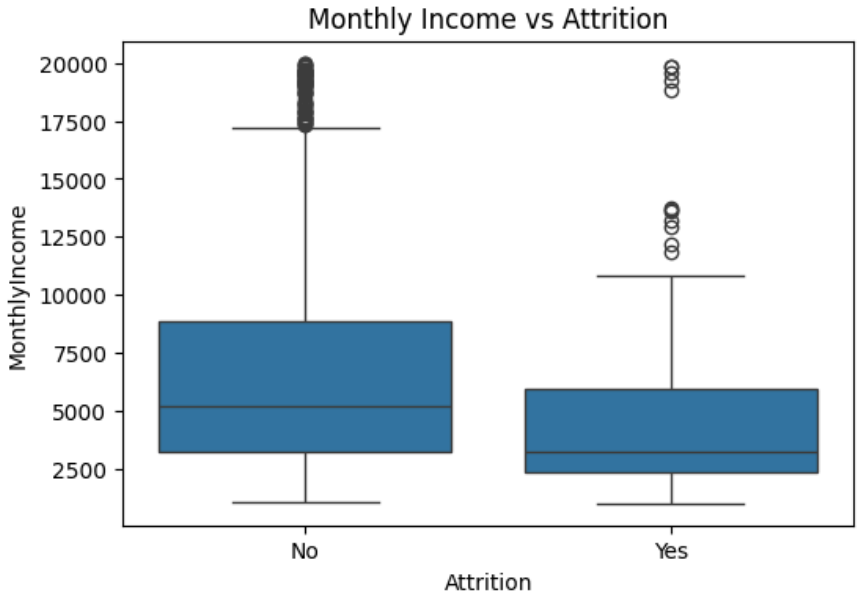
```
attrition_rate = (df['Attrition'] == 'Yes').mean()
print(f"Overall Attrition Rate: {attrition_rate:.2%}")

plt.figure(figsize=(6,4))
sns.countplot(x='Attrition', data=df)
plt.title("Attrition Count"); plt.show()

plt.figure(figsize=(8,4))
sns.barplot(x='Department', y=(df['Attrition']=='Yes').astype(int), data=df)
plt.title("Attrition Rate by Department"); plt.show()
```

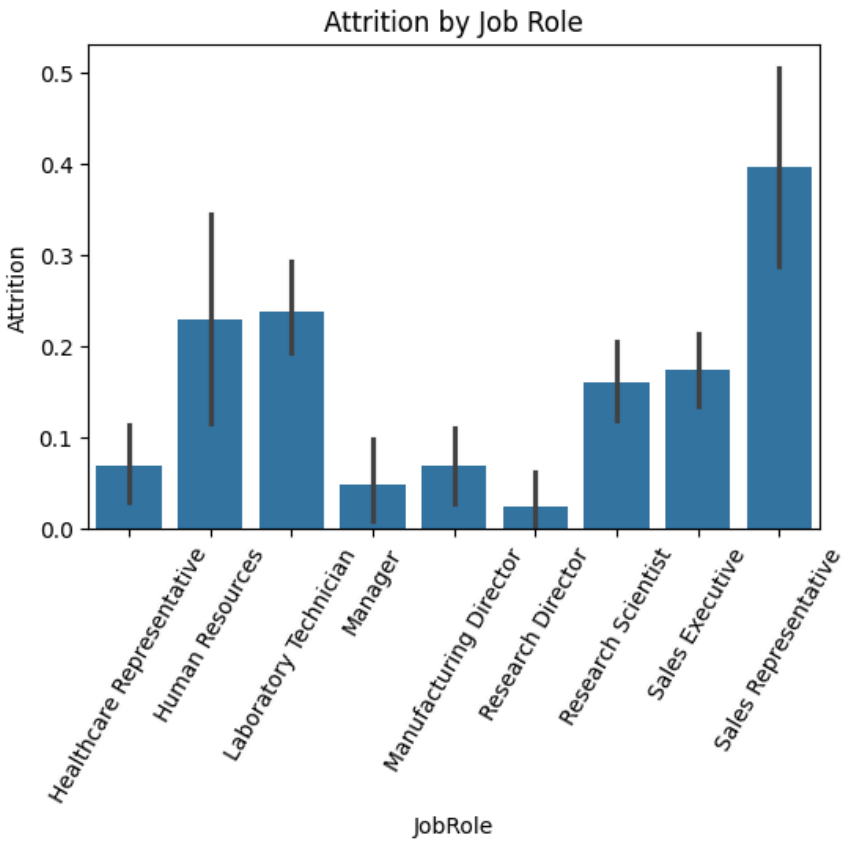
```
Overall Attrition Rate: 16.12%
```






```
plt.figure(figsize=(6,4))
sns.boxplot(x='Attrition', y='MonthlyIncome', data=df)
plt.title("Monthly Income vs Attrition"); plt.show()

plt.figure(figsize=(6,4))
sns.barplot(x='JobRole', y=(df['Attrition']=='Yes').astype(int), data=df)
plt.xticks(rotation=60)
plt.title("Attrition by Job Role"); plt.show()
```






```
y = (df['Attrition'] == 'Yes').astype(int)
X = df.drop(columns=['Attrition','EmployeeNumber','EmployeeCount','Over18','StandardHours'])

num_features = X.select_dtypes(include=[np.number]).columns.tolist()
cat_features = X.select_dtypes(exclude=[np.number]).columns.tolist()

pre = ColumnTransformer([
    ('num', 'passthrough', num_features),
    ('cat', OneHotEncoder(handle_unknown='ignore'), cat_features)
])

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, stratify=y, random_state=42
```

```
)
len(X_train), len(X_test)
```

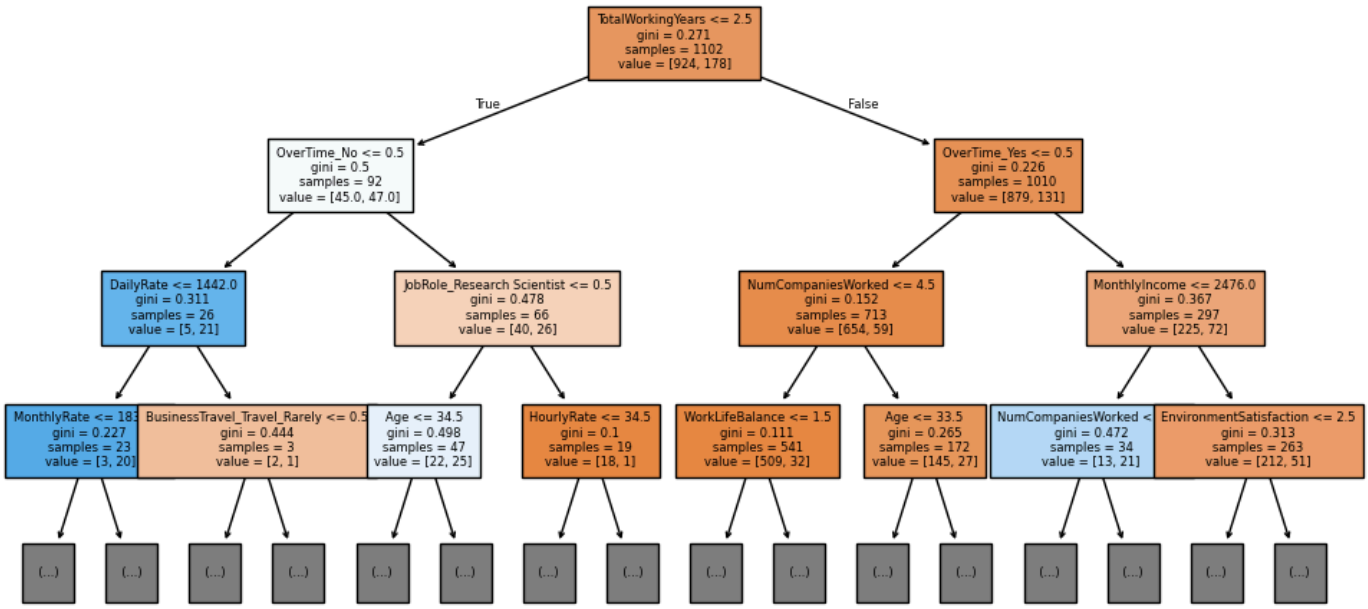
```
(1102, 368)
```

```
log_reg = Pipeline(steps=[
    ('prep', pre),
    ('clf', LogisticRegression(max_iter=500))
])
log_reg.fit(X_train, y_train)

pred_lr = log_reg.predict(X_test)
print("Accuracy (Logistic Regression):", accuracy_score(y_test, pred_lr))
print("\nClassification Report:\n", classification_report(y_test, pred_lr))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, pred_lr))
```

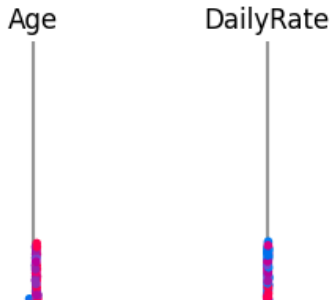
```
Accuracy (Logistic Regression): 0.8559782608695652

Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.99      0.92       309
           1       0.80      0.14      0.23        59

    accuracy                           0.86       368
   macro avg       0.83      0.56      0.58       368
weighted avg       0.85      0.86      0.81       368


Confusion Matrix:
 [[307   2]
 [ 51   8]]
```

```
dt = Pipeline(steps=[
    ('prep', pre),
    ('clf', DecisionTreeClassifier(max_depth=6, random_state=42))
])
dt.fit(X_train, y_train)

pred_dt = dt.predict(X_test)
print("Accuracy (Decision Tree):", accuracy_score(y_test, pred_dt))
print("\nClassification Report:\n", classification_report(y_test, pred_dt))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, pred_dt))

plt.figure(figsize=(12,6))
ohe = dt.named_steps['prep'].named_transformers_['cat']
cat_names = ohe.get_feature_names_out(cat_features)
feature_names = num_features + list(cat_names)
plot_tree(dt.named_steps['clf'], feature_names=feature_names, filled=True, max_depth=3, fontsize=6)
plt.show()
```

```
Accuracy (Decision Tree): 0.7989130434782609

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.92      0.88       309
           1       0.29      0.17      0.21        59

    accuracy                           0.80       368
   macro avg       0.57      0.54      0.55       368
weighted avg       0.76      0.80      0.78       368


Confusion Matrix:
 [[284  25]
 [ 49  10]]
```




```
X_train_enc = dt.named_steps['prep'].fit_transform(X_train)
X_test_enc  = dt.named_steps['prep'].transform(X_test)
tree = DecisionTreeClassifier(max_depth=6, random_state=42).fit(X_train_enc, y_train)

explainer = shap.TreeExplainer(tree)
shap_values = explainer(X_test_enc, check_additivity=False)

shap.summary_plot(shap_values, X_test_enc, feature_names=feature_names, show=True)
```


Age
DailyRate