In [24]:

In [25]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import warnings
import os
from scipy.stats import chi2_contingency
warnings.filterwarnings('ignore')
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
plt.rcParams.update({'font.size': 12})
palette_colors = sns.color_palette("tab10")
color1 = palette_colors[0]
color2 = palette_colors[1]
colors = [color1, color2]
```

In [47]:
```
df = pd.read_csv(r"C:\Users\avani\Downloads\HR_DS.csv")
df.head(3)
```

Out[47]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2
1	49	No	Travel_Frequently	279	Research & Development	8	1
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2

3 rows × 35 columns

In [49]:
```
df = df.dropna(subset=['Department', 'MonthlyIncome', 'YearsAtCompany', 'YearsSince
```

In [51]:
```
categorical_cols = ['Department', 'Attrition', 'Gender', 'JobRole']
for col in categorical_cols:
    df[col] = df[col].astype('category')
```

In [54]:
```
numerical_cols = ['MonthlyIncome', 'YearsAtCompany', 'YearsSinceLastPromotion', 'Ag
for col in numerical_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

In [61]:
```
df.columns = df.columns.str.strip()
income_col = 'MonthlyIncome'
if income_col in df.columns:
    income_cap = df[income_col].quantile(0.99)
    df[income_col] = df[income_col].clip(upper=income_cap)
else:
    print(f" Column '{income_col}' not found in DataFrame.")
```

In [62]:
```
df.shape
```

Out[62]: (1470, 35)

In [63]:
```
df.duplicated().any()
```

Out[63]: False

In [64]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   category
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   category
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   category
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   category
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   float64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 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: category(4), float64(1), int64(25), object(5)
memory usage: 362.6+ KB
```

In [65]:
```
pd.options.display.max_rows = len(df)
pd.options.display.max_columns = len(df.columns)
```

```
df.describe()
```

Out[65]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	Employ
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1
mean	36.923810	802.485714	9.192517	2.912925	1.0	1
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	1
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1
max	60.000000	1499.000000	29.000000	5.000000	1.0	2

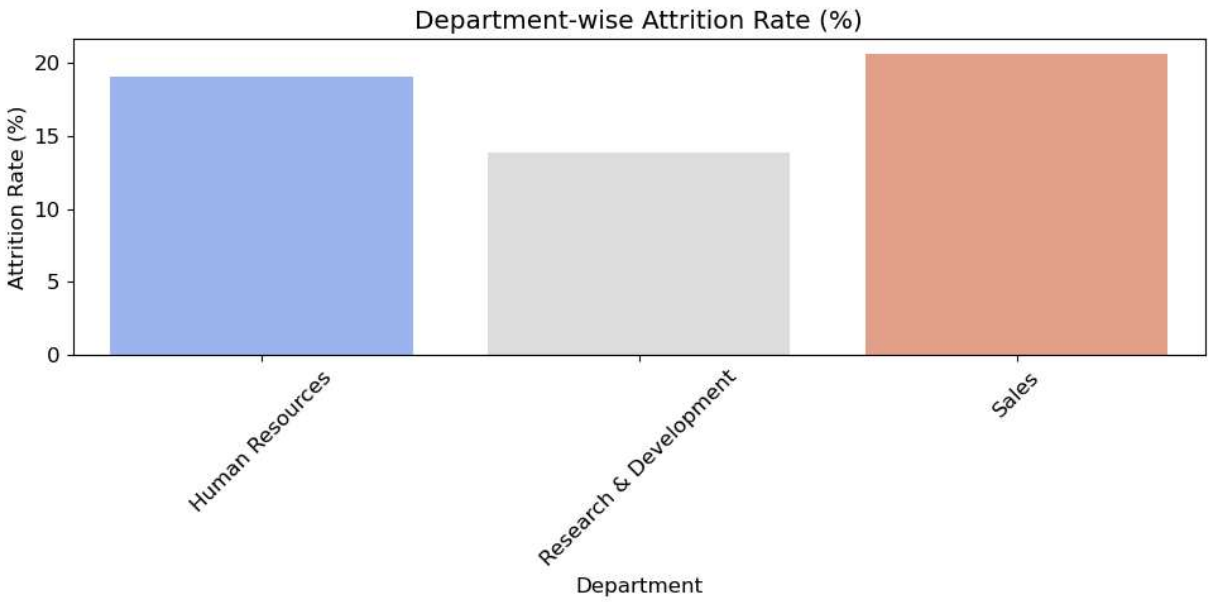
In [69]:
```
summary_stats = df.groupby('Department').agg({
    'MonthlyIncome': ['mean', 'median', 'std'],
    'YearsAtCompany': ['mean', 'median'],
    'YearsSinceLastPromotion': ['mean', 'median'],
    'Attrition': lambda x: (x == 'Yes').mean() * 100
}).round(2)

print("\nSummary Statistics by Department:")
print(summary_stats)
```

```
Summary Statistics by Department:
                        MonthlyIncome                  YearsAtCompany         \
                                 mean  median      std           mean median
Department
Human Resources              6652.40  3886.0  5783.92           7.24    5.0
Research & Development       6279.38  4374.0  4890.67           6.86    5.0
Sales                        6957.72  5754.5  4054.16           7.28    6.0

                        YearsSinceLastPromotion        Attrition
                                           mean median  <lambda>
Department
Human Resources                            1.78    1.0     19.05
Research & Development                     2.14    1.0     13.84
Sales                                      2.35    1.0     20.63
```

Department-wise Attrition Rate¶

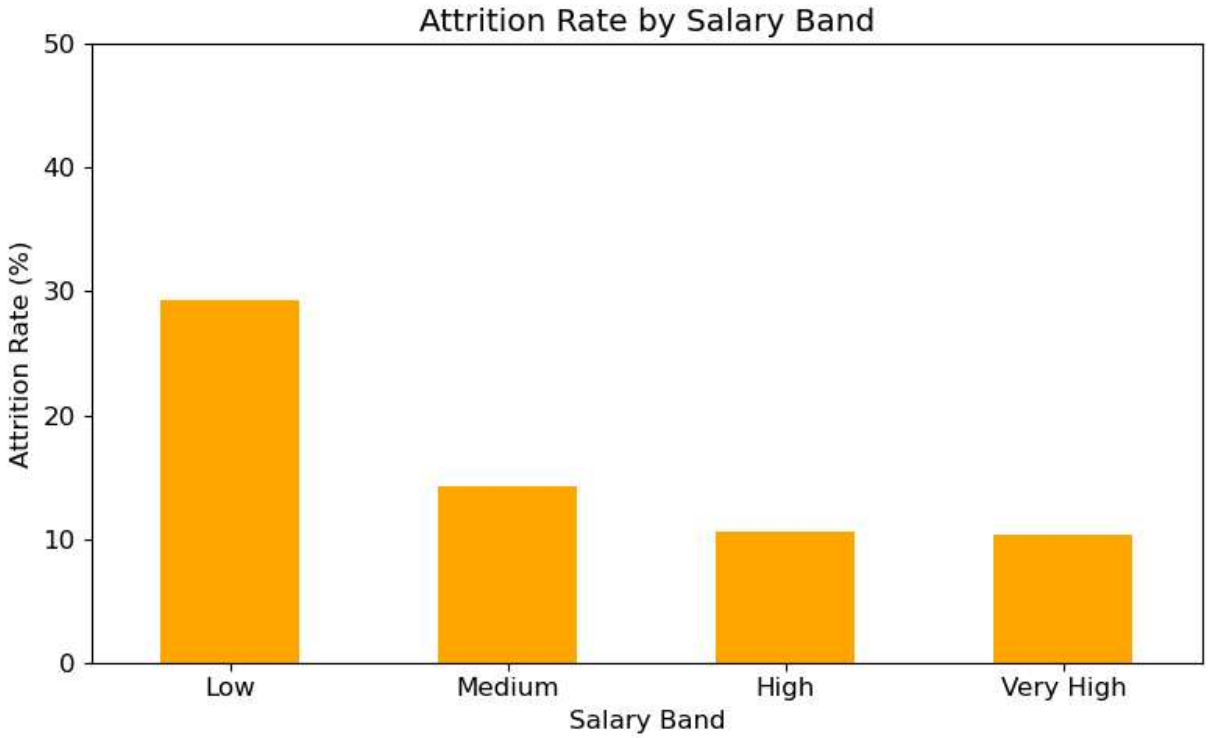
In [71]:
```
dept_attrition_counts = df[df['Attrition'] == 'Yes']['Department'].value_counts()
dept_total_counts = df['Department'].value_counts()
dept_attrition_rate = (dept_attrition_counts / dept_total_counts * 100).sort_values

plt.figure(figsize=(10, 5))
sns.barplot(x=dept_attrition_rate.index, y=dept_attrition_rate.values, palette='coo
plt.title('Department-wise Attrition Rate (%)')
```

```
plt.ylabel('Attrition Rate (%)')
plt.xlabel('Department')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



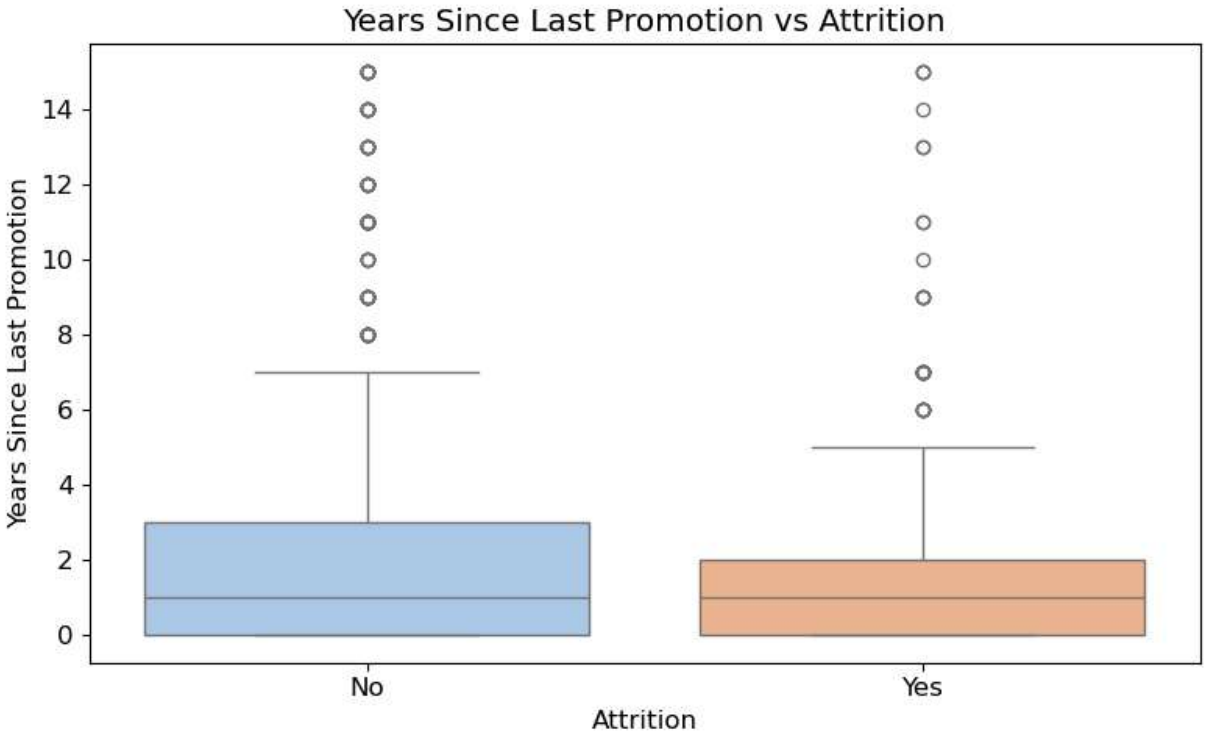
Salary Band vs Attrition

In [74]:
```
df.columns = df.columns.str.strip()
df['Salary Band'] = pd.qcut(df['MonthlyIncome'], q=4, labels=['Low', 'Medium', 'Hig
salary_attrition = pd.crosstab(df['Salary Band'], df['Attrition'], normalize='index
import matplotlib.pyplot as plt
salary_attrition['Yes'].plot(kind='bar', color='orange', figsize=(8, 5))
plt.title('Attrition Rate by Salary Band')
plt.ylabel('Attrition Rate (%)')
plt.xlabel('Salary Band')
plt.xticks(rotation=0)
plt.ylim(0, 50)
plt.tight_layout()
plt.show()
```



Promotions (Years Since Last Promotion vs Attrition)

In [77]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(8, 5))
sns.boxplot(x='Attrition', y='YearsSinceLastPromotion', data=df, palette='pastel')
plt.title("Years Since Last Promotion vs Attrition")
plt.xlabel("Attrition")
plt.ylabel("Years Since Last Promotion")
plt.tight_layout()
plt.show()
```



Result

In [79]:
```
df = pd.read_csv(r"C:\Users\avani\Downloads\HR_DS.csv")
df.head(3)
```

Out[79]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education
0	41	Yes	Travel_Rarely	1102	Sales	1	2
1	49	No	Travel_Frequently	279	Research & Development	8	1
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2

In [80]:
```
df.drop(['Employee Number', 'emp no'], axis=1, errors='ignore', inplace=True)
```

In [81]:
```
df['Attrition'] = df['Attrition'].map({'Yes': 1, 'No': 0})
```

In [82]:
```
label_encoder = LabelEncoder()
for col in df.select_dtypes(include='object').columns:
    df[col] = label_encoder.fit_transform(df[col])
```

In [84]:
```
X = df.drop('Attrition', axis=1)
y = df['Attrition']
```

In [88]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [90]:
```
from sklearn.model_selection import train_test_split
```

In [91]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, ran
```

In [93]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=2000)
model.fit(X_train, y_train)
```

Out[93]:

LogisticRegression

LogisticRegression(max_iter=2000)

In [95]:
```
y_pred = model.predict(X_test)
```

In [100…
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.8741496598639455

Confusion Matrix:
 [[239   8]
 [ 29  18]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.97      0.93       247
           1       0.69      0.38      0.49        47

    accuracy                           0.87       294
   macro avg       0.79      0.68      0.71       294
weighted avg       0.86      0.87      0.86       294
```

classification model (Logistic Regression)

In [103…
```
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.linear_model import LogisticRegression
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import confusion_matrix, roc_curve, auc, classification_report
```

Confusion Matrix and ROC Curve

In [14]:
```
df = pd.read_csv(r"C:\Users\avani\Downloads\HR_DS.csv")
```

In [125…
```
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", ax=ax1)
ax1.set_title("Confusion Matrix")
ax1.set_xlabel("Predicted")
ax1.set_ylabel("Expected")
```

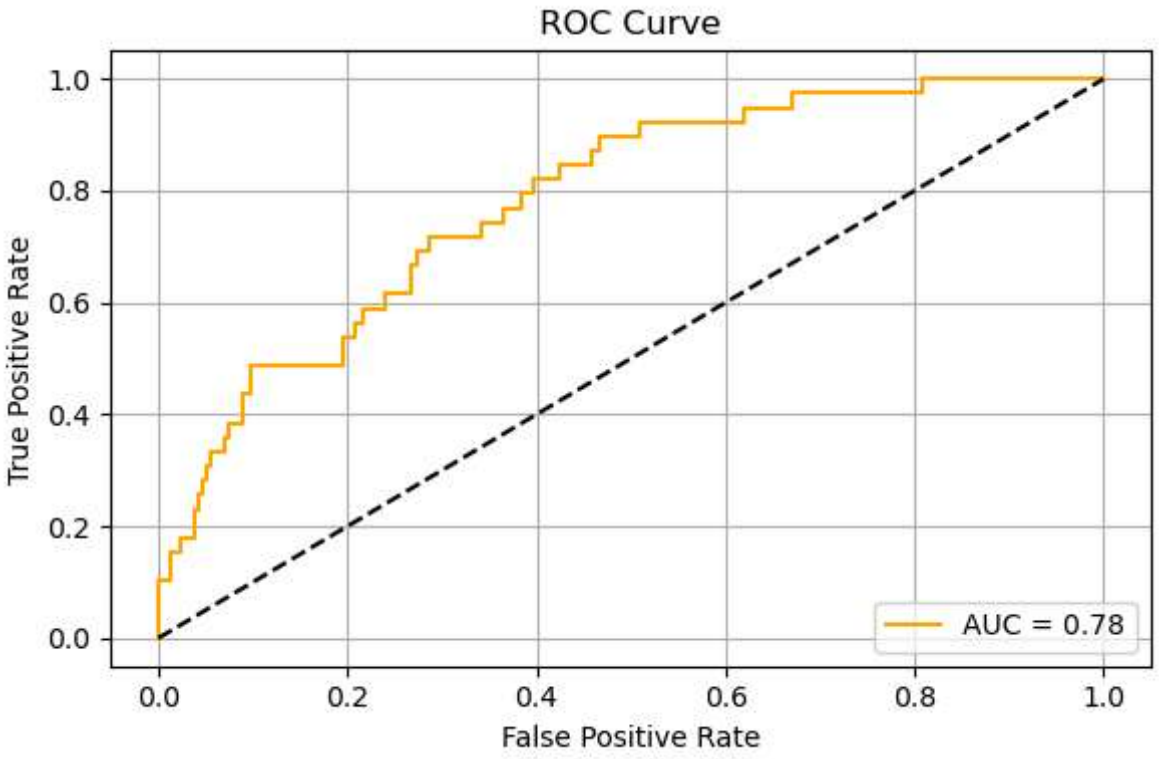
Out[125…
```
Text(144.72222222222223, 0.5, 'Expected')
```

In [19]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
```

In [20]:
```
y_probs = model.predict_proba(X_test)[:, 1]
```

In [21]:
```
fpr, tpr, _ = roc_curve(y_test, y_probs)
roc_auc = auc(fpr, tpr)
```

In [22]:
```
fig, ax2 = plt.subplots(figsize=(6, 4))
# Plot the ROC curve
ax2.plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}', color='orange')
ax2.plot([0, 1], [0, 1], 'k--')  # Diagonal line
ax2.set_title("ROC Curve")
ax2.set_xlabel("False Positive Rate")
ax2.set_ylabel("True Positive Rate")
ax2.legend(loc='lower right')
ax2.grid(True)
plt.tight_layout()
plt.show()
```



In [135…
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns

dt_model = DecisionTreeClassifier(max_depth=5, random_state=42)
dt_model.fit(X_train, y_train)
```

Out[135…
```
DecisionTreeClassifier(max_depth=5, random_state=42)
```

In [137…
```
y_pred = dt_model.predict(X_test)
y_prob = dt_model.predict_proba(X_test)[:, 1]
```

In [139…
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
```

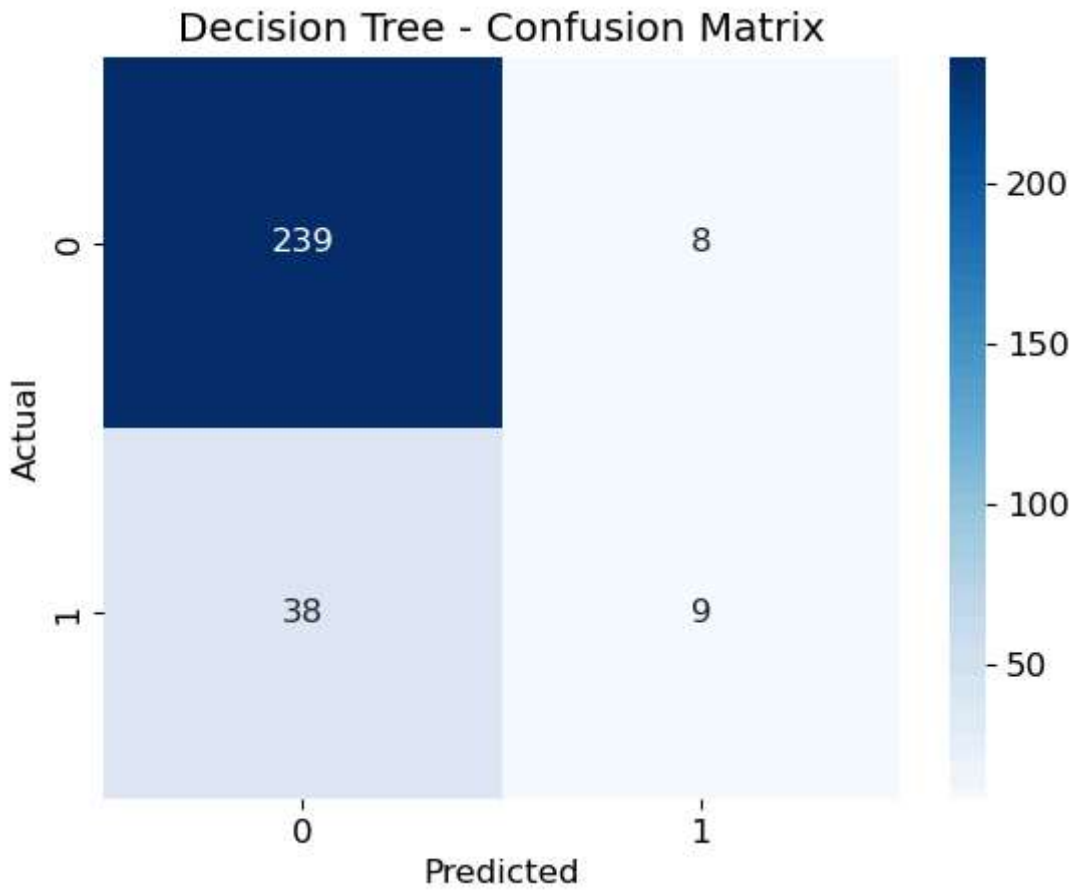
In [140…
```
dt_model = DecisionTreeClassifier(max_depth=5, random_state=42)
dt_model.fit(X_train, y_train)
```

Out[140…
```
DecisionTreeClassifier(max_depth=5, random_state=42)
```

In [141…
```
y_pred = dt_model.predict(X_test)
```

```
y_prob = dt_model.predict_proba(X_test)[:, 1]
```

In [143…]
```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Decision Tree - Confusion Matrix")
plt.show()
```



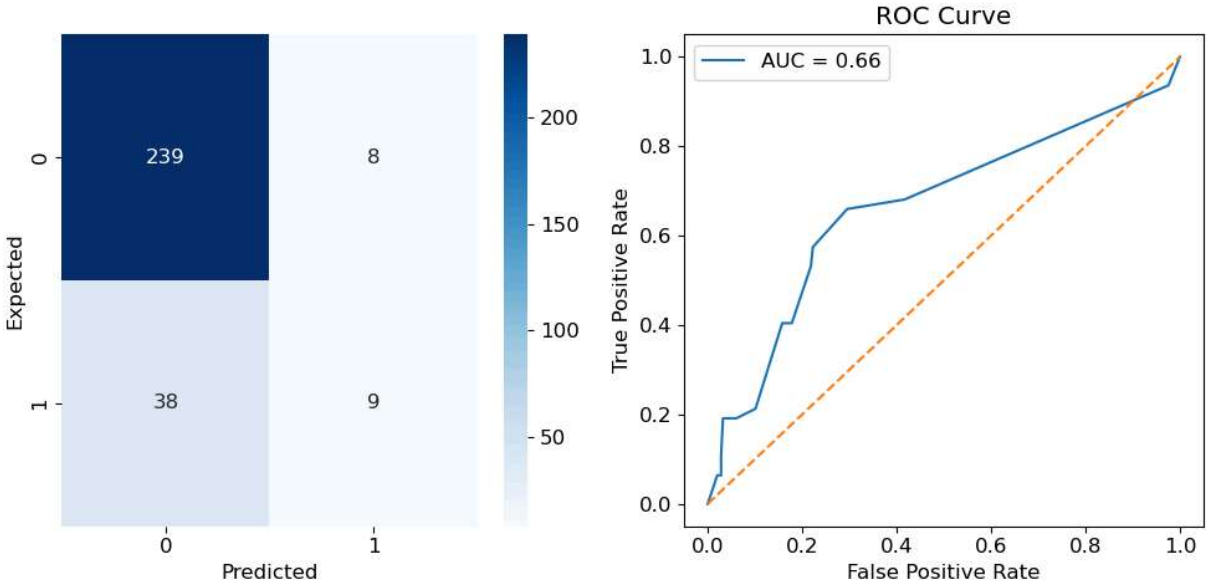
In [144…]
```
# Classification report
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.86      0.97      0.91       247
           1       0.53      0.19      0.28        47

    accuracy                           0.84       294
   macro avg       0.70      0.58      0.60       294
weighted avg       0.81      0.84      0.81       294
```

In [145…]
```
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
```

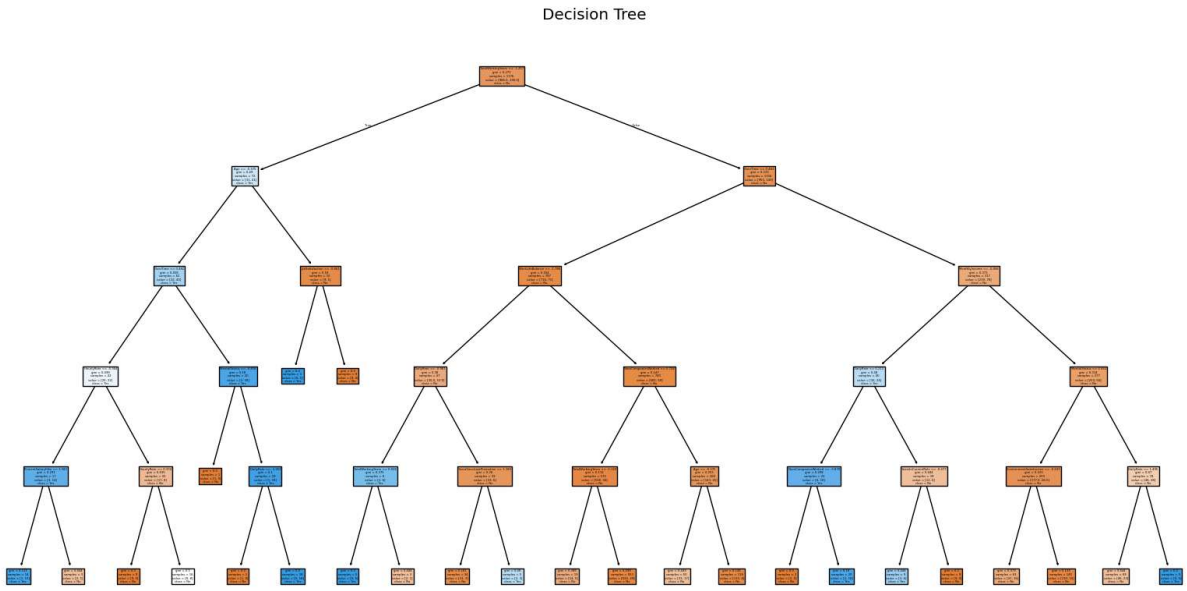
In [147…

```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Expected")
plt.subplot(1, 2, 2)
plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.2f}")
plt.plot([0, 1], [0, 1], linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.tight_layout()
plt.show()
```



In [149…

```
plt.figure(figsize=(20, 10))
plot_tree(dt_model, feature_names=X.columns, class_names=["No", "Yes"], filled=True
plt.title("Decision Tree")
plt.show()
```



SHAP value analysis

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from xgboost import XGBClassifier
import shap
import matplotlib.pyplot as plt
```

In [2]:
```
try:
    data = pd.read_csv(r"C:\Users\avani\Downloads\HR_DS.csv")
except FileNotFoundError:
    print("Error: File not found. Please check the file path and name.")
    exit()
except Exception as e:
    print(f"Error loading file: {e}")
    exit()
```

In [3]:
```
data = data.drop(columns=['emp no', 'Employee Number', 'Over18', 'Employee Count',
```

In [4]:
```
data['Attrition'] = data['Attrition'].map({'Yes': 1, 'No': 0})
```

In [5]:
```
X = data.drop(columns=['Attrition'])
y = data['Attrition']
```

In [6]:
```
for col in X.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [8]:
```
model = XGBClassifier(random_state=42, eval_metric='logloss')
model.fit(X_train, y_train)
```

Out[8]:

XGBClassifier

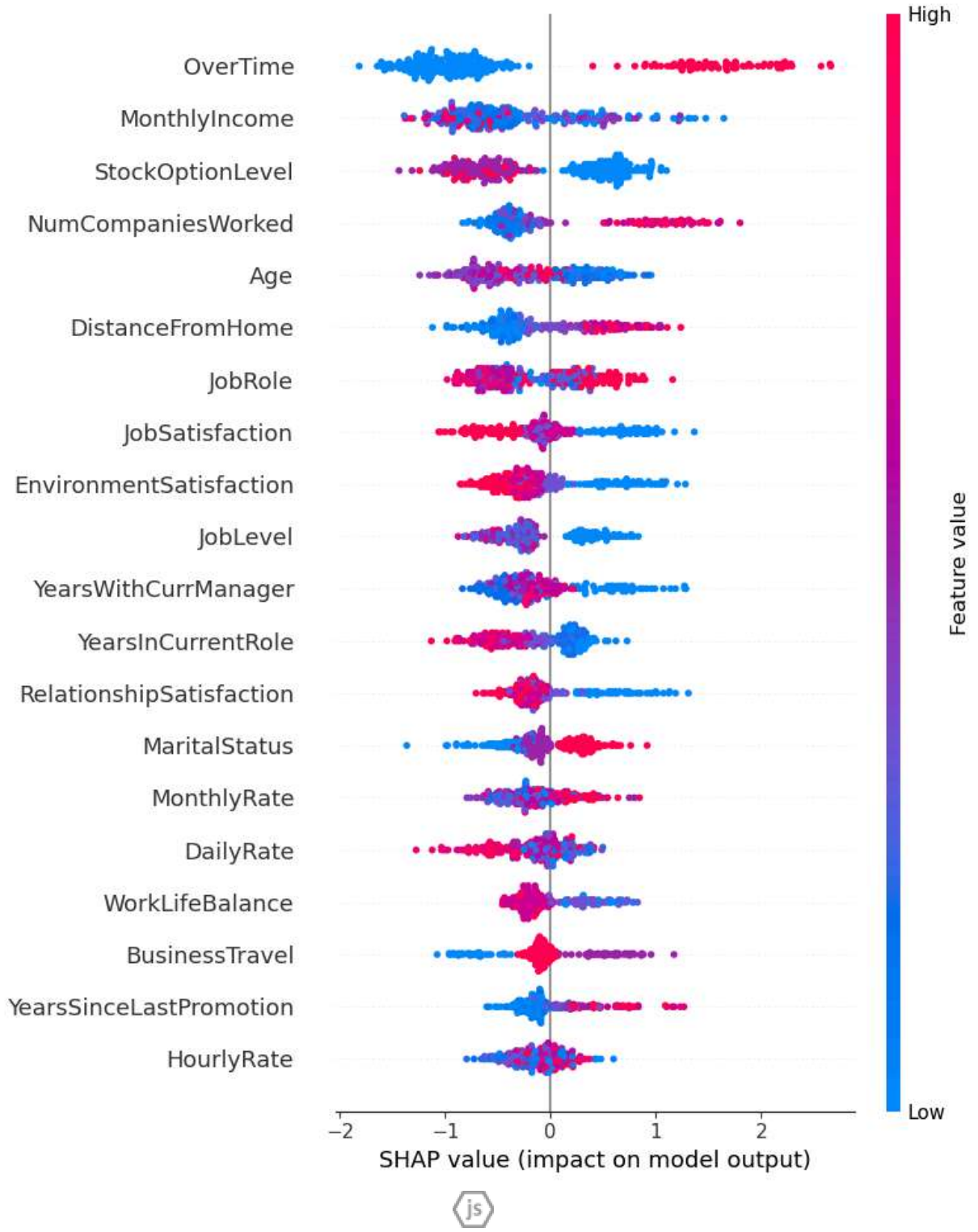
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds
=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, feature_weights=None, gamma=None,
              grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin
=None,
```

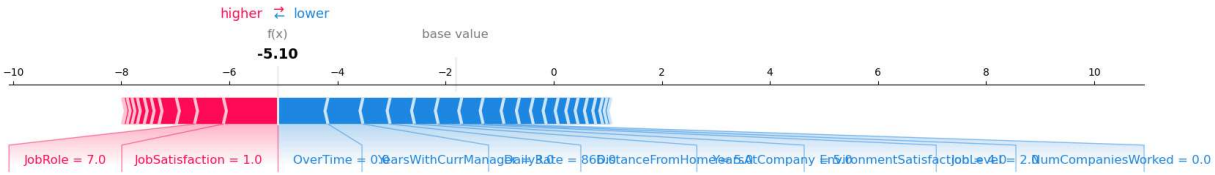
In [9]:
```
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)
```

In [10]:
```
shap_df = pd.DataFrame({'Feature': X_test.columns, 'SHAP Value': shap_values[0]})
print(shap_df.sort_values(by='SHAP Value', ascending=False))
```

	Feature	SHAP Value
15	JobSatisfaction	1.008890
14	JobRole	0.533935
16	MaritalStatus	0.335950
25	StockOptionLevel	0.320514
18	MonthlyRate	0.145573
27	TrainingTimesLastYear	0.128742
10	Gender	0.128037
11	HourlyRate	0.099076
0	Age	0.088067
3	Department	0.073713
5	Education	0.013795
22	PerformanceRating	0.000000
7	EmployeeCount	0.000000
24	StandardHours	0.000000
12	JobInvolvement	-0.072693
17	MonthlyIncome	-0.079976
30	YearsInCurrentRole	-0.128943
1	BusinessTravel	-0.154831
6	EducationField	-0.171549
21	PercentSalaryHike	-0.172695
28	WorkLifeBalance	-0.196053
26	TotalWorkingYears	-0.222691
23	RelationshipSatisfaction	-0.237364
8	EmployeeNumber	-0.257236
31	YearsSinceLastPromotion	-0.283193
19	NumCompaniesWorked	-0.386937
13	JobLevel	-0.396398
9	EnvironmentSatisfaction	-0.433106
29	YearsAtCompany	-0.440304
4	DistanceFromHome	-0.451310
2	DailyRate	-0.501992
32	YearsWithCurrManager	-0.647198
20	OverTime	-0.926470

In [11]:
```
plt.figure()
shap.summary_plot(shap_values, X_test)
plt.savefig('shap_summary.png')
plt.close()

shap.initjs()
shap.force_plot(explainer.expected_value, shap_values[0], X_test.iloc[0], matplotli
plt.savefig('shap_force.png')
plt.close()
```


OverTime
MonthlyIncome
StockOptionLevel
NumCompaniesWorked
Age
DistanceFromHome
JobRole
JobSatisfaction
EnvironmentSatisfaction
JobLevel
YearsWithCurrManager
YearsInCurrentRole
RelationshipSatisfaction
MaritalStatus
MonthlyRate
DailyRate
WorkLifeBalance
BusinessTravel
YearsSinceLastPromotion
HourlyRate
−2
−1
0
1
2
SHAP value (impact on model output)
High
Feature value
Low


higher ⇄ lower
f(x)
base value
-5.10
−10
−8
−6
−4
−2
0
2
4
6
8
10
JobRole = 7.0
JobSatisfaction = 1.0
OverTime = 0.0
YearsWithCurrManager = 8.0
DailyRate = 865.0
DistanceFromHome = 5.0
YearsAtCompany = 5.0
EnvironmentSatisfaction = 4.0
JobLevel = 2.0
NumCompaniesWorked = 0.0

In [ ]: