In [1]:
```
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, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_mat
import joblib

%matplotlib inline
sns.set(style="whitegrid")
```

In [3]:
```
FILE = "WA_Fn-UseC_-HR-Employee-Attrition.csv"
df = pd.read_csv(FILE)

print("Dataset shape:", df.shape)
df.head()
```

```
Dataset shape: (1470, 35)
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educat
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 [5]:
```
df.info()


print("\nMissing values per column:\n", df.isnull().sum())


print("\nAttrition value counts:\n", df['Attrition'].value_counts())
print("\nAttrition %:\n", df['Attrition'].value_counts(normalize=True)*100)
```

```
<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   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 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   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 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: int64(26), object(9)
memory usage: 402.1+ KB

Missing values per column:
 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

Attrition value counts:
 Attrition
No     1233
Yes     237
Name: count, dtype: int64

Attrition %:
 Attrition
No     83.877551
Yes    16.122449
Name: proportion, dtype: float64
```

In [7]:
```
single_val_cols = [c for c in df.columns if df[c].nunique() == 1]
print("Single-value columns (will drop):", single_val_cols)


to_drop = ['EmployeeNumber'] + single_val_cols
to_drop = [c for c in to_drop if c in df.columns]
df.drop(columns=to_drop, inplace=True)
print("After drop shape:", df.shape)
```

```
Single-value columns (will drop): ['EmployeeCount', 'Over18', 'StandardHours']
After drop shape: (1470, 31)
```

In [9]:
```
df['Attrition'] = df['Attrition'].map({'Yes':1, 'No':0})
df['Attrition'].value_counts()
```

Out[9]:
```
Attrition
0    1233
1     237
Name: count, dtype: int64
```

In [11]:
```
cat_cols = df.select_dtypes(include=['object']).columns.tolist()
print("Categorical columns:", cat_cols)


df_encoded = pd.get_dummies(df, columns=cat_cols, drop_first=True)
print("Encoded shape:", df_encoded.shape)
```

```
Categorical columns: ['BusinessTravel', 'Department', 'EducationField', 'Gender',
'JobRole', 'MaritalStatus', 'OverTime']
Encoded shape: (1470, 45)
```

In [13]:
```
X = df_encoded.drop('Attrition', axis=1)
y = df_encoded['Attrition']

print("X shape:", X.shape, "y shape:", y.shape)


print("Class distribution:\n", y.value_counts())
```

```
X shape: (1470, 44) y shape: (1470,)
Class distribution:
 Attrition
0    1233
1     237
Name: count, dtype: int64
```

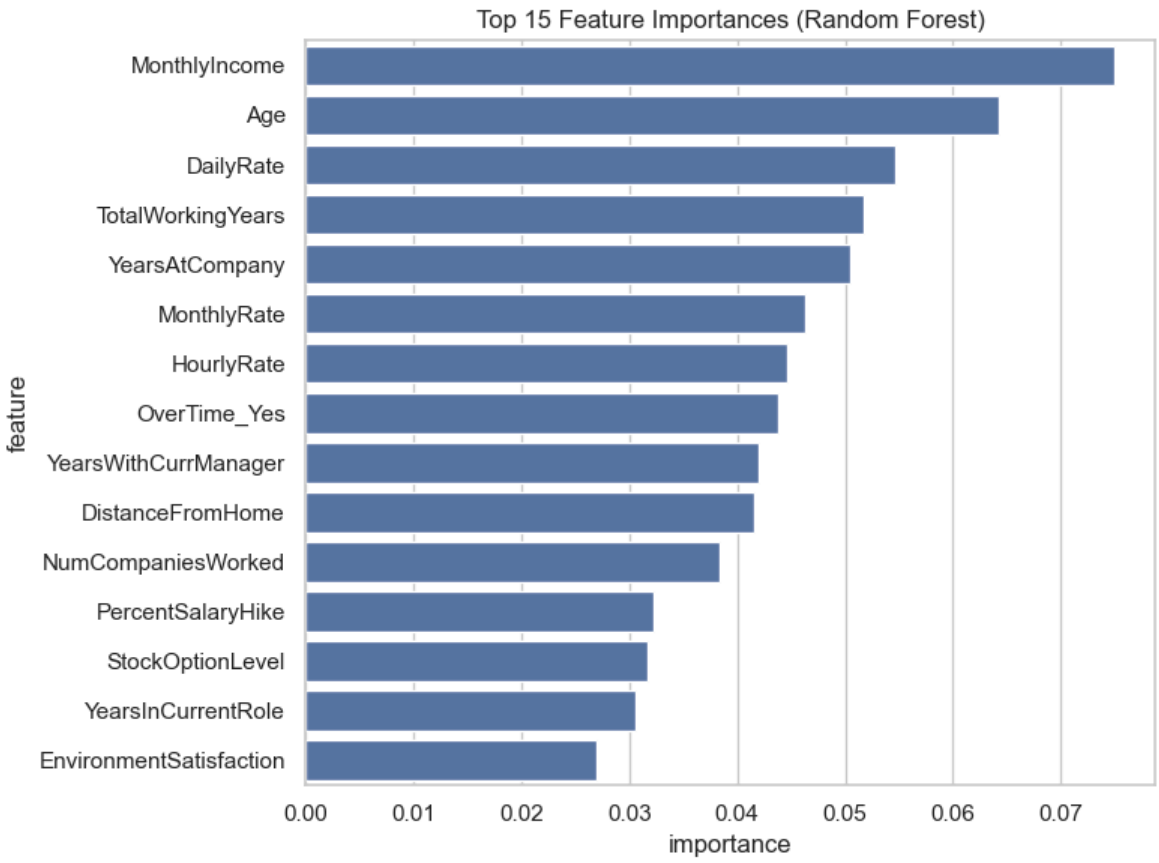
In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print("Train shape:", X_train.shape, "Test shape:", X_test.shape)
```

```
Train shape: (1176, 44) Test shape: (294, 44)
```

In [17]:
```
rf = RandomForestClassifier(n_estimators=200, random_state=42, class_weight='bal
rf.fit(X_train, y_train)

y_pred = rf.predict(X_test)
y_proba = rf.predict_proba(X_test)[:,1]

print("Accuracy:", round(accuracy_score(y_test, y_pred),4))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nROC AUC:", round(roc_auc_score(y_test, y_proba),4))


cv_scores = cross_val_score(rf, X_train, y_train, cv=5, scoring='roc_auc', n_job
print("\nCV ROC-AUC scores:", np.round(cv_scores,4), "mean:", round(cv_scores.me
```

```
Accuracy: 0.8367

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.98      0.91       247
           1       0.44      0.09      0.14        47

    accuracy                           0.84       294
   macro avg       0.65      0.53      0.53       294
weighted avg       0.78      0.84      0.79       294


Confusion Matrix:
 [[242   5]
 [ 43   4]]

ROC AUC: 0.7707

CV ROC-AUC scores: [0.6978 0.8601 0.8008 0.7827 0.81  ] mean: 0.7903
```

In [19]:
```
feat_imp = pd.DataFrame({
    'feature': X.columns,
    'importance': rf.feature_importances_
}).sort_values('importance', ascending=False)

top_n = 15
plt.figure(figsize=(8,6))
sns.barplot(data=feat_imp.head(top_n), x='importance', y='feature')
plt.title(f"Top {top_n} Feature Importances (Random Forest)")
plt.tight_layout()
plt.show()


feat_imp.head(top_n)
```

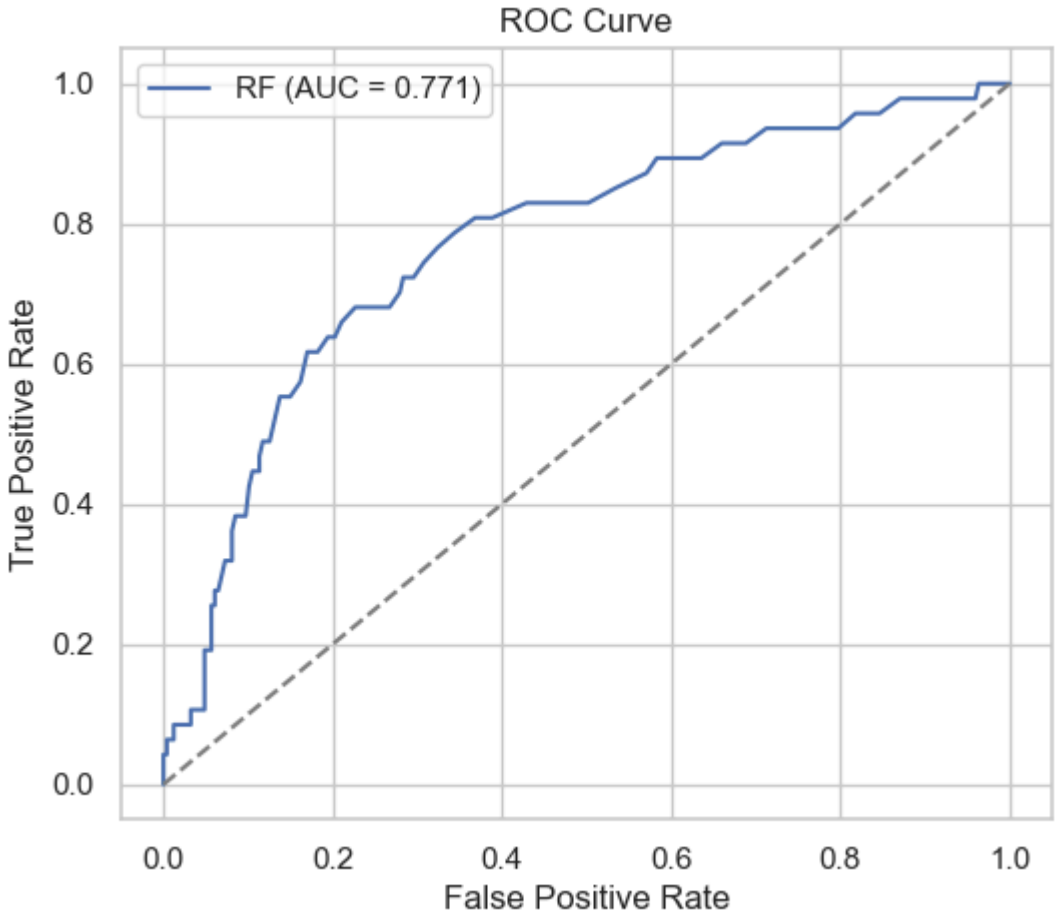


Out[19]:

	feature	importance
9	MonthlyIncome	0.074990
0	Age	0.064265
1	DailyRate	0.054629
16	TotalWorkingYears	0.051717
19	YearsAtCompany	0.050516
10	MonthlyRate	0.046254
5	HourlyRate	0.044628
43	OverTime_Yes	0.043875
22	YearsWithCurrManager	0.042047
2	DistanceFromHome	0.041556
11	NumCompaniesWorked	0.038319
12	PercentSalaryHike	0.032239
15	StockOptionLevel	0.031730
20	YearsInCurrentRole	0.030598
4	EnvironmentSatisfaction	0.027015

In [21]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_proba)
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f"RF (AUC = {roc_auc_score(y_test, y_proba):.3f})")
plt.plot([0,1],[0,1],'--', color='grey')
```

```
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.grid(True)
plt.show()
```



In [23]:
```
from sklearn.pipeline import make_pipeline

pipe = make_pipeline(StandardScaler(), LogisticRegression(max_iter=1000, class_w
pipe.fit(X_train, y_train)
y_pred_lr = pipe.predict(X_test)
y_proba_lr = pipe.predict_proba(X_test)[:,1]

print("Logistic Regression Accuracy:", round(accuracy_score(y_test, y_pred_lr),4
print("LR ROC-AUC:", round(roc_auc_score(y_test, y_proba_lr),4))
print("\nClassification Report (LR):\n", classification_report(y_test, y_pred_lr
```

```
Logistic Regression Accuracy: 0.7517
LR ROC-AUC: 0.7983

Classification Report (LR):
              precision    recall  f1-score   support

           0       0.91      0.78      0.84       247
           1       0.35      0.62      0.44        47

    accuracy                           0.75       294
   macro avg       0.63      0.70      0.64       294
weighted avg       0.82      0.75      0.78       294
```

In [25]:
```
joblib.dump(rf, 'rf_attrition_model.joblib')
print("Saved rf_attrition_model.joblib")


pd.Series(X.columns).to_csv('features_list.csv', index=False)
print("Saved features_list.csv")
```

```
Saved rf_attrition_model.joblib
Saved features_list.csv
```

In [ ]:
```

```