```python
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, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

#Step 1: Data Preprocessing
data = pd.read_csv("Employee-Attrition.csv")

# Explore the dataset
print(data.head())
print(data.info())
```

```
   Age Attrition     BusinessTravel  DailyRate              Department
\
0   41       Yes      Travel_Rarely       1102                   Sales

1   49        No  Travel_Frequently        279  Research & Development

2   37       Yes      Travel_Rarely       1373  Research & Development

3   33        No  Travel_Frequently       1392  Research & Development

4   27        No      Travel_Rarely        591  Research & Development


   DistanceFromHome  Education EducationField  EmployeeCount
EmployeeNumber  \
0                 1          2  Life Sciences              1
1
1                 8          1  Life Sciences              1
2
2                 2          2          Other              1
4
3                 3          4  Life Sciences              1
5
4                 2          1        Medical              1
7

   ...  RelationshipSatisfaction StandardHours  StockOptionLevel  \
0  ...                         1            80                 0
1  ...                         4            80                 1
2  ...                         2            80                 0
3  ...                         3            80                 0
```

```
4  ...                          4            80                   1

   TotalWorkingYears  TrainingTimesLastYear WorkLifeBalance
YearsAtCompany  \
0                  8                      0               1
6
1                 10                      3               3
10
2                  7                      3               3
0
3                  8                      3               3
8
4                  6                      3               3
2

  YearsInCurrentRole  YearsSinceLastPromotion  YearsWithCurrManager
0                  4                         0                     5
1                  7                         1                     7
2                  0                         0                     0
3                  7                         3                     0
4                  2                         2                     2

[5 rows x 35 columns]
<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
```

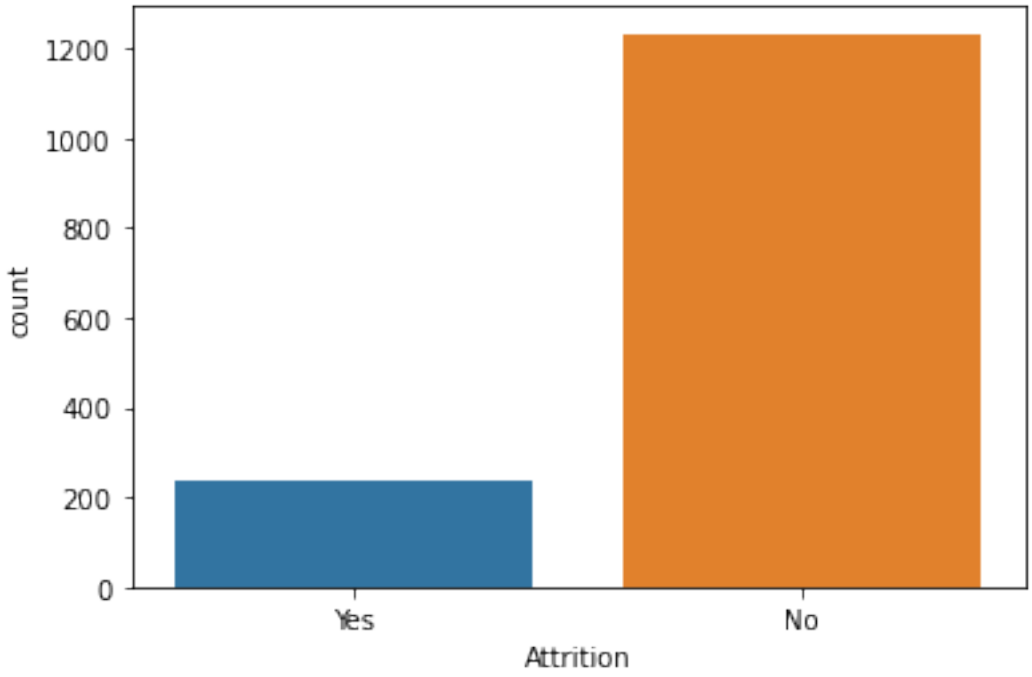
```
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          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
None
```

```
# Check for missing values
print(data.isnull().sum())

# Explore the dataset
print(data.describe())

# Data Visualization
sns.countplot(x='Attrition', data=data)
plt.show()
```

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

       EmployeeNumber  EnvironmentSatisfaction   HourlyRate
JobInvolvement  \
count     1470.000000              1470.000000  1470.000000
1470.000000
mean      1024.865306                 2.721769    65.891156
2.729932
std        602.024335                 1.093082    20.329428
0.711561
min          1.000000                 1.000000    30.000000
1.000000
25%        491.250000                 2.000000    48.000000
2.000000
50%       1020.500000                 3.000000    66.000000
3.000000
75%       1555.750000                 4.000000    83.750000
3.000000
max       2068.000000                 4.000000   100.000000
```

```
4.000000

          JobLevel  ...  RelationshipSatisfaction  StandardHours  \
count  1470.000000  ...               1470.000000         1470.0
mean      2.063946  ...                  2.712245           80.0
std       1.106940  ...                  1.081209            0.0
min       1.000000  ...                  1.000000           80.0
25%       1.000000  ...                  2.000000           80.0
50%       2.000000  ...                  3.000000           80.0
75%       3.000000  ...                  4.000000           80.0
max       5.000000  ...                  4.000000           80.0

       StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
count       1470.000000        1470.000000            1470.000000
mean           0.793878          11.279592               2.799320
std            0.852077           7.780782               1.289271
min            0.000000           0.000000               0.000000
25%            0.000000           6.000000               2.000000
50%            1.000000          10.000000               3.000000
75%            1.000000          15.000000               3.000000
max            3.000000          40.000000               6.000000

       WorkLifeBalance  YearsAtCompany  YearsInCurrentRole  \
count      1470.000000     1470.000000         1470.000000
mean          2.761224        7.008163            4.229252
std           0.706476        6.126525            3.623137
min           1.000000        0.000000            0.000000
25%           2.000000        3.000000            2.000000
50%           3.000000        5.000000            3.000000
75%           3.000000        9.000000            7.000000
max           4.000000       40.000000           18.000000

       YearsSinceLastPromotion  YearsWithCurrManager
count              1470.000000           1470.000000
mean                  2.187755              4.123129
std                   3.222430              3.568136
min                   0.000000              0.000000
25%                   0.000000              2.000000
50%                   1.000000              3.000000
75%                   3.000000              7.000000
max                  15.000000             17.000000

[8 rows x 26 columns]
```




```
# Encode categorical variables
label_encoder = LabelEncoder()
categorical_columns = data.select_dtypes(include=['object']).columns

for column in categorical_columns:
    data[column] = label_encoder.fit_transform(data[column])

# Split data into features (X) and target (y)
X = data.drop('Attrition', axis=1)
y = data['Attrition']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#Step 2: Model Building
# Initialize and train the logistic regression model
logistic_regression_model = LogisticRegression()
logistic_regression_model.fit(X_train, y_train)

LogisticRegression()

# Initialize and train the decision tree classifier
decision_tree_model = DecisionTreeClassifier()
decision_tree_model.fit(X_train, y_train)
```

```
DecisionTreeClassifier()

# Initialize and train the random forest classifier
random_forest_model = RandomForestClassifier()
random_forest_model.fit(X_train, y_train)

RandomForestClassifier()

#Step 3: Calculate Performance Metrics
# Predict using logistic regression model
y_pred_lr = logistic_regression_model.predict(X_test)

# Calculate accuracy
accuracy_lr = accuracy_score(y_test, y_pred_lr)

# Generate classification report
classification_report_lr = classification_report(y_test, y_pred_lr)

# Generate confusion matrix
confusion_matrix_lr = confusion_matrix(y_test, y_pred_lr)

# Predict using decision tree model
y_pred_dt = decision_tree_model.predict(X_test)

# Calculate accuracy
accuracy_dt = accuracy_score(y_test, y_pred_dt)

# Generate classification report
classification_report_dt = classification_report(y_test, y_pred_dt)

# Generate confusion matrix
confusion_matrix_dt = confusion_matrix(y_test, y_pred_dt)

# Predict using random forest model
y_pred_rf = random_forest_model.predict(X_test)

# Calculate accuracy
accuracy_rf = accuracy_score(y_test, y_pred_rf)

# Generate classification report
classification_report_rf = classification_report(y_test, y_pred_rf)

# Generate confusion matrix
confusion_matrix_rf = confusion_matrix(y_test, y_pred_rf)

#Step 4: Display Results
# Display results for Logistic Regression
print("Logistic Regression Metrics:")
print("Accuracy:", accuracy_lr)
print("Classification Report:\n", classification_report_lr)
print("Confusion Matrix:\n", confusion_matrix_lr)
```

```python
# Display results for Decision Tree
print("\nDecision Tree Metrics:")
print("Accuracy:", accuracy_dt)
print("Classification Report:\n", classification_report_dt)
print("Confusion Matrix:\n", confusion_matrix_dt)

# Display results for Random Forest
print("\nRandom Forest Metrics:")
print("Accuracy:", accuracy_rf)
print("Classification Report:\n", classification_report_rf)
print("Confusion Matrix:\n", confusion_matrix_rf)
```

```
Logistic Regression Metrics:
Accuracy: 0.891156462585034
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.98      0.94       255
           1       0.68      0.33      0.45        39

    accuracy                           0.89       294
   macro avg       0.79      0.65      0.69       294
weighted avg       0.88      0.89      0.87       294

Confusion Matrix:
 [[249   6]
 [ 26  13]]

Decision Tree Metrics:
Accuracy: 0.7857142857142857
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.88      0.88       255
           1       0.18      0.18      0.18        39

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

Confusion Matrix:
 [[224  31]
 [ 32   7]]

Random Forest Metrics:
Accuracy: 0.8775510204081632
Classification Report:
              precision    recall  f1-score   support

           0       0.88      1.00      0.93       255
```

```
           1       0.80      0.10      0.18        39

    accuracy                           0.88       294
   macro avg       0.84      0.55      0.56       294
weighted avg       0.87      0.88      0.83       294

Confusion Matrix:
 [[254   1]
 [ 35   4]]
```