Importing pandas

In [1]:
```
import pandas as pd
```

Loading the dataset into a DataFrame

In [2]:
```
df = pd.read_csv('Employee-Attrition.csv')
```

In [3]:
```
df
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
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	33	No	Travel_Frequently	1392	Research & Development	3	4	
4	27	No	Travel_Rarely	591	Research & Development	2	1	
...	...	...	...	...	...	...	...	...
1465	36	No	Travel_Frequently	884	Research & Development	23	2	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	
1468	49	No	Travel_Frequently	1023	Sales	2	3	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	

1470 rows × 35 columns

Data Preprocessing

Importing the required from scikit learn library

In [4]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_
```

Eliminating Null values

In [5]:
```
df = df.drop(['EmployeeNumber', 'EmployeeCount'], axis=1)
df = pd.get_dummies(df, columns=['Department', 'EducationField', 'Gender'], dr
non_numeric_columns = ['EmployeeNumber', 'EmployeeCount', 'Attrition', 'Busine
```

Split the data into features (X) and the target variable (y)

In [6]:
```
X = df.drop('Attrition', axis=1)
y = df['Attrition']
```

Split the data into training and testing sets

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

In [8]:
```
numeric_columns = [col for col in X_train.columns if col not in non_numeric_cc
```

Feature scaling (standardization)

In [9]:
```
scaler = StandardScaler()
X_train[numeric_columns] = scaler.fit_transform(X_train[numeric_columns])
X_test[numeric_columns] = scaler.transform(X_test[numeric_columns])
```

Logistic Regression Model

In [10]:
```
logistic_reg_model = LogisticRegression(random_state=42)
logistic_reg_model.fit(X_train[numeric_columns], y_train)
```

Out[10]:
```
LogisticRegression
LogisticRegression(random_state=42)
```

Make predictions

In [11]:
```
y_pred_logistic = logistic_reg_model.predict(X_test[numeric_columns])
```

Calculate performance metrics for Logistic Regression

Calculate Accuracy

In [12]:
```
accuracy_logistic = accuracy_score(y_test, y_pred_logistic)
print("Accuracy: " + str(round(accuracy_logistic, 2)))
```

```
Accuracy: 0.89
```

Calculate precision for yes and no class

In [13]:
```
precision_yes_logistic = precision_score(y_test, y_pred_logistic, pos_label='Y
print("Precision (Yes): " + str(round(precision_yes_logistic, 2)))

precision_no_logistic = precision_score(y_test, y_pred_logistic, pos_label='No
print("Precision (No): " + str(round(precision_no_logistic, 2)))
```

```
Precision (Yes): 0.83
Precision (No): 0.9
```

Calculate recall for yes and no class

In [14]:
```
recall_yes_logistic = recall_score(y_test, y_pred_logistic, pos_label='Yes')
print("Recall (Yes): " + str(round(recall_yes_logistic, 2)))

recall_no_logistic = recall_score(y_test, y_pred_logistic, pos_label='No')
print("Recall (No): " + str(round(recall_no_logistic, 2)))
```

```
Recall (Yes): 0.26
Recall (No): 0.99
```

Calculate f1 score for yes and no class

In [15]:
```
f1_score_yes_logistic = f1_score(y_test, y_pred_logistic, pos_label='Yes')
print("F1 Score (Yes): " + str(round(f1_score_yes_logistic, 2)))

f1_score_no_logistic = f1_score(y_test, y_pred_logistic, pos_label='No')
print("F1 Score (No): " + str(round(f1_score_no_logistic, 2)))
```

```
F1 Score (Yes): 0.39
F1 Score (No): 0.94
```

Confusion matrix

In [16]:
```
confusion_matrix_logistic = confusion_matrix(y_test, y_pred_logistic)
print("Confusion Matrix:")
print(confusion_matrix_logistic)
```

```
Confusion Matrix:
[[253   2]
 [ 29  10]]
```

Classification Report

In [17]:
```
print("\nClassification Report:")
print(classification_report(y_test, y_pred_logistic))
```

```
Classification Report:
              precision    recall  f1-score   support

          No       0.90      0.99      0.94       255
         Yes       0.83      0.26      0.39        39

    accuracy                           0.89       294
   macro avg       0.87      0.62      0.67       294
weighted avg       0.89      0.89      0.87       294
```

Decision Tree Model

In [18]:
```
decision_tree_model = DecisionTreeClassifier(random_state=42)
decision_tree_model.fit(X_train[numeric_columns], y_train)
```

Out[18]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

Make predictions

In [19]:
```
y_pred_tree = decision_tree_model.predict(X_test[numeric_columns])
```

Calculate performance metrics for Decision Tree

Calculate Accuracy

In [20]:
```
accuracy_tree = accuracy_score(y_test, y_pred_tree)
print("Accuracy: " + str(round(accuracy_tree, 2)))
```

```
Accuracy: 0.81
```

Calculate precision for yes and no class

In [21]:
```
precision_yes_tree = precision_score(y_test, y_pred_tree, pos_label='Yes')
print("Precision (Yes): " + str(round(precision_yes_tree, 2)))

precision_no_tree = precision_score(y_test, y_pred_tree, pos_label='No')
print("Precision (No): " + str(round(precision_no_tree, 2)))
```

```
Precision (Yes): 0.3
Precision (No): 0.9
```

Calculate recall for yes and no class

In [22]:
```
recall_yes_tree = recall_score(y_test, y_pred_tree, pos_label='Yes')
print("Recall (Yes): " + str(round(recall_yes_tree, 2)))

recall_no_tree = recall_score(y_test, y_pred_tree, pos_label='No')
print("Recall (No): " + str(round(recall_no_tree, 2)))
```

```
Recall (Yes): 0.33
Recall (No): 0.88
```

Calculate f1 score for yes and no class

In [23]:
```
f1_score_yes_tree = f1_score(y_test, y_pred_tree, pos_label='Yes')
print("F1 Score (Yes): " + str(round(f1_score_yes_tree, 2)))

f1_score_no_tree = f1_score(y_test, y_pred_tree, pos_label='No')
print("F1 Score (No): " + str(round(f1_score_no_tree, 2)))
```

```
F1 Score (Yes): 0.31
F1 Score (No): 0.89
```

Confusion matrix

In [24]:
```
confusion_matrix_tree = confusion_matrix(y_test, y_pred_tree)
print("Confusion Matrix:")
print(confusion_matrix_tree)
```

```
Confusion Matrix:
[[224  31]
 [ 26  13]]
```

Classification Report

In [25]:
```
print("\nClassification Report:")
print(classification_report(y_test, y_pred_tree))
```

```
Classification Report:
              precision    recall  f1-score   support

          No       0.90      0.88      0.89       255
         Yes       0.30      0.33      0.31        39

    accuracy                           0.81       294
   macro avg       0.60      0.61      0.60       294
weighted avg       0.82      0.81      0.81       294
```