```
1.Download the Employee Attrition Dataset
https://www.kaggle.com/datasets/patelprashant/employee-attrition
2.Perfrom Data Preprocessing
3.Model Building using Logistic Regression and Decision Tree and Random Forest
4.Calculate Performance metrics
```

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
dataset = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
```

In [6]:
```
dataset.shape
```

Out[6]: (1470, 35)

In [7]:
```
dataset.describe()
```

Out[7]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	Er
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	

8 rows × 26 columns

In [8]:
```
dataset.isnull().any()
```

Out[8]:
```
Age                         False
Attrition                   False
BusinessTravel              False
DailyRate                   False
Department                  False
DistanceFromHome            False
Education                   False
EducationField              False
EmployeeCount               False
EmployeeNumber              False
EnvironmentSatisfaction     False
Gender                      False
HourlyRate                  False
JobInvolvement              False
JobLevel                    False
JobRole                     False
JobSatisfaction             False
MaritalStatus               False
MonthlyIncome               False
```

In [9]:
```
#there is no null values to get the correlation we have to make sure about the r
dataset.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   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
```

In [10]:
```
dataset.nunique()
```

Out[10]:
```
Age                         43
Attrition                    2
BusinessTravel               3
DailyRate                  886
Department                   3
DistanceFromHome            29
Education                    5
EducationField               6
EmployeeCount                1
EmployeeNumber            1470
EnvironmentSatisfaction      4
Gender                       2
HourlyRate                  71
JobInvolvement               4
JobLevel                     5
JobRole                      9
JobSatisfaction              4
MaritalStatus                3
MonthlyIncome             1349
MonthlyRate               1427
NumCompaniesWorked          10
Over18                       1
OverTime                     2
PercentSalaryHike           15
PerformanceRating            2
RelationshipSatisfaction     4
StandardHours                1
StockOptionLevel             4
TotalWorkingYears           40
TrainingTimesLastYear        7
WorkLifeBalance              4
YearsAtCompany              37
YearsInCurrentRole          19
YearsSinceLastPromotion     16
YearsWithCurrManager        18
dtype: int64
```

```
Here EmployeeCount, Over18, StandardHours, EmployeeNumber we don't need these
for analysis
```

```
2 As the EMployeeCount is 1, and StandardHours, Over18 are same for every company
  and we don't need EmployeeNUmber because of indexes
```

In [11]:
```
1 dataset = dataset.drop(columns = ["EmployeeCount","Over18","EmployeeNumber","Sta
```

In [12]:
```
1 dataset.head()
```

Out[12]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationFiel
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Science
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Science
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Othe
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Science
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medica

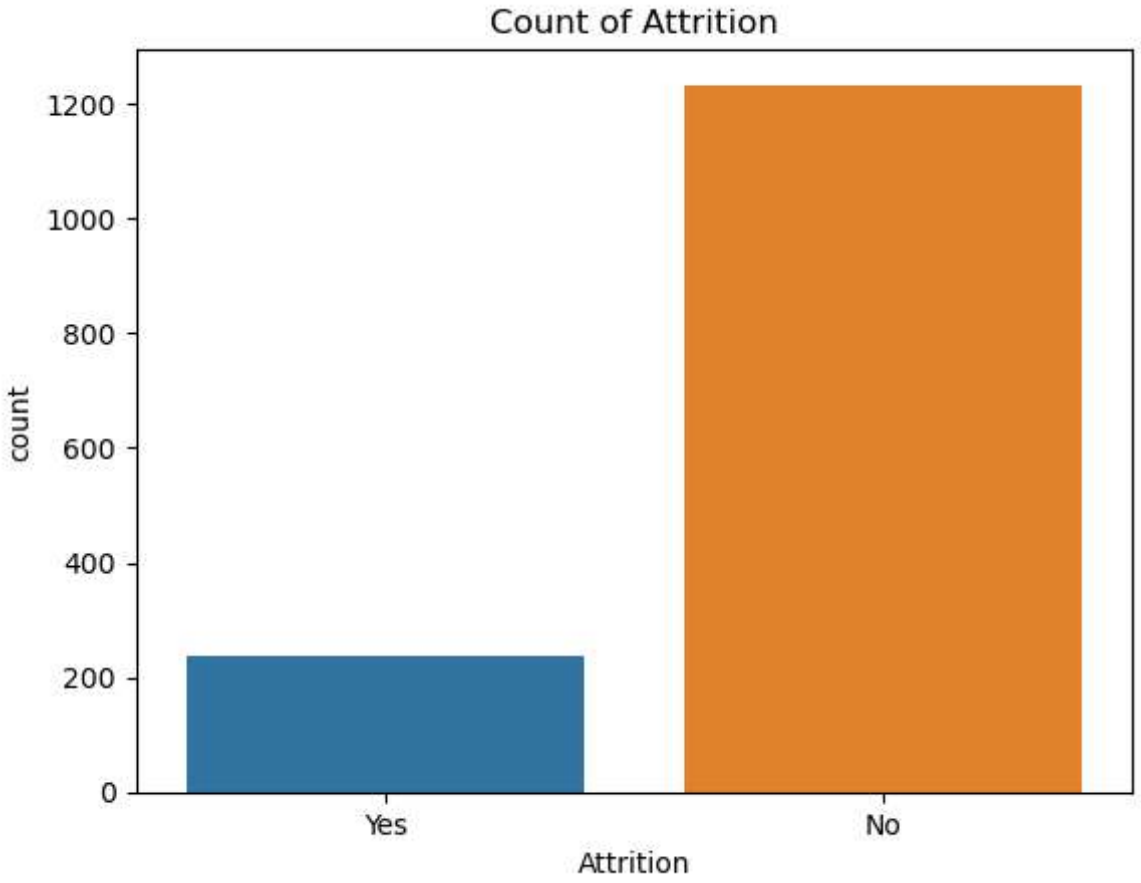
5 rows × 31 columns

```
1 numeric_columns =
  ['DailyRate','Age','DistanceFromHome','MonthlyIncome','MonthlyRate','PercentSal
  aryHike','TotalWorkingYears',
2                       'YearsAtCompany','NumCompaniesWorked','HourlyRate',
3
  'YearsInCurrentRole','YearsSinceLastPromotion','YearsWithCurrManager','Training
  TimesLastYear']
4
5 # Our output variable is Attrition: Which is a categorical Variable.
6 categorical_columns = ['Attrition','OverTime','BusinessTravel',
  'Department','Education',
7
  'EducationField','JobSatisfaction','EnvironmentSatisfaction','WorkLifeBalance',
8                         'StockOptionLevel','Gender', 'PerformanceRating',
  'JobInvolvement',
9                         'JobLevel', 'JobRole',
  'MaritalStatus','RelationshipSatisfaction']
```

Data Visualization

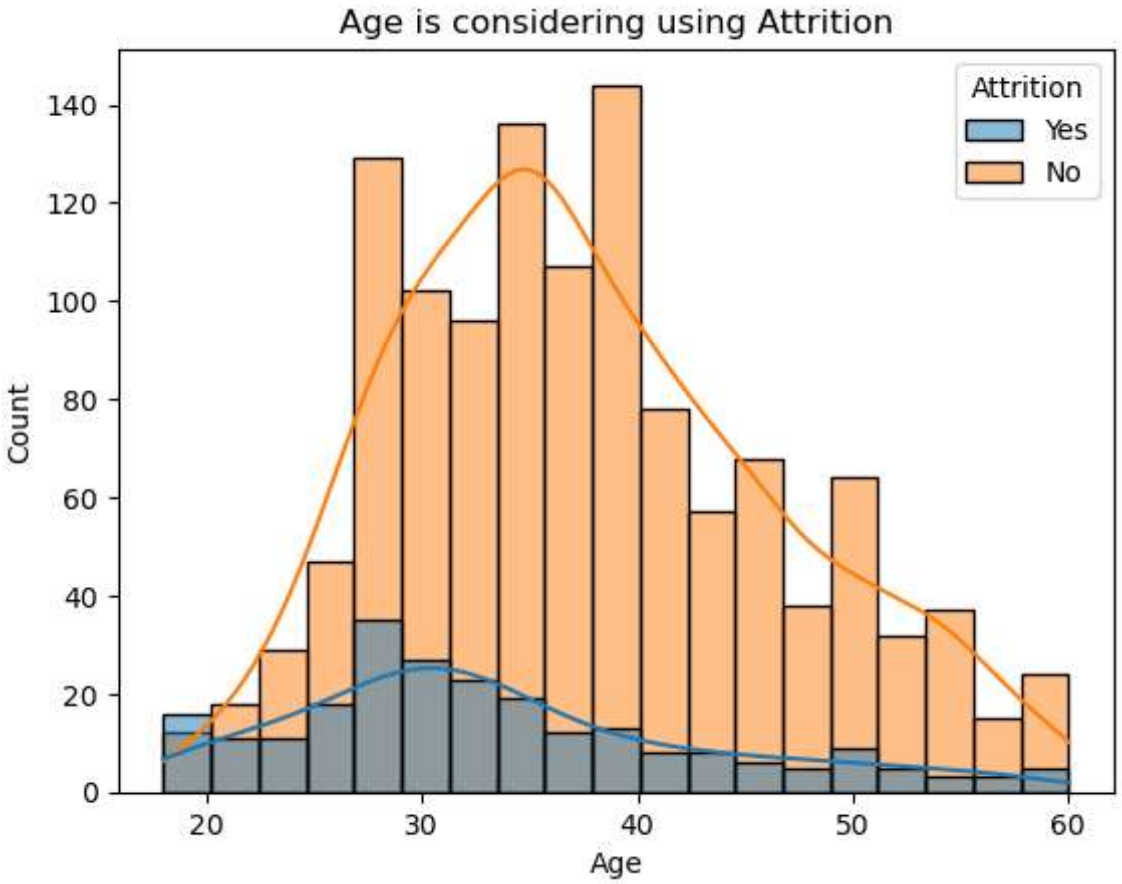
In [13]:
```
sns.countplot(x = "Attrition", data = dataset)
plt.title("Count of Attrition")
```

Out[13]: Text(0.5, 1.0, 'Count of Attrition')



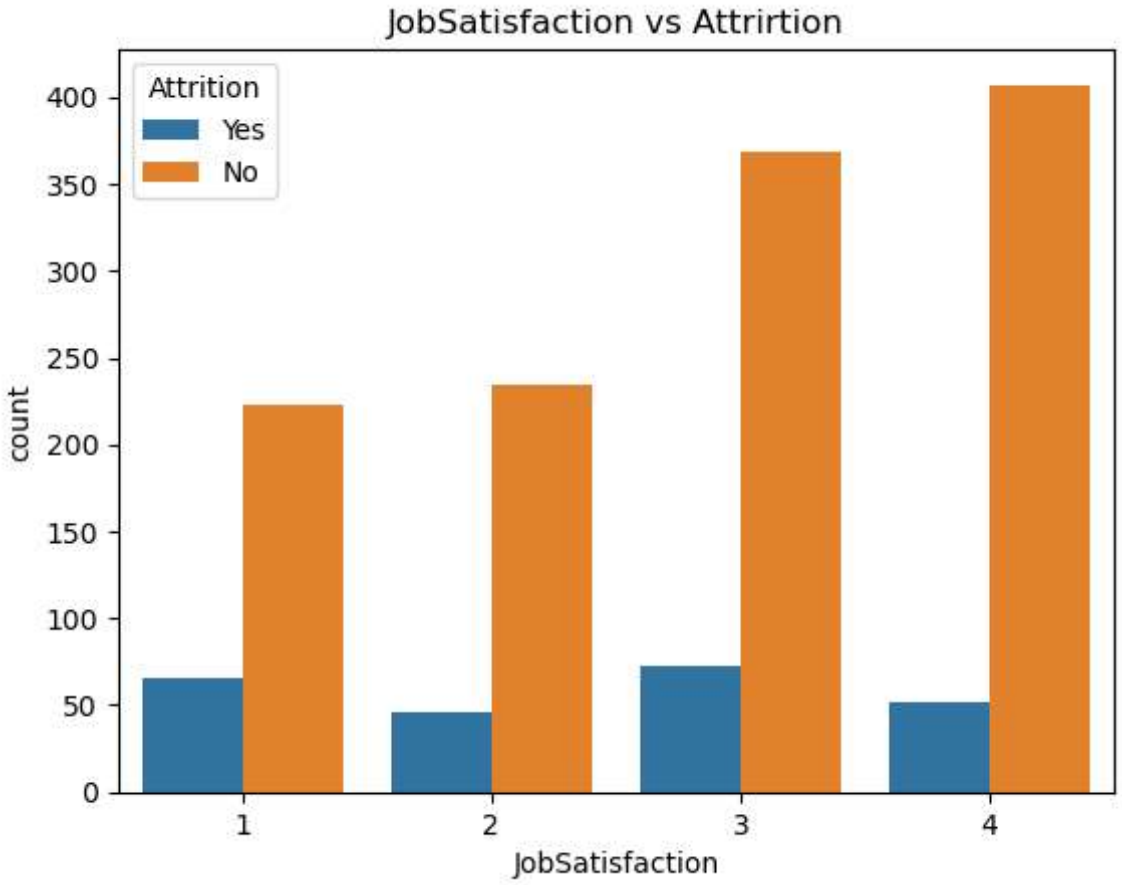
In [14]:
```
# checking the relation between AGe and Attrition
sns.histplot(data = dataset, x ='Age', hue = 'Attrition', kde = True)
plt.title('Age is considering using Attrition')
```

Out[14]: Text(0.5, 1.0, 'Age is considering using Attrition')



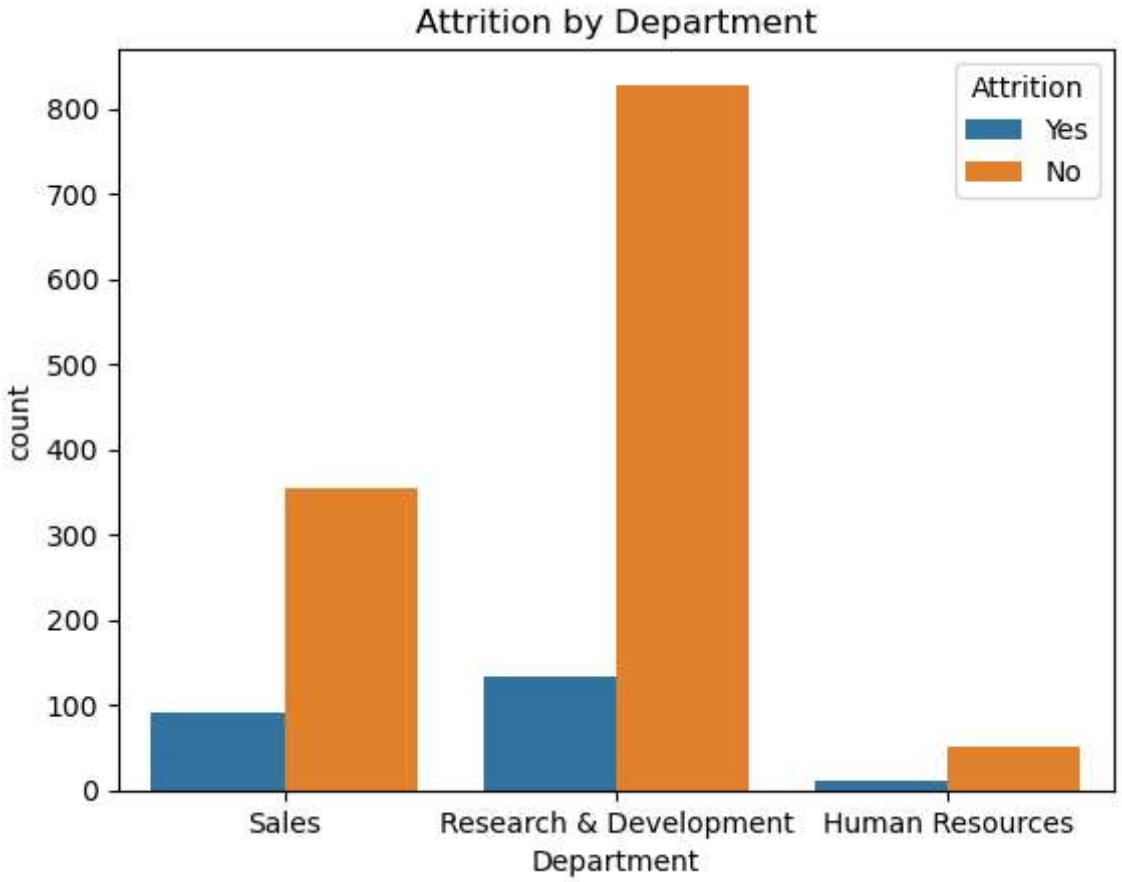
In [15]:
```
# for less count variables we are using countplots instead histplots
sns.countplot(x = 'JobSatisfaction', hue = 'Attrition', data = dataset)
plt.title('JobSatisfaction vs Attrirtion')

```

Out[15]:
```
Text(0.5, 1.0, 'JobSatisfaction vs Attrirtion')
```



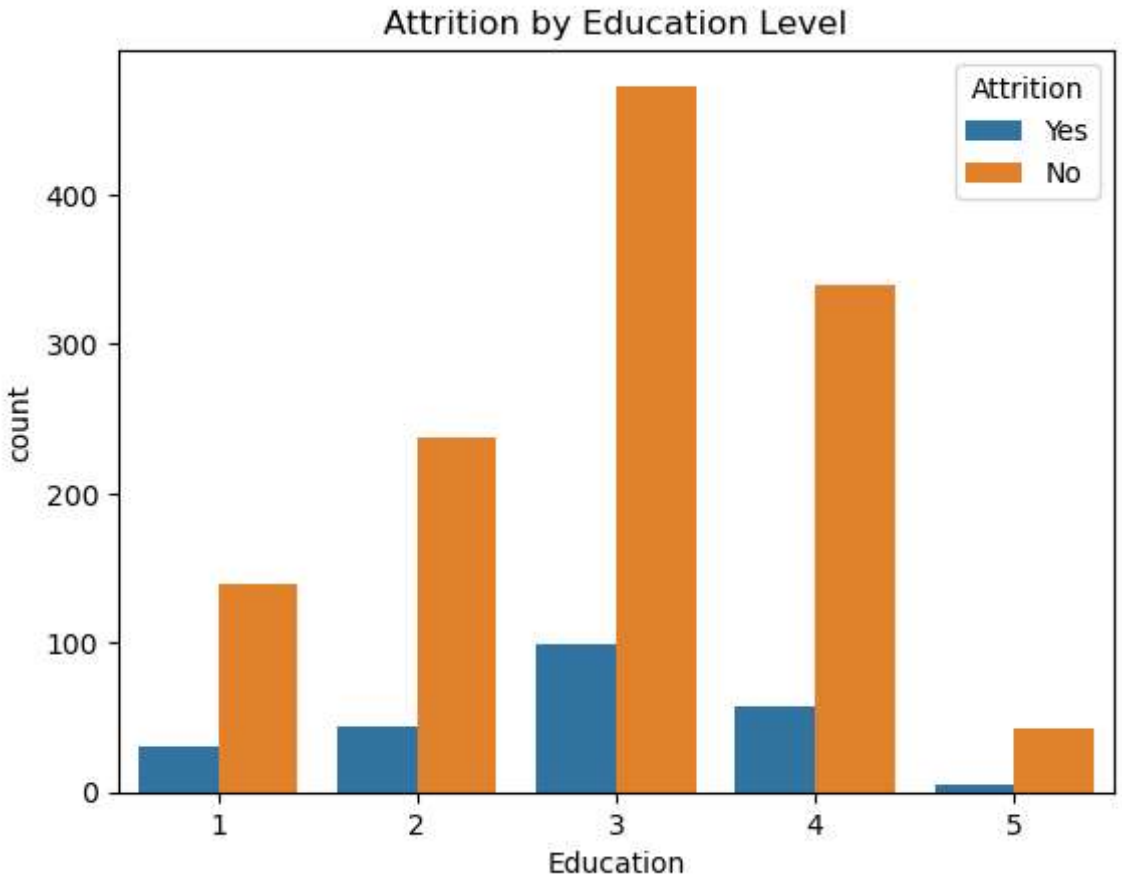
In [16]:
```
sns.countplot(x='Department', hue='Attrition', data=dataset)
plt.title('Attrition by Department')
```

Out[16]: Text(0.5, 1.0, 'Attrition by Department')



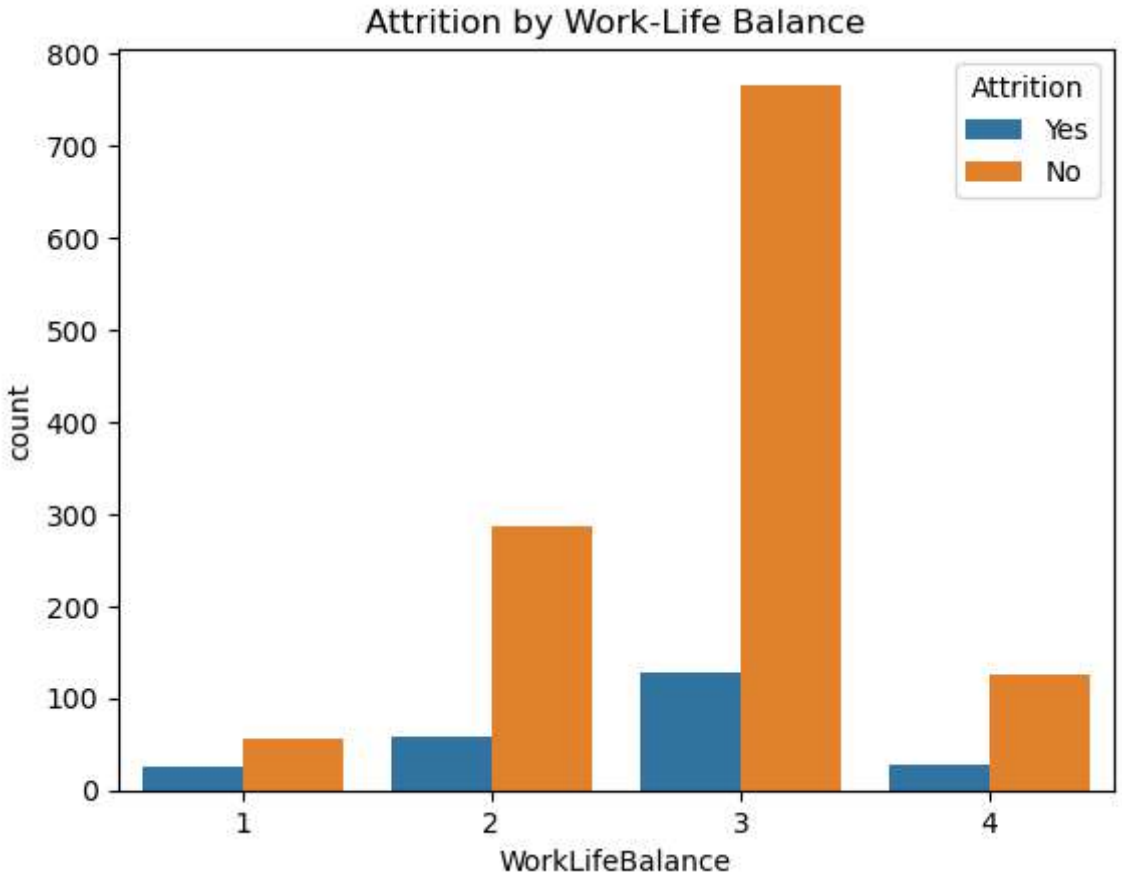
In [17]:
```
sns.countplot(x='Education', hue='Attrition', data=dataset)
plt.title('Attrition by Education Level')
```

Out[17]: Text(0.5, 1.0, 'Attrition by Education Level')



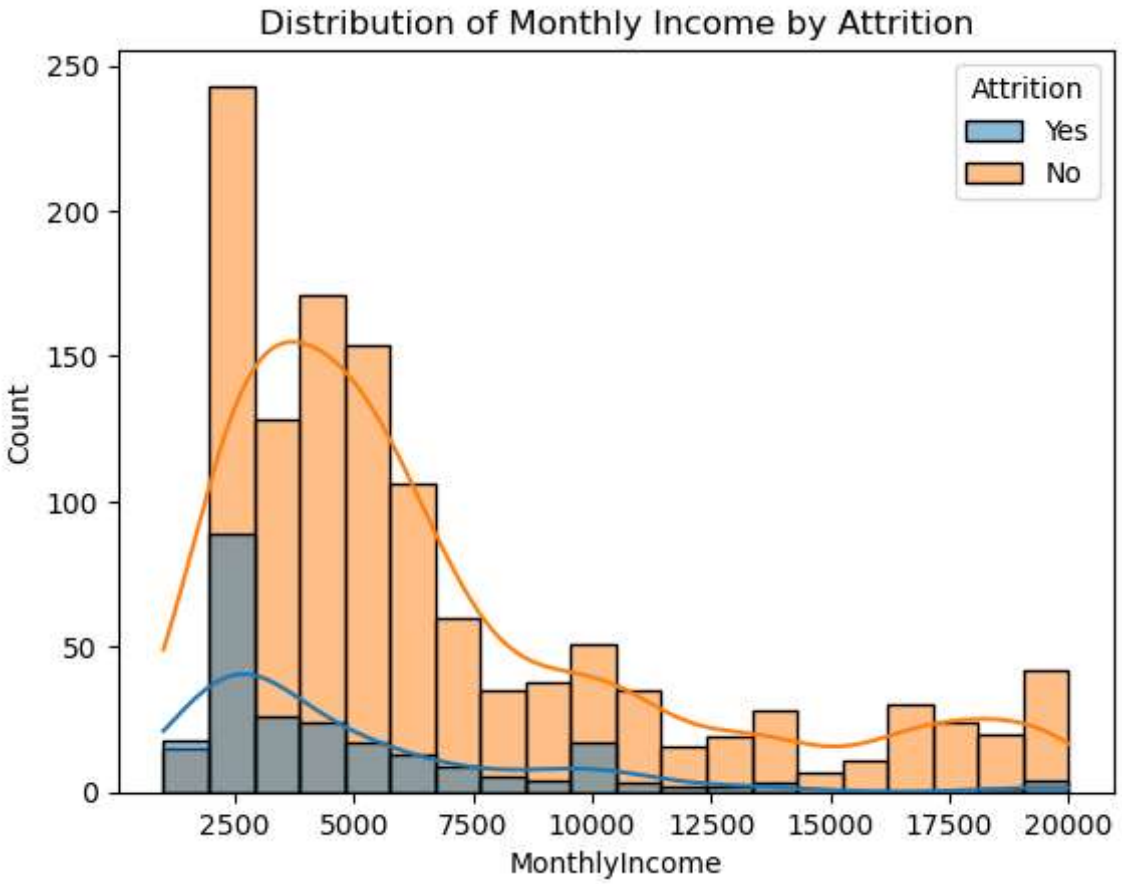
In [18]:
```
sns.countplot(x='WorkLifeBalance', hue='Attrition', data=dataset)
plt.title('Attrition by Work-Life Balance')
```

Out[18]: Text(0.5, 1.0, 'Attrition by Work-Life Balance')



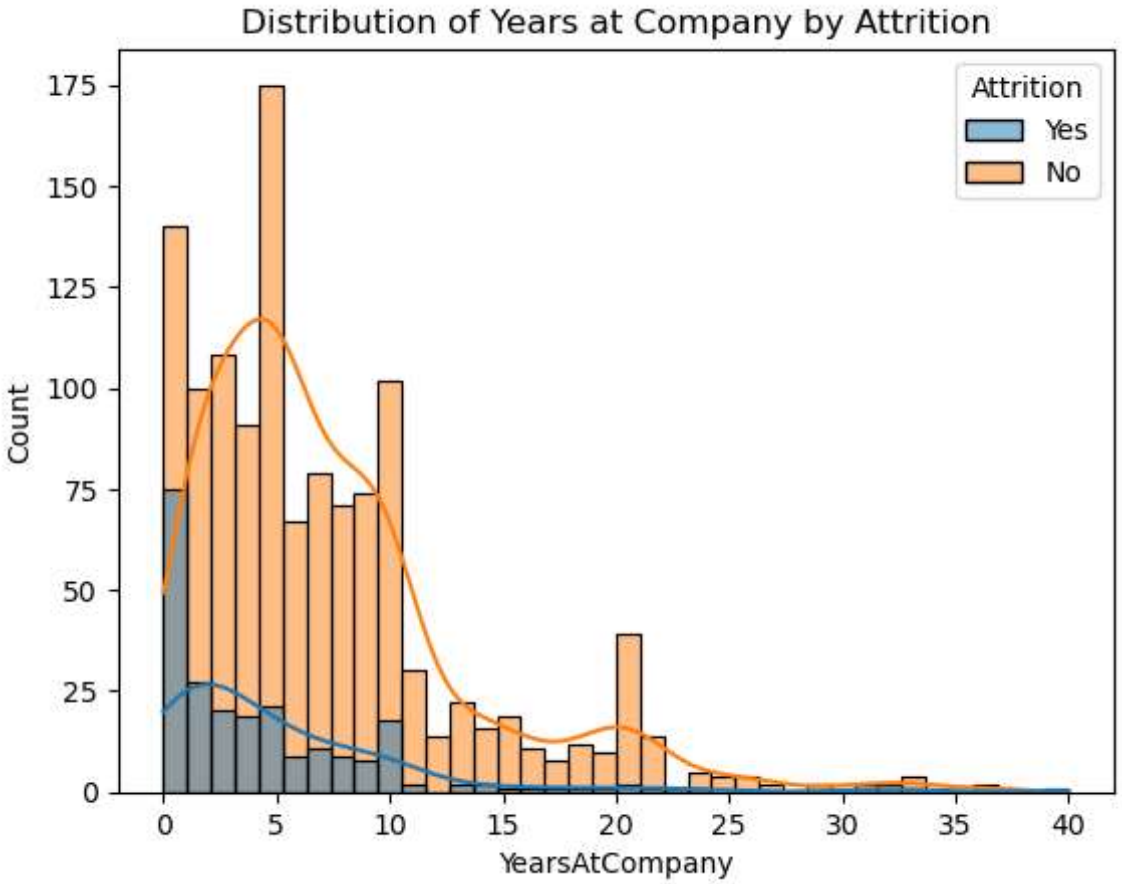
In [19]:
```
sns.histplot(data=dataset, x='MonthlyIncome', hue='Attrition', kde=True)
plt.title('Distribution of Monthly Income by Attrition')
```

Out[19]: Text(0.5, 1.0, 'Distribution of Monthly Income by Attrition')



In [20]:
```
sns.histplot(data=dataset, x='YearsAtCompany', hue='Attrition', kde=True)
plt.title('Distribution of Years at Company by Attrition')
```

Out[20]: Text(0.5, 1.0, 'Distribution of Years at Company by Attrition')

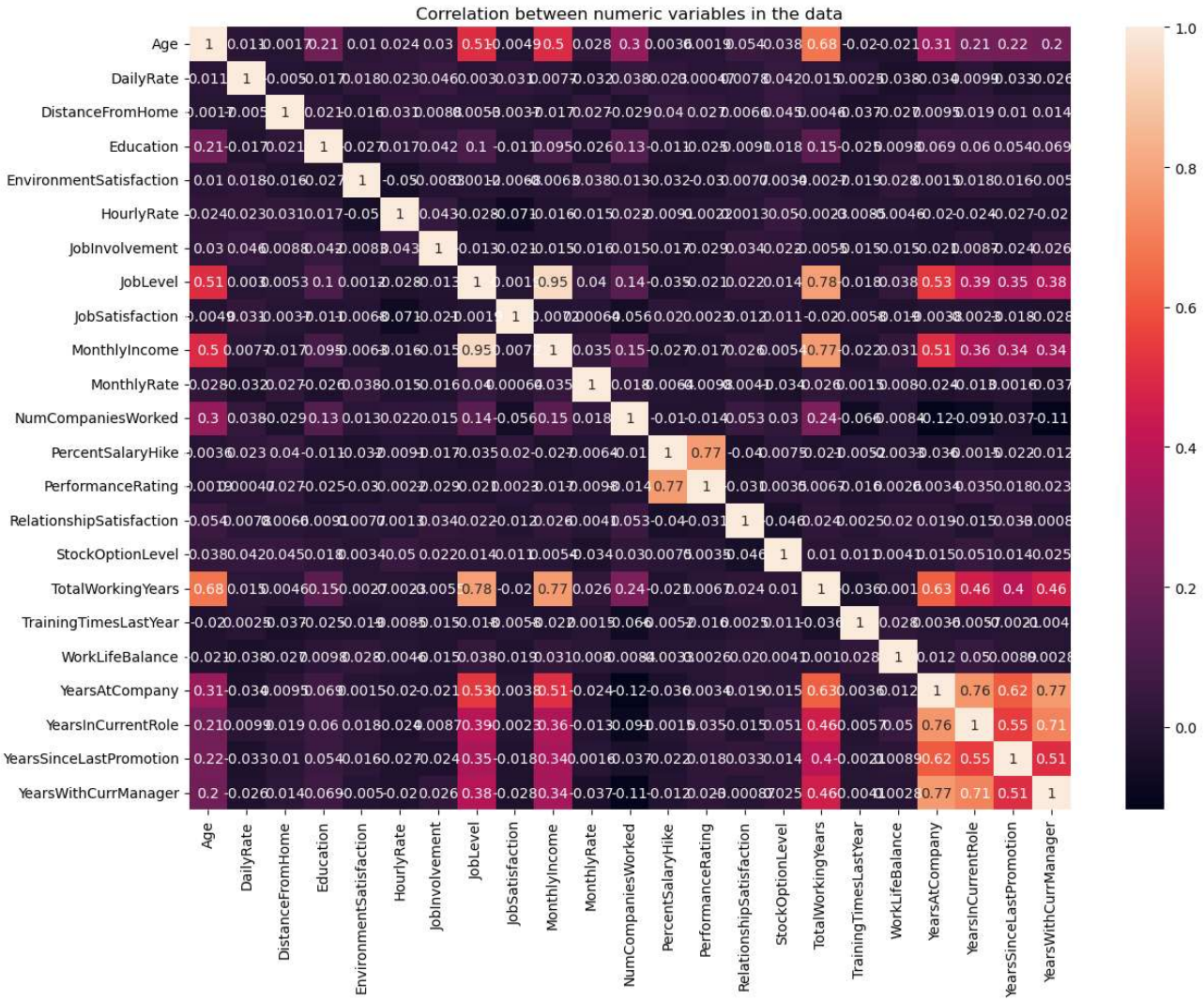


In [21]:

```
#Correlation Matrix (HEatmap)
correlation = dataset.corr(numeric_only = True)
plt.figure(figsize = (14,10))
sns.heatmap(correlation, annot =True)
plt.title('Correlation between numeric variables in the data')
```

Out[21]:

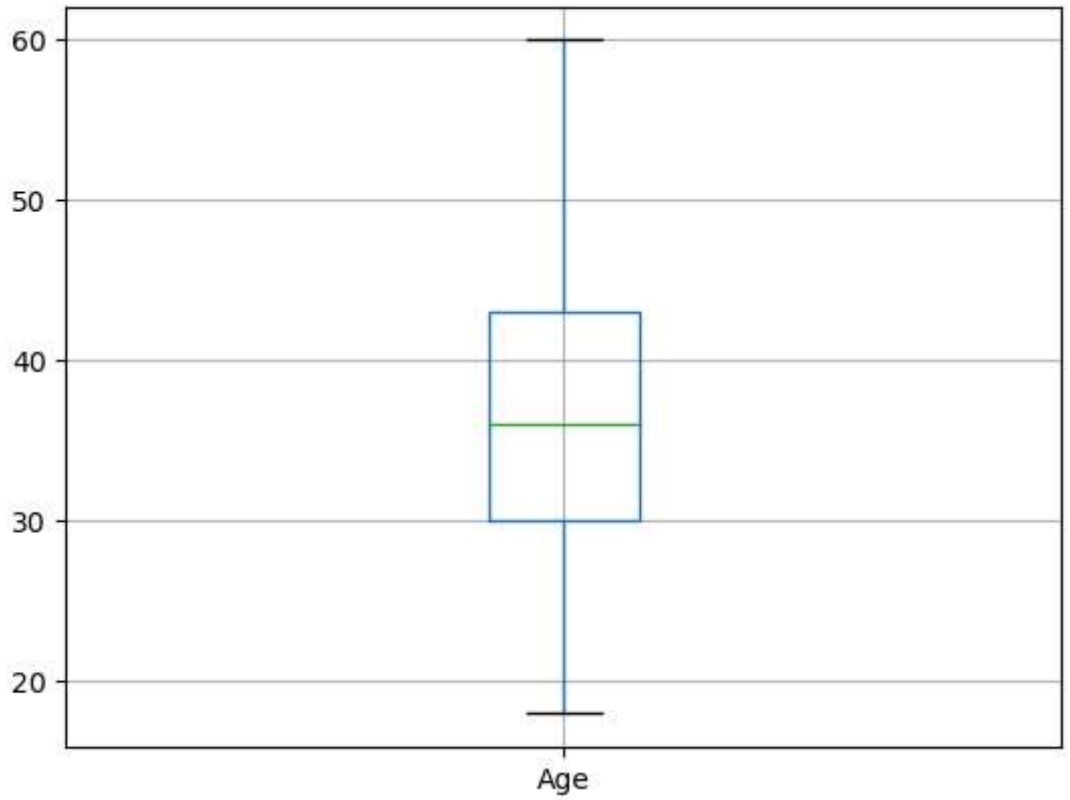
```
Text(0.5, 1.0, 'Correlation between numeric variables in the data')
```



Checking Outliers

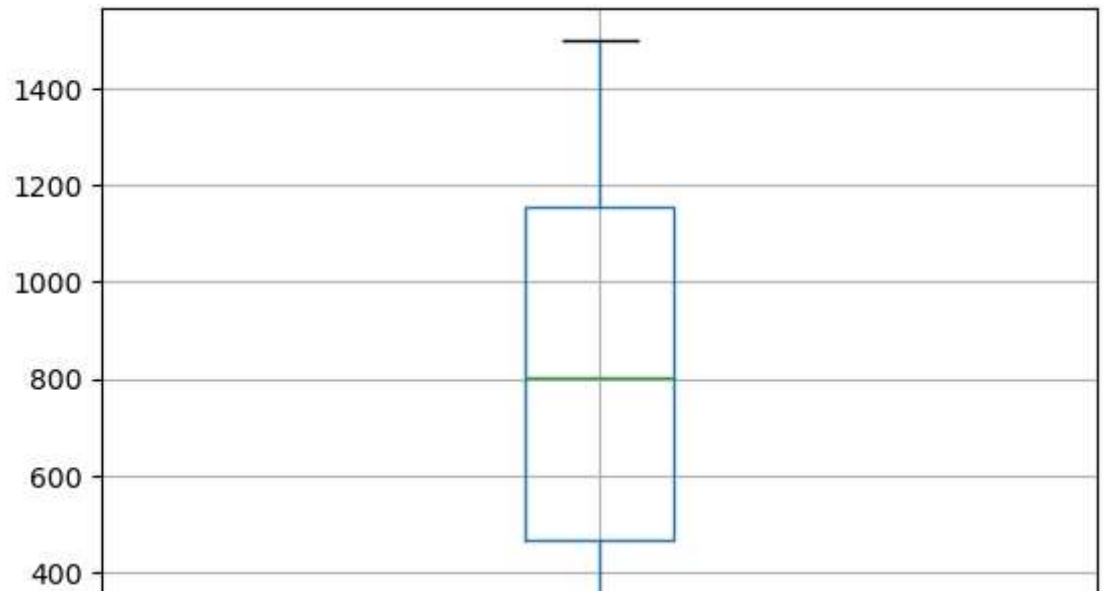
In [22]:
```
print(dataset.boxplot("Age"))

```

```
Axes(0.125,0.11;0.775x0.77)
```



In [23]:
```
print(dataset.boxplot("DailyRate"))
```

```
Axes(0.125,0.11;0.775x0.77)
```



In [41]:
```
# Define the target variable (dependent variable)
y = dataset['Attrition']
y = pd.DataFrame(y)
# Define the independent variables (features)
X = dataset.drop('Attrition', axis=1)
```

In [42]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y.Attrition = le.fit_transform(y.Attrition)
```

In [43]:
```
# Perform one-hot encoding on categorical columns
X_encoded = pd.get_dummies(X, drop_first=True)
X_encoded.head()
```

Out[43]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvement	J
0	41	1102	1	2	2	94	3	
1	49	279	8	1	3	61	2	
2	37	1373	2	2	4	92	2	
3	33	1392	3	4	4	56	3	
4	27	591	2	1	1	40	3	

5 rows × 44 columns

In [44]:
```
from sklearn.preprocessing import MinMaxScaler

# Initialize the scaler
scaler = MinMaxScaler()

# Fit and transform the scaled features
X_scaled = scaler.fit_transform(X_encoded)

# Convert the scaled features back to a DataFrame (optional)
X_scaled_df = pd.DataFrame(X_scaled, columns=X_encoded.columns)
```

In [45]:
```
X_scaled_df.head()
```

Out[45]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvem
0	0.547619	0.715820	0.000000	0.25	0.333333	0.914286	0.666
1	0.738095	0.126700	0.250000	0.00	0.666667	0.442857	0.333
2	0.452381	0.909807	0.035714	0.25	1.000000	0.885714	0.333
3	0.357143	0.923407	0.071429	0.75	1.000000	0.371429	0.666
4	0.214286	0.350036	0.035714	0.00	0.000000	0.142857	0.666

5 rows × 44 columns

In [46]:
```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets (e.g., 80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,

```

In [58]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, roc_curve
import joblib  # For model saving
```

In [59]:
```
# Initialize the models
logistic_model = LogisticRegression(random_state=42)
decision_tree_model = DecisionTreeClassifier(random_state=42)
random_forest_model = RandomForestClassifier(random_state=42)
```

In [60]:
```
# Training and testing the Logistic Regression model
logistic_model.fit(X_train, y_train)
logistic_predictions = logistic_model.predict(X_test)

# Training and testing the Decision Tree model
decision_tree_model.fit(X_train, y_train)
decision_tree_predictions = decision_tree_model.predict(X_test)

# Training and testing the Random Forest model
random_forest_model.fit(X_train, y_train)
random_forest_predictions = random_forest_model.predict(X_test)
```

```
C:\Users\pbalu\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataCo
nversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\pbalu\AppData\Local\Temp\ipykernel_3672\4153721992.py:10: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  random_forest_model.fit(X_train, y_train)
```

In [61]:
```
# Evaluation of Logistic Regression model
logistic_accuracy = accuracy_score(y_test, logistic_predictions)
logistic_report = classification_report(y_test, logistic_predictions)

print("Logistic Regression Model Accuracy:", logistic_accuracy)
print("Logistic Regression Model Classification Report:")
print(logistic_report)
```

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

           0       0.91      0.97      0.94       255
           1       0.67      0.36      0.47        39

    accuracy                           0.89       294
   macro avg       0.79      0.67      0.70       294
weighted avg       0.88      0.89      0.88       294
```

In [62]:

```
# Evaluation of Decision Tree model
decision_tree_accuracy = accuracy_score(y_test, decision_tree_predictions)
decision_tree_report = classification_report(y_test, decision_tree_predictions)

print("Decision Tree Model Accuracy:", decision_tree_accuracy)
print("Decision Tree Model Classification Report:")
print(decision_tree_report)
```

```
Decision Tree Model Accuracy: 0.7585034013605442
Decision Tree Model Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       255
           1       0.15      0.18      0.16        39

    accuracy                           0.76       294
   macro avg       0.51      0.51      0.51       294
weighted avg       0.78      0.76      0.77       294
```

In [63]:

```
# Evaluation of Random Forest model
random_forest_accuracy = accuracy_score(y_test, random_forest_predictions)
random_forest_report = classification_report(y_test, random_forest_predictions)

print("Random Forest Model Accuracy:", random_forest_accuracy)
print("Random Forest Model Classification Report:")
print(random_forest_report)
```

```
Random Forest Model Accuracy: 0.8775510204081632
Random Forest Model 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
```

In [64]:

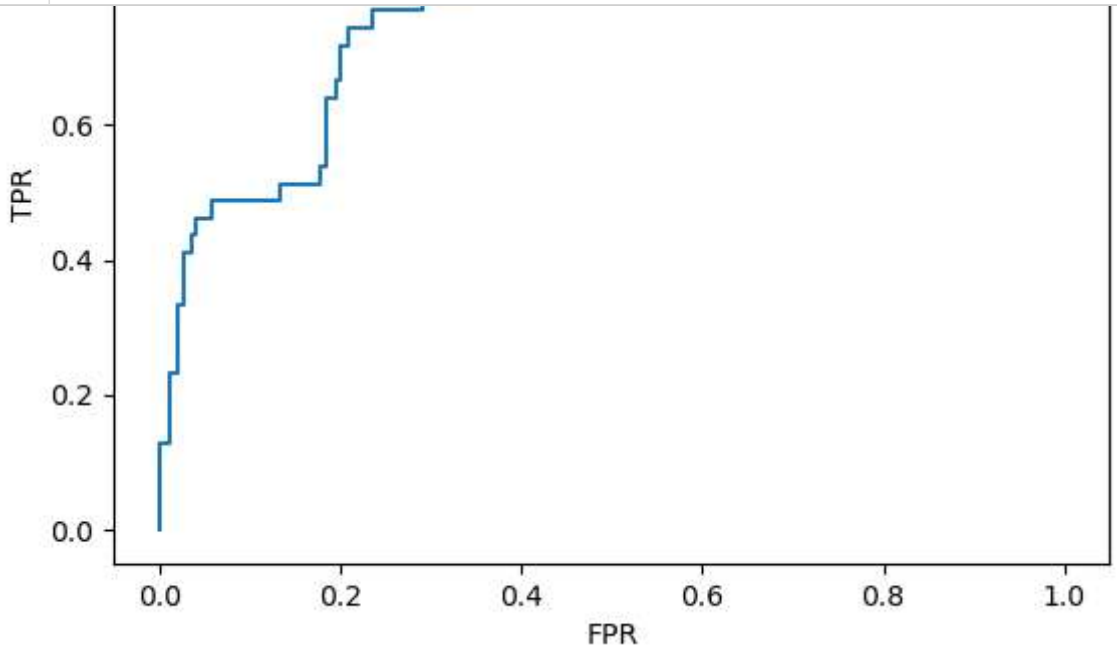
```
probability=logistic_model.predict_proba(X_test)[:,1]
probability
```

Out[64]:
```
array([8.20073192e-02, 4.57610775e-03, 2.84764545e-01, 2.46620134e-02,
       7.47106590e-02, 3.35893969e-01, 3.76598534e-01, 4.35268844e-02,
       5.75428136e-02, 1.91548056e-02, 4.82605900e-01, 5.31004390e-02,
       1.09161135e-01, 7.33923982e-02, 3.59403058e-02, 1.48126811e-01,
       2.65510432e-01, 1.59133872e-01, 1.84830439e-01, 4.91485926e-02,
       5.54327555e-01, 5.28354516e-03, 4.93097782e-02, 3.10364306e-01,
       1.34206572e-01, 1.31069154e-02, 8.22176238e-02, 1.87420735e-02,
       8.79626036e-02, 4.16035868e-02, 1.45945490e-02, 3.80982342e-03,
       5.70977884e-03, 3.23467257e-02, 5.58533146e-01, 9.30108837e-03,
       2.02220261e-03, 2.15510371e-01, 8.00164005e-01, 2.18623190e-02,
       6.17767824e-02, 9.44164078e-02, 5.97231133e-02, 5.92778741e-02,
       6.64210992e-01, 1.53625484e-02, 7.50024521e-01, 4.92698546e-01,
       5.65245414e-01, 5.04691723e-01, 1.65603669e-02, 1.59596535e-01,
       1.54532482e-02, 1.01034186e-01, 1.11615339e-01, 1.55139363e-01,
       4.06199017e-01, 4.40304306e-03, 5.83340416e-02, 1.02118488e-01,
       4.17111979e-02, 4.31616734e-01, 3.50897695e-02, 2.66784979e-03,
       2.12603753e-01, 4.83236708e-01, 3.01168692e-02, 1.79844614e-01,
       1.39346934e-02, 1.28180635e-01, 1.92381167e-01, 1.39543844e-01,
       2.77098459e-01, 9.73048198e-02, 6.29107449e-02, 1.04698197e-02,
       2.27995772e-02, 9.73917066e-02, 2.45968040e-01, 1.03898276e-01,
       7.71787154e-04, 1.64191596e-01, 7.93309023e-02, 1.73529996e-01,
       8.19847865e-02, 6.88774428e-02, 3.50864195e-01, 2.00312867e-01,
       2.39586049e-02, 5.88181809e-02, 7.85423485e-03, 7.14073544e-02,
       3.61802220e-01, 3.37510187e-01, 5.38354521e-02, 8.80094441e-02,
       2.77974011e-02, 1.74729416e-01, 1.80735795e-01, 1.19905041e-01,
       4.15610981e-02, 2.58866778e-03, 8.21565865e-03, 1.09812302e-01,
       3.01858107e-02, 1.31743313e-01, 4.95778154e-03, 3.44684822e-01,
       4.23869539e-02, 2.30056581e-03, 6.61567838e-01, 6.70242509e-01,
       1.53174831e-02, 4.92800309e-02, 2.16739871e-01, 1.21197060e-01,
       2.70654448e-01, 1.90386686e-02, 1.64451858e-02, 2.08360629e-01,
       1.71286565e-02, 3.39733400e-03, 3.00153368e-01, 3.34308649e-02,
       5.15192449e-01, 4.72266207e-01, 7.33870964e-02, 4.29612893e-02,
       8.26967693e-02, 1.32563434e-01, 4.63068816e-02, 8.83400334e-02,
       9.40770788e-02, 1.89878961e-01, 2.26684119e-02, 2.76570923e-01,
       7.48266951e-02, 1.79806960e-01, 2.79998011e-01, 2.36060318e-02,
       6.77280987e-01, 8.39748658e-02, 3.29381015e-03, 7.27200357e-02,
       1.08704569e-02, 7.91547766e-02, 4.83082857e-02, 7.19385088e-01,
       8.97109071e-02, 6.34218963e-01, 3.09052619e-02, 3.92334894e-02,
       1.09122869e-01, 6.38021680e-01, 2.44855864e-01, 2.33864819e-01,
       1.14747640e-01, 1.32894006e-01, 3.88601097e-03, 3.02350626e-02,
       2.45934302e-02, 4.63260000e-02, 6.45993888e-03, 4.48342428e-01,
       1.64772245e-02, 2.75186604e-01, 1.87404561e-01, 2.66229919e-02,
       4.36054609e-02, 1.90006272e-02, 1.45593790e-01, 2.12428561e-02,
       4.74360321e-01, 4.71179087e-01, 5.76003467e-02, 3.75002572e-01,
       7.20942293e-02, 7.14744191e-02, 6.82274033e-02, 6.91101869e-01,
       1.28960480e-01, 8.17195383e-02, 3.68826773e-03, 5.13457865e-02,
       3.47524821e-02, 6.49371473e-03, 2.99678824e-01, 5.22152996e-02,
       6.38140758e-03, 1.71842384e-02, 2.56216642e-03, 1.48696943e-01,
       3.05488265e-01, 3.20641554e-01, 5.93149233e-02, 7.98172093e-02,
       2.85665019e-01, 2.78677873e-02, 2.10361087e-02, 2.97983450e-02,
       2.97185063e-01, 3.19481683e-02, 4.45026877e-01, 9.55695626e-03,
       1.92501193e-03, 1.28293658e-01, 9.81177617e-02, 2.92527467e-02,
       1.53361928e-01, 2.83815840e-02, 1.68610315e-01, 2.10707430e-01,
       4.06520793e-02, 2.37902858e-01, 3.71022012e-02, 3.68726932e-02,
       2.18343436e-02, 6.20603212e-03, 4.46789365e-01, 2.63330802e-03,
       8.18173972e-02, 4.64869204e-01, 1.26173772e-02, 8.45869142e-01,
       1.32547763e-01, 3.80501709e-01, 6.91755514e-02, 3.80784774e-02,
       1.04594576e-01, 6.74221165e-02, 6.17572824e-01, 1.86583352e-01,
       7.60292528e-02, 1.00512888e-01, 5.42433610e-01, 1.54948350e-02,
       1.36929171e-03, 2.11350434e-01, 8.61048198e-02, 1.60937255e-02,
       1.30308944e-01, 6.18643161e-02, 5.95714470e-01, 4.99242439e-03,
       9.79340488e-03, 3.48689548e-01, 1.82483314e-02, 6.32801339e-01,
       1.89666695e-02, 3.93791265e-02, 2.81756067e-01, 6.59266369e-02,
       3.20343837e-01, 1.35538343e-02, 1.47151375e-01, 4.45011797e-03,
       1.35060009e-01, 1.03906895e-01, 4.50771030e-01, 3.38248450e-03,
```

```
7.59856525e-02, 7.30183735e-02, 3.94143808e-02, 1.71844121e-01,
1.23617934e-01, 5.07344667e-02, 3.44269189e-03, 2.69923756e-02,
3.53366535e-01, 9.79760293e-02, 8.53245153e-02, 4.94447346e-03,
1.73729664e-01, 7.83438559e-01, 7.86236024e-02, 4.31031841e-02,
9.99057041e-02, 2.66800338e-02, 4.38753833e-02, 6.99612307e-02,
7.59968738e-03, 4.46879796e-01, 1.33800547e-01, 2.21784919e-02,
2.57789402e-01, 1.65568655e-02, 2.24504799e-01, 7.10630716e-02,
1.34460845e-02, 2.25161299e-02, 3.20533260e-02, 7.14196741e-02,
4.44089210e-02, 9.72385427e-02])
```

In [65]:
```
# roc_curve
fpr,tpr,threshsholds = roc_curve(y_test,probability)
```

In [66]:
```
plt.plot(fpr,tpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC CURVE')
plt.show()
```



In [67]:
```

from sklearn.model_selection import GridSearchCV
```

In [ ]:
```

```

In [69]:

```
#alpha should never be zero
parametres={"C":[0.1,0.2,0.3,0.01,0.002,1,2,3,5,10,20,30,40,60,70,80,90],
             'penalty': ['l1', 'l2'],
             'solver': ['liblinear', 'lbfgs']
            }
grid_search=GridSearchCV(logistic_model,parametres,scoring="neg_mean_squared_err
grid_search.fit(X_train,y_train)
```

```
ase change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\pbalu\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Ple
ase change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\pbalu\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Ple
ase change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\pbalu\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Ple
ase change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\pbalu\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Ple
ase change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\pbalu\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Ple
```

In [70]:

```
print(grid_search.best_params_)
```

```
{'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}
```

In [71]:

```
print(grid_search.best_score_)
```

```
-0.12413992066354129
```

In [73]:

```
y_pred=grid_search.predict(X_test)
y_pred
```

Out[73]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0])
```

In [74]:

```
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
print(accuracy)
```

```
0.8945578231292517
```

In [75]:
```
1 y_test
```

Out[75]:

	Attrition
1041	0
184	0
1222	1
67	0
220	0
...	...
567	0
560	0
945	0
522	0
651	0

In [76]:
```
1 grid_search = accuracy_score(y_test, y_pred)
2 logistic_report = classification_report(y_test, y_pred)
3
4
5 print("Logistic Regression Model Accuracy:", logistic_accuracy)
6 print("Logistic Regression Model Classification Report:")
7 print(logistic_report)
```

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

           0       0.91      0.98      0.94       255
           1       0.70      0.36      0.47        39

    accuracy                           0.89       294
   macro avg       0.80      0.67      0.71       294
weighted avg       0.88      0.89      0.88       294
```

In [78]:
```
1 grid_search = accuracy_score(y_test, y_pred)
2 logistic_report = classification_report(y_test, y_pred)
3
4
5 print("Logistic Regression Model Accuracy:", logistic_accuracy)
6 print("Logistic Regression Model Classification Report:")
7 print(logistic_report)
```

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

           0       0.91      0.98      0.94       255
           1       0.70      0.36      0.47        39

    accuracy                           0.89       294
   macro avg       0.80      0.67      0.71       294
weighted avg       0.88      0.89      0.88       294
```

In [ ]:
```
1
```

In [79]:
```
1 from sklearn import metrics
2 # R- Square
3 # evaluating testing accuracy
4 print(metrics.r2_score(y_test,y_pred))
5
```

```
0.08355957767722466
```

In [83]:
```
1 import numpy as np
2 #mean squared error
3 print(metrics.mean_squared_error(y_test,y_pred))
4 # RMSE (Root Mean Square Error)
5 print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
6 #mean absolute error
7 print(metrics.mean_absolute_error(y_test,y_pred))
```

```
0.1054421768707483
0.32471861183299655
0.1054421768707483
```

In [ ]:
```
1
```

In [ ]:
```
1
```