```
In [1]: # Step 1: Import Libraries
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
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve, auc
from sklearn.model_selection import cross_val_score, GridSearchCV
import joblib
from sklearn.metrics import roc_auc_score
```

```
In [2]: # Step 2: Load Data
data=pd.read_csv('archive (85).zip')
print(data.tail(4))
```

```
      Age Attrition     BusinessTravel  DailyRate              Department  \
1466   39        No      Travel_Rarely        613  Research & Development
1467   27        No      Travel_Rarely        155  Research & Development
1468   49        No  Travel_Frequently       1023                   Sales
1469   34        No      Travel_Rarely        628  Research & Development

      DistanceFromHome  Education EducationField  EmployeeCount  \
1466                 6          1        Medical              1
1467                 4          3  Life Sciences              1
1468                 2          3        Medical              1
1469                 8          3        Medical              1

      EmployeeNumber  ...  RelationshipSatisfaction StandardHours  \
1466            2062  ...                         1            80
1467            2064  ...                         2            80
1468            2065  ...                         4            80
1469            2068  ...                         1            80

      StockOptionLevel  TotalWorkingYears  TrainingTimesLastYear  \
1466                 1                  9                      5
1467                 1                  6                      0
1468                 0                 17                      3
1469                 0                  6                      3

     WorkLifeBalance  YearsAtCompany YearsInCurrentRole  \
1466               3               7                  7
1467               3               6                  2
1468               2               9                  6
1469               4               4                  3

      YearsSinceLastPromotion  YearsWithCurrManager
1466                        1                     7
1467                        0                     3
1468                        0                     8
1469                        1                     2

[4 rows x 35 columns]
```

```
In [3]: # Step 3: Exploratory Data Analysis (EDA)
# Check basic information
data.info()

# Display the first few rows
data.head()
```

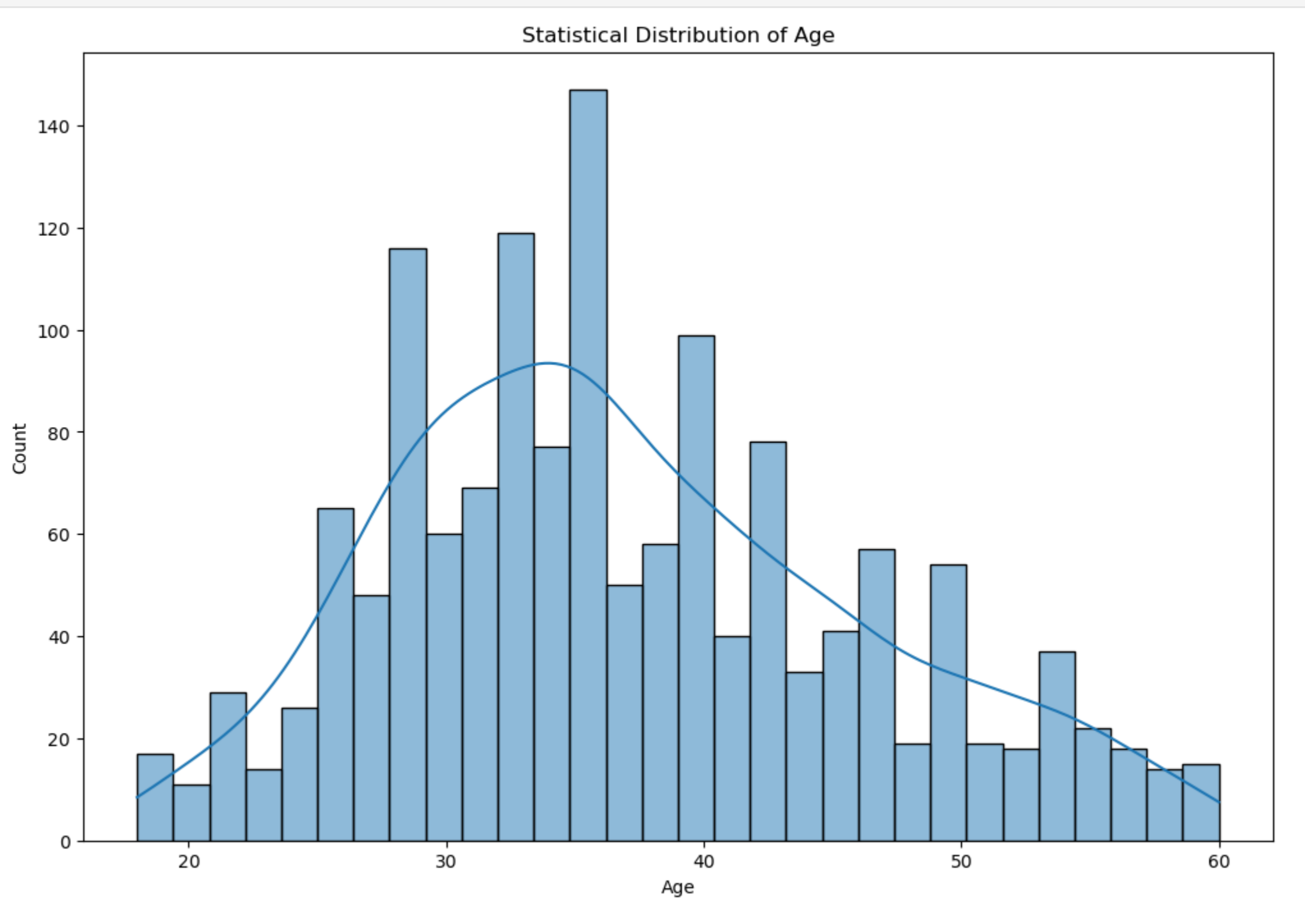
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 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
```

Out[3]:

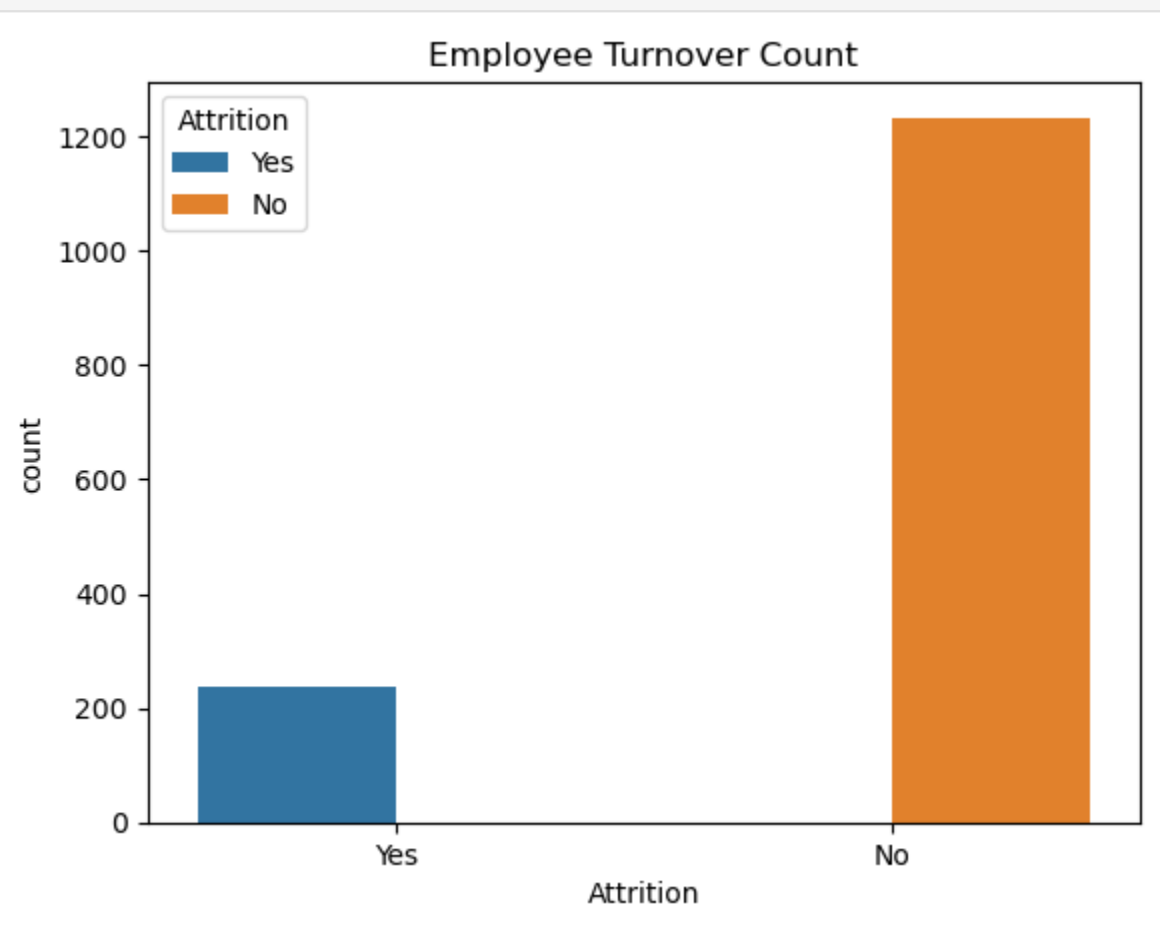
	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipSatisfaction	StandardHours
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	1	80
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	4	80
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	2	80
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	3	80
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	4	80

5 rows × 35 columns

```
In [4]: # Visualize distributions, correlations, etc.
# Add EDA tasks as needed


plt.figure(figsize=(12, 8))

# Distribution of Age
sns.histplot(data['Age'], bins=30, kde=True)
plt.title('Statistical Distribution of Age')
plt.xlabel('Age')
plt.show()
```

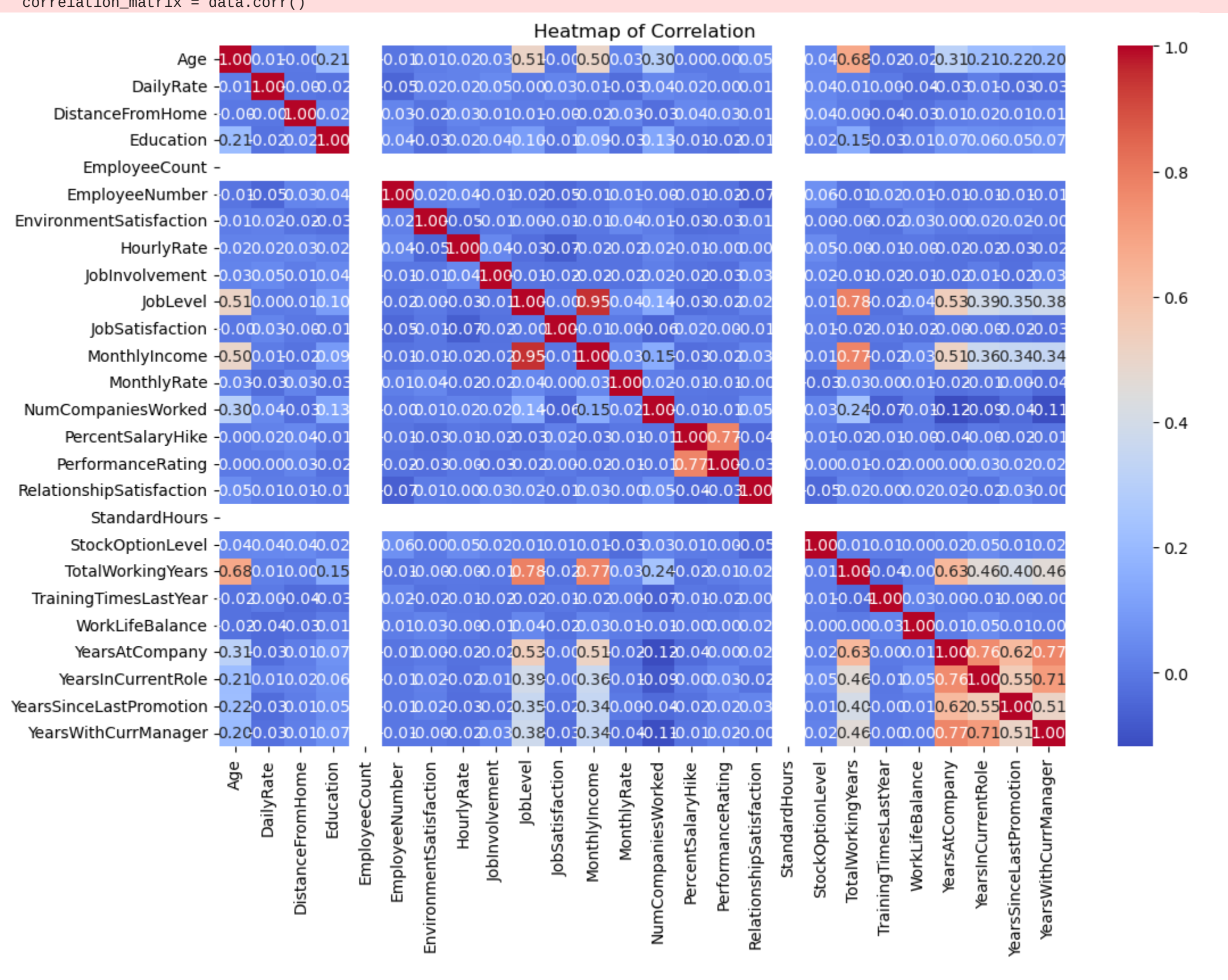



```
In [5]: # Assuming 'Attrition' is a column in your DataFrame
sns.countplot(x='Attrition', data=data, hue='Attrition')
plt.title("Employee Turnover Count" )
plt.xlabel('Attrition')
plt.show()
```




```
In [6]: # Correlation Heatmap
plt.figure(figsize=(12, 8))
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title(' Heatmap of Correlation')
plt.show()
```

```
C:\Users\nishi\AppData\Local\Temp\ipykernel_18576\3397502437.py:3: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecate
d. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = data.corr()
```

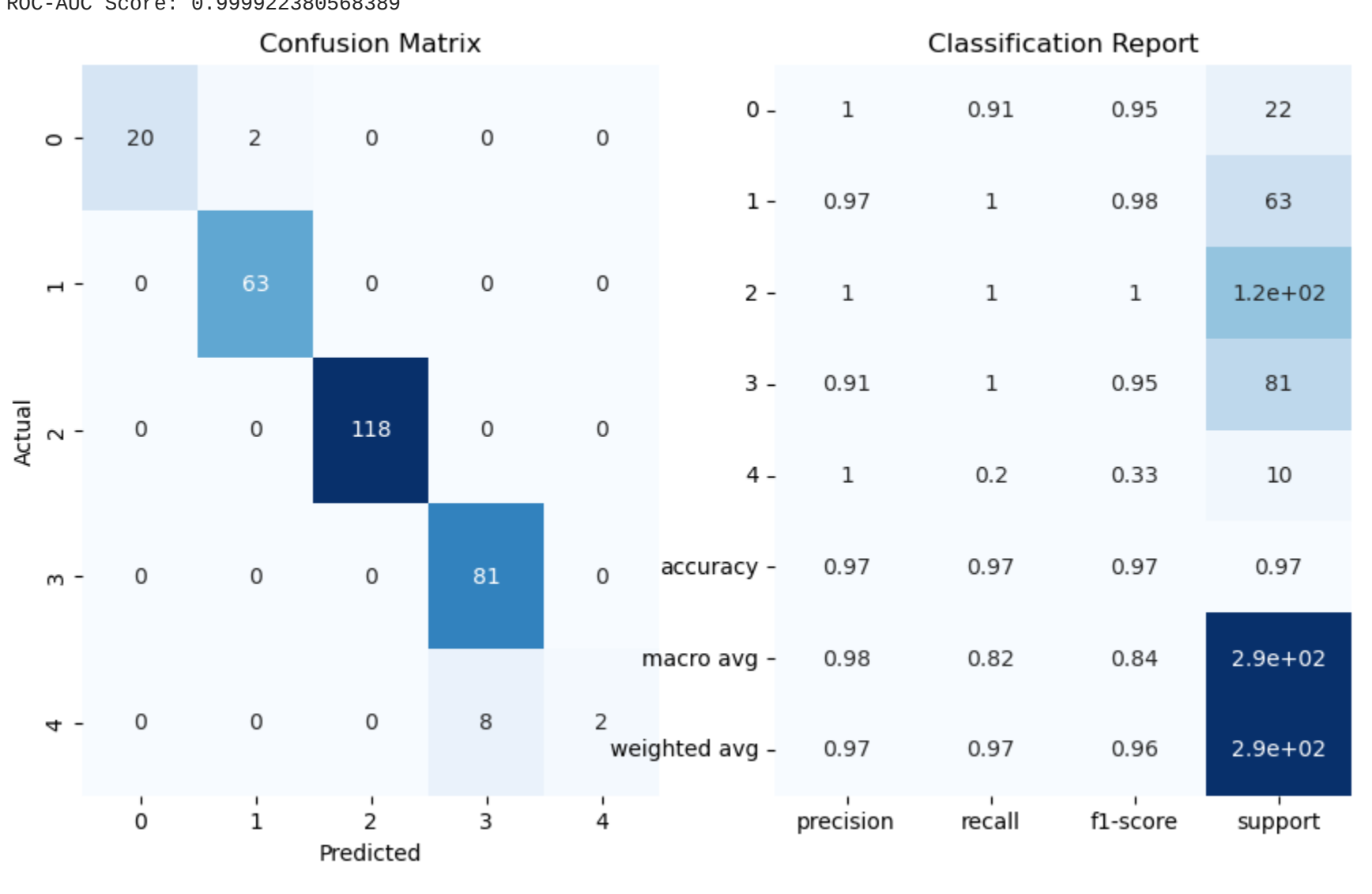


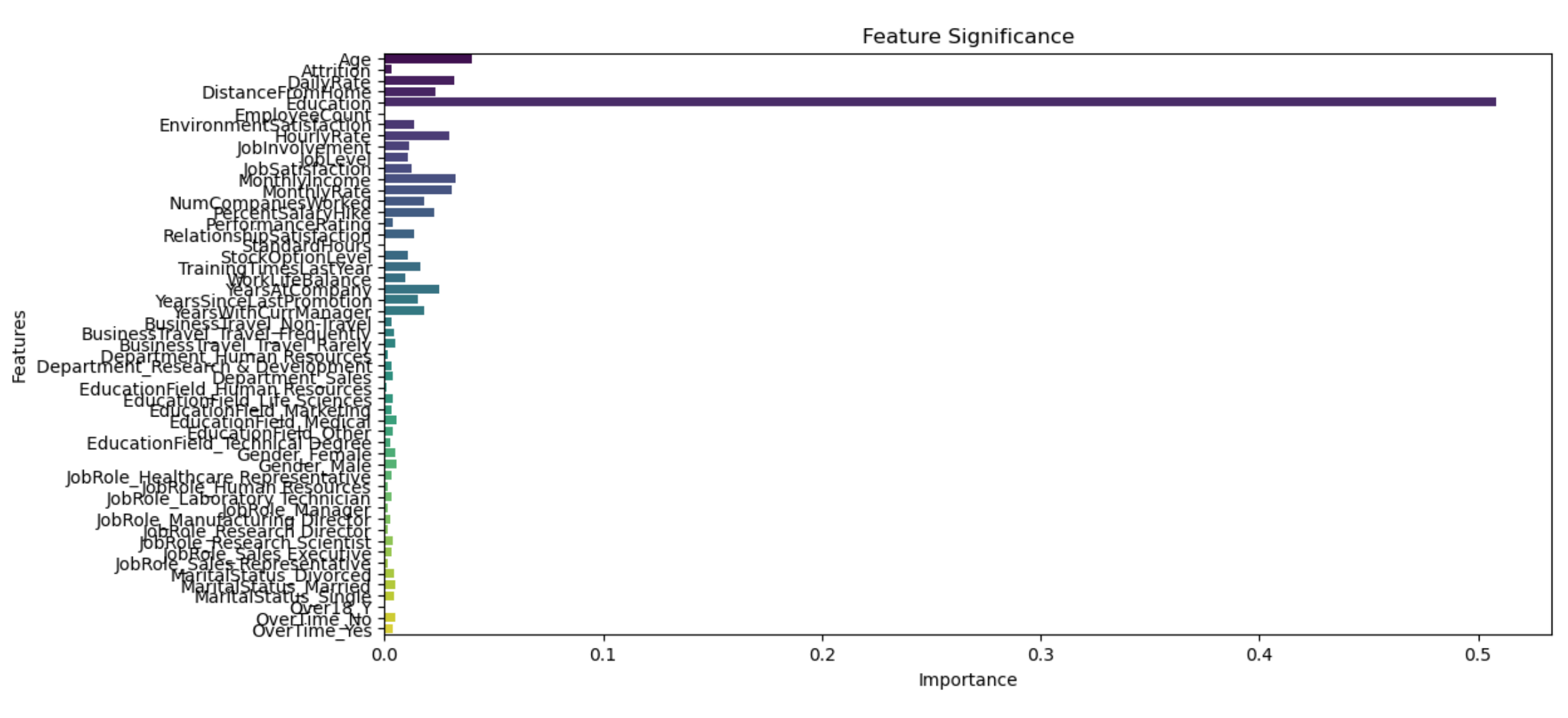

```
In [7]: import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Assuming 'data' is your DataFrame
# Load your dataset
data = pd.read_csv('archive (85).zip')  # Adjust the filename

# Display the original columns
print("Original Columns:", data.columns)

# Step 4: Preprocessing and Feature Engineering
# Handle missing values (if any)

# Encode categorical variables
la = LabelEncoder()
data['Attrition'] = la.fit_transform(data['Attrition'])

# Display columns after encoding
print("Columns after encoding:", data.columns)

# Drop uninformative features
try:
    data.drop('EmployeeNumber', axis=1, inplace=True)
    print("Column 'EmployeeNumber' dropped successfully.")
except KeyError as e:
    print(f"Error: {e}")

# Perform one-hot encoding for categorical variables if needed

# Split the data into features and targe
Label = data['Attrition']
Input_variables = data.drop('Attrition', axis=1)

# Display the final columns and first few rows of the modified DataFrame
print("Final Columns:", data.columns)
print("Modified DataFrame:")
print(data.head())
```

```
Original Columns: Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
Columns after encoding: Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
Column 'EmployeeNumber' dropped successfully.
Final Columns: Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement',
       'JobLevel', 'JobRole', 'JobSatisfaction', 'MaritalStatus',
       'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'Over18',
       'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
Modified DataFrame:
   Age  Attrition     BusinessTravel  DailyRate              Department  \
0   41          1      Travel_Rarely       1102                   Sales
1   49          0  Travel_Frequently        279  Research & Development
2   37          1      Travel_Rarely       1373  Research & Development
3   33          0  Travel_Frequently       1392  Research & Development
4   27          0      Travel_Rarely        591  Research & Development

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

   EnvironmentSatisfaction  ... RelationshipSatisfaction  StandardHours  \
0                        2  ...                        1             80
1                        3  ...                        4             80
2                        4  ...                        2             80
3                        4  ...                        3             80
4                        1  ...                        4             80

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

  YearsAtCompany  YearsInCurrentRole  YearsSinceLastPromotion  \
0              6                   4                        0
1             10                   7                        1
2              0                   0                        0
3              8                   7                        3
4              2                   2                        2

   YearsWithCurrManager
0                     5
1                     7
2                     0
3                     0
4                     2

[5 rows x 34 columns]
```

```
In [8]: # Assuming you have a DataFrame 'data' with columns representing features and the target variable
features = data.drop(columns=['TotalWorkingYears', 'YearsInCurrentRole'])
target = data['Education']

# Encode the target variable for multiclass classification
label_encoder = LabelEncoder()
target_encoded = label_encoder.fit_transform(target)

# Step 10: Handle categorical variables (one-hot encoding)
features = pd.get_dummies(features)

# Step 11: Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(features, target_encoded, test_size=0.2, random_state=42)

# Step 12: Build a RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Step 13: Test the model
y_pred = model.predict(X_test)

# Step 14: Evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Additional metrics
classification_rep = classification_report(y_test, y_pred, zero_division=1)  # Set zero_division=1 to avoid warnings
confusion_mat = confusion_matrix(y_test, y_pred)

# Visualizations for model evaluation metrics
plt.figure(figsize=(16, 6))

# Confusion Matrix
plt.subplot(1, 3, 1)
sns.heatmap(confusion_mat, annot=True, cmap='Blues', fmt='g', cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')

# Classification Report
plt.subplot(1, 3, 2)
df_classification_rep = pd.DataFrame.from_dict(classification_report(y_test, y_pred, output_dict=True)).transpose()
sns.heatmap(df_classification_rep.iloc[:, :4], annot=True, cmap='Blues', cbar=False)
plt.title('Classification Report')

# ROC-AUC Score
try:
    roc_auc = roc_auc_score(pd.get_dummies(y_test), model.predict_proba(X_test), multi_class='ovr')
    print(f"ROC-AUC Score: {roc_auc}")
except ValueError:
    print("ROC-AUC score is not applicable for multiclass classification")

# Feature Importance
features_importance = model.feature_importances_
feature_names = features.columns

plt.figure(figsize=(12, 6))
sns.barplot(x=features_importance, y=feature_names, palette='viridis')
plt.title('Feature Significance')
plt.xlabel('Importance')
plt.ylabel('Features')
plt.show()
```

```
Accuracy: 0.9659863945578231
ROC-AUC Score: 0.999922380568389
```






```
In [9]: # Assuming 'features' is your feature matrix
featuresencoded = pd.get_dummies(features)
from sklearn.model_selection import cross_val_score

# Use cross-validation to evaluate the model's performance
cross_val = cross_val_score(model, featuresencoded, target, cv=5)
print(f"Cross_validation scores: {cross_val}")
```

```
Cross_validation scores: [0.94217687 0.94897959 0.92857143 0.95238095 0.93877551]
```

```
In [10]: # Step 15: Model Selection
# Choose the best model based on evaluation metrics
# In this case, we are focusing on accuracy, but consider other metrics based on your project goals
bestmodel = model
```

```
In [11]: # Step 1: Import necessary libraries
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

# Step 2: Assuming 'data' is your DataFrame with the relevant columns
# Ensure the 'model' variable is defined with the appropriate classifier
model = RandomForestClassifier()

# Step 3: Define the preprocessing steps using ColumnTransformer
# Include only the columns that are present in your DataFrame
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), ['DistanceFromHome', 'RelationshipSatisfaction'])
    ],
    remainder='passthrough'
)

# Step 4: Create the pipeline by combining the preprocessing and the classifier
line = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', model)
])

# Step 5: Define the parameter grid to search
param_grid = {
    'classifier__n_estimators': [50, 100, 200],
    'classifier__max_depth': [None, 10, 20],
    'classifier__min_samples_split': [2, 5, 10],
    'classifier__min_samples_leaf': [1, 2, 4]
}

# Step 6: Replace 'data' with your actual DataFrame and adjust features and target
Covariates = data[['DistanceFromHome', 'RelationshipSatisfaction']]
Dependent_variable = data['Education']
```

```
In [12]: # Step 7: Create the GridSearchCV object
grid_search = GridSearchCV(line, param_grid, cv=5)

# Step 8: Fit the GridSearchCV object to the data
grid_search.fit(features, target)

# Step 9: Access the best hyperparameters and best model from the result
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
```

```
In [13]: # Test the best model
y_pred_best = best_model.predict(X_test)

# Evaluate the performance of the best model
accuracy_best = accuracy_score(y_test, y_pred_best)

print(f"Accuracy of the best model: {accuracy_best}")
```

```
Accuracy of the best model: 0.0
```

HR Analytics Project Documentation

Project Summary: The HR Analytics Project goal to analyze and understand employee attrition within an organization. Employee attrition, the gradual loss of employees over time, poses challenges for industry, including enhance costs and the loss of organizational knowledge and experience. HR Analytics is employed to apply analytical processes to the human resource department, providing cognition into employee performance and assisting in delibrate-making to improve HR processes.

Project Targets: Understand Attrition Impact:

Explore how attrition affects industry, to incressed costs, disruptions, and potential adverse impacts on customer interactions. Apply HR Analytics:

Use HR Analytics to analyze magnetic pattern within the organization. Employ data-driven percepition to make informed deliberate for improving HR processes and employee retention strategies. Steps to Achieve Objectives:

1. Data Exploration: Analyze the provided dataset to understand its structure and attributes. Identify key features relevant to attrition analysis.
2. Attrition Impact Analysis: Find the cost implications of high employee attrition. Explore the hundles associated with constant turnover, such as the loss of knowledge and experience.
3. HR Analytics Implementation: Employ data analysis techniques to find patterns and trends related to attrition. employ visualization tools to present meaningful insights. Apply statistical analysis to understand the factors contributing to attrition.
4. Decision Making: Based on analytics output, make informed decisions to improve HR processes and Implement successful employee retention strategies. Consider adjustments to compensation programs, work culture, and motivation systems.

Conclusion: The HR Analytics Project revolves around understanding and addressing employee attrition challenges. By analyzing data and emplying HR Analytics techniques, the project goals to provide actionable insights for organizations to enhance employee retention and mitigate the negative impacts of attrition. The final outcome includes informed decision-making strategies to make more stable and efficient workforce.