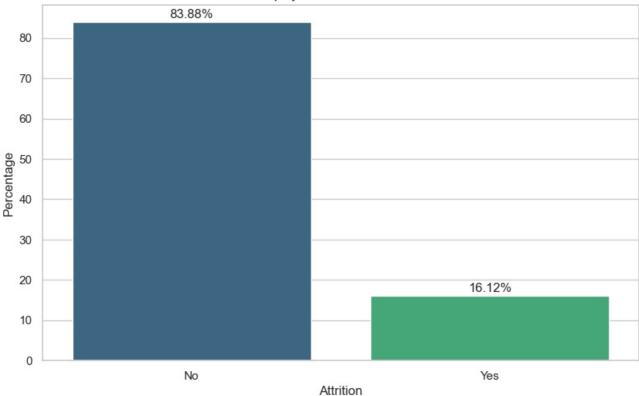
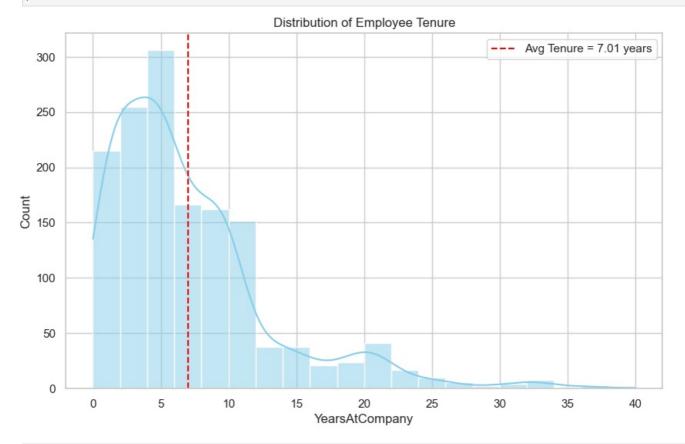
```
In [1]: # IBM HR Analytics Employee Attrition & Performance Project
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        from sklearn.model selection import train test split
        from sklearn.linear model import LogisticRegression
        from sklearn.metrics import classification report, confusion matrix, accuracy score
        warnings.filterwarnings('ignore')
        sns.set(style='whitegrid')
        plt.rcParams['figure.figsize'] = (10, 6)
In [3]: #Load Dataset
        df = pd.read csv("C:\\Users\\HP\\Downloads\\WA Fn-UseC -HR-Employee-Attrition.csv")
        # Data Overview
        df.drop(['EmployeeCount', 'EmployeeNumber', 'StandardHours', 'Over18'], axis=1, inplace=True)
        print(f"Shape: {df.shape}")
        print(f"Missing values: \n{df.isnull().sum().sum()}")
        print(f"Duplicated rows: {df.duplicated().sum()}")
       print(df.dtypes)
       Shape: (1470, 31)
       Missing values:
      Duplicated rows: 0
                                  int64
       Attrition
                                  object
       BusinessTravel
                                  object
       DailyRate
                                  int64
       Department
                                  object
       DistanceFromHome
                                  int64
       Education
                                   int64
       EducationField
                                 object
       EnvironmentSatisfaction
                                   int64
       Gender
                                 object
       HourlyRate
                                   int64
       JobInvolvement
                                  int64
       Jobl evel
                                  int64
       JobRole
                                  object
       JobSatisfaction
                                  int64
       MaritalStatus
                                 object
                                  int64
       MonthlyIncome
       MonthlyRate
                                   int64
      NumCompaniesWorked
                                   int64
       OverTime
                                 object
                                  int64
       PercentSalaryHike
       PerformanceRating
                                   int64
      RelationshipSatisfaction int64
       StockOptionLevel
                                  int64
                                  int64
       TotalWorkingYears
       TrainingTimesLastYear
                                   int64
      WorkLifeBalance
                                  int64
       YearsAtCompany
                                  int64
       YearsInCurrentRole
                                  int64
       YearsSinceLastPromotion
                                   int64
       YearsWithCurrManager
                                   int64
       dtype: object
In [4]: # Attrition Rate
        attrition percentage = df['Attrition'].value counts(normalize=True) * 100
        sns.barplot(x=attrition_percentage.index, y=attrition_percentage.values, palette='viridis')
        for index, value in enumerate(attrition percentage.values):
            plt.text(index, value + 1, f"{value:.2f}%", ha='center')
        plt.title("Employee Attrition Distribution")
        plt.xlabel("Attrition")
        plt.ylabel("Percentage")
        plt.show()
```

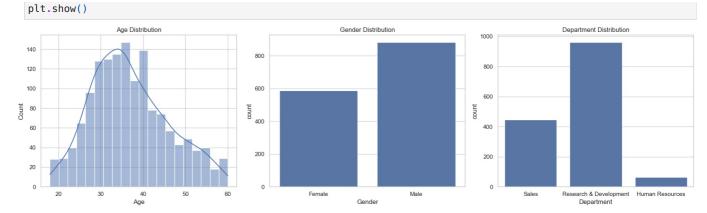
Employee Attrition Distribution



```
In [5]: # Average Tenure
    avg_tenure = df['YearsAtCompany'].mean()
    sns.histplot(df['YearsAtCompany'], kde=True, bins=20, color='skyblue')
    plt.axvline(avg_tenure, color='red', linestyle='--', label=f"Avg Tenure = {avg_tenure:.2f} years")
    plt.title("Distribution of Employee Tenure")
    plt.legend()
    plt.show()
```



```
In [6]: # Demographic Distribution
fig, axes = plt.subplots(1, 3, figsize=(18, 5))
sns.histplot(df['Age'], kde=True, bins=20, ax=axes[0])
sns.countplot(data=df, x='Gender', ax=axes[1])
sns.countplot(data=df, x='Department', ax=axes[2])
axes[0].set_title('Age Distribution')
axes[1].set_title('Gender Distribution')
axes[2].set_title('Department Distribution')
plt.tight_layout()
```



```
In [8]: # Attrition by Age and Gender

df['Attrition_Flag'] = df['Attrition'].apply(lambda x: 1 if x == 'Yes' else 0)

sns.kdeplot(data=df[df['Attrition'] == 'Yes'], x='Age', fill=True, label='Attrition=Yes', color='salmon')

sns.kdeplot(data=df[df['Attrition'] == 'No'], x='Age', fill=True, label='Attrition=No', color='lightblue')

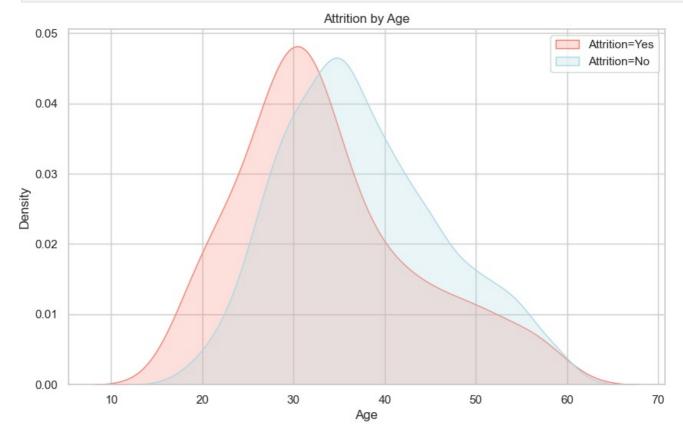
plt.title("Attrition by Age")

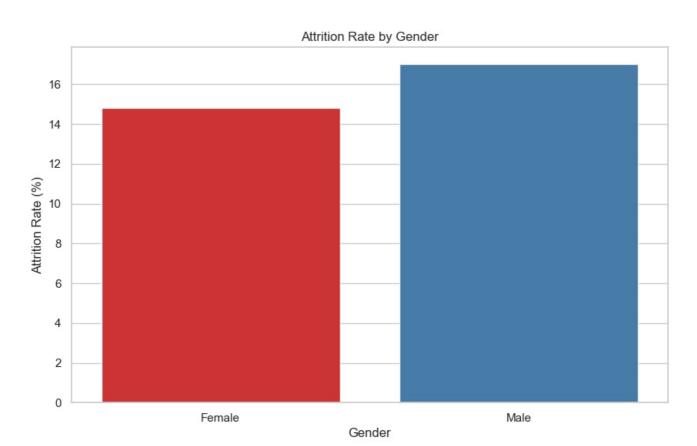
plt.legend()

plt.show()

def calculate_attrition_rate(df, column):
    attrition_counts = df.groupby([column, 'Attrition']).size().unstack(fill_value=0)
    attrition_rate = (attrition_counts['Yes'] / attrition_counts.sum(axis=1)) * 100
    return attrition_rate.reset_index().rename(columns={0: 'AttritionRate'})

gender_attr = calculate_attrition_rate(df, 'Gender')
    sns.barplot(data=gender_attr, x='Gender', y='AttritionRate', palette='Set1')
    plt.title("Attrition Rate by Gender")
    plt.ylabel("Attrition Rate (%)")
    plt.show()
```



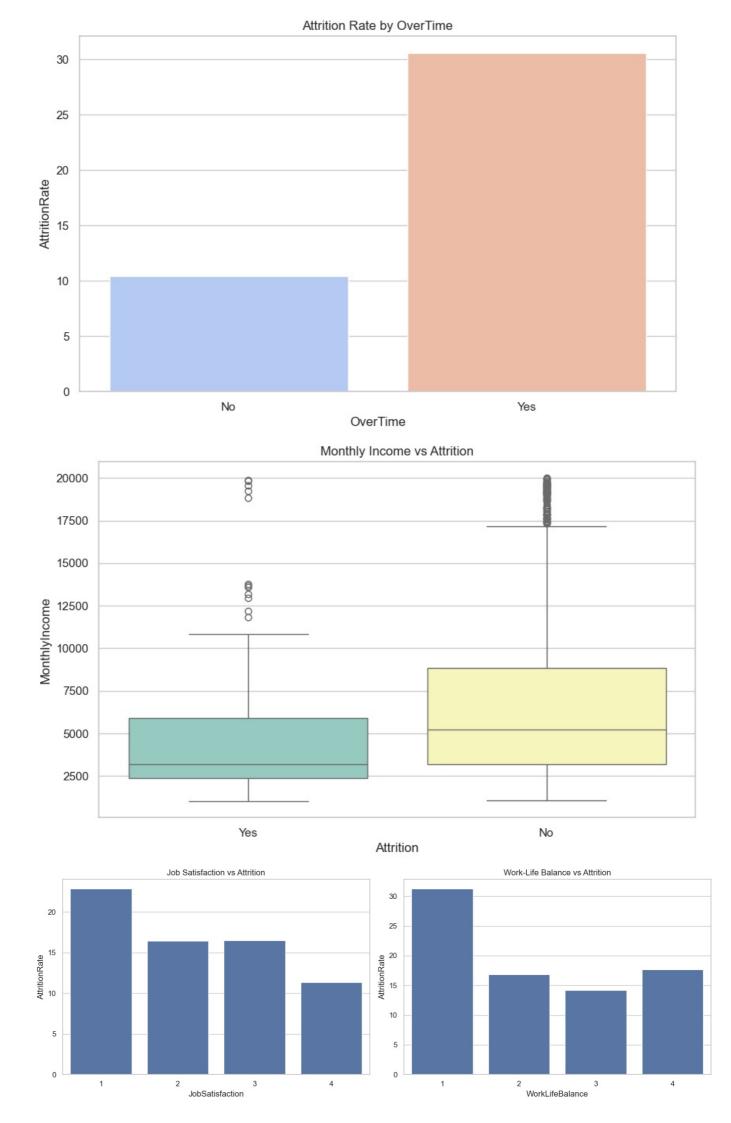


```
In [9]: # Key Factors Analysis
    overtime_attr = calculate_attrition_rate(df, 'OverTime')
    sns.barplot(data=overtime_attr, x='OverTime', y='AttritionRate', palette='coolwarm')
    plt.title("Attrition Rate by OverTime")
    plt.show()

sns.boxplot(data=df, x='Attrition', y='MonthlyIncome', palette='Set3')
    plt.title("Monthly Income vs Attrition")
    plt.show()

satisfaction_attr = calculate_attrition_rate(df, 'JobSatisfaction')
    wlb_attr = calculate_attrition_rate(df, 'WorkLifeBalance')

fig, axes = plt.subplots(1, 2, figsize=(14, 5))
    sns.barplot(data=satisfaction_attr, x='JobSatisfaction', y='AttritionRate', ax=axes[0])
    sns.barplot(data=wlb_attr, x='WorkLifeBalance', y='AttritionRate', ax=axes[1])
    axes[0].set_title("Job Satisfaction vs Attrition")
    axes[1].set_title("Work-Life Balance vs Attrition")
    plt.tight_layout()
    plt.show()
```



```
In [10]: # Correlation Heatmap
          numeric df = df.select dtypes(include=['int64', 'float64'])
          sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
          plt.title("Correlation Heatmap")
          plt.show()
                                                                  Correlation Heatmap
                                                                                                                                1.0
                                  <mark>1.00</mark>.040.00.20.010.020.00<mark>0.51</mark>0.00<mark>0.50</mark>.00<mark>0.30</mark>.000.000.050.04<mark>0.68</mark>0.020.000.310.210.220.240.10
                             Age
                                   DailyRate
                                  0.0<del>0</del>.0<mark>0.00</mark>.020.02.02.030.010.040.000.02.030.030.030.040.030.010.040.000.040.030.010.020.010.010.08
              DistanceFromHome
                       Education
                                  0.210.02.02<mark>1.00</mark>0.03.020.040.140.00.090.00.130.040.040.040.00.020.150.030.010.070.060.050.070.03
                                                                                                                                0.8
          EnvironmentSatisfaction
                                  HourlyRate
                                  0.020.020.030.020.0<mark>5.00</mark>0.040.030.070.020.020.020.040.000.000.050.040.040.040.020.020.020.0
                                  Joblnvolvement
                        JobLevel
                                                                                                                               - 0.6
                   JobSatisfaction
                  MonthlyIncome 0.5(0.04).02.090.040.020.02.01.000.00.150.030.02.030.01.770.020.00.510.360.340.340.1
                                  0.030.09.030.08.040.020.02.040.000.03<mark>1.00</mark>0.020.040.040.040.08.030.0<u>0</u>0.040.022.040.040.0
                     MonthlyRate
                                  0.30.040.00.13.010.020.020.140.00.150.02<mark>1.04</mark>0.040.00.050.030.240.040.040.120.090.040.10.04
           NumCompaniesWorked
                                                                                                                              -04
               PercentSalaryHike
                                  0.000.000.030.042.043.040.043.012.040.042.040.01<mark>0.777.04</mark>0.013.000.040.012.000.000.030.0120.0120.01
0.0150.010.040.010.010.000.030.0420.010.040.0150.040.013.040.0150.040.0150.040.0150.040.0150.040.0150.040.0150
               PerformanceRating
          RelationshipSatisfaction
                                   0.040.040.040.020.000.050.020.010.010.040.038.030.010.040.0<mark>5.00</mark>0.010.010.000.020.050.010.020.1
                StockOptionLevel
                                  -0.2
                TotalWorkingYears
            TrainingTimesLastYear
                                  WorkLifeBalance
                 YearsAtCompany
                                  YearsInCurrentRole
                                                                                                                               - 0.0
         YearsSinceLastPromotion 0.220.08.010.050.020.030.00.350.00.340.090.040.020.030.00.440.000.010.620.551.00.510.00
           YearsWithCurrManager 0.200.09.010.070.09.02.00.340.04.140.00.020.00.02.460.00.00.70.710.511.000.10
                    TotalWorkingYears
                                                                                                                   Attrition Flag
                                                    HourlyRate
                                                           JobLevel
                                                                  MonthlyIncome
                                                                      MonthlyRate
                                                                             PercentSalaryHike
                                                                                PerformanceRating
                                                                                                         YearsInCurrentRole
                                      DailyRate
                                          DistanceFromHome
                                                        Joblnvolvement
                                                               JobSatisfaction
                                                                         NumCompaniesWorked
                                                                                       StockOptionLevel
                                                                                                            YearsSinceLastPromotion
                                             Education
                                                 EnvironmentSatisfaction
                                                                                    RelationshipSatisfaction
                                                                                              Training Times Last Year
                                                                                                 WorkLifeBalance
                                                                                                     YearsAtCompany
                                                                                                               YearsWithCurrManager
In [11]: # Predictive Modeling
          df encoded = pd.get dummies(df.drop('Attrition', axis=1), drop first=True)
          X = df_encoded.drop('Attrition_Flag', axis=1)
          y = df encoded['Attrition Flag']
          X_{\text{train}}, X_{\text{test}}, y_{\text{train}}, y_{\text{test}} = \text{train\_test\_split}(X, y, \text{test\_size=0.2}, \text{random\_state=42})
          model = LogisticRegression(max_iter=1000)
          model.fit(X_train, y_train)
          y pred = model.predict(X test)
In [12]: # Evaluation
          print("Classification Report:\n", classification report(y test, y pred))
          print("Accuracy:", accuracy_score(y_test, y_pred))
          print("Confusion Matrix:\n", confusion matrix(y test, y pred))
         Classification Report:
                                       recall f1-score
                         precision
                                                            support
                                                    0.92
                     0
                             0.88
                                        0.97
                                                                255
                     1
                             0.42
                                        0.13
                                                   0.20
                                                                 39
                                                   0.86
                                                                294
             accuracy
                                                    0.56
                                                                294
            macro avg
                             0.65
                                        0.55
         weighted avg
                                                   0.83
                                                                294
                             0.82
                                        0.86
        Accuracy: 0.8605442176870748
         Confusion Matrix:
          [[248
                 71
          [ 34
                 5]]
```