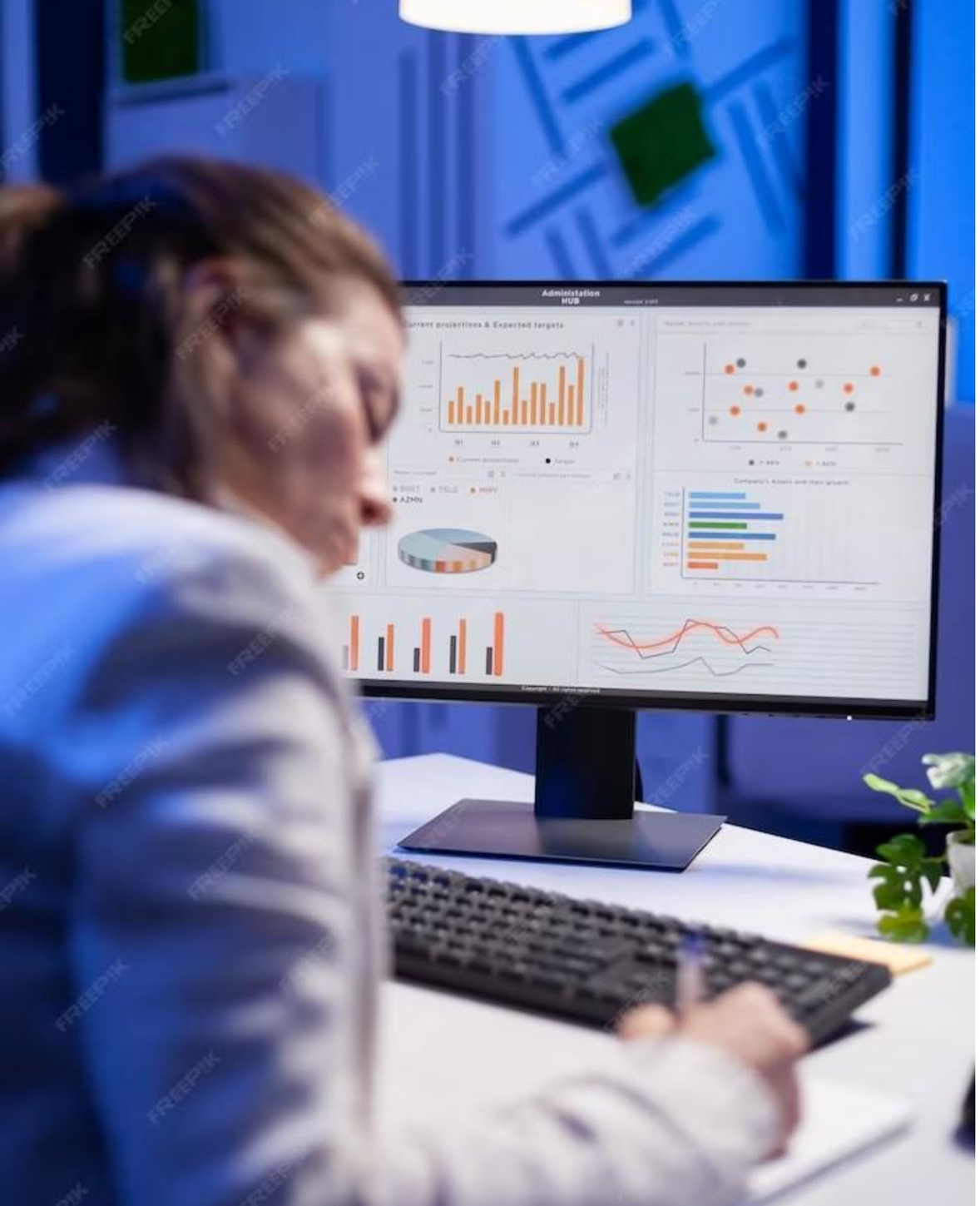
IBM HR Analytics for Employee Attrition Prediction

Presented by -

Faymin Chen



Introduction

Attrition is a major problem for all businesses regardless of location, size, and industry sector. A well-managed human resources department is critical to any organization. Companies spend a lot of time and money to nurture their most valuable asset: their employees. When employees leave, companies face huge losses. High turnover means high cost. Hence, it is important to predict turnover rates. The Human Resources Department would greatly benefit from the ability of data scientists to predict employee attrition risk and identify the underlying factors contributing to it, such as inadequate compensation, job dissatisfaction, and commuting distance. This insight will enable them to implement targeted support measures aimed at enhancing employee retention and safeguarding our valuable workforce. In this report, we summarize the findings and insights gained from analyzing the IBM HR Analytics Employee Attrition & Performance dataset.

OBJECTIVE

Employee attrition is the rate at which employees leave a company. The goal of this analysis is to create a model that can determine the most dominant contributing factors affecting attrition. Through this kind of analysis, we can understand how many employees are likely to leave, while also determining which employees are at the highest risk and for what reasons.

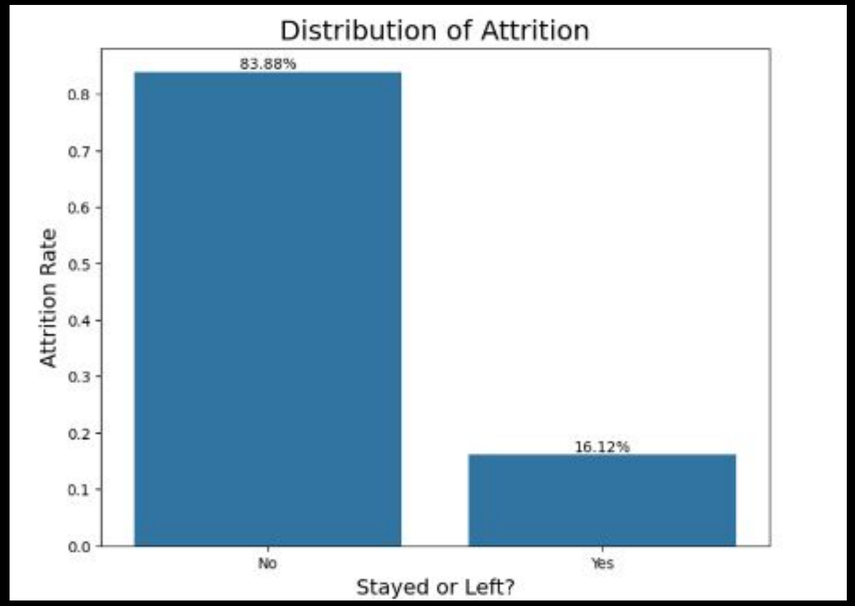
METHODOLOGY

Load the Dataset: The IBM HR Analytics Attrition Dataset is loaded using the pd.read_csv() function. The head() and info() methods are used to display the first few rows and get information about the dataset, respectively.
Knowing the Dataset: Basic Information about the dataset is generated; numerical and categorical attributes are enlisted.
Data Cleaning: no duplicate values found. Drop unnecessary uniform columns: EmployeeCount, EmployeeNumber, StandardHours, and Over18
Data Visualization: Seaborn library is used to visualize the data.
Data Preprocessing: The target variable 'Attrition' is mapped to binary values (1 for 'Yes' and 0 for 'No'). Selected features are extracted from the dataset and one-hot encoded using the get_dummies() function.
- Splitting the Dataset: The dataset is split into training and testing sets using the train_test_split() method from scikit-learn.
Implementing Machine Learning Algorithms: Logistic Regression, Gradient Boostin, and Random Forest classifiers are initialized and trained using the training data.
Model Evaluation: AUC ROC score and confusion matrix (precision, recall, F1 score) are computed to evaluate the performance of each algorithm on the testing data.
Results: The results, including the AUC ROC score and confusion matrix, are printed for each algorithm.
Model Performance Comparison:

DATASET

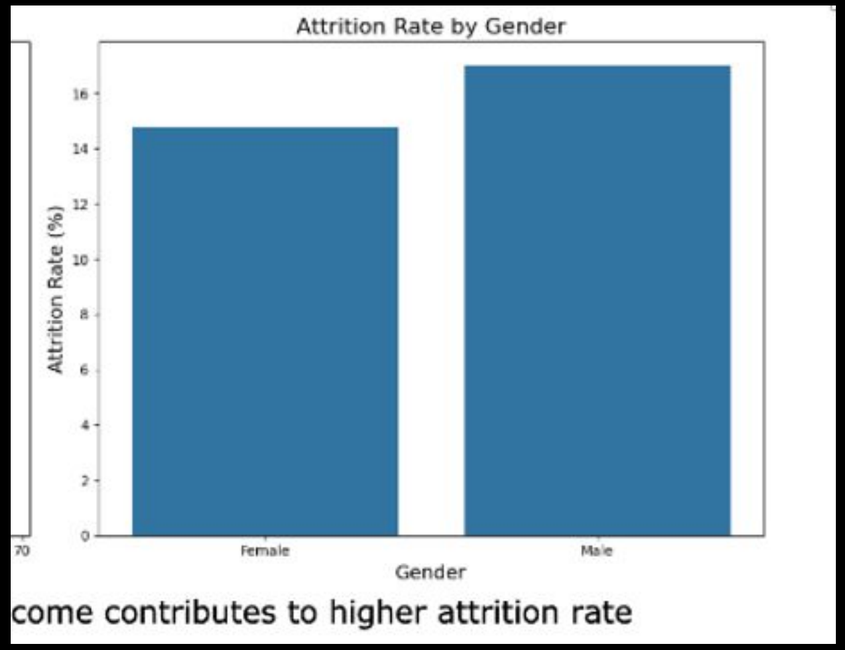
This is a hypothetical dataset created by IBM data scientists. The dataset has 1470 observations and 35 features that contains numeric and categorical data types describing each employee's background and characteristics, and labeled with whether they are still in the company or whether they have left to work somewhere else.

01] Age
02] Attrition
03] BusinessTravel
04] DailyRate
05] Department
06] DistanceFromHome
07] Education
08] EducationField
09] EmployeeCount
10] EmployeeNumber
11] EnvironmentSatisfaction
12] Gender
13] HourlyRate
14] JobInvolvement
15] JobLevel
16] JobRole
17] JobSatisfaction
18] MaritalStatus
19] MonthlyIncome
20] MonthlyRate
21] NumCompaniesWorked
22] Over18
23] OverTime
24] PercentSalaryHike
25] PerformanceRating
26] RelationshipSatisfaction
27] StandardHours
28] StockOptionLevel
29] TotalWorkingYears
30] TrainingTimesLastYear
31] WorkLifeBalance
32] YearsAtCompany
33] YearsInCurrentRole
34] YearsSinceLastPromotion
35] YearsWithCurrManager



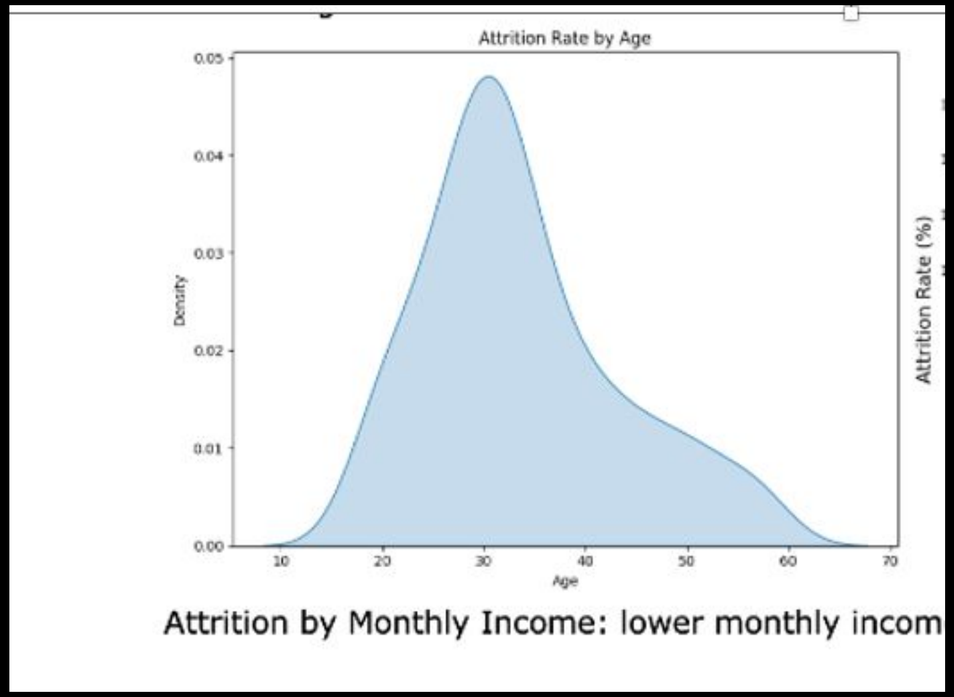
Inference:

1. The employee attrition rate of this organization is 16.12%.
2. According to experts in the field of Human Resources, says that the attrition rate 4% to 6% is normal in organization.
3. So we can say the attrition rate of the organization is at a dangerous level.
4. Therefore the organization should take measures to reduce the attrition rate.



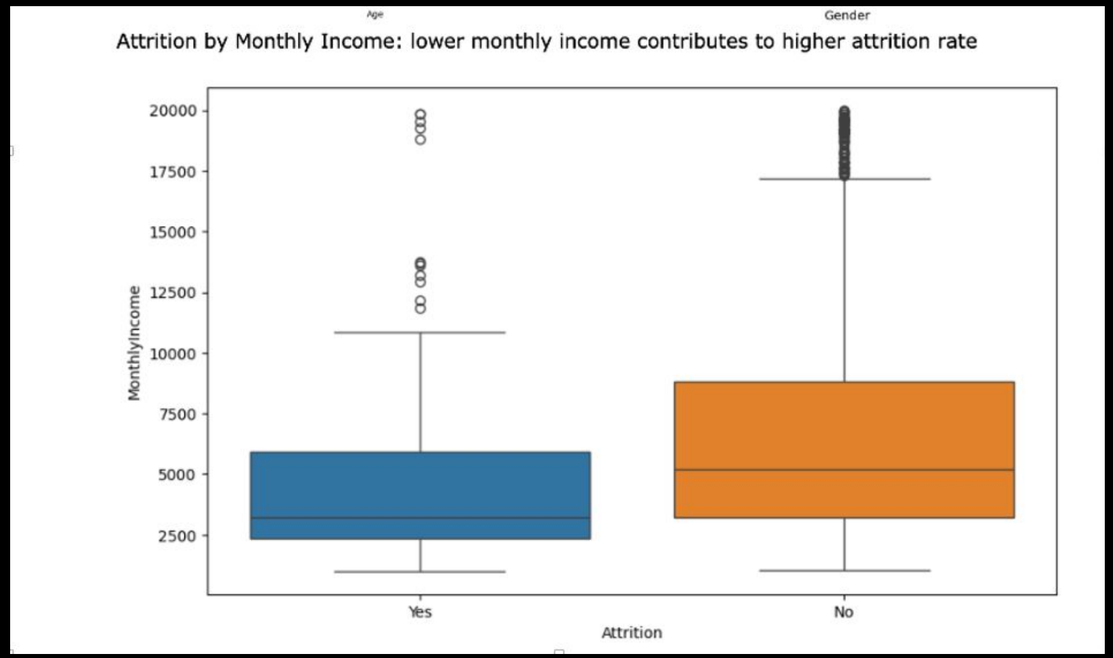
Inference:

1. The number of male employees in the organization accounts for a higher proportion than female employees by more than 20%.
2. Male employees are leaving more from the organization compared to female employees.



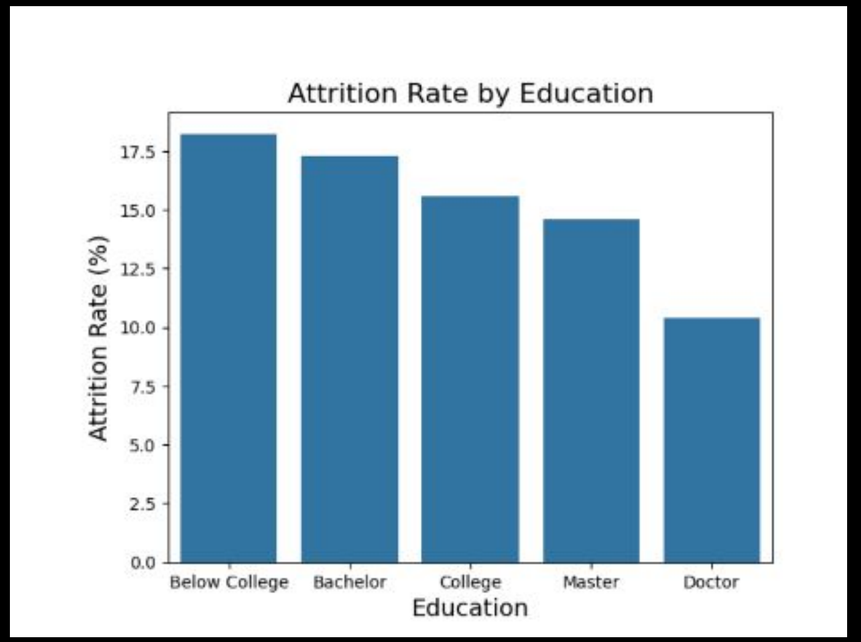
Inference:

1. Most of the employees are between ages 30 to 40.
2. We can clearly observe a trend that as the age is increasing the attrition is decreasing.
3. From the boxplot we can also observe that the median age of employee who left the organization is less than the employees who are working in the organization.
4. Employees with young age leaves the company more compared to elder employees.

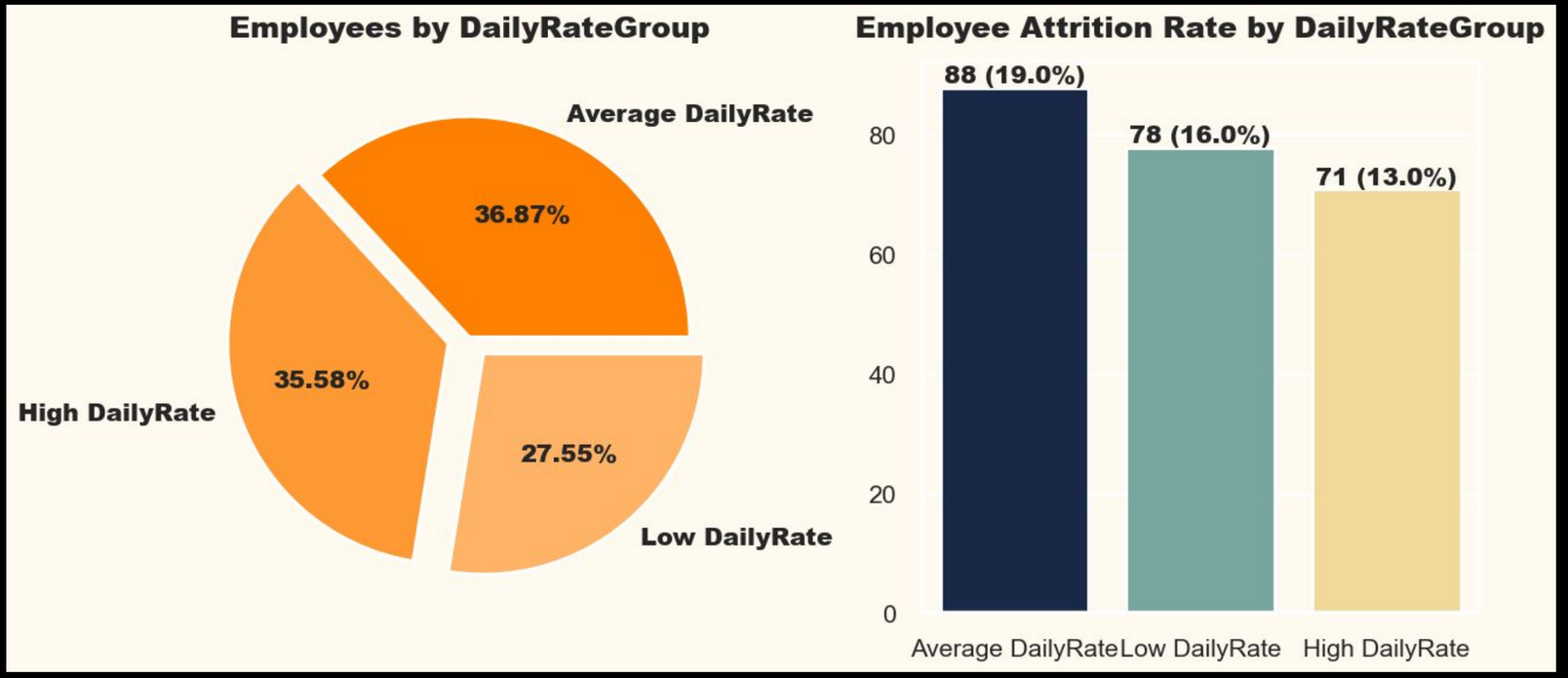


Inference:

1. Most of the employees are getting paid less than 10000 in the organization.
2. The average monthly income of employee who has left is comparatively low with employee who is still working.
3. As the Monthly Income increases the attrition decreases.

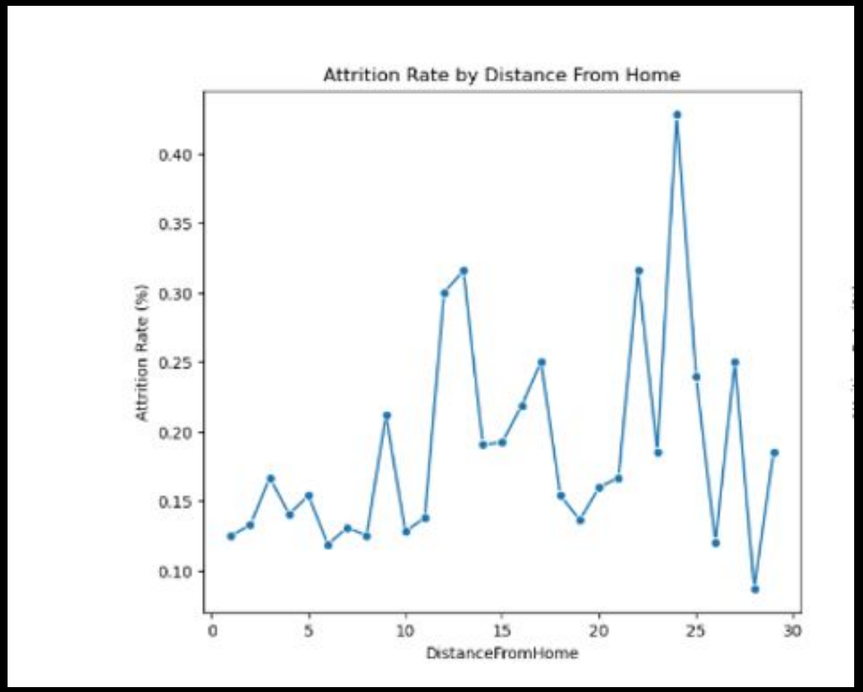


Education includes five levels, from no college experience to having a master's or doctorate degree. Attrition rate breakdown by education level shows that the highest education (doctorate) has the lowest attrition rate, while below college has the highest attrition rate.¶



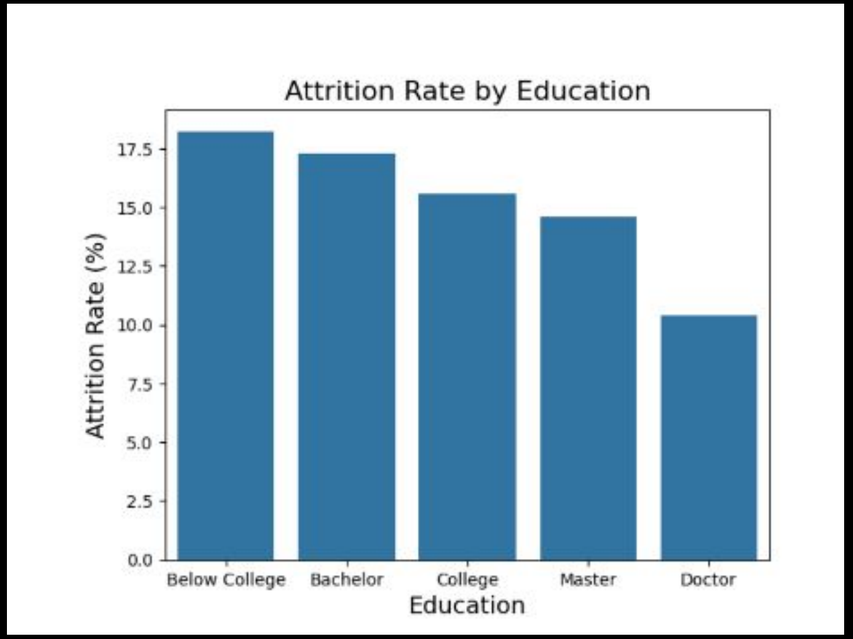
Inference:

1. Employees with Average DailyRate & High Daily Rate are approximately equal.
2. But the attrition rate is very high of employees with average Daily Rate compared to the employees with High DailyRate.
3. The attrition rate is also high of employees with low DailyRate.
4. Employees who are not getting High Daily Rate are mostly leaving the organization.



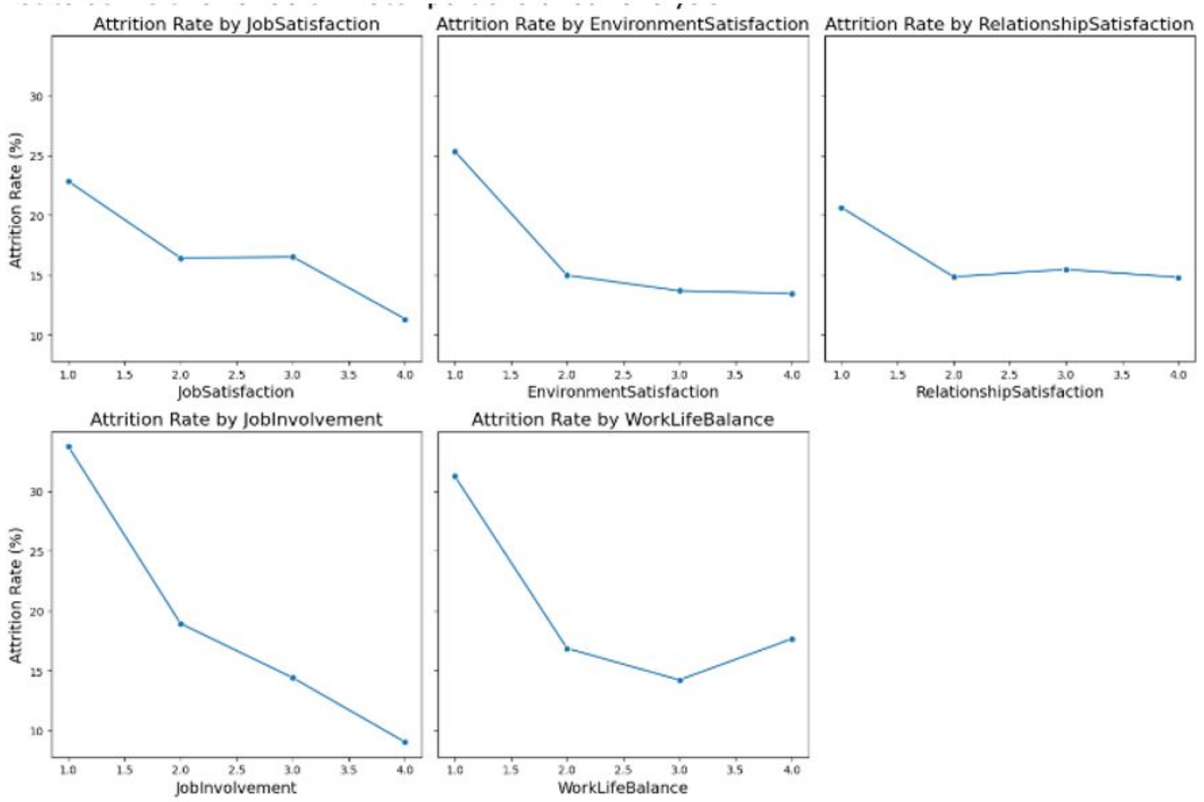
Inference:

1. In the organization there is all kind of employees staying close or staying far from the office.
2. The feature Distance from Home doesn't follow any trend in attrition rate.
3. Employees staying close to the organization are mostly leaving compared to employees staying far from the organization.



Inference:

1. Most of the employees in the organization have completed Bachelors or Masters as their education qualification.
2. Very few employees in the organization have completed Doctorate degree as their education qualification.
3. We can observe a trend of decreasing in attrition rate as the education qualification increases.



Inference:

1. Most of the employees have rated the organization environment satisfaction High & Very High.
2. Though the organization environment satisfaction is high still there's very high attrition in this env
3. Attritior

Inference:

1. More than 60% of employees are having a better work life balance.
2. Employees with Bad Work Life Balance are having Very High Attrition Rate.
3. Other Categories is also having High attrition Rate.

Inference:

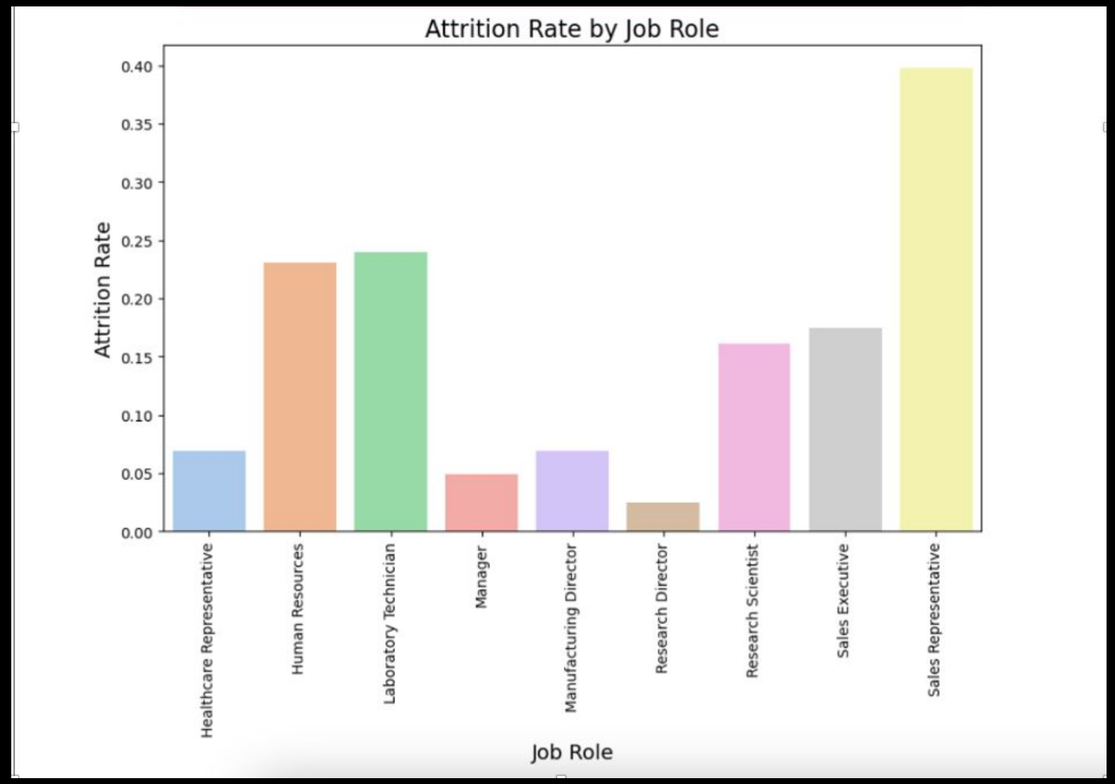
1. Most of
2. Employees who rated their job satisfaction low are mostly leaving the organization.
3. All the categories in job satisfaction is having high attrition rate.

Inference:

1. Most of the employees are having high or very high relationship satisfaction.
2. Though the relationship satisfaction is high there's a high attrition rate.
3. All the categories in this feature are having a high attrition rate.

Inference:

1. More than 60% of employees are having a better work life balance.
2. Employees with Bad Work Life Balance are having Very High Attrition Rate.
3. Other Categories is also having High attrition Rate.

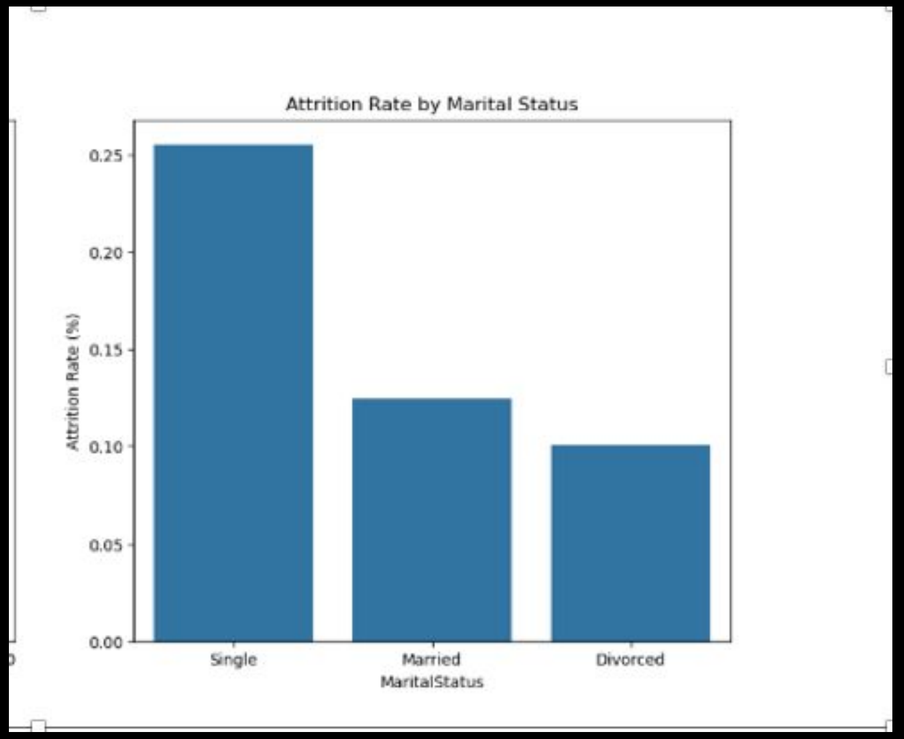


Inference:

1. Most employees are working as Sales executive, Research Scientist or Laboratory Technician in this organization.
2. Highest attrition rates are in sector of Research Director, Sales Executive, and Research Scientist.

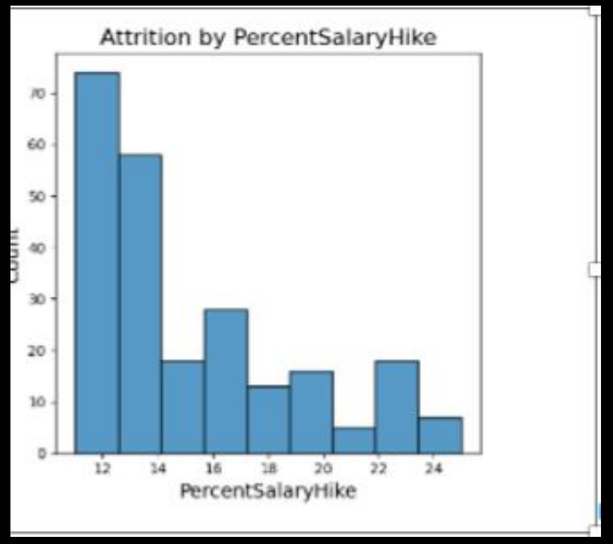
Inference:

1. Most of the employees in the organization are at Entry Level or Junior Level.
2. Highest Attrition is at the Entry Level.
3. As the level increases the attrition rate decreases.



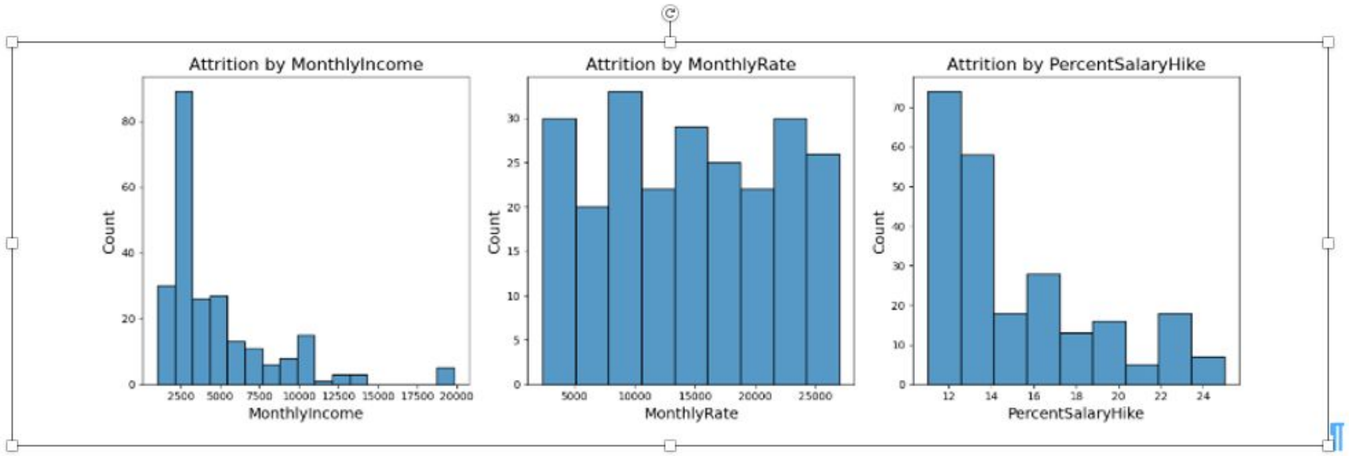
Inference:

1. Most of the employees are married in the organization.
2. The attrition rate is very high of employees who are divorced.
3. The attrition rate is low for employees who are single.



Inference:

1. Very Few employees are getting a high percent salary hike.
2. As the amount of percent salary increases the attrition rate decreases.

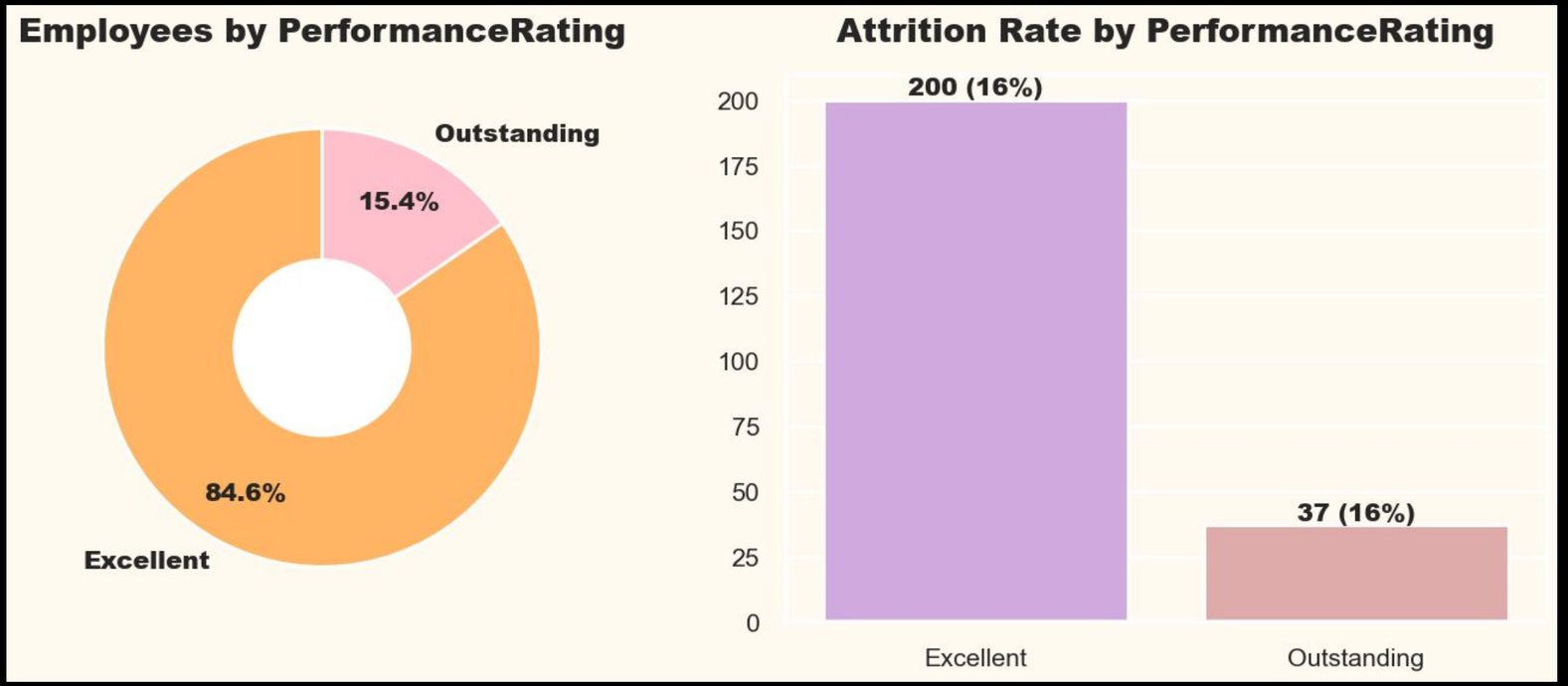


Turnover by Monthly Income:

This chart shows that most of the employees who left had a monthly income in the range of 5,000 to 7,500. There is a significant decrease in the turnover rate for employees with a monthly income above 7,500, indicating that employees with higher salaries tend to stay with the company longer. Turnover by Monthly Rate: The Turnover by Monthly Rate graph does not show a clear pattern between salary levels and turnover rates. Turnover fluctuates randomly across different salary ranges.

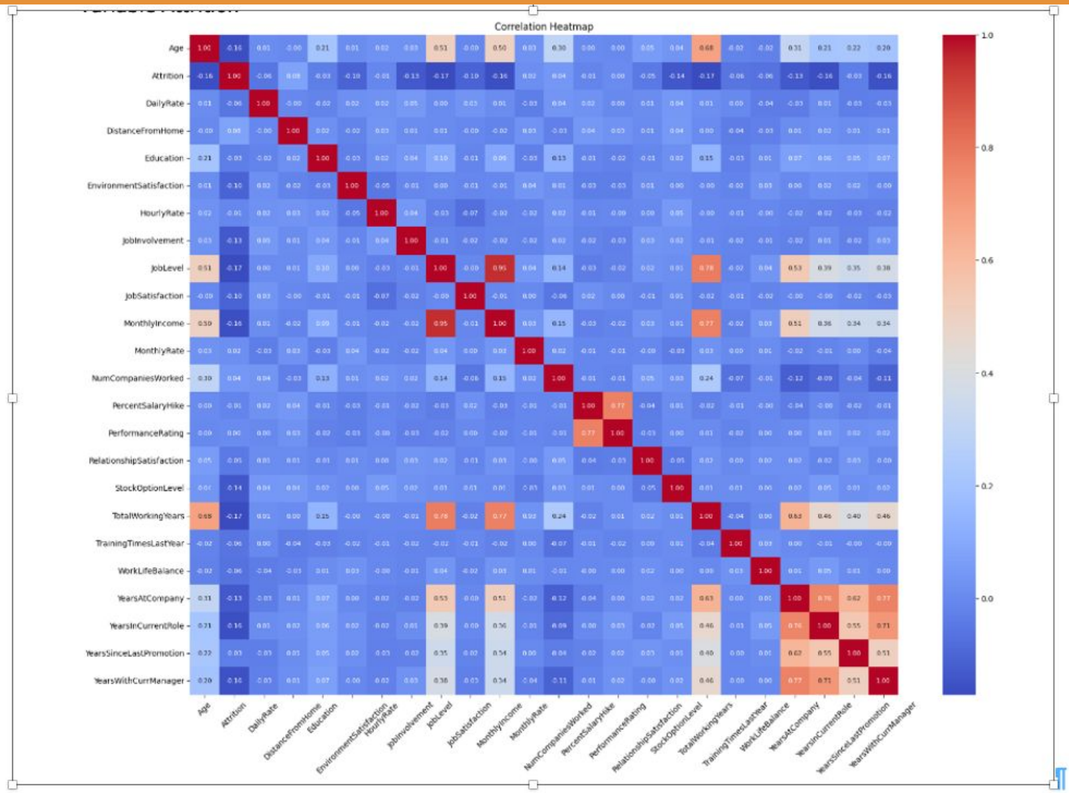
Turnover by Percent Salary Increase:

This chart shows that employees who receive lower salary increases (below 16%) tend to have higher turnover rates. The higher the percentage increase, the lower the turnover rate. This shows that a significant salary increase can be an effective retention factor.



Inference:

1. Most of the employees are having excellent performance rating.
2. Both the categories in this field is having same attrition rate.
3. That's why we can't generate any meaningful insights.



Here we discover some highly correlated features: "YearsWithCurrManager" has a high positive correlation with 'YearsAtCompany'(0.77), and 'YearsInCurrentRole' (0.71) -- this may indicate employees tend to stay with the same manager in the same role. For the above reason, using a cutoff of 0.7 correlation coefficient, the columns MonthlyIncome, TotalWorkingYears, YearsInCurrentRole, and YearsWithCurrManager will be removed, and the columns JobLevel and YearsAtCompany will be retained. It will reduce the possibility of features multicollinearity and will improve model accuracy as well. For MonthlyIncome, the high correlation with job level and sensitivity around salary makes it both redundant and hard to work with in policy change management. For TotalWorkingYears, the data will be encapsulated in other features like YearsAtCompany. For YearsInCurrentRole and YearsWithCurrManager, the data will not pose much additional insights beyond YearsAtCompany, considering organisational structures/functional specializations (e.g. Accounting) are what keep individuals in similar Departments and roles.

Label encoding

```
cat_cols=data_df.select_dtypes(include=['object','category']).columns
dummy_cols=pd.get_dummies(data_df,columns=cat_cols,drop_first=True)
```

In [41]:

```
# Adding these dummy variable to input X
#X = pd.concat([X, dummy_BusinessTravel,dummy_Department,dummy_EducationField,dummy_Gender,dummy_JobRole,dummy_
#dummy_OverTime], axis = 1)
X=pd.concat([X,dummy_cols],axis=1)
```

In [47]:

```
#Split data into train and test set
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_state = 42, stratify=y)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)
X_std = scaler.transform(X)
```

TRAIN AND TEST DATASET

LOGISTIC REGRESSION

In [48]:
```
LR_model=LogisticRegression(solver='liblinear',penalty='l1')
LR_model.fit(X_train_std,y_train)
y_test_pred=LR_model.predict(X_test_std)
y_train_pred=LR_model.predict(X_train_std)

train_accuracy=accuracy_score(y_train,y_train_pred)
train_cf_matrix=confusion_matrix(y_train,y_train_pred)
train_cl_report=classification_report(y_train,y_train_pred)

test_accuracy=accuracy_score(y_test,y_test_pred)
test_cf_matrix=confusion_matrix(y_test,y_test_pred)
test_cl_report=classification_report(y_test,y_test_pred)
```

In [49]:
```
print("TRAINING RESULTS: \n==============================")
print("Accuracy:",train_accuracy)
print("Confusion Matrix:\n",train_cf_matrix)
print("Classification Report:\n",train_cl_report)
print("TEST RESULTS: \n==============================")
print("Accuracy:",test_accuracy)
print("Confusion Matrix:\n",test_cf_matrix)
print("Classification Report:\n",test_cl_report)
```

```
TRAINING RESULTS:
==============================
Accuracy: 0.8979591836734694
Confusion Matrix:
 [[845  18]
 [ 87  79]]
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.98      0.94       863
           1       0.81      0.48      0.60       166

    accuracy                           0.90      1029
   macro avg       0.86      0.73      0.77      1029
weighted avg       0.89      0.90      0.89      1029

TEST RESULTS:
==============================
Accuracy: 0.8820861678004536
Confusion Matrix:
 [[359  11]
 [ 41  30]]
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.97      0.93       370
           1       0.73      0.42      0.54        71

    accuracy                           0.88       441
   macro avg       0.81      0.70      0.73       441
weighted avg       0.87      0.88      0.87       441
```

RANDOM FOREST CLASSIFIER

In [52]:

```
from sklearn.ensemble import RandomForestClassifier
#X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_state = 42, stratify=y)
#X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.25, random_state = 42, stratify=y)
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state = 42, stratify=y)

#from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
#scaler = RobustScaler()
scaler = MinMaxScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)
X_std = scaler.transform(X)

RF_model=RandomForestClassifier(n_estimators=100, bootstrap=False)
RF_model.fit(X_train_std,y_train)
y_test_pred=RF_model.predict(X_test_std)
y_train_pred=RF_model.predict(X_train_std)

train_accuracy=accuracy_score(y_train,y_train_pred)
train_cf_matrix=confusion_matrix(y_train,y_train_pred)
train_cl_report=classification_report(y_train,y_train_pred)

test_accuracy=accuracy_score(y_test,y_test_pred)
test_cf_matrix=confusion_matrix(y_test,y_test_pred)
test_cl_report=classification_report(y_test,y_test_pred)
```

In [53]:

```
print("TRAINING RESULTS: \n================================")
print("Accuracy:",train_accuracy)
print("Confusion Matrix:\n",train_cf_matrix)
print("Classification Report:\n",train_cl_report)
print("TEST RESULTS: \n================================")
print("Accuracy:",test_accuracy)
print("Confusion Matrix:\n",test_cf_matrix)
print("Classification Report:\n",test_cl_report)
```

```
TRAINING RESULTS:
================================
Accuracy: 1.0
Confusion Matrix:
 [[986   0]
 [  0 190]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       986
           1       1.00      1.00      1.00       190

    accuracy                           1.00      1176
   macro avg       1.00      1.00      1.00      1176
weighted avg       1.00      1.00      1.00      1176

TEST RESULTS:
================================
Accuracy: 0.8333333333333334
Confusion Matrix:
 [[238   9]
 [ 40   7]]
Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.96      0.91       247
           1       0.44      0.15      0.22        47

    accuracy                           0.83       294
   macro avg       0.65      0.56      0.56       294
weighted avg       0.79      0.83      0.80       294
```

GRADIENT BOOSTING

Threshold Precision

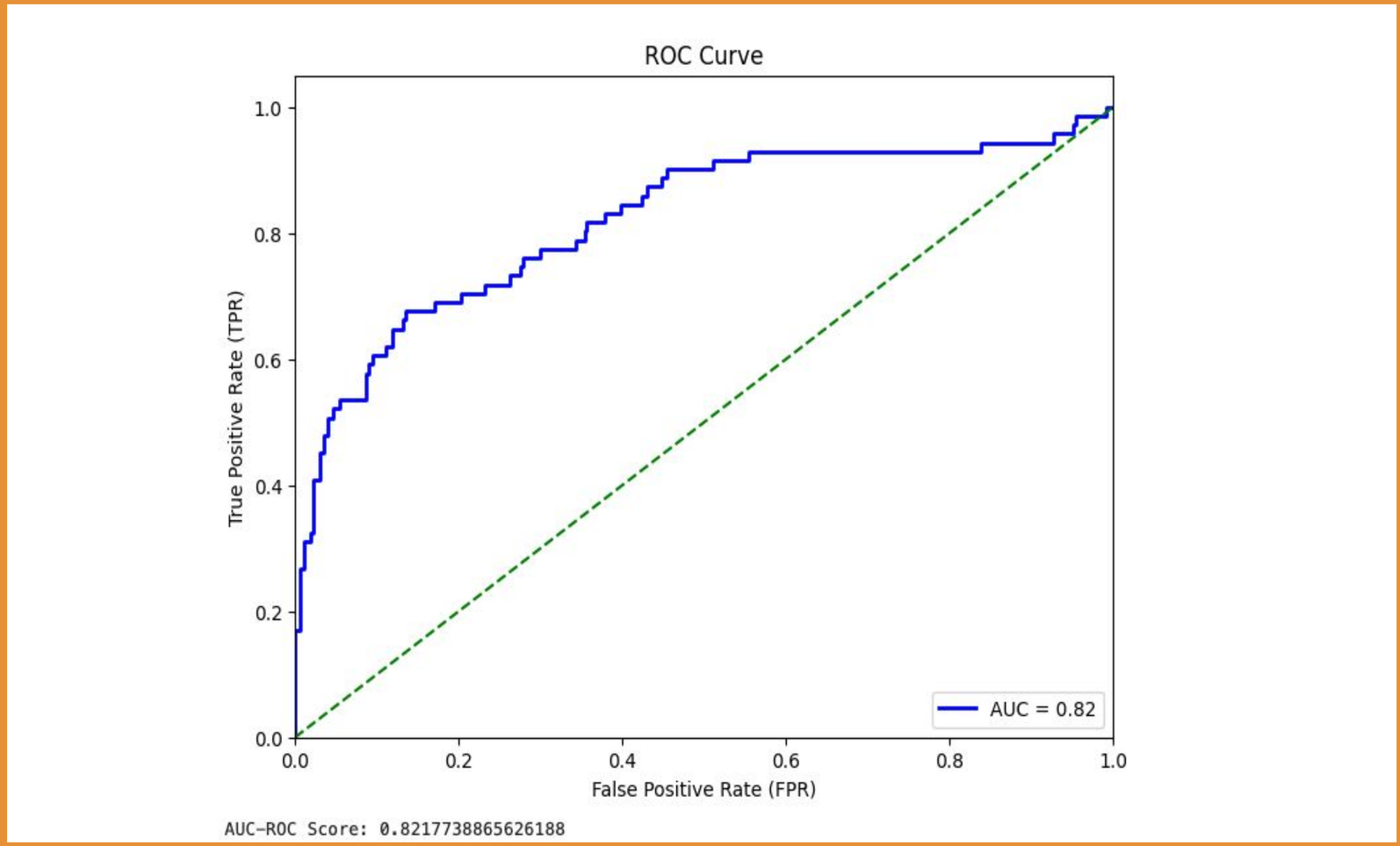
In [58]:

```
# Gradient Boosting
# Hyperparameter Tuning
param_grid_gb = {
    'n_estimators': [100, 200, 300],
    #'learning_rate': [0.01, 0.1, 0.2],
    'learning_rate': [0.01, 0.1],
    'max_depth': [3, 5]
}
cross_validation = StratifiedKFold(n_splits=10,shuffle=True)
# GridSearchCV
grid_search_gb = GridSearchCV(estimator=GradientBoostingClassifier(),param_grid=param_grid_gb,
                cv=cross_validation, n_jobs=-1, scoring = 'roc_auc')
grid_search_gb.fit(X_train_std, y_train)
best_params_gb = grid_search_gb.best_params_
best_estimator_gb = grid_search_gb.best_estimator_
best_estimator_gb.fit(X_train_std, y_train)

y_pred_tuned_gb = best_estimator_gb.predict(X_test_std)
accuracy_tuned_gb = accuracy_score(y_test, y_pred_tuned_gb)
cf_matrix_tuned_gb = confusion_matrix(y_test, y_pred_tuned_gb)
cl_report_tuned_gb = classification_report(y_test, y_pred_tuned_gb)

print("\nBest Parameters (Gradient Boosting):", best_params_gb)
print("Accuracy (Tuned Gradient Boosting):", accuracy_tuned_gb)
print("Confusion Matrix (Tuned Gradient Boosting):\n", cf_matrix_tuned_gb)
print("Classification Report (Tuned Gradient Boosting):\n", cl_report_tuned_gb)

# AUC-ROC
y_prob_gb = best_estimator_gb.predict_proba(X_test_std)[:, 1]
fpr_gb, tpr_gb, thresholds_gb = roc_curve(y_test, y_prob_gb)
roc_auc_gb = roc_auc_score(y_test, y_prob_gb)

plt.figure(figsize=(8, 6))
plt.plot(fpr_gb, tpr_gb, color='blue', lw=2, label=f'AUC = {roc_auc_gb:.2f}')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()

print("AUC-ROC Score (Gradient Boosting):", roc_auc_gb)
```

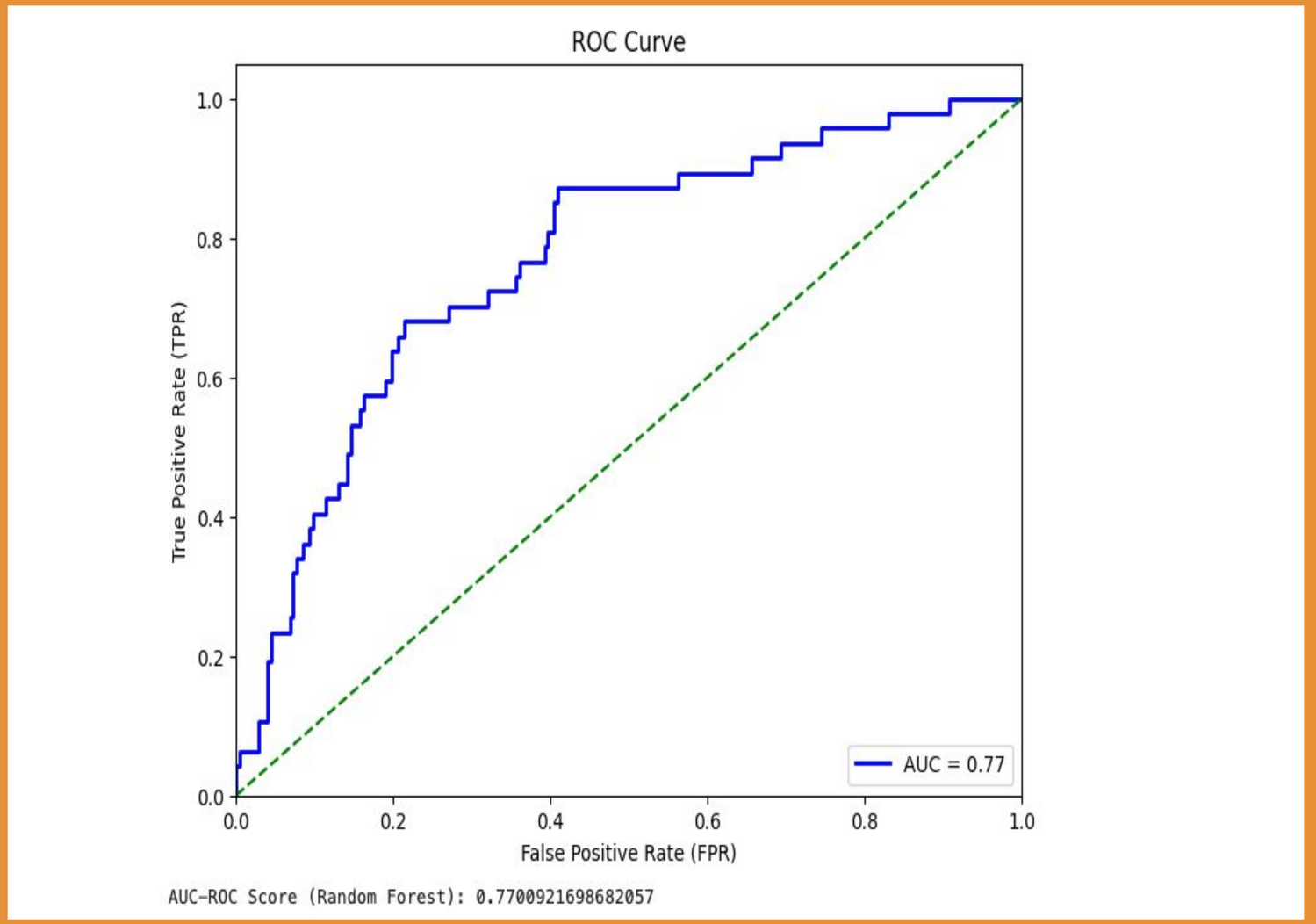
```
Best Parameters (Gradient Boosting): {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 200}
Accuracy (Tuned Gradient Boosting): 0.8401360544217688
Confusion Matrix (Tuned Gradient Boosting):
 [[236  11]
 [ 36  11]]
Classification Report (Tuned Gradient Boosting):
              precision    recall  f1-score   support

           0       0.87      0.96      0.91       247
           1       0.50      0.23      0.32        47

    accuracy                           0.84       294
   macro avg       0.68      0.59      0.61       294
weighted avg       0.81      0.84      0.82       294
```

ROC Curve

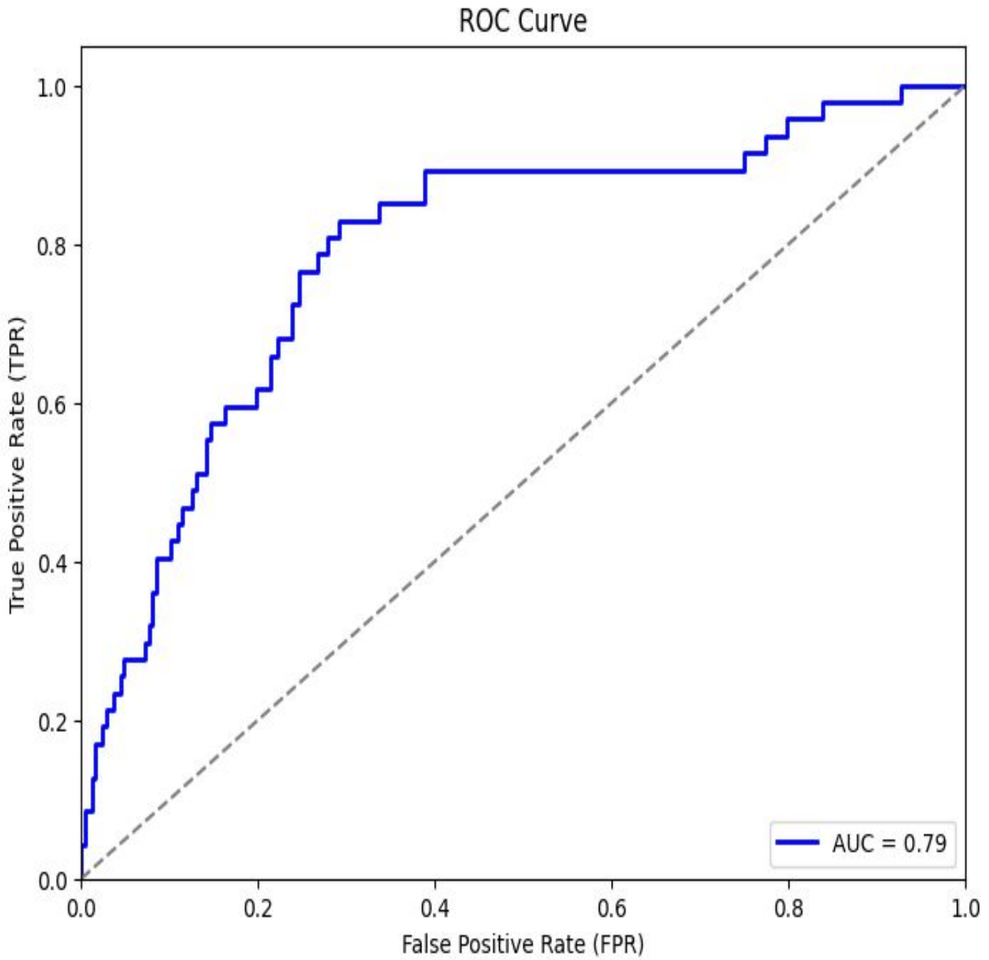
ROC CURVE
(Logistic Regression)



ROC CURVE
(Random Forest Classifier)

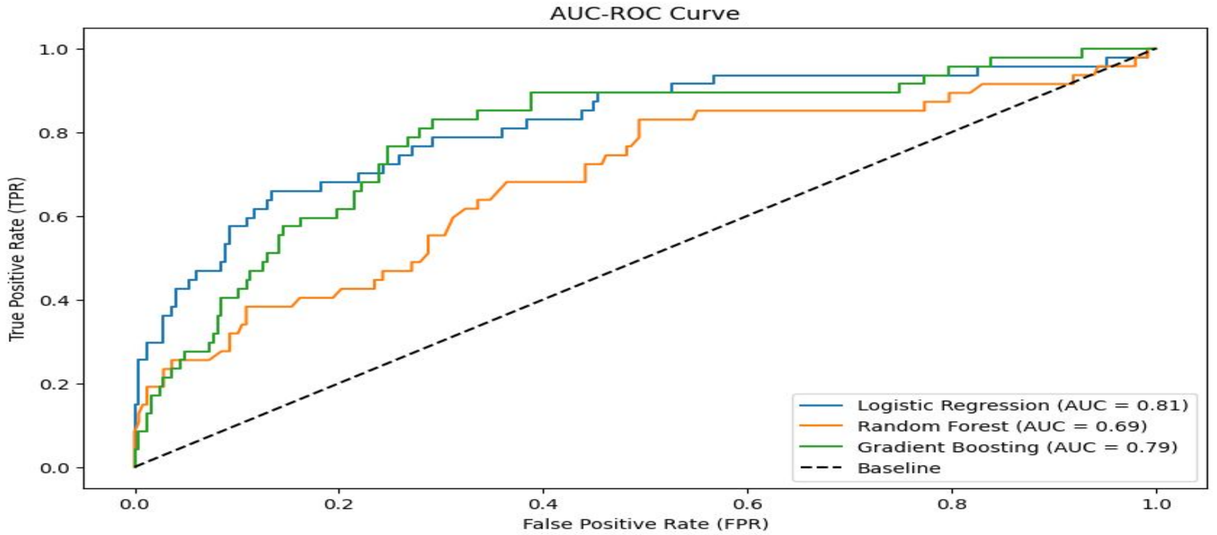


ROC CURVE (Gradient Boosting)



AUC-ROC Score (Gradient Boosting): 0.7924024463778103

Model Comparison



Model Evaluation Metrics:

	Model	AUC	Precision	Recall	F1 Score
0	Logistic Regression	0.814627	0.666667	0.425532	0.519481
1	Random Forest	0.686407	0.227273	0.851064	0.358744
2	Gradient Boosting	0.792402	0.500000	0.234043	0.318841

Looking at the data collectively, Logistic Regression model has the highest AUC score

we can tell again that the logistic regression model emerged as the top performer, achieving an F1 score of approximately 0.5 for Attrition['Yes'] and an AUC of 0.81. This outcome suggests that there is likely the existence of linear relationships between Attrition and the features and Attrition is less likely to depend on non-linear or more complex relationships with other features. Also, when dealing with imbalanced datasets, it is important to point out again that the F1 score can be a better metric than AUC. This is because F1 score balances precision and recall and is less affected by class imbalance. While accuracy score is more commonly used, it is less useful in this context as there is lesser emphasis on the minority class (Attrition - 'Yes') due to the overemphasis of better results in the majority class. Overall, considering the relatively small sample of Attrition['Yes'] data and lack of longitudinal employee data, the results from the logistic regression model is satisfactory and decent in aiding the company to predict Attrition and work on tailored interventions. On the other hand, Random Forest and Gradient Boosting algorithms may not have shined here because of the relatively small/moderate sized dataset. For these 2 ensemble methods, it is likely that we will see better predictive statistics when there is a larger dataset in order to fully exploit their capabilities. In our context, their complexity in building the model might have led to overfitting issues with limited data size.

Conclusion

In conclusion, we embarked on a comprehensive analysis of the IBM HR Analytics Attrition Dataset, from data loading to model evaluation. By implementing and evaluating various machine learning algorithms, we gained insights into which models are effective for predicting employee attrition. The results and visualizations generated throughout the process provide valuable information for decision-makers and HR professionals seeking to understand and mitigate employee attrition within the organization. This project showcases the power of data analysis and machine learning in addressing real-world business challenges.