

[Show code](#)

[Show code](#)

✓ Executive Summary

Employee turnover costs businesses time, money, and talent. This project analyzes IBM's HR dataset to identify key factors behind employee attrition and builds predictive models to support proactive HR decisions. Using Python and popular data science libraries, I created a streamlined workflow for cleaning data, performing EDA, and building models to detect patterns behind voluntary exits — providing a data-driven foundation for improving employee retention.

Business Understanding

- Attrition, the loss of employees over time, can damage productivity, morale, and a company's bottom line. HR teams must understand why employees leave in order to reduce turnover and improve engagement.
- In this analysis, I'll seek to answer key questions:
 - What features are most strongly associated with employee attrition?
 - Can we build a model to help predict which employees are at risk of leaving?
 - How do factors like job satisfaction, income, and workload affect attrition?
 - What insights can support more effective retention strategies?

Answering these questions helps organizations develop smarter, more data-informed HR practices.

[Show code](#)

[Show code](#)

✓ Data Understanding

The dataset contains demographic, job-related, and satisfaction-level data on 1,470 IBM employees. Each row represents an individual employee, with the target variable being `Attrition` (Yes/No).

Key variable categories include:

- **Personal Demographics:** `Age`, `Gender`, `MaritalStatus`, `DistanceFromHome`

- **Job Role and Satisfaction:** JobRole, Department, JobSatisfaction, EnvironmentSatisfaction, WorkLifeBalance
- **Performance and Tenure:** MonthlyIncome, YearsAtCompany, PerformanceRating, YearsSinceLastPromotion
- **Training and Growth:** TrainingTimesLastYear, TotalWorkingYears, EducationField

This diverse set of features enables both descriptive and predictive analysis to understand patterns behind employee attrition.

[Show code](#)

- There are 1,470 records and 35 columns in the dataset.
 - There are no missing values, which makes preprocessing smoother.
- Several categorical columns such as Attrition, BusinessTravel, Department, and EducationField will need encoding before modeling.
- A few columns such as EmployeeNumber, EmployeeCount, Over18, and StandardHours may not provide useful variance and can be dropped later.

[Show code](#)

[Show code](#)

[Show code](#)

[Show code](#)

- Attrition is an **imbalanced binary target**, with approximately 83% of employees staying and 17% leaving.
- Certain job roles like Sales Representative have disproportionately high attrition.
- Lower work-life balance scores and lower monthly incomes appear correlated with higher attrition rates.

[Show code](#)

[Show code](#)

The strongest negative correlations with Attrition include Age, TotalWorkingYears, and YearsAtCompany, suggesting longer-tenured, older employees are more likely to stay. Stronger positive correlations (e.g.,

OverTime, DistanceFromHome) may be red flags contributing to turnover risk.

▼ Feature Engineering

Feature engineering is the process of creating, selecting, or transforming variables to improve model performance and reveal deeper insights. In this attrition project, it's especially important because:

- **It reduces noise and redundancy:** Some variables may contain overlapping or irrelevant information. Feature engineering helps highlight what's truly predictive.
- **It boosts model accuracy:** Thoughtful transformations can reveal hidden patterns—like how work-life balance, promotion frequency, or distance from home interacts with attrition.
- **It supports explainability:** Engineered features often have real-world meaning, which is essential for communicating insights to HR teams and leadership.

The goal is not just to fit a model, but to understand the “why” behind attrition—and feature engineering bridges that gap between raw data and actionable insight.

[Show code](#)

▼ Step 4: Model Building

Now that I've explored the data and engineered key features, it's time to build predictive models that can help identify employees likely to leave the company.

The primary goal is to create a classification model that can predict whether an employee will attrite (Attrition = Yes) based on their characteristics and work experience.

I'll compare the performance of several models:

- **Logistic Regression** – a baseline model that's easy to interpret.
- **Random Forest Classifier** – a powerful ensemble model that handles non-linear relationships well.
- **XGBoost Classifier** – a gradient boosting model known for strong performance in classification problems.

Evaluation Strategy

I'll evaluate model performance using:

- **Accuracy** – overall correctness
- **Precision & Recall** – for class imbalance handling (important since attrition is a minority class)
- **F1-Score** – the harmonic mean of precision and recall
- **ROC-AUC** – to understand how well the model separates the classes

To ensure reliability, I'll also use:

- **Train/Test Split**
- **Cross-validation (optional)**
- **Confusion Matrix & Classification Report**

[Show code](#)

[Show code](#)

▼ Method 1: Logistic Regression with Class Weights

Logistic regression is a strong baseline model for binary classification tasks. However, since attrition is imbalanced (fewer "Yes" than "No"), I'll apply `class_weight='balanced'` to penalize misclassifications of the minority class more heavily.

This approach ensures the model treats both classes fairly. After fitting the model, I'll evaluate its performance and extract feature importance based on the model coefficients.

[Show code](#)

▼ Method 2: Random Forest Classifier

Random Forest is an ensemble learning method that builds multiple decision trees and merges their results for more robust and accurate predictions. It's particularly useful for capturing nonlinear relationships and handling datasets with both numerical and categorical variables.

This method is less sensitive to feature scaling and naturally handles class imbalance to some extent, although tuning hyperparameters can further improve this. After training, I'll evaluate its performance and examine feature importance based on mean decrease in impurity across all trees.

[Show code](#)

▼ Method 3: XGBoost Classifier

XGBoost (Extreme Gradient Boosting) is a powerful and efficient implementation of gradient-boosted decision trees. It is often the go-to algorithm in structured data competitions due to its accuracy, speed, and ability to handle class imbalance with built-in techniques.


XGBoost uses boosting to sequentially improve weak learners, focusing on difficult-to-classify samples. In this step, I'll fit the model, assess performance, and extract feature importance from the trained boosting

ensemble.


[Show code](#)

Model Comparison Summary


Logistic Regression (with Class Weights)

- Simple and easy to interpret
- Fast to train and great as a baseline
- Automatically adjusts for class imbalance using `class_weight='balanced'`
- Performs decently on both classes, with **62% recall** for attrition cases
-  Best when interpretability matters (e.g., in business presentations)

Random Forest Classification

- Captures nonlinear relationships and feature interactions
- Handles both numerical and categorical data well
- Tends to favor the majority class without additional tuning
- Had **very low recall (11%)** for predicting attrition — missed most actual "Yes" cases
-  Less ideal here unless rebalanced or tuned further

XGBoost

- Highly accurate and robust for tabular data
- Built-in ability to handle class imbalance (can tune `scale_pos_weight`)
- Showed the best balance between **precision (71%) and recall (26%)** for predicting attrition
- More complex to train and tune, but worth it for performance
-  Best when **predictive performance is critical**

Final Recommendation:

- Use **Logistic Regression** for quick insights and when model explainability is key.
- Use **Random Forest** for exploring data patterns or building ensembles.
- Use **XGBoost** for optimizing for performance if comfortable tuning model parameters.

Step 6: Model Interpretation & Business Insights

After building and comparing three different models (Logistic Regression with class weights, Random Forest, and XGBoost), I focused on understanding **why employees may leave** and how a company might intervene proactively.

Key Findings

- `OverTime`, `JobRole_Sales Representative`, and `DistanceFromHome` were consistent predictors of attrition across multiple models.
- `MonthlyIncome` and `StockOptionLevel` also had strong influence — employees with lower income or fewer stock options were more likely to leave.
- `JobInvolvement` and `WorkLifeBalance` showed meaningful effects, suggesting workplace experience impacts attrition risk.

Model Interpretability Summary

- **Logistic Regression** gave the clearest insight into *how* features impact attrition. It's great for stakeholders who want transparency and explanation.
- **Random Forest** was highly accurate but struggled to capture the minority class (attrition). It's useful when precision on the majority class is critical.
- **XGBoost** offered a good balance between performance and insight. It outperformed others in recall for the attrition class and can be tuned for even better results.

Business Recommendation

If I were advising HR leadership, I'd recommend:

- **Monitoring and supporting employees working overtime** or who live far from the office.
- **Re-evaluating compensation and benefits packages** (especially for Sales Representatives and lower-income roles).
- **Prioritizing job involvement and work-life balance initiatives**, possibly via regular pulse surveys or team-level reviews.

This approach could reduce attrition risk and boost employee satisfaction, especially among at-risk groups.

