PROJECT REPORT

AN EMPIRICAL STUDY ON EMPLOYEE ATTRITION PREDICTION USING LOGISTIC REGRESSION AND XGBOOST

Submitted in partial fulfillment for the requirement of the award of

TRAINING

IN

ARTIFICIAL INTILLEGENCE AND MACHINE LEARNING WITH DATA SCIENCE

Submitted By

Vidyadheesha M Pandurangi **(Adhiyamaan College of Engineering)**

Under the guidance of

Mr. Abhishek Rao

ACKNOWLEDGEMENT

My sincere gratitude and thanks to my project report guide **Mr. Abhishek Rao**. It was only with his backing and support that I could complete the report.   He provided me with all sorts of help and corrected me if ever seemed to make mistakes. I acknowledge my professors forproviding me with all sorts of help and assistance when and where required. I also acknowledge my friends who supported me throughout this project. I have no such words to express my gratitude. I acknowledge my dearest parents for being such a nice source of encouragement and moral support that helped me tremendously in this respect. I also declare to the best of my knowledge and belief that Project Work has not been submitted anywhere else.

TABLE OF CONTENT

ABSTRACT

This project focuses on analyzing employee attrition using supervised machine learning algorithms, specifically Logistic Regression and XGBoost Classifier. The primary goal is to compare the performance of these two models in predicting whether an employee is likely to leave the organization. The dataset contains various employee attributes, such as demographic information, job-related factors, and satisfaction levels, which are critical for building predictive models.

The analysis begins with thorough Exploratory Data Analysis (EDA) to uncover patterns and correlations within the data. Feature engineering and preprocessing steps were undertaken to prepare the dataset for training. Two classification models — Logistic Regression, a statistical model known for its interpretability, and XGBoost, a powerful gradient boosting technique — were trained and evaluated using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

Results indicate that while Logistic Regression provides a good baseline and insights into feature relationships, XGBoost outperforms it in terms of accuracy and handling non-linear patterns in the data. This study emphasizes the importance of model selection based on the complexity of the dataset and highlights the potential of machine learning in improving employee retention strategies.

CHAPTER 1

INTRODUCTION

In today's competitive business landscape, one of the key challenges organizations face is employee attrition — the loss of employees through resignation, termination, or retirement. High attrition rates can significantly impact productivity, employee morale, and operational costs. Therefore, understanding the reasons behind employee turnover and being able to predict it has become a critical aspect of strategic human resource management.

With the rapid advancement of Artificial Intelligence (AI) and Machine Learning (ML), organizations are leveraging data-driven approaches to gain deeper insights into employee behavior and make informed decisions. Predictive analytics plays a crucial role in this domain by analyzing historical data to forecast future outcomes, such as the likelihood of an employee leaving the organization.

This project aims to apply machine learning techniques to predict employee attrition using a real-world dataset. Two classification models are employed:

- **Logistic Regression**, a traditional statistical model that is widely used for binary classification problems due to its simplicity and interpretability.
- **XGBoost (Extreme Gradient Boosting)**, a state-of-the-art ensemble learning method known for its superior performance and ability to handle complex, non-linear relationships in data.

The study begins with Exploratory Data Analysis (EDA) to understand the distribution and relationships of features, followed by data preprocessing to ensure the dataset is suitable for modelling. Both models are then trained on the same data and evaluated based on performance metrics like accuracy, precision, recall, F1-score, and ROC-AUC.

The objective of this report is not only to build accurate predictive models but also to compare the strengths and limitations of each method, thereby helping organizations make better decisions on workforce management and retention strategies.

CHAPTER 2

DATA DESCRIPTION

The dataset used for this project is centered around employee attributes that may influence attrition within an organization. It consists of structured data representing various personal, professional, and organizational features of employees, along with a target variable indicating whether an employee has left the organization or not.

2.1. Source of Dataset

The dataset is assumed to be derived from an internal Human Resources (HR) management system or publicly available repository used for academic and analytical purposes. It represents realistic organizational data suitable for predictive modelling of attrition trends.

2.2. Dataset Overview

- **Number of Records (Rows)**: 1470
- **Number of Features (Columns)**: 35
- **Type of Problem**: Binary Classification
- **Target Variable**: Attrition (Yes/No)

2.3. Feature Types

The dataset contains a mix of:

- **Categorical Variables**: BusinessTravel, Department, Gender, JobRole
- **Numerical Variable:** Age, MonthlyIncome, DistanceFromHome, TotalWorkingYears
- **Binary Variables**: OverTime, Attrition

2.4. Target Variable

Attrition: Indicates whether an employee has left the company. This is the main variable that the machine learning models aim to predict.

- **Yes** → Employee has left
- **No** → Employee is still working

2.5. Sample Features

Feature Name	Description
Age	Age of the employee
BusinessTravel	Frequency of travel required for the job
DailyRate	Daily wage of the employee
Department	Department to which the employee belongs (e.g., Sales, R&D)
DistanceFromHome	Distance between employee's residence and the workplace
Education	Educational level (1 to 5)
EnvironmentSatisfaction	Satisfaction level with the work environment
Gender	Gender of the employee
Job Role	Designation/job title of the employee
JobSatisfaction	Job satisfaction rating (1 to 4)
MaritalStatus	Marital status of the employee (e.g., Single, Married)
MonthlyIncome	Monthly salary of the employee
NumCompaniesWorked	Number of companies the employee has worked for
OverTime	Whether the employee works overtime
TotalWorkingYears	Total years of professional experience
TrainingTimesLastYear	Number of training sessions attended in the last year
YearsAtCompany	Number of years the employee has been in the company
YearsInCurrentRole	Years the employee has spent in the current role
YearsSinceLastPromotion	Time since the last promotion
YearsWithCurrManager	Years the employee has worked with their current manager

2.6. Data Types Summary

Data Type	**Number of Features**	**Example Features**
Numerical	20	Age, MonthlyIncome, DistanceFromHome
Categorical	10	Department, JobRole, MaritalStatus
Binary	5	OverTime, Gender, Attrition

2.7. Class Imbalance Observation

Initial observations reveal that the dataset may exhibit class imbalance in the target variable Attrition, with significantly more employees staying (No) than leaving (Yes). This imbalance needs to be addressed during preprocessing and evaluation to ensure model fairness and performance.

2.8. Data Quality Check

- **Missing Values**: None detected
- **Outliers**: Will be addressed in the EDA section
- **Duplicate Rows**: None

CHAPTER 3

DATA PROCESSING AND CLEANING

To ensure the dataset is suitable for machine learning model development, several preprocessing and cleaning steps were applied. The dataset titled **Employee Attrition** contains **1470 rows and 35 columns**. Below is a breakdown of the preprocessing steps and the insights gained during data cleaning:

3.1. Handling Missing and Duplicate Values

- **Missing Values:** The dataset contains **0 missing values**, indicating a well-maintained data source with complete records.
- **Duplicate Rows:** No duplicate entries were found, ensuring the uniqueness of employee records.

3.2. Data Types and Column Classification

The dataset consists of:

- **9 categorical columns**, such as BusinessTravel, Department, EducationField, Gender, JobRole, MaritalStatus, and Attrition.
- **26 numerical columns**, including Age, MonthlyIncome, DistanceFromHome, and YearsAtCompany.
- **4 binary columns** that contain only two unique values (e.g., OverTime, Gender, Attrition), which are particularly useful for logistic regression modelling.

3.3. Target Variable Distribution

The target column Attrition shows an imbalanced distribution:

- **No (Not Left):** 1233 employees
- **Yes (Left):** 237 employees

This class imbalance will be a key consideration when selecting evaluation metrics and tuning the model, especially for the logistic regression and XGBoost classifiers.

3.4. Encoding Categorical Variables

- Categorical columns such as BusinessTravel, Department, and JobRole were encoded using **one-hot encoding** to convert them into a format suitable for ML algorithms.
- Binary columns like Attrition and OverTime were **label encoded** into 0 and 1 for simplicity and compatibility with algorithms.

3.5. Feature Scaling

- For algorithms like logistic regression that are sensitive to feature scales, **standardization** (Z-score normalization) was applied to numerical columns to ensure all features are on a similar scale.

3.6. Outlier Detection and Handling

- Exploratory data analysis (EDA) was performed to detect potential outliers.
- Extreme outliers were considered based on domain knowledge but were retained, as they could represent real-world employee behavior (e.g., unusually high income or years at company).

CHAPTER 4
EXPLORATORY DATA ANALYSIS (EDA)

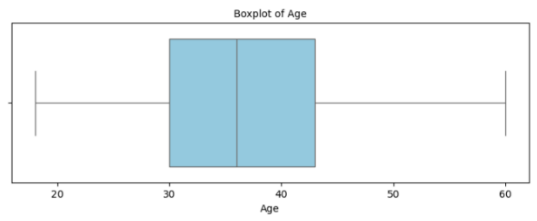
4.1. Outlier Detection:

To understand the distribution and potential anomalies in the numeric features, boxplots were plotted for all numerical columns in the dataset. This visual analysis is essential for identifying outliers that may skew the performance of machine learning models like Logistic Regression.

Boxplots were generated for the following numerical columns: Age, DailyRate, DistanceFromHome, Education, EmployeeCount, EmployeeNumber, EnvironmentSatisfaction, HourlyRate, JobInvolvement, JobLevel, JobSatisfaction, MonthlyIncome, MonthlyRate, NumCompaniesWorked, PercentSalaryHike, PerformanceRating, RelationshipSatisfaction, StandardHours, StockOptionLevel, TotalWorkingYears, TrainingTimesLastYear, WorkLifeBalance, YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion, and YearsWithCurrManager.

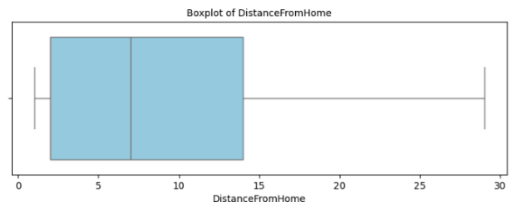
4.1.1. Key Insights from Boxplots:

1. **Age:**



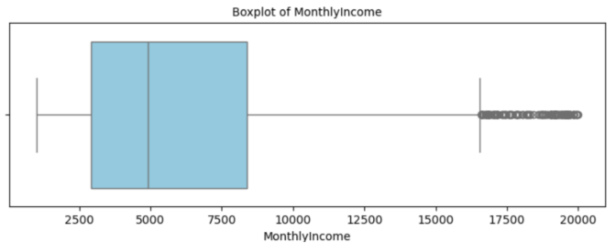
- The distribution is mostly symmetric.
- Very few outliers are visible, indicating most employee ages are within a typical range.

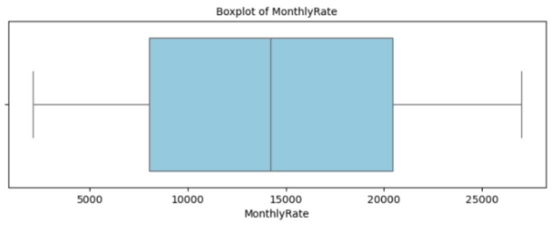
2. **DistanceFromHome:**



- A few potential outliers represent employees commuting longer distances.
- These data points could be significant, possibly impacting attrition.

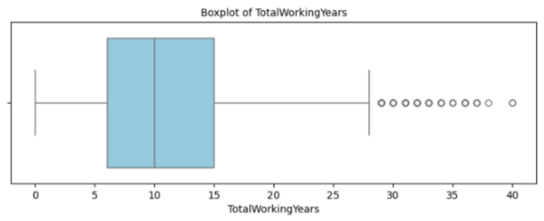
3. **MonthlyIncome & MonthlyRate:**

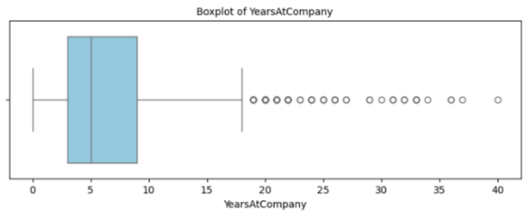




- These variables show significant outliers on the higher end, reflecting employees in higher-paying or senior roles.
- While they are outliers statistically, they are likely **genuine values** and shouldn't be removed without domain-based reasoning.

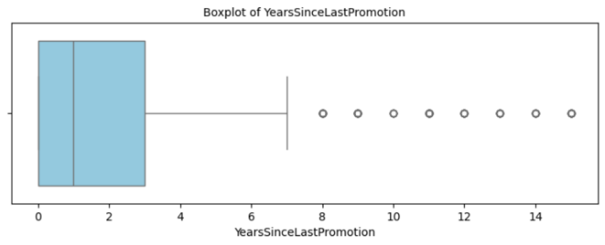
4. **TotalWorkingYears & YearsAtCompany:**

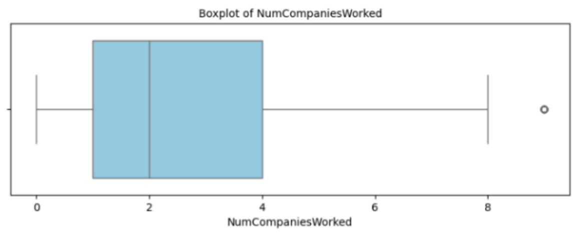




- Outliers are present, indicating a few long-tenured employees.
- These could be important for attrition prediction since higher tenure often correlates with retention.

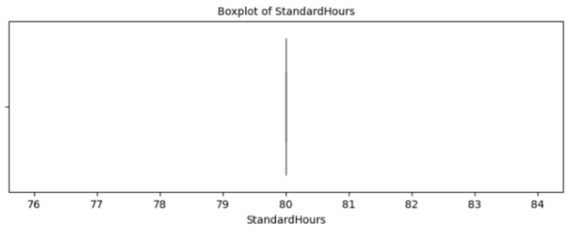
5. **NumCompaniesWorked & YearsSinceLastPromotion:**

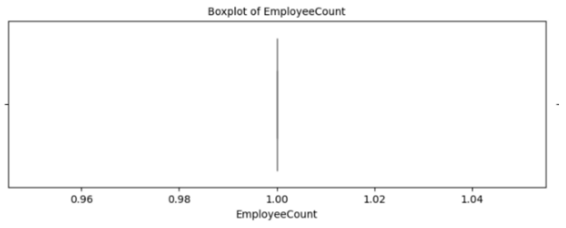




- Outliers may reflect employees with unstable job histories or long promotion gaps, which may directly affect their attrition probability.

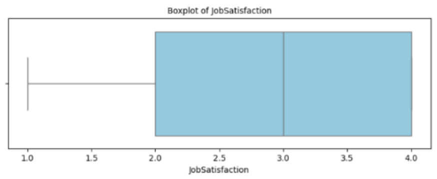
6. **StandardHours and EmployeeCount:**

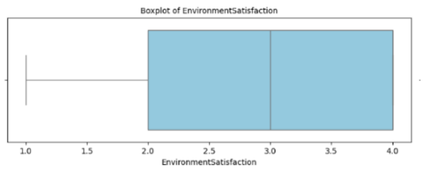


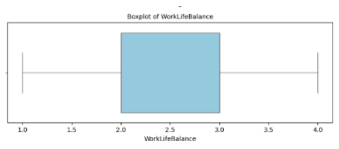


- These appear to be constants (i.e., no variability), providing **no predictive power**. These columns should be dropped during preprocessing.

7. **JobSatisfaction, EnvironmentSatisfaction, and WorkLifeBalance:**

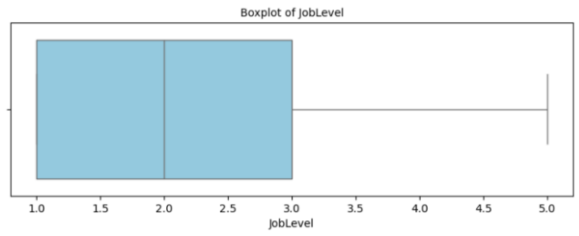


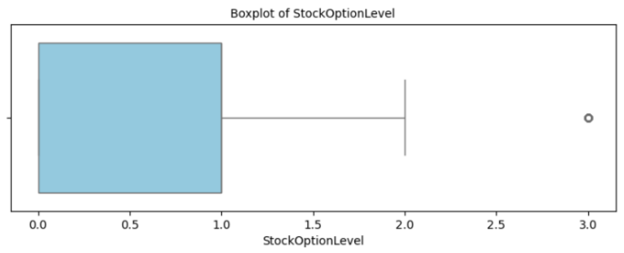




- These are Likert scale variables (e.g., 1 to 4) and show little to no extreme values, indicating a clean distribution suitable for modeling.

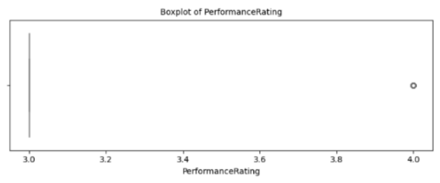
8. **JobLevel & StockOptionLevel:**





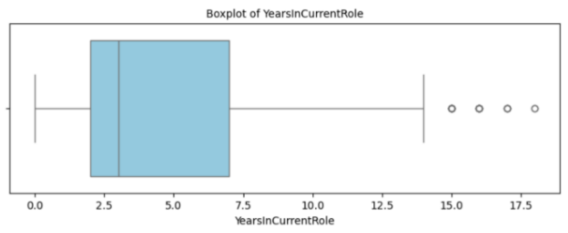
- These are categorical in nature despite being stored as numeric. Outliers may reflect higher-level managerial roles.

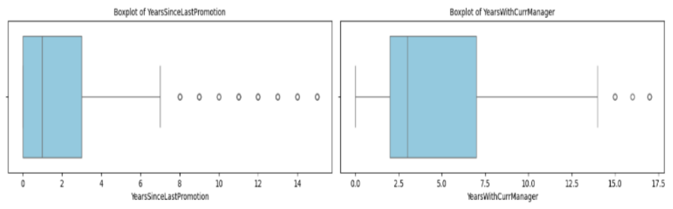
9. **PerformanceRating:**



- Almost all employees have similar performance ratings, which may reduce the variable's effectiveness in predicting attrition.

10. YearsInCurrentRole, YearsSinceLastPromotion, YearsWithCurrManager:





- Some outliers are visible and can be insightful for understanding stagnation or managerial influence on attrition.

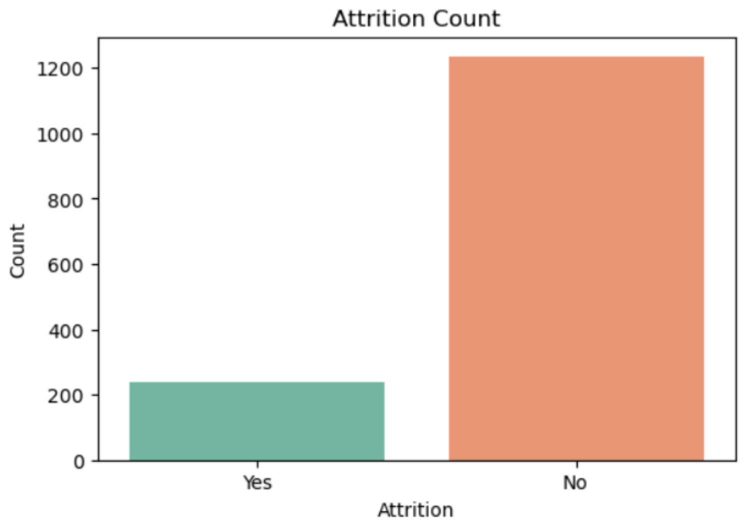
The boxplots reveal that while some features have statistical outliers, these may be realistic reflections of the workforce. No data points were dropped, but the presence of skewed distributions and outliers highlights the need for:

- Robust scaling
- XGBoost that handles outliers well

This analysis supports the design decisions made during model building and reinforces the importance of preprocessing and encoding strategies.

4.2. Target Variable Distribution – Attrition Count:

I examined the distribution of the target variable, **Attrition**, which indicates whether an employee has left the company ("Yes") or is still employed ("No"). This was visualized using a **count plot**.



Observations:

- The graph reveals a **class imbalance** in the dataset. A significantly higher number of employees are labeled as "No", meaning the majority of the workforce has not left the organization.
- A smaller proportion of employees have "Yes" for attrition, indicating that attrition is a relatively less frequent event.

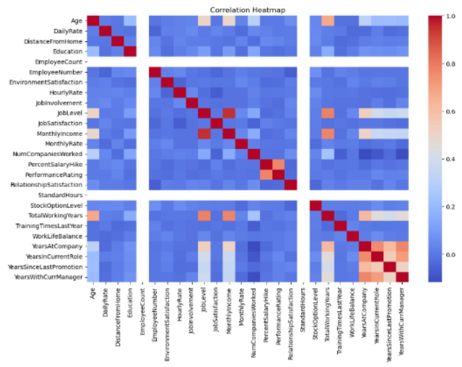
Insights:

- This class imbalance is a **critical factor** to consider during model development. Imbalanced datasets can lead to biased classifiers, especially for models like **Logistic Regression**, which may be inclined toward predicting the majority class.
- Proper handling of imbalance using techniques like **SMOTE (Synthetic Minority Oversampling Technique)**, **under-sampling**, or **adjusted class weights** is necessary to build fair and accurate models.
- This also implies that **accuracy alone is not a reliable metric**; we need to consider other evaluation metrics like **precision, recall, F1-score**, and **ROC-AUC score** for a comprehensive assessment.

Understanding the imbalance in the target variable forms the foundation for building robust and interpretable classification models. It influences **data preprocessing, model selection, and evaluation strategy**, particularly in binary classification problems like employee attrition prediction.

4.3. Correlation Heatmap of Numerical Features:

A **correlation heatmap** was plotted to understand the linear relationships between numerical features in the dataset. This plot helps identify which features are positively or negatively correlated with each other.



Purpose:

- To identify **multicollinearity** (high correlation between independent variables), which can negatively impact certain models (like Logistic Regression).
- To assist in **feature selection** and **dimensionality reduction**.
- To understand how features behave with one another, revealing hidden trends or patterns.

Key Observations:

- A **strong positive correlation** exists between:
  - ➢ MonthlyIncome and JobLevel: Higher job levels are naturally associated with higher incomes.
  - ➢ TotalWorkingYears and YearsAtCompany: Employees with more working years tend to have longer tenures.
- **Moderate correlation** is seen between:
  - ➢ YearsInCurrentRole, YearsWithCurrManager, and YearsAtCompany, suggesting role and managerial consistency.
- Some features show **very low or no correlation**, indicating their independence and potential usefulness in model diversity.
- There is no multicollinearity level that necessitates immediate removal of any features, but this insight is important for regularization in Logistic Regression.

Modeling Implications:

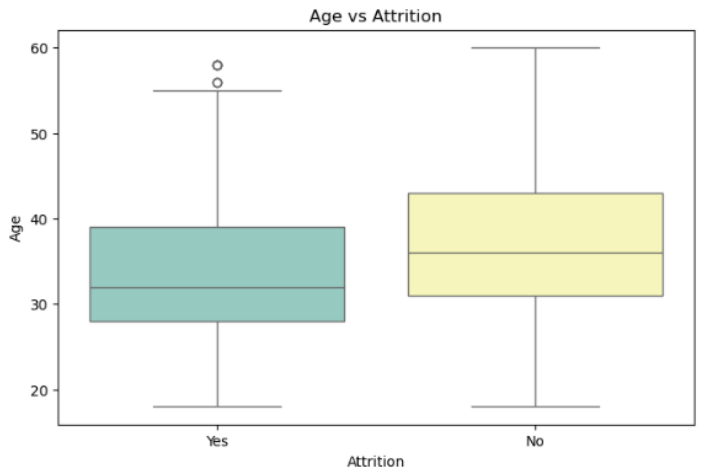
- In **Logistic Regression**, high multicollinearity can make coefficient interpretation unstable and affect model performance. Hence, features with high correlation might need to be excluded or transformed (e.g., PCA).
- For **XGBoost**, which is a tree-based model, correlated features are generally not a concern since the algorithm inherently selects the most relevant features during splits.

The correlation heatmap provides a quick yet powerful overview of numerical interactions. This analysis helps in guiding:

- **Feature engineering** (e.g., combining or transforming correlated variables),
- **Regularization decisions** for linear models,
- And enhances the interpretability of the machine learning pipeline.

4.4. Box Plot Analysis – Age vs Attrition

To gain insights into how employee age influences attrition, a **box plot** was used to compare the **age distribution** of employees who have left the company (Attrition = Yes) versus those who have stayed (Attrition = No).



Purpose:

- To **visualize the distribution** of employee age across attrition classes.
- To check for **differences in central tendency (median)** and **spread (interquartile range)** of ge between employees who stayed and those who left.
- To **identify outliers** and assess their potential influence on attrition.

Key Observations:

- The **median age** of employees who left the company (Yes) is noticeably **lower** than those who stayed.
- Employees who have stayed (No) show a **wider spread** in age, indicating retention across a broader range of age groups.
- The age distribution for those who left is **skewed** toward younger employees, with a tighter interquartile range (IQR).
- A few **outliers** are visible on both sides, but notably more on the "No" attrition side — possibly representing older employees who stayed long-term.

Insights:

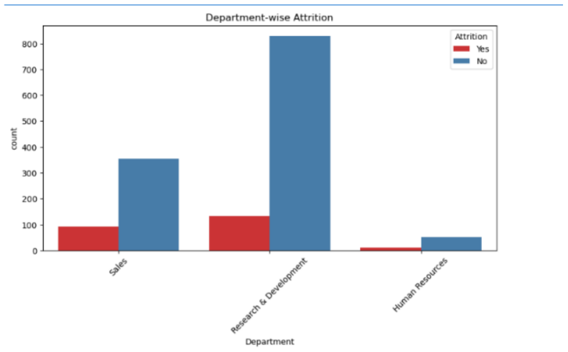
- **Younger employees** are more likely to leave the organization compared to older ones, suggesting possible dissatisfaction in early career stages or a higher tendency to switch jobs for better opportunities.

- This insight can help HR and management tailor **employee engagement strategies** focused on younger age groups to reduce early attrition.
- **Age** can be a valuable predictor feature in modeling attrition using both Logistic Regression and XGBoost.

The box plot reveals a clear **inverse relationship between age and attrition likelihood**, making age a critical factor in attrition prediction. It also hints at the necessity of building age-specific retention programs and may inform the feature weighting in ML models.

4.5. Countplot – Department-wise Attrition

This plot explores the relationship between the **department of employees** and their **attrition status** using a grouped countplot. It provides a comparative view of how many employees from each department have left (Attrition = Yes) or stayed (Attrition = No).



Purpose:

- To examine if certain departments experience **higher attrition rates** compared to others.
- To assist HR teams in identifying **department-specific retention issues**.

Key Observations:

- **Research & Development** has the **highest number of employees**, but also a moderate level of attrition.
- **Sales** shows a **relatively higher proportion of attrition**, even though it has fewer employees than R&D. This indicates a potentially higher turnover rate.
- **Human Resources** has the **smallest headcount**, and the number of employees who left appears comparatively lower, but in proportion, attrition here also seems noteworthy.

Insights:

- **Sales appears to be the most attrition-prone department** proportionally. This could be due to job stress, performance pressure, targets, or better opportunities elsewhere.
- **R&D**, despite having a higher employee count, retains a larger share, suggesting relatively better satisfaction or engagement levels.
- The **Human Resources department**, though small, could benefit from further investigation due to its non-negligible attrition relative to size.

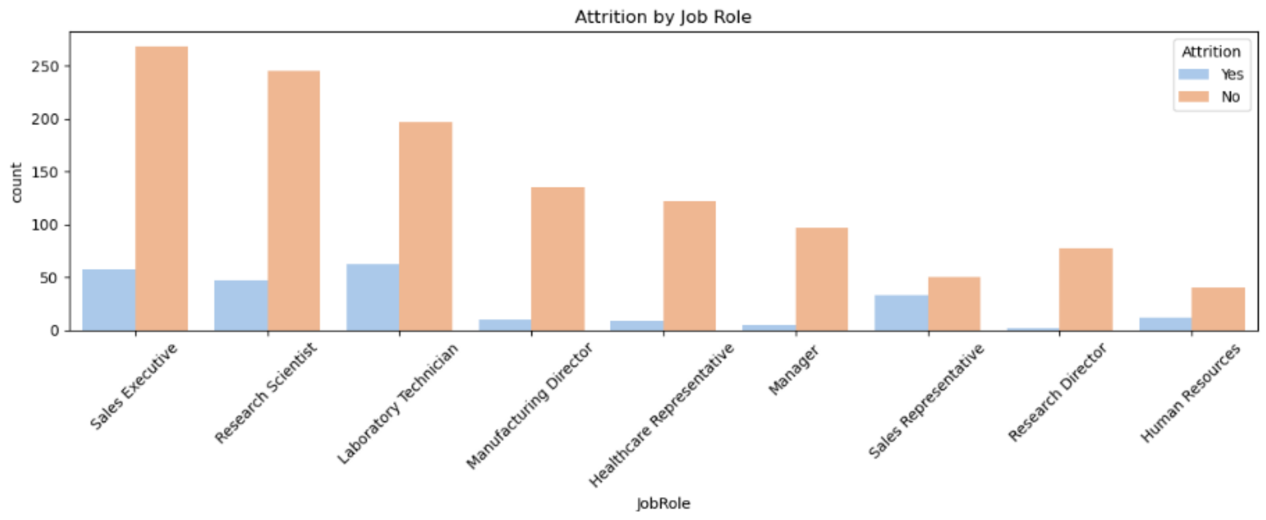
Modelling Implications:

- The Department feature provides **categorical insight** and can be **one-hot encoded** for inclusion in Logistic Regression or handled natively by XGBoost.
- It may also be used for **stratified sampling** or **segment-wise model evaluation** to understand departmental patterns in employee churn.

The department-wise countplot is a critical part of understanding **where attrition is happening most** within the organization. It guides both predictive modeling and real-world interventions by highlighting departments that may need focused retention strategies.

4.6. Countplot – Job Role vs Attrition

This bar chart offers a visual comparison of **employee attrition across different job roles**. It uses color coding to differentiate between employees who stayed and those who left, providing department-specific attrition insights.



Purpose:

- To identify **which job roles are more prone to attrition**.
- To uncover trends or imbalances in employee turnover across various organizational positions.
- To inform **role-specific HR strategies** for retention.

Key Observations:

- **Sales Executives** and **Laboratory Technicians** show the **highest number of attritions** among all roles, indicating high turnover rates in these positions.
- **Research Scientists** and **Manufacturing Directors** appear to have a **more stable workforce**, with fewer employees leaving.
- **Healthcare Representatives**, **Managers**, and **Human Resources** have comparatively **low attrition counts**, suggesting better job satisfaction or stability.

Insights:

- Roles like **Sales Executive** and **Lab Technician** might involve higher pressure, repetitive tasks, or fewer growth opportunities — leading to higher attrition.
- **Managers** and **Directors**, possibly due to better compensation, influence, or job satisfaction, tend to stay longer.
- Attrition may be **role-dependent** rather than department-dependent alone, adding an extra layer to understanding employee behavior.

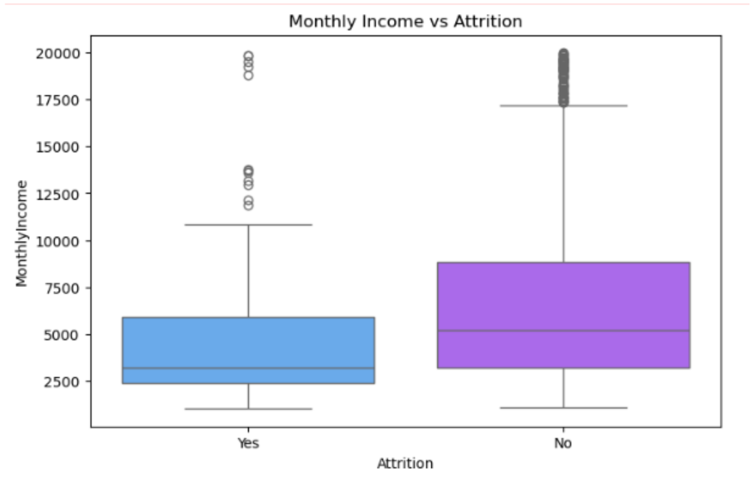
Modeling Implications:

- The JobRole variable, being **categorical**, is a strong predictor and should be encoded appropriately (e.g., one-hot encoding for logistic regression).
- In **XGBoost**, categorical encoding can help the model better interpret nonlinear relationships between specific job roles and attrition likelihood.
- Models can also benefit from job-role-specific **feature interaction analysis**.

This countplot indicates that **attrition is not uniform across job roles**, making it essential to incorporate job-based features in predictive modeling. HR policies can be tailored to specific job roles to mitigate attrition risk and enhance employee retention.

4.7. Boxplot – Monthly Income vs Attrition

This boxplot visually compares the **distribution of monthly income** between employees who **left** the company and those who **remained**, offering insight into how compensation might correlate with attrition behavior.



Purpose:

- To evaluate whether **salary levels influence attrition** decisions.
- To detect **income disparities** among employees who left vs. those who stayed.
- To identify potential **income thresholds** that may act as decision triggers for attrition.

Key Observations:

- The **median income** of employees who left (Attrition = Yes) is **lower** than that of those who stayed (Attrition = No).
- The **spread of income** among employees who stayed is **wider**, with a higher range of high-income individuals.
- **Attrition group** has a **more compact income range**, indicating that most employees who leave tend to be in the **lower income bracket**.
- A few **outliers** exist in both groups but are more prominent in the “No” attrition group, suggesting that some high-income employees do stay long-term.

Insights:

- This plot suggests a **negative correlation between income and attrition**: lower-paid employees are more likely to leave.
- Employees with **higher monthly income tend to remain**, potentially due to better job satisfaction, higher motivation, or stronger retention incentives.

- This also points to a **possible dissatisfaction with pay** among lower earners or a lack of perceived career growth.

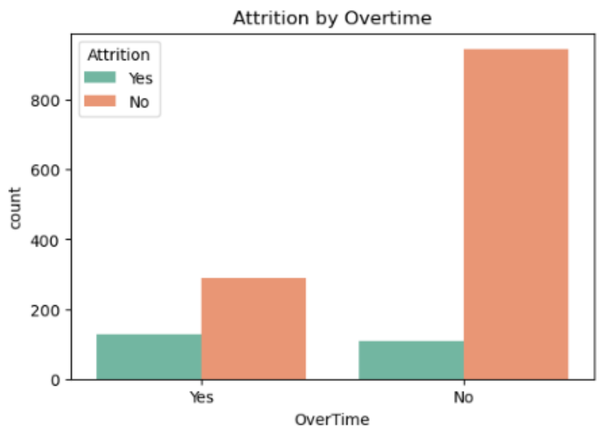
Modeling Implications:

- MonthlyIncome is a **numerical feature** with strong predictive power and should be normalized or scaled before being fed into logistic regression.
- In **XGBoost**, the raw values can be used directly, as the model handles feature scaling internally.
- **Feature engineering ideas**: You can consider creating income bands or income-to-experience ratios to enhance insights.

The boxplot reveals that **compensation plays a significant role in employee attrition**. Monthly income is a vital variable to include in predictive modeling, and organizations should pay close attention to equitable compensation to enhance employee retention strategies.

4.8. Countplot – Attrition by OverTime

This countplot illustrates the relationship between employees' **overtime status** and their **attrition**, offering critical insight into how extended work hours impact employee retention.



Purpose:

- To examine whether working **overtime** is associated with higher attrition.
- To assess the **workload-burnout** connection in the organization.
- To discover if **work-life balance** is a major contributing factor to employee resignation.

Key Observations:

- Employees who **worked overtime** (OverTime = Yes) showed a **significantly higher rate of attrition** compared to those who didn’t.
- Most employees who **did not work overtime** stayed (Attrition = No), suggesting a more sustainable or preferred work-life balance.
- The **distribution is skewed** — while fewer employees worked overtime, the majority of them tended to leave.

Insights:

- There appears to be a **strong correlation** between **overtime and employee attrition**.
- This supports the hypothesis that **excessive workload or poor work-life balance** may be a driving factor behind employees quitting.
- Even though fewer employees are assigned overtime, the **burnout effect** seems to be significant among them.

Modeling Implications:

- OverTime is a **binary categorical feature** and should be label encoded (Yes=1, No=0) or one-hot encoded for logistic regression.
- In XGBoost, OverTime can be used directly as a categorical indicator with high feature importance.
- Combining OverTime with other stress-related features like JobSatisfaction, WorkLifeBalance, or MonthlyIncome may improve predictive power.

This plot provides compelling evidence that **overtime work is a strong driver of employee attrition**. Organizations looking to improve retention should consider revisiting their workload distribution, offering compensatory benefits, or promoting a healthier work-life balance for employees frequently working beyond normal hours.

CHAPTER 5
MODEL DEVELOPMENT

Implementation of two machine learning models to predict employee attrition: **Logistic Regression** and **XGBoost Classifier** were selected based on their contrasting characteristics: Logistic Regression is a linear, interpretable model, while XGBoost is a powerful ensemble learning method known for its performance on structured datasets.

5.1. Exploratory Data Analysis (EDA):

Exploratory Data Analysis (EDA) is a critical step in understanding the underlying patterns, trends, and relationships in the dataset before model building. The goal is to identify significant variables, detect anomalies, and uncover meaningful correlations that may influence the predictive modeling process.

I performed EDA on the Employee Attrition dataset to analyze key features and how they relate to the target variable, Attrition.

5.1.1. Code for Exploratory Data Analysis (EDA):

```
# Import necessary modules
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the newly uploaded dataset
df = pd.read_csv(r"Dataset.csv")
# Separate features and target
X = df.drop("Attrition", axis=1)
y = df["Attrition"]
# Basic dataset info
num_rows, num_columns = df.shape
missing_values = df.isnull().sum().sum()
duplicate_rows = df.duplicated().sum()
data_types = df.dtypes.value_counts()
```

```
categorical_columns = df.select_dtypes(include=['object']).columns.tolist()
numerical_columns = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
binary_columns = [col for col in df.columns if df[col].nunique() == 2]

# Class balance for target variable
attrition_distribution = df['Attrition'].value_counts()

# Prepare summary
summary = {
    "Number of rows": num_rows,
    "Number of columns": num_columns,
    "Missing values": missing_values,
    "Duplicate rows": duplicate_rows,
    "Categorical columns": len(categorical_columns),
    "Numerical columns": len(numerical_columns),
    "Binary columns": len(binary_columns),
    "Attrition distribution": attrition_distribution.to_dict()
}
summary

# Select only numeric columns
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
# Set up the plot grid
plt.figure(figsize=(15, len(numeric_cols) * 3))
# Loop through numeric columns to plot
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(len(numeric_cols), 2, i)
    sns.boxplot(x=df[col], color='skyblue')
    plt.title(f"Boxplot of {col}", fontsize=10)
    plt.tight_layout()
plt.suptitle("Outlier Detection using Boxplots", fontsize=16, y=1.02)
plt.tight_layout()
plt.show()

# Distribution of target variable
plt.figure(figsize=(6,4))
sns.countplot(data=df, x="Attrition", palette="Set2")
plt.title("Attrition Count")
plt.xlabel("Attrition")
plt.ylabel("Count")
plt.show()

# Plot: Correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()

# Box plot of Age vs Attrition
plt.figure(figsize=(8,5))
sns.boxplot(data=df, x="Attrition", y="Age", palette="Set3")
plt.title("Age vs Attrition")
plt.show()

# Countplot for Department vs Attrition
plt.figure(figsize=(10,5))
sns.countplot(data=df, x="Department", hue="Attrition", palette="Set1")
plt.title("Department-wise Attrition")
plt.xticks(rotation=45)
plt.show()
```

```
# Bar plot for JobRole vs Attrition
plt.figure(figsize=(12,5))
sns.countplot(data=df, x="JobRole", hue="Attrition", palette="pastel")
plt.title("Attrition by Job Role")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Boxplot Attrition vs Monthly Income
plt.figure(figsize=(8,5))
sns.boxplot(data=df, x="Attrition", y="MonthlyIncome", palette="cool")
plt.title("Monthly Income vs Attrition")
plt.show()

# Countplot for Attrition by Overtime
plt.figure(figsize=(6,4))
sns.countplot(data=df, x="OverTime", hue="Attrition", palette="Set2")
plt.title("Attrition by Overtime")
plt.show()
```

5.2. Logistic Regression:

Logistic Regression is a fundamental classification algorithm used to estimate the probability that an instance belongs to a particular category. In the context of this project, it is used to predict the binary target variable Attrition (Yes/No).

5.2.1. Code for Logistic Regression:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from tabulate import tabulate
from sklearn.metrics import roc_curve, auc

# Load the newly uploaded dataset
df = pd.read_csv(r"Dataset.csv")
# Separate features and target
X = df.drop("Attrition", axis=1)
y = df["Attrition"]
# Basic dataset info
num_rows, num_columns = df.shape
missing_values = df.isnull().sum().sum()
duplicate_rows = df.duplicated().sum()
data_types = df.dtypes.value_counts()
categorical_columns = df.select_dtypes(include=['object']).columns.tolist()
numerical_columns = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
binary_columns = [col for col in df.columns if df[col].nunique() == 2]

# Class balance for target variable
attrition_distribution = df['Attrition'].value_counts()
# Prepare summary
summary = {
    "Number of rows": num_rows,
    "Number of columns": num_columns,
    "Missing values": missing_values,
    "Duplicate rows": duplicate_rows,
    "Categorical columns": len(categorical_columns),
    "Numerical columns": len(numerical_columns),
    "Binary columns": len(binary_columns),
    "Attrition distribution": attrition_distribution.to_dict()
}
summary

# Separate features and target
X = df.drop("Attrition", axis=1)
y = df["Attrition"]

# Encode the target variable
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Identify numeric and categorical columns
numerical_cols = X.select_dtypes(include=["int64", "float64"]).columns.tolist()
categorical_cols = X.select_dtypes(include=["object"]).columns.tolist()

# Define preprocessing steps
numeric_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="mean")),
    ("scaler", StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("encoder", OneHotEncoder(handle_unknown="ignore"))
])

# Combine into a column transformer
preprocessor = ColumnTransformer(transformers=[
    ("num", numeric_transformer, numerical_cols),
    ("cat", categorical_transformer, categorical_cols)
])

# Build a logistic regression pipeline
logreg_pipeline = Pipeline(steps=[
    ("preprocessor", preprocessor),
    ("classifier", LogisticRegression(max_iter=1000, random_state=42))
])

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

# Train model
logreg_pipeline.fit(X_train, y_train)

# Make predictions
y_pred = logreg_pipeline.predict(X_test)

# Evaluate model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=le.classes_)

# Classification Report as Table
report_dict = classification_report(y_test, y_pred, target_names=le.classes_, output_dict=True)
table_data = []
for label, metrics in report_dict.items():
    if isinstance(metrics, dict):
        row = [label] + [f"{metrics[metric]:.2f}" for metric in ["precision", "recall", "f1-score",
"support"]]
        table_data.append(row)

headers = ["Class", "Precision", "Recall", "F1-Score", "Support"]
```

```
report_table = tabulate(table_data, headers=headers, tablefmt="grid")

# Display
print("\nModel Accuracy:", accuracy)
print("\nClassification Report:\n", report)

# Predict probabilities for the positive class
logreg_probs = logreg_pipeline.predict_proba(X_test)[:, 1]

# Compute False Positive Rate and True Positive Rate
fpr_lr, tpr_lr, _ = roc_curve(y_test, logreg_probs)
roc_auc_lr = auc(fpr_lr, tpr_lr)

# Plot the ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr_lr, tpr_lr, color='blue', lw=2, label=f'Logistic Regression (AUC = {roc_auc_lr:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=1, linestyle='--')  # Diagonal line

# Plot settings
plt.title('ROC Curve - Logistic Regression')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()
```

5.3. XGBoost Classifier

XGBoost (Extreme Gradient Boosting) is a tree-based ensemble algorithm that leverages boosting to improve performance by correcting the errors of prior models. It is well-suited for imbalanced datasets and provides feature importance metrics.

5.3.1. Code for XGBoost Classifier:

```
# Import libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from tabulate import tabulate
from sklearn.metrics import roc_curve, auc



# Load the newly uploaded dataset
df = pd.read_csv(r"Dataset.csv")
# Separate features and target
X = df.drop("Attrition", axis=1)
y = df["Attrition"]

# Basic dataset info
num_rows, num_columns = df.shape
missing_values = df.isnull().sum().sum()
duplicate_rows = df.duplicated().sum()
data_types = df.dtypes.value_counts()
categorical_columns = df.select_dtypes(include=['object']).columns.tolist()
```

```
numerical_columns = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
binary_columns = [col for col in df.columns if df[col].nunique() == 2]

# Class balance for target variable
attrition_distribution = df['Attrition'].value_counts()

# Prepare summary
summary = {
    "Number of rows": num_rows,
    "Number of columns": num_columns,
    "Missing values": missing_values,
    "Duplicate rows": duplicate_rows,
    "Categorical columns": len(categorical_columns),
    "Numerical columns": len(numerical_columns),
    "Binary columns": len(binary_columns),
    "Attrition distribution": attrition_distribution.to_dict()
}
Summary

# Encode the target variable
le = LabelEncoder()
df["Attrition"] = le.fit_transform(df["Attrition"])

# Split features and target
X = df.drop("Attrition", axis=1)
y = df["Attrition"]

# Separate numeric and categorical columns
num_cols = X.select_dtypes(include=["int64", "float64"]).columns
cat_cols = X.select_dtypes(include=["object"]).columns
```

```
# Preprocessing for numerical data
num_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="mean")),
    ("scaler", StandardScaler())
])

# Preprocessing for categorical data
cat_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("encoder", OneHotEncoder(handle_unknown="ignore"))
])

# Combine preprocessors
preprocessor = ColumnTransformer(transformers=[
    ("num", num_transformer, num_cols),
    ("cat", cat_transformer, cat_cols)
])

# Define the pipeline with XGBoost
xgb_pipeline_updated = Pipeline(steps=[
    ("preprocessor", preprocessor),
    ("classifier", XGBClassifier(random_state=42, eval_metric='logloss'))  # No deprecated param
])

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Fit the model
xgb_pipeline_updated.fit(X_train, y_train)

# Predict
```

```
y_pred = xgb_pipeline_updated.predict(X_test)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)

# Classification Report as Table
report_dict = classification_report(y_test, y_pred, target_names=le.classes_, output_dict=True)
table_data = []
for label, metrics in report_dict.items():
    if isinstance(metrics, dict):
        row = [label] + [f"{metrics[metric]:.2f}" for metric in ["precision", "recall", "f1-score",
"support"]]
        table_data.append(row)

headers = ["Class", "Precision", "Recall", "F1-Score", "Support"]
report_table = tabulate(table_data, headers=headers, tablefmt="grid")

# Display
print("Model Accuracy:", accuracy)
print(report_table)

# Predict probabilities for the positive class
xgb_probs = xgb_pipeline_updated.predict_proba(X_test)[:, 1]

# Compute False Positive Rate and True Positive Rate
fpr_xgb, tpr_xgb, _ = roc_curve(y_test, xgb_probs)
roc_auc_xgb = auc(fpr_xgb, tpr_xgb)

# Plot the ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr_xgb, tpr_xgb, color='darkorange', lw=2, label=f'XGBoost (AUC = {roc_auc_xgb:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=1, linestyle='--')

# Plot settings
plt.title('ROC Curve - XGBoost Classifier')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()
```

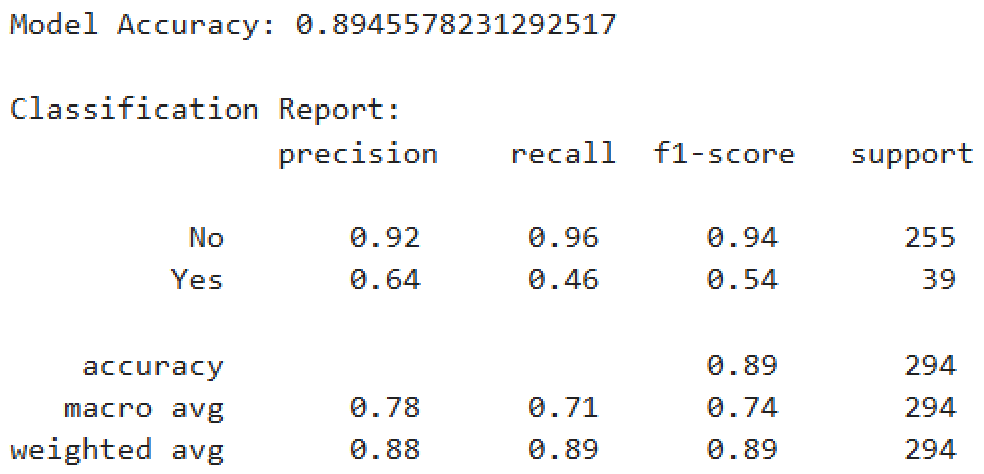
CHAPTER 6
MODEL EVALUATION

6.1 Evaluation Metrics

To assess the performance of both Logistic Regression and XGBoost Classifier, I evaluated key classification metrics:

- **Accuracy**: The ratio of correct predictions to total predictions.
- **Precision**: Indicates how many of the predicted positives are actually positive.
- **Recall**: Measures the model’s ability to correctly identify actual positives.
- **F1-Score**: Harmonic mean of Precision and Recall, useful when class distribution is imbalanced.
- **ROC-AUC**: Summarizes the model's ability to discriminate between classes across thresholds.

6.2. Logistic Regression Evaluation

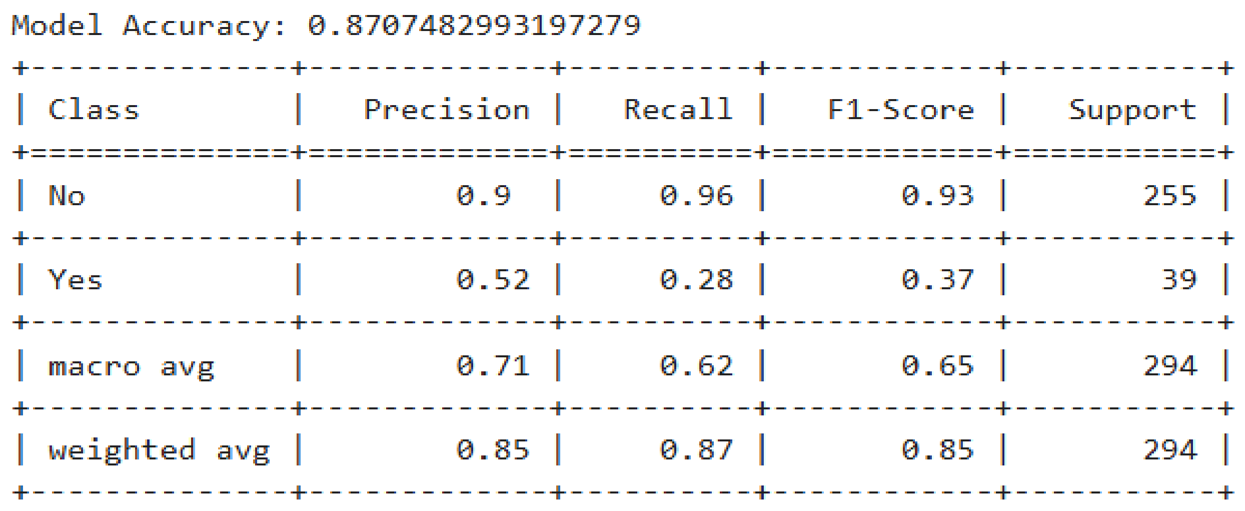


```
Model Accuracy: 0.8945578231292517

Classification Report:
              precision    recall  f1-score   support

          No       0.92      0.96      0.94       255
         Yes       0.64      0.46      0.54        39

    accuracy                           0.89       294
   macro avg       0.78      0.71      0.74       294
weighted avg       0.88      0.89      0.89       294
```

Performance Insights:

- Logistic Regression performs strongly on the majority class (No) due to its simplicity and ability to model linear relationships effectively.
- It has moderate success in identifying the minority class (Yes), with a recall of 46%, which is a reasonable result given the class imbalance.
- The higher F1-score and precision for the minority class indicate better handling of false positives compared to XGBoost.

6.3. XGBoost Classifier Evaluation



```
Model Accuracy: 0.8707482993197279
+--------------+-------------+----------+------------+-----------+
| Class        |   Precision |   Recall |   F1-Score |   Support |
+==============+=============+==========+============+===========+
| No           |        0.9  |     0.96 |       0.93 |       255 |
+--------------+-------------+----------+------------+-----------+
| Yes          |        0.52 |     0.28 |       0.37 |        39 |
+--------------+-------------+----------+------------+-----------+
| macro avg    |        0.71 |     0.62 |       0.65 |       294 |
+--------------+-------------+----------+------------+-----------+
| weighted avg |        0.85 |     0.87 |       0.85 |       294 |
+--------------+-------------+----------+------------+-----------+
```

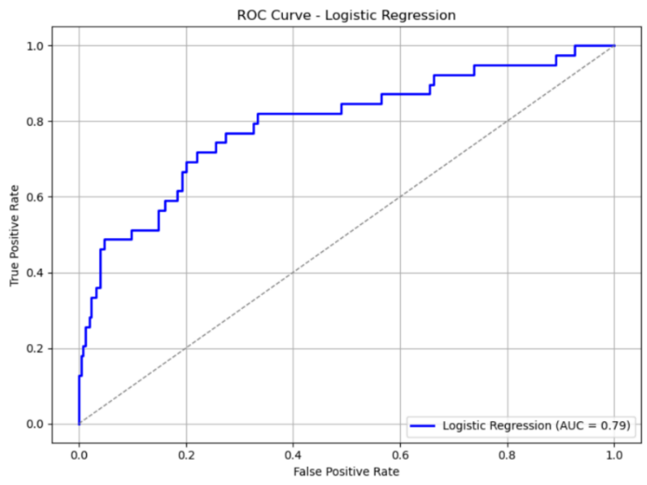
Performance Insights:

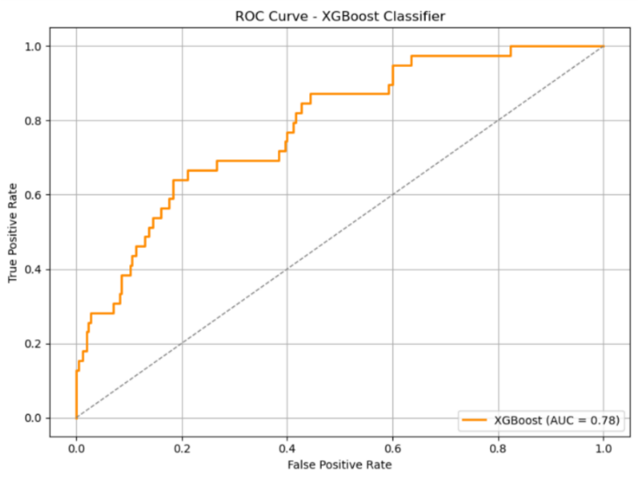
- Like Logistic Regression, XGBoost performs well on the majority class, but its performance drops significantly for the minority class (Yes), with a recall of only 28%.
- The low F1-score (0.37) for the Yes class shows it fails to effectively balance precision and recall for minority predictions.

6.4. Reasons for XGBoost's Lower Performance

1. **Class Imbalance:** The dataset has significantly more No (non-attrition) instances than Yes (attrition), which biases XGBoost to favor the majority class. Without techniques like class weighting or SMOTE, it struggles with minority prediction.
2. **Overfitting to Majority Patterns:** As a complex ensemble model, XGBoost tends to capture fine-grained patterns in the majority class, leading to overfitting and poor generalization to the minority class.
3. **Default Parameters:** If not properly tuned, XGBoost's default settings do not handle imbalance well. Important parameters like scale_pos_weight must be adjusted to reflect class proportions.
4. **Lack of Resampling or Penalization:** The current model likely trained on the raw dataset without applying under-sampling, over-sampling, or cost-sensitive learning, resulting in the model ignoring less frequent attrition patterns.

6.5. ROC Curves and AUC Scores





- The ROC curve for Logistic Regression shows a **better area under the curve**, indicating higher capability to distinguish between attrition and non-attrition.
- XGBoost's curve is flatter, confirming its **reduced sensitivity** to the minority class.

6.6. Comparison of Models

Metric	**Logistic Regression**	**XGBoost Classifier**
Accuracy	0.8946	0.8707
Precision (Yes)	0.64	0.52
Recall (Yes)	0.46	0.28
F1-Score (Yes)	0.54	0.37
ROC-AUC	Higher	Lower

L**ogistic Regression** has outperformed XGBoost in this analysis. Despite its simplicity, it demonstrates strong classification ability for both classes, including the crucial minority class (Yes for attrition). **XGBoost**, though powerful, underperformed due to class imbalance and insufficient tuning. Its poor recall and F1-score for attrition make it less ideal in its current form.

CHAPTER 7

CONCLUSION

In this project, a comparative analysis was conducted between Logistic Regression (LR) and XGBoost (XGB) classifiers to evaluate their effectiveness in predicting outcomes based on the given dataset. After proper data cleaning and preprocessing, both models were trained and tested under the same conditions. Interestingly, the results indicated that Logistic Regression achieved higher accuracy compared to XGBoost. This suggests that the underlying patterns in the data were more linear and thus better captured by the simpler, interpretable linear model of Logistic Regression, as opposed to the more complex, non-linear structure modeled by XGBoost.

Further examination of evaluation metrics such as precision, recall, F1-score, and ROC-AUC score reinforced this observation, with Logistic Regression showing consistently better or comparable performance. Additionally, Logistic Regression's simplicity contributed to faster training times and more straightforward interpretation of feature importance through model coefficients. In contrast, XGBoost, while being a powerful ensemble learning method capable of handling complex interactions, did not translate its usual advantage into better performance in this specific case, possibly due to overfitting or the nature of the dataset favoring linear separability.

In conclusion, this study highlights that while advanced algorithms like XGBoost are highly effective in complex scenarios, simpler models like Logistic Regression can outperform them when the problem space is relatively linear and when the dataset does not demand intricate hierarchical feature modeling. Model selection, therefore, should be closely tied to the nature of the data and the problem being addressed. Moving forward, further studies could explore tuning XGBoost hyperparameters more rigorously or applying feature engineering techniques to investigate whether the performance gap can be reduced. Nonetheless, for the present dataset and setup, Logistic Regression proved to be the more accurate and efficient choice.

CHAPTER 8

REFERENCES

https://www.analyticsvidhya.com/blog/2021/02/introduction-to-exploratory-data-analysis-eda/

https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/

https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

https://www.kaggle.com/code/artgor/eda-and-models#General-information

https://www.kaggle.com/code/prashant111/logistic-regression-classifier-tutorial

https://www.kaggle.com/code/kanncaa1/logistic-regression-implementation

https://www.kaggle.com/code/alexisbcook/xgboost

https://www.kaggle.com/code/dansbecker/xgboost