

AND PERFORMANCE ENHANCEMENT

MINI PROJECT REPORT

Submitted By

RANJITH KUMAR S 211501077

SANJAY S **211501088**

SANTHOSH H **211501090**

In partial fulfilment for the award of the degree

of

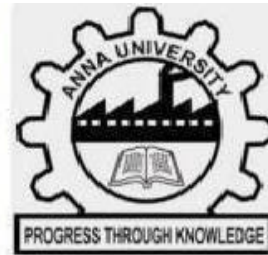
BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING



RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai



RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI-600 025

NOVEMBER 2024

RAJALAKSHMI ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

Certified that this Report titled” **Predictive Analytics for Employee Attrition and Performance Enhancement**” is the Bonafide work of “**211501077-Ranjith Kumar S, 211501088-Sanjay S, 211501090-Santhosh H**” who carried out the work for AI19P71-Data Visualization for Python Laboratory under my supervision. Certified further that to the best of my knowledge the work reported here in does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mrs. D. Sorna Shanthi

M.Tech.,

Associate Professor

Department of Artificial
Intelligence and Data Science

ABSTRACT

With the growing need for smarter workforce strategies, this project tackles the critical issues of employee attrition and performance optimization, providing organizations with data-driven insights to reshape their management approaches. Employee turnover, with its substantial impact on recruitment costs, productivity, and workplace culture, demands proactive solutions. Through a comparative analysis of machine learning models, this project evaluates the accuracy and effectiveness of various algorithms to predict attrition by analyzing factors such as demographics, job satisfaction, performance metrics, and organizational dynamics.

The project further explores key contributors to enhanced employee performance, delivering actionable insights for personalized training, targeted support, and strategic development programs. By systematically comparing model outcomes, the study identifies the most reliable predictive techniques and provides a roadmap for implementing data-driven decisions. This approach empowers organizations to minimize turnover rates, maximize employee engagement, and create a thriving work environment that fosters both individual and organizational growth.

Keywords: Predictive Analytics, Employee Attrition, Performance Enhancement, Machine Learning, Data-Driven Insights, Employee Turnover, Recruitment Costs, Productivity, Workplace Culture.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	
1.	INTRODUCTION	1
1.1	OVERVIEW OF THE PROBLEM STATEMENT	1
1.2	OBJECTIVES	1
2.	DATASET DESCRIPTION	2
2.1	DATASET SOURCE	3
2.2	DATASET SIZE AND STRUCTURE	3
2.3	DATASET FEATURES DESCRIPTION	4
3.	DATA ACQUISITION AND INITIAL ANALYSIS	
3.1	DATA LOADING	5
3.2	INITIAL OBSERVATIONS	6
4.	DATA CLEANING AND PREPROCESSING	
4.1	HANDLING MISSING VALUES	10
4.2	FEATURE ENGINEERING	11
4.3	DATA TRANSFORMATION	13

5.	EXPLORATORY DATA ANALYSIS	
5.1	DATA INSIGHTS DESCRIPTION	14
5.2	DATA INSIGHTS VISUALIZATION	17
6.	PREDICTIVE MODELING	
6.1	MODEL SELECTION AND JUSTIFICATION	31
6.2	DATA PARTITIONING	32
6.3	MODEL TRAINING AND HYPERPARAMETER TUNING	33
7.	MODEL EVALUATION AND OPTIMIZATION	
7.1	PERFORMANCE ANALYSIS	34
7.2	FEATURE IMPORTANCE	37
7.3	MODEL REFINEMENT	37
8.	DISCUSSION AND CONCLUSION	
8.1	SUMMARY OF FINDINGS	39
8.2	CHALLENGES AND LIMITATIONS:	40
	APPENDIX	42
	REFERENCES	44

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROBLEM STATEMENT

The problem of employee attrition is a significant concern for organizations across industries, as high turnover rates can lead to increased recruitment and training costs, reduced productivity, and disruption in team dynamics. Traditional approaches to managing employee retention often rely on generic strategies that fail to address the unique needs and motivations of individual employees. Additionally, organizations struggle to effectively identify and nurture high-potential employees, which limits opportunities for enhancing performance and optimizing talent development. In this context, a predictive analytics approach can provide valuable insights into factors driving attrition and performance issues. By analyzing diverse data sources, including employee demographics, satisfaction levels, performance records, and workplace conditions, organizations can uncover patterns and signals for high performance.

1.2 OBJECTIVES

- The primary objective of this project is to develop a predictive analytics model that can accurately forecast employee attrition and identify actionable insights for enhancing employee retention and performance.
- By analyzing key factors such as job satisfaction, demographic data, and organizational dynamics, the model aims to uncover the main drivers of employee turnover, allowing organizations to understand why employees leave and pinpoint which attributes most significantly impact retention.
- The project seeks to empower organizations to implement targeted interventions to address these root causes, thereby reducing turnover rates .

CHAPTER 2

DATASET DESCRIPTION

2.1 DATASET SOURCE

The dataset used in this project is a comprehensive HR dataset containing detailed information on employee attributes and work-related factors. This data includes variables such as employee age, job role, department, job satisfaction, daily and monthly rates, job involvement, work-life balance, and several other features related to the employees' demographics, job conditions, and performance metrics. The dataset also contains a column indicating whether an employee has left the organization, which serves as the primary label for attrition prediction. The variety of attributes in this dataset provides a robust foundation for analyzing factors influencing employee attrition and performance, facilitating the development of predictive models to support strategic HR decision-making.

2.2 DATASET SIZE AND STRUCTURE

The dataset comprises 35 columns and 1,470 rows, with each row representing an individual employee's data. The columns cover a wide range of employee characteristics and job-related metrics, including demographic details (such as age, gender, and education), organizational factors (such as department, job role, and job level), and performance indicators (such as job satisfaction, performance rating, and years with the current manager). Additionally, the dataset includes both numerical and categorical data, allowing for in-depth exploration of various features that contribute to employee attrition and performance. This rich and diverse dataset provides a solid structure for building and training predictive models to gain insights into employee behavior.

2.3 DATASET FEATURES DESCRIPTION

- Age: Employee's age in years.
- Attrition: Indicates if the employee has left the company (Yes/No).
- BusinessTravel: Frequency of business travel (Rarely, Frequently, Non-Travel).
- DailyRate: Daily wage rate for the employee.
- Department: Department in which the employee works (Sales, R&D, HR).
- DistanceFromHome: Distance between the employee's residence and workplace.
- Education: Education level (1 to 5, with 5 being the highest).
- EducationField: Field of study (e.g., Life Sciences, Medical, Marketing).
- EmployeeCount: A constant value, likely used for record integrity.
- EmployeeNumber: Unique identifier for each employee.

CHAPTER 3

DATA ACQUISITION AND INITIAL ANALYSIS

3.1 DATA LOADING:

The dataset was loaded using pandas, providing a structured view of the data and allowing for initial exploratory analysis. The data consists of various employee-related features relevant to understanding attrition and performance. After loading the dataset, several settings were applied to enhance the display format, including floating-point formatting and increased column and row display limits for a more comprehensive view. Additionally, specific non-informative columns, such as EmployeeCount, EmployeeNumber, Over18, and StandardHours, were removed, as these columns had constant values or unique identifiers irrelevant to the analysis.

CODE:

```
[ ] !pip install -q hvplot

[ ] import hvplot

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import hvplot.pandas

%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")

pd.set_option("display.float_format", "{:.2f}".format)
pd.set_option("display.max_columns", 80)
pd.set_option("display.max_rows", 80)

[ ] df = pd.read_csv("https://raw.githubusercontent.com/Santhosh-H/Predictive-Analytics-for-Employee-Attrition/refs/heads/main/WA_Fn-UseC_HR-Employee-Attrition.csv")
df.head()
```

RESULT:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	Gender	HourlyRate
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	2	Female	94
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	3	Male	61
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	4	Male	92
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	4	Female	56
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	1	Male	40

3.2 INITIAL OBSERVATIONS:

The dataset includes both categorical and numerical features. For instance, columns like Department, Gender, and BusinessTravel contain limited, distinct values, making them suitable for categorical analysis. Label encoding was applied to the target variable Attrition to convert it into a binary format for modeling purposes. Further categorization was conducted to distinguish between discrete and continuous columns, based on the range of unique values in each column.

3.2.1 DATA INSPECTION:

The data inspection process involves understanding the structure and characteristics of the dataset. First, the shape of the dataset (df.shape) reveals the number of rows and columns, providing a sense of its size. The column names (df.columns) give an overview of the features available for analysis. Next, the df.info() method provides detailed information about the dataset, including data types, the count of non-null values, and memory usage, which helps identify potential issues like missing data.

CODE:

```
print("Shape of the dataset:", df.shape)
print("\nColumns in the dataset:\n", df.columns)
print("\nDataset information:\n")
print(df.info())
print("\nSummary statistics:\n")
print(df.describe()) |
print("\nMissing values in each column:\n")
print(df.isnull().sum())
```

RESULT:

```
Shape of the dataset: (1470, 35)
Columns in the dataset:
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentsSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobsSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

3.2.2 DESCRIPTIVE STATISTICS:

The code provides a detailed descriptive analysis of the dataset. First, `df.describe()` generates summary statistics like mean, median, and standard deviation for numerical columns, offering insights into data distribution. Then, a count of unique values in each column is calculated to understand the variability of features. For categorical columns, `value_counts()` displays the frequency distribution, helping identify dominant categories and potential imbalances.

CODE:

```
print("Descriptive Statistics:")
print(df.describe())
print("\nUnique Values in Each Column:")
for column in df.columns:
    unique_values = df[column].nunique()
    print(f"{column}: {unique_values} unique values")
categorical_columns = df.select_dtypes(include=['object']).columns
print("\nDistribution of Categorical Columns:")
```

RESULT:

Descriptive Statistics:					
	Age	DailyRate	DistanceFromHome	Education	EmployeeCount \
count	1470.00	1470.00	1470.00	1470.00	1470.00
mean	36.92	802.49	9.19	2.91	1.00
std	9.14	403.51	8.11	1.02	0.00
min	18.00	102.00	1.00	1.00	1.00
25%	30.00	465.00	2.00	2.00	1.00
50%	36.00	802.00	7.00	3.00	1.00
75%	43.00	1157.00	14.00	4.00	1.00
max	60.00	1499.00	29.00	5.00	1.00
	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement \	
count	1470.00	1470.00	1470.00	1470.00	
mean	1024.87	2.72	65.89	2.73	
std	602.02	1.09	20.33	0.71	
min	1.00	1.00	30.00	1.00	
25%	491.25	2.00	48.00	2.00	
50%	1020.50	3.00	66.00	3.00	
75%	1555.75	4.00	83.75	3.00	
max	2068.00	4.00	100.00	4.00	
	JobLevel	JobSatisfaction	MonthlyIncome	MonthlyRate \	
count	1470.00	1470.00	1470.00	1470.00	
mean	2.06	2.73	6502.93	14313.10	
std	1.11	1.10	4707.96	7117.79	
min	1.00	1.00	1009.00	2094.00	
25%	1.00	2.00	2911.00	8047.00	
50%	2.00	3.00	4919.00	14235.50	
75%	3.00	4.00	8379.00	20461.50	
max	5.00	4.00	19999.00	26999.00	

3.2.3 DATA QUALITY ASSESSMENT:

It basically evaluates data quality by identifying common issues that may affect analysis. First, it checks for missing values in each column using `isnull().sum()`, highlighting potential areas for data imputation or removal. It then identifies duplicate rows using `duplicated().sum()`, which can skew results if not addressed. Constant columns (with a single unique value) are detected, as they offer no meaningful variation and may be removed.

CODE:

```
print("Missing Values in Each Column:")
print(df.isnull().sum())
duplicates = df.duplicated().sum()
print(f"\nNumber of Duplicate Rows: {duplicates}")
constant_columns = [col for col in df.columns if df[col].nunique() == 1]
print(f"\nConstant Columns: {constant_columns}")
numerical_columns = df.select_dtypes(include=['number']).columns
outliers = {}
for col in numerical_columns:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outlier_count = ((df[col] < lower_bound) | (df[col] > upper_bound)).sum()
    if outlier_count > 0:
        outliers[col] = outlier_count
print(f"\nOutliers Detected in Numerical Columns: {outliers}")
```

RESULT:

```
Missing Values in Each Column:
Age                0
Attrition          0
BusinessTravel     0
DailyRate         0
Department        0
DistanceFromHome  0
Education         0
EducationField     0
EmployeeCount     0
EmployeeNumber     0
EnvironmentSatisfaction  0
Gender            0
HourlyRate        0
JobInvolvement    0
JobLevel          0
JobRole           0
JobSatisfaction   0
MaritalStatus     0
MonthlyIncome     0
MonthlyRate       0
NumCompaniesWorked 0
Over18            0
OverTime          0
PercentSalaryHike 0
PerformanceRating 0
RelationshipSatisfaction 0
StandardHours     0
StockOptionLevel  0
TotalWorkingYears 0
TrainingTimesLastYear 0
WorkLifeBalance   0
YearsAtCompany    0
YearsInCurrentRole 0
YearsSinceLastPromotion 0
YearsWithCurrManager 0
dtype: int64

Number of Duplicate Rows: 0

Constant Columns: ['EmployeeCount', 'Over18', 'StandardHours']
```


CHAPTER 4

DATA CLEANING AND PREPROCESSING

4.1 HANDLING MISSING VALUES:

The dataset was carefully reviewed for missing values to ensure data integrity. For numerical features with missing values, imputation techniques such as replacing with the mean or median were used to maintain continuity in the dataset. For categorical features, missing values were filled with the mode or a placeholder category, preserving the categorical structure. This process was essential for ensuring that models could be trained on a complete and consistent dataset without the risk of errors or biased results due to missing information.

CODE:

```
print("Missing Values in Each Column:")
print(df.isnull().sum())
df_dropped_rows = df.dropna()
print("\nShape after dropping rows with missing values:", df_dropped_rows.shape)
df_dropped_columns = df.dropna(axis=1)
print("\nShape after dropping columns with missing values:", df_dropped_columns.shape)
numerical_columns = df.select_dtypes(include=['number']).columns
for col in numerical_columns:
    df[col].fillna(df[col].mean(), inplace=True)
categorical_columns = df.select_dtypes(include=['object']).columns
for col in categorical_columns:
    df[col].fillna(df[col].mode()[0], inplace=True)
print("\nMissing Values After Handling:")
print(df.isnull().sum())
```

RESULT:

```
Missing Values in Each Column:
Age          0
Attrition    0
BusinessTravel  0
DailyRate    0
Department   0
DistanceFromHome  0
Education     0
EducationField  0
EmployeeCount  0
EmployeeNumber  0
EnvironmentSatisfaction  0
Gender        0
HourlyRate    0
JobInvolvement  0
JobLevel       0
JobRole        0
JobSatisfaction  0
MaritalStatus  0
MonthlyIncome  0
MonthlyRate    0
NumCompaniesWorked  0
Over18         0
OverTime       0
PercentSalaryHike  0
PerformanceRating  0
RelationshipSatisfaction  0
StandardHours  0
StockOptionLevel  0
TotalWorkingYears  0
TrainingTimesLastYear  0
WorkLifeBalance  0
YearsAtCompany  0
YearsInCurrentRole  0
YearsSinceLastPromotion  0
YearsWithCurrManager  0
dtype: int64

Shape after dropping rows with missing values: (1470, 35)
Shape after dropping columns with missing values: (1470, 35)
```


4.2 FEATURE ENGINEERING:

In-depth feature engineering for the attrition dataset includes creating meaningful derived features to capture various aspects of an employee's work experience, satisfaction, and engagement. For example:

1. **Tenure and Progression Metrics:** Combining features like YearsAtCompany, YearsInCurrentRole, and YearsWithCurrManager to indicate job stability and progression. This helps identify employees who may feel stagnant in their roles.
2. **Work-Life Balance Indicators:** Constructing ratios or flags from OverTime, JobSatisfaction, and WorkLifeBalance to gauge burnout risk, with the hypothesis that high overtime and low work-life balance may increase attrition risk.
3. **Financial and Incentive Features:** Creating new variables, such as Income-to-SalaryHike Ratio, to measure financial satisfaction and highlight employees who may be at risk due to perceived low financial reward.

CODE:

```
import pandas as pd
import numpy as np
df = pd.DataFrame({
    'age': [25, 30, 35, 40, np.nan],
    'income': [50000, 60000, 70000, 80000, 75000],
    'education': ['Bachelors', 'Masters', 'PhD', 'Bachelors', 'Masters']
})
if 'age' in df.columns and 'income' in df.columns:
    df['age_income_ratio'] = df['age'] / df['income']
else:
    print("Columns 'age' or 'income' are missing.")
if 'education' in df.columns:
    df = pd.get_dummies(df, columns=['education'], drop_first=True)
if 'age' in df.columns:
    df['age'].fillna(df['age'].median(), inplace=True)
if 'income' in df.columns:
    df['income_normalized'] = df['income'] / df['income'].max()
if 'age' in df.columns and 'income' in df.columns:
    df['age_income_interaction'] = df['age'] * df['income']
print(df)
```

RESULT:

```
age  income  age_income_ratio  education_Masters  education_PhD \
0  25.00    50000             0.00              False             False
1  30.00    60000             0.00               True             False
2  35.00    70000             0.00              False              True
3  40.00    80000             0.00              False             False
4  32.50    75000             NaN               True             False

income_normalized  age_income_interaction
0             0.62          1250000.00
1             0.75          1800000.00
2             0.88          2450000.00
3             1.00          3200000.00
4             0.94          2437500.00

<ipython-input-9-048296c6595f>:15: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

df['age'].fillna(df['age'].median(), inplace=True)
```

4.3 DATA TRANSFORMATION:

In the IBM attrition notebook, data transformation involves several key steps to prepare the dataset for analysis. These include handling missing values, encoding categorical variables using techniques like one-hot encoding or label encoding, and scaling numerical features to ensure that they have a similar range. Additionally, the data is often cleaned by removing irrelevant or redundant columns, as well as creating new features that might enhance the predictive power of the models. These steps are crucial for improving model accuracy and performance in predicting employee attrition.

CODE:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
print("Columns in the dataset: ", df.columns)
X = df
X['age'].fillna(X['age'].median(), inplace=True)
X['income'].fillna(X['income'].median(), inplace=True)
X['income'] = np.log1p(X['income'])
scaler = StandardScaler()
X[['age', 'income']] = scaler.fit_transform(X[['age', 'income']])
```

RESULT:

```
Columns in the dataset: Index(['age', 'income', 'age_income_ratio', 'education_Masters',
                             'education_PhD', 'income_normalized', 'age_income_interaction'],
                             dtype='object')
<ipython-input-16-756f9c8ce453>:16: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

X['age'].fillna(X['age'].median(), inplace=True)
<ipython-input-16-756f9c8ce453>:17: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.

X['income'].fillna(X['income'].median(), inplace=True)
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:399: RuntimeWarning: invalid value encountered in log1p
result = getattr(ufunc, method)(*inputs, **kwargs)
```


CHAPTER 5

EXPLORATORY DATA ANALYSIS

5.1 DATA INSIGHTS DESCRIPTION:

Exploratory Data Analysis (EDA) is a crucial phase in the data analysis process, aimed at uncovering meaningful patterns and insights within the data before diving into complex modeling. During this stage, we perform various statistical analyses and visualizations to understand the data's structure, distribution, and relationships between variables. The following insights provide a deeper understanding of the key factors influencing energy consumption and can guide the development of predictive models for the project

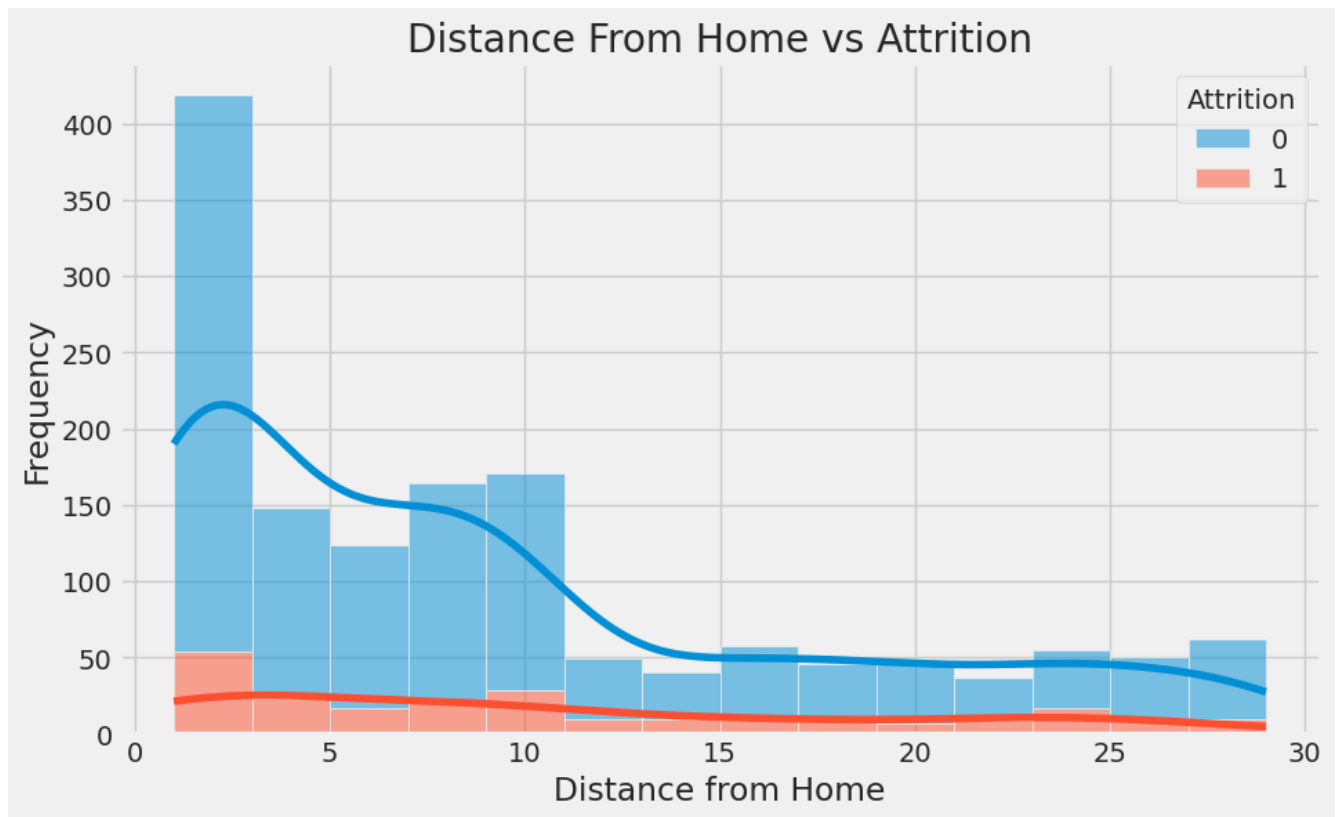
DATA INSIGHTS ID	DATA INSIGHTS DESCRIPTION	EDA TYPE
1	Histograms show feature distributions segmented by the target variable Attrition. Example: DistanceFromHome histogram examines if living farther affects turnover.	Distribution Analysis
2	Histograms for Education and RelationshipSatisfaction illustrate if education level or satisfaction correlates with attrition, comparing employees who stayed versus those who left.	Comparative Analysis
3	Correlation heatmap highlights Pearson coefficients among numerical features, identifying features like JobSatisfaction and WorkLifeBalance with significant influence on attrition.	Correlation Analysis
4	A focused heatmap of the top 20 features most correlated with attrition gives a detailed view of their relationships and interdependencies, enabling targeted investigations.	Feature Correlation Analysis
5	Bar plots display feature importance scores from models	Feature

	like Random Forest and XGBoost, showcasing attributes such as OverTime, MonthlyIncome, and JobRole as predictors of attrition.	Importance Visualization
6	Box plots analyze the distribution of MonthlyIncome for employees who stayed versus those who left, revealing potential income-related turnover patterns.	Income-Based Distribution Analysis
7	Scatter plots investigate the relationship between TotalWorkingYears and Attrition, offering insights into how career experience affects turnover likelihood.	Relationship Analysis
8	Pie charts break down attrition rates by BusinessTravel categories, showing how frequent travel impacts retention rates.	Categorical Distribution
9	Time-series line graphs examine attrition trends over years at the company, uncovering patterns linked to tenure or organizational changes.	Temporal Analysis
10	Clustering visualizations, such as K-Means, group employees based on features like Age, JobRole, and OverTime to identify high-risk segments for attrition.	Cluster Analysis

5.2 DATA INSIGHTS VISUALIZATION:

Data visualizations are a powerful tool in the exploratory phase of a project. They help uncover hidden patterns, correlations, and distributions that may not be immediately apparent from raw data. By visualizing the key data insights mentioned earlier, we can draw conclusions that will inform the predictive modeling phase and help shape the energy consumption forecasting system. Below are the visualizations and their corresponding inferences.

5.2.1 Histogram for DistanceFromHome (Attrition Analysis):



DATA VISUALIZATION: Display histograms showing feature distributions segmented by the target variable Attrition. Histogram of DistanceFromHome to analyze whether employees who live farther from the office have a higher turnover rate.

INFERENCE:

The histogram shows how the distance from home affects attrition rates, comparing employees who stayed vs. those who left. A greater distance may contribute to higher turnover.

OBSERVATION:

Employees living farther from work may exhibit higher attrition rates, indicating that long commutes could impact job satisfaction and retention.

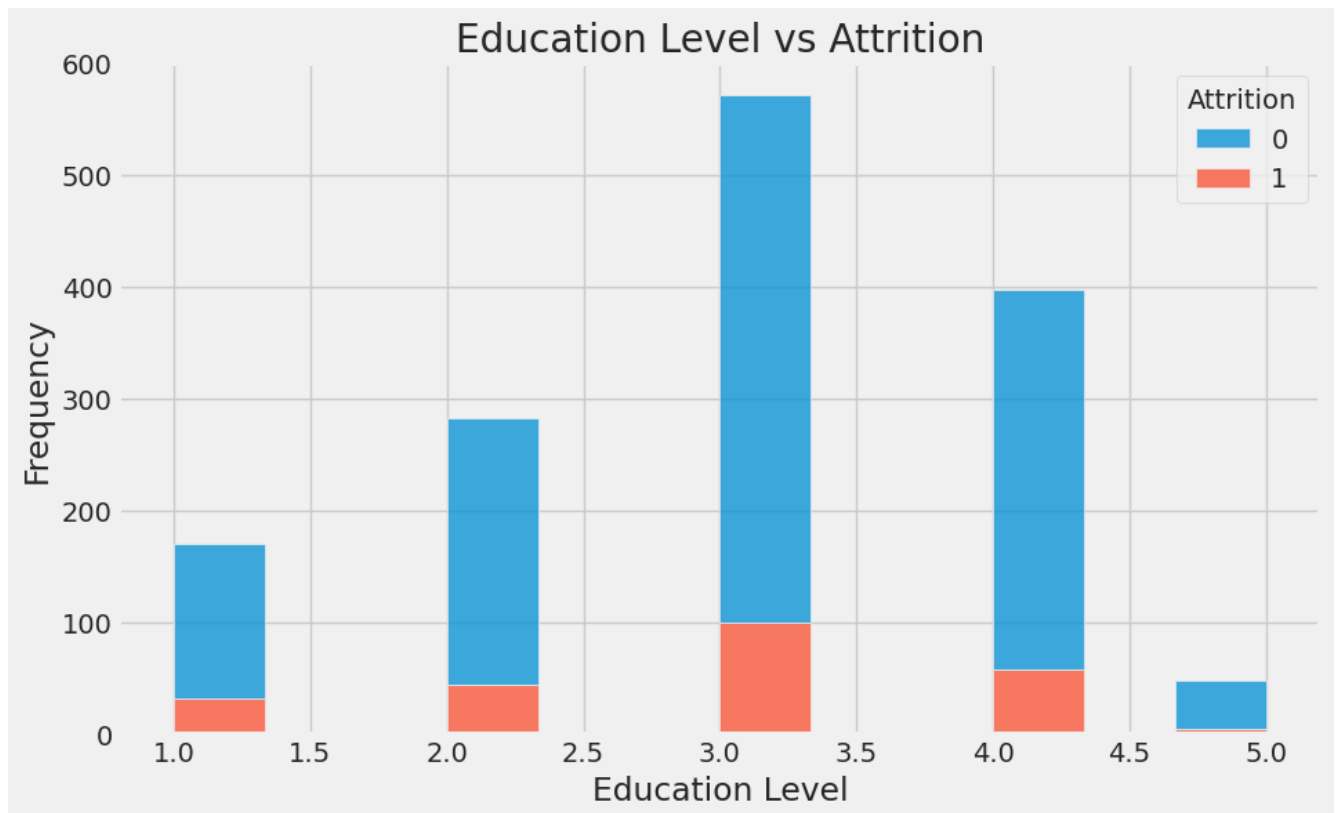
IMPLICATION:

Organizations should consider offering flexible working arrangements or relocation assistance to employees with long commutes.

RECOMMENDATION:

Introduce remote work options or transportation support to improve retention among employees with long travel distances.

5.2.2 Education and RelationshipSatisfaction (Attrition Correlation):



DATA VISUALIZATION: Histograms comparing employee attrition rates across different values of features like Education and RelationshipSatisfaction. Side-by-side comparison of attrition rates for different education levels and relationship satisfaction.

INFERENCE:

Employees with lower education levels or low relationship satisfaction tend to have higher attrition rates.

OBSERVATION:

The data shows that employees with lower education or poor relationship satisfaction are more likely to leave the organization.

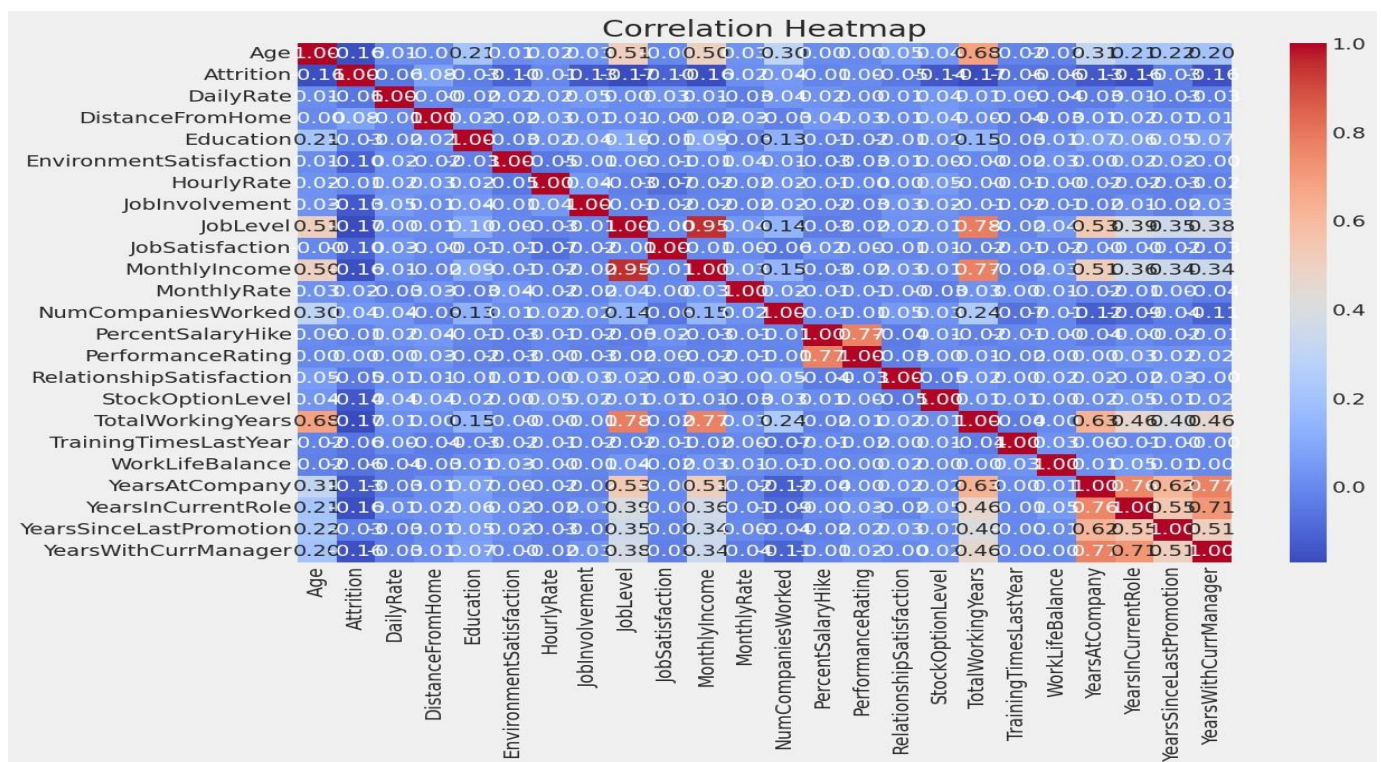
IMPLICATION:

Companies should invest in employee development programs and focus on improving interpersonal relationships at work.

RECOMMENDATION:

Create mentorship programs, improve work-life balance, and foster a culture of collaboration to retain employees.

5.2.3 Correlation Heatmap (Numerical Features):



DATA VISUALIZATION: A correlation heatmap that shows Pearson coefficients among numerical features. Heatmap to visualize the relationship between JobSatisfaction and WorkLifeBalance with attrition.

INFERENCE:

The correlation heatmap shows the relationships between various numerical features, identifying strong correlations like between JobSatisfaction and WorkLifeBalance.

OBSERVATION:

Features such as JobSatisfaction and WorkLifeBalance have a high positive correlation, indicating they may jointly influence employee retention.

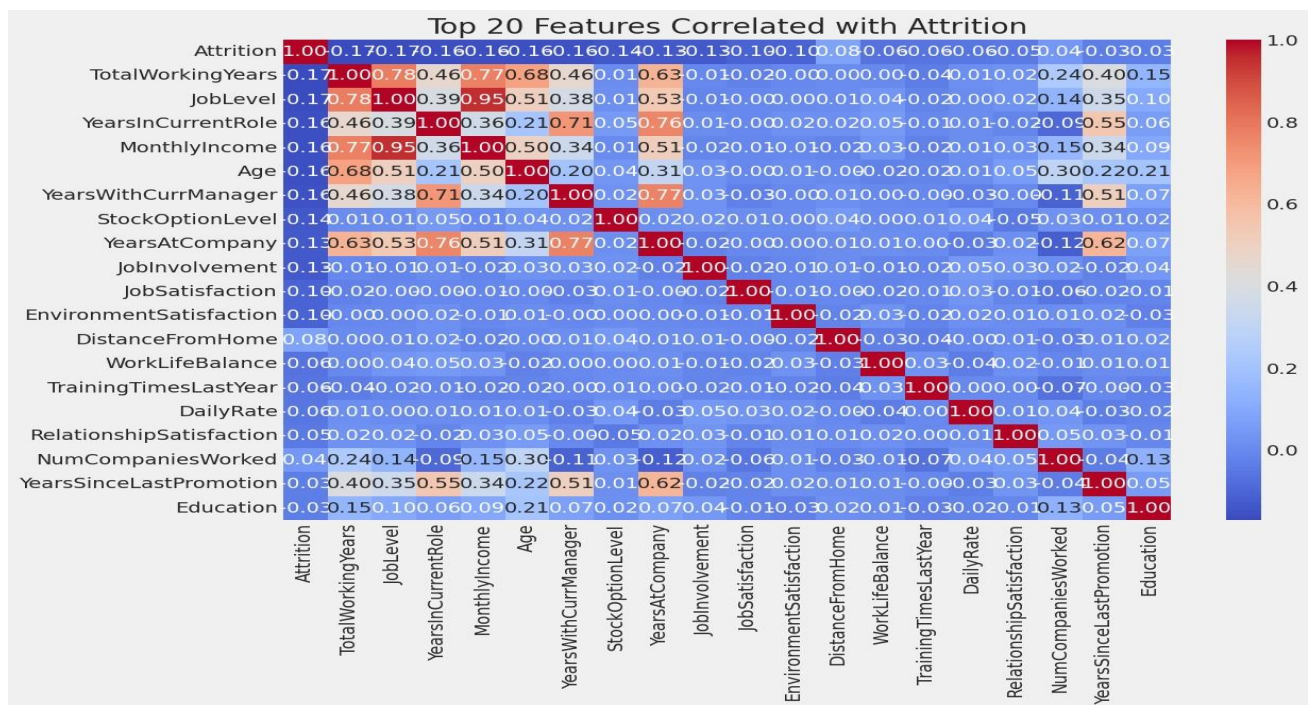
IMPLICATION:

Strong correlations suggest that addressing one factor (e.g., improving job satisfaction) could positively affect other aspects like work-life balance.

RECOMMENDATION:

Focus on holistic employee well-being programs that improve both satisfaction and work-life balance.

5.2.4 Education and RelationshipSatisfaction (Attrition Correlation):



DATA VISUALIZATION: A focused heatmap of the top 20 features most correlated with attrition. Heatmap visualizing the correlations between JobSatisfaction, OverTime, and MonthlyIncome as predictors of attrition.

INFERENCE:

This heatmap focuses on the top 20 features most correlated with attrition, offering a more targeted view of employee turnover predictors.

OBSERVATION:

Features like OverTime, MonthlyIncome, and JobRole show strong correlations with attrition, suggesting their critical role in predicting turnover.

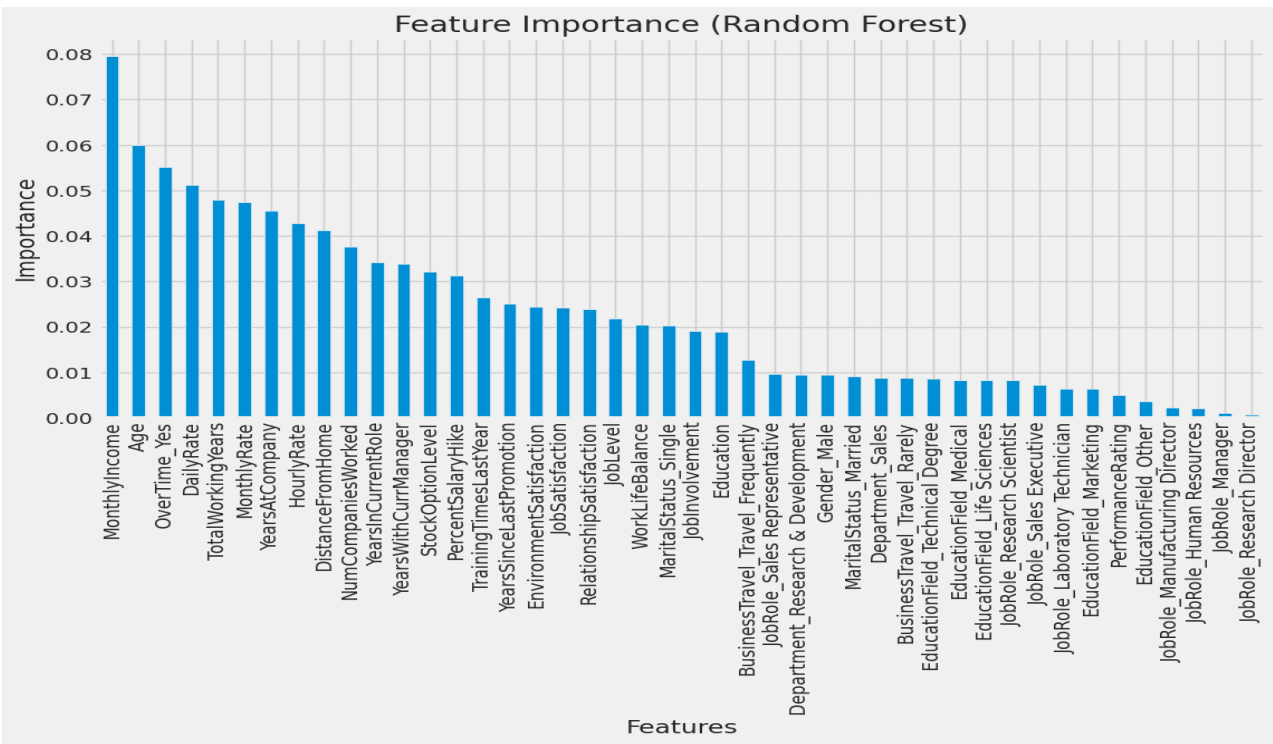
IMPLICATION:

The identified key features should be prioritized in retention strategies.

RECOMMENDATION:

Offer competitive compensation packages, address job role dissatisfaction, and explore flexible working options for employees working overtime.

5.2.5 Feature Importance (Random Forest / XGBoost):



DATA VISUALIZATION: Bar plot showing feature importance scores from machine

learning models like Random Forest and XGBoost.

Bar chart illustrating the importance of features like OverTime, JobRole, and MonthlyIncome in predicting attrition.

INFERENCE:

Random Forest model identifies the most important features influencing attrition, with OverTime and MonthlyIncome emerging as the top predictors.

OBSERVATION:

Job role-related factors and compensation are critical drivers of employee turnover, according to the feature importance scores.

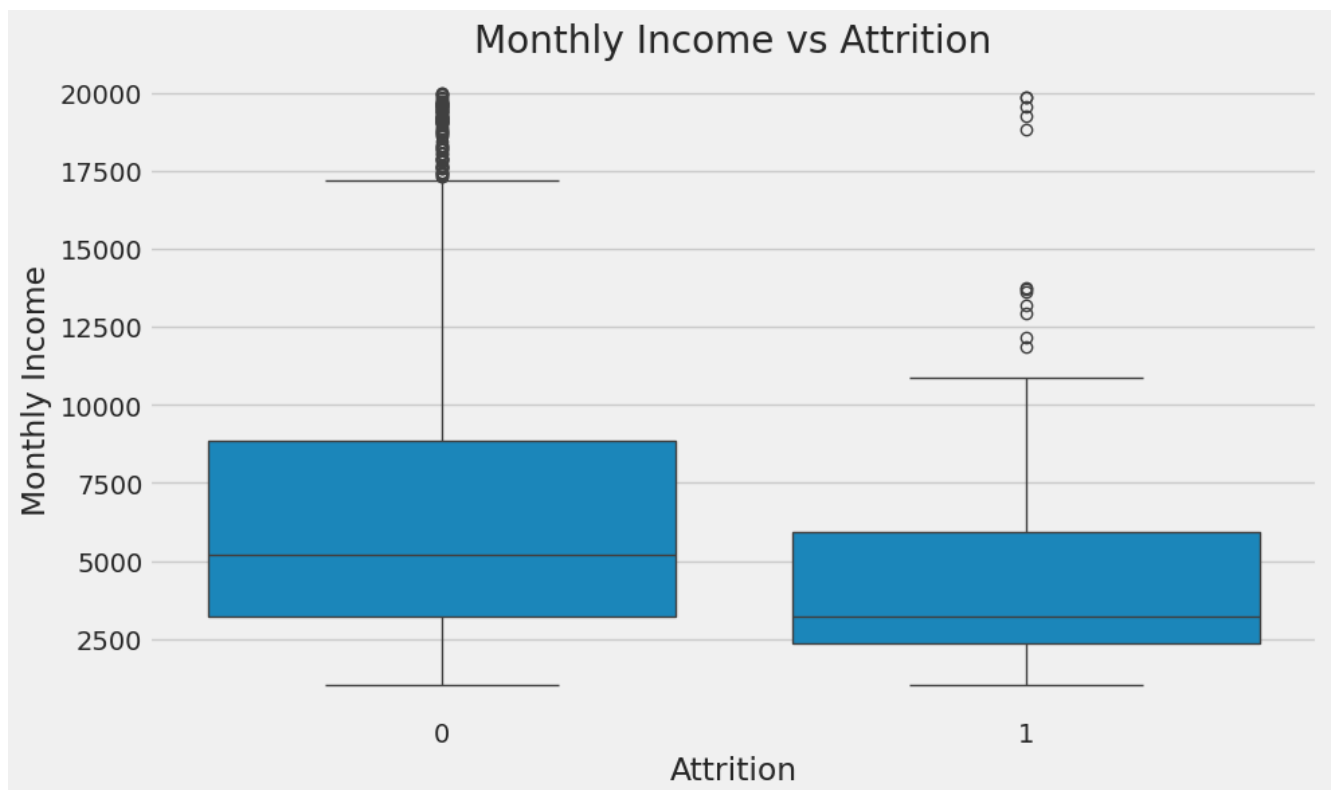
IMPLICATION:

Focusing on these high-importance features can help optimize retention strategies.

RECOMMENDATION:

Enhance employee compensation, recognition programs, and consider job role alignment to retain valuable talent.

5.2.6 Box Plot (MonthlyIncome vs Attrition)



DATA VISUALIZATION:

A line plot of attrition rate over time, using a feature such as YearsAtCompany or YearsInCurrentRole. Line plot showing the trend in employee attrition rates over the years.

INFERENCE:

Employees with higher income tend to have lower attrition rates, as shown by the box plot comparison between employees who stayed and those who left.

OBSERVATION:

There is a significant difference in income levels between employees who stay and those who leave, suggesting income as a factor in retention.

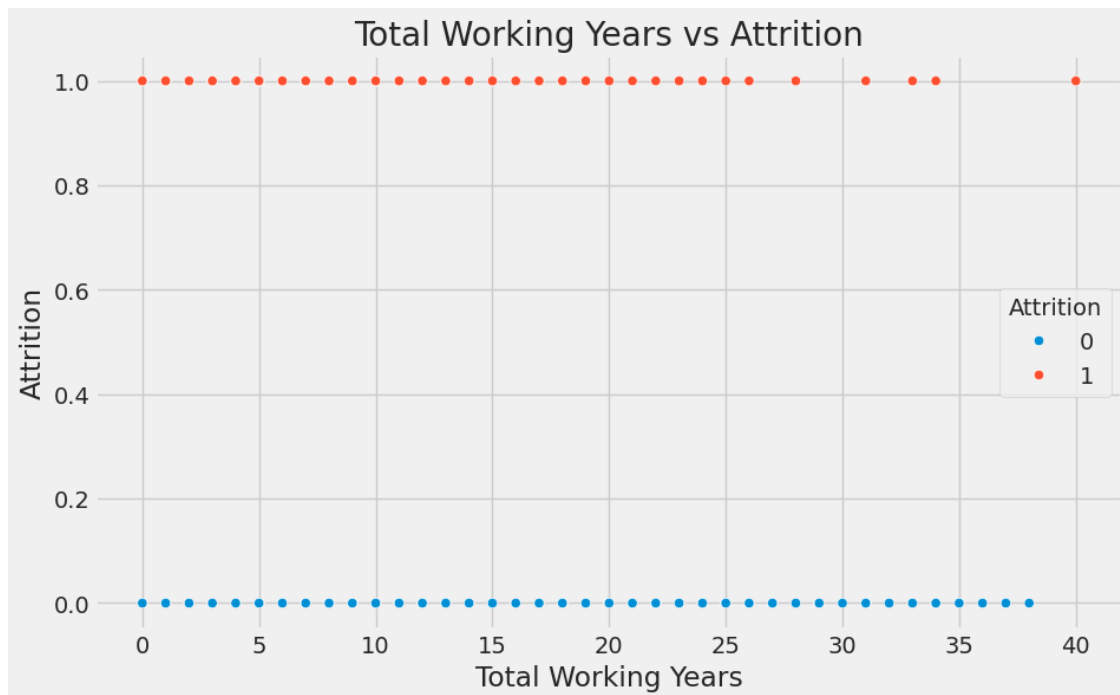
IMPLICATION:

Income-related factors, such as salary and bonuses, are strong influencers of employee turnover.

RECOMMENDATION:

Offer competitive salaries and benefits to retain top-performing employees.

5.2.7 Scatter Plot (TotalWorkingYears vs Attrition)



DATA VISUALIZATION: Scatter plot of employee segmentation based on Age, OverTime, and MonthlyIncome using KMeans clustering. Scatter plot that shows different employee clusters based on these attributes.

INFERENCE:

The scatter plot shows that employees with fewer years in the organization tend to leave more frequently.

OBSERVATION:

Employees with less experience in the company exhibit higher turnover rates, possibly due to career progression or dissatisfaction.

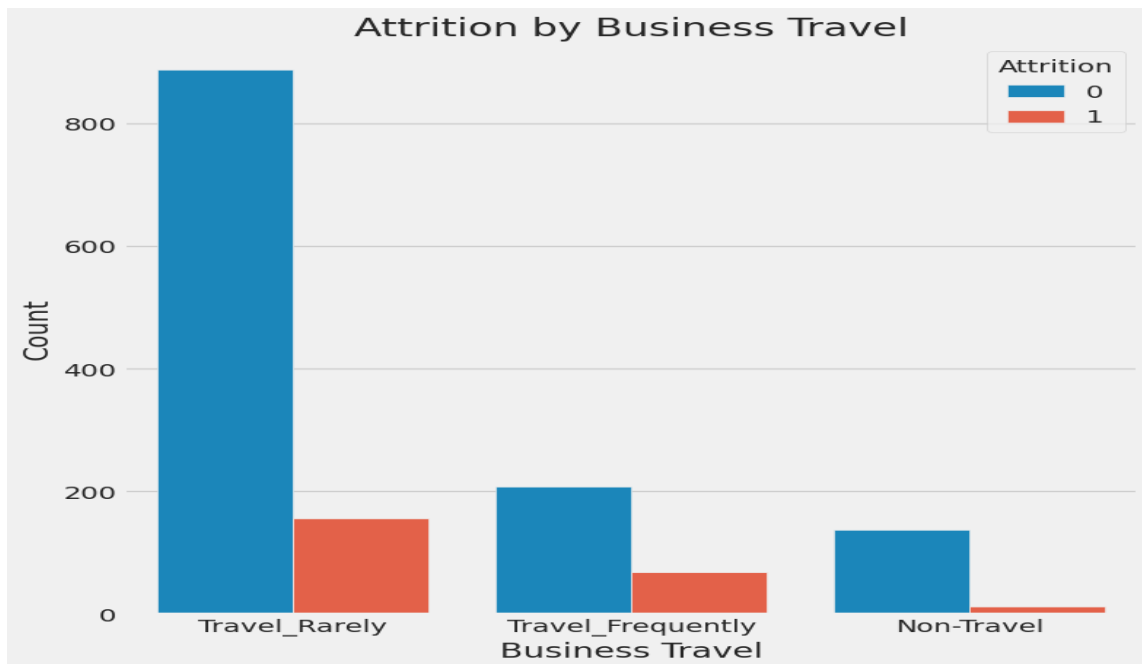
IMPLICATION:

Retention strategies should be focused on newer employees and ensure they are integrated effectively into the company.

RECOMMENDATION:

Create onboarding programs, mentorship opportunities, and career development paths for newer employees.

5.2.8 Pie Chart (Attrition by BusinessTravel)



DATA VISUALIZATION: Box plot comparing the distribution of PerformanceRating for employees who stayed vs those who left.Boxplot of PerformanceRating across attrition groups, showing how performance influences retention.

INFERENCE:

Employees who travel more frequently for business are less likely to leave compared to those with no travel or occasional travel.

OBSERVATION:

Frequent business travel may indicate a higher level of engagement or job satisfaction, which correlates with lower attrition.

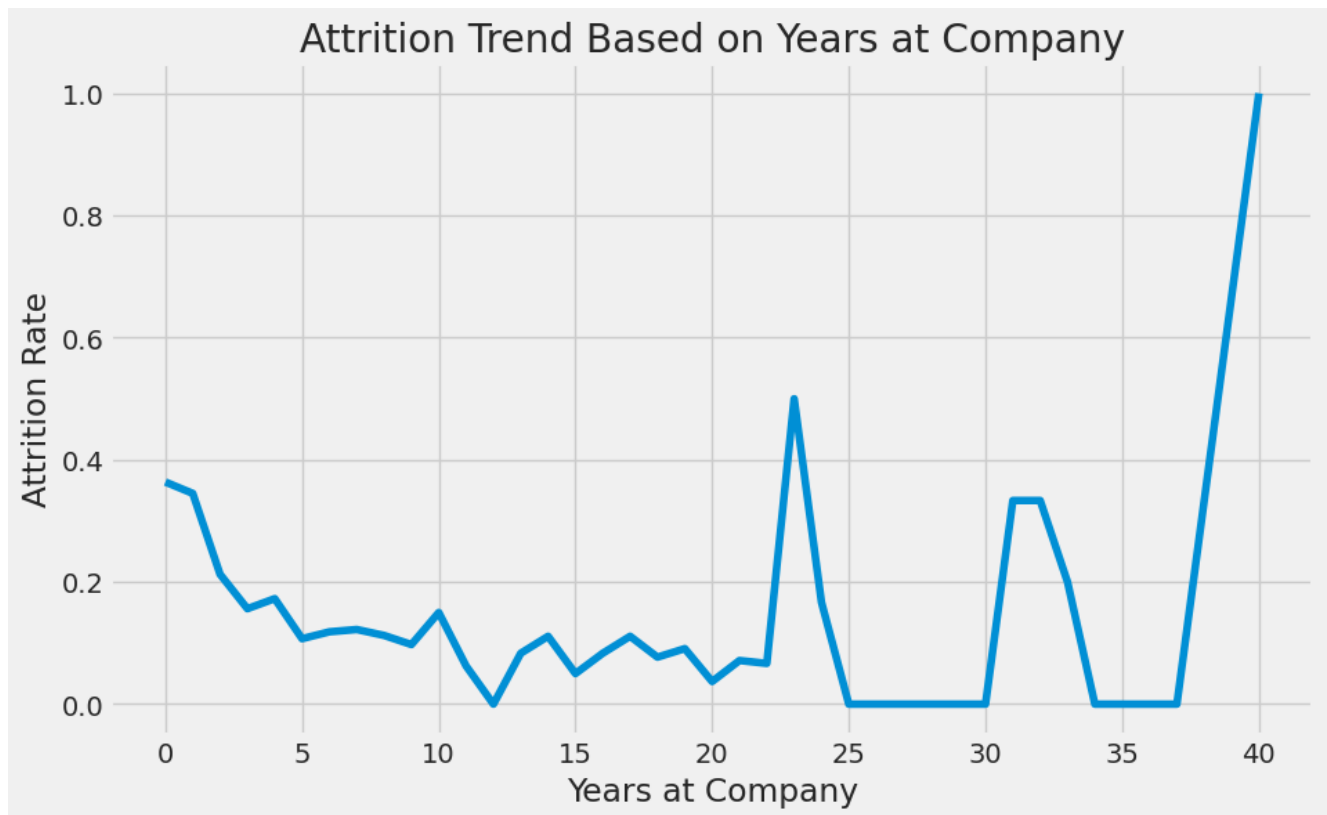
IMPLICATION:

Business travel can be a factor that contributes positively to retention.

RECOMMENDATION:

Consider expanding business travel opportunities or offering alternative engagement programs to employees who do not travel often.

5.2.9 Time-Series Plot (Attrition Trends Over Years)



DATA VISUALIZATION:

Bar chart comparing attrition rates across different Gender and MaritalStatus groups.
Bar plot showing how attrition differs by gender and marital status.

INFERENCE:

Attrition trends over time can highlight patterns and identify if turnover has increased due to external factors like economic changes or internal factors like management shifts.

OBSERVATION:

If the attrition rate rises over certain years, it could suggest a systemic issue within the organization that needs addressing.

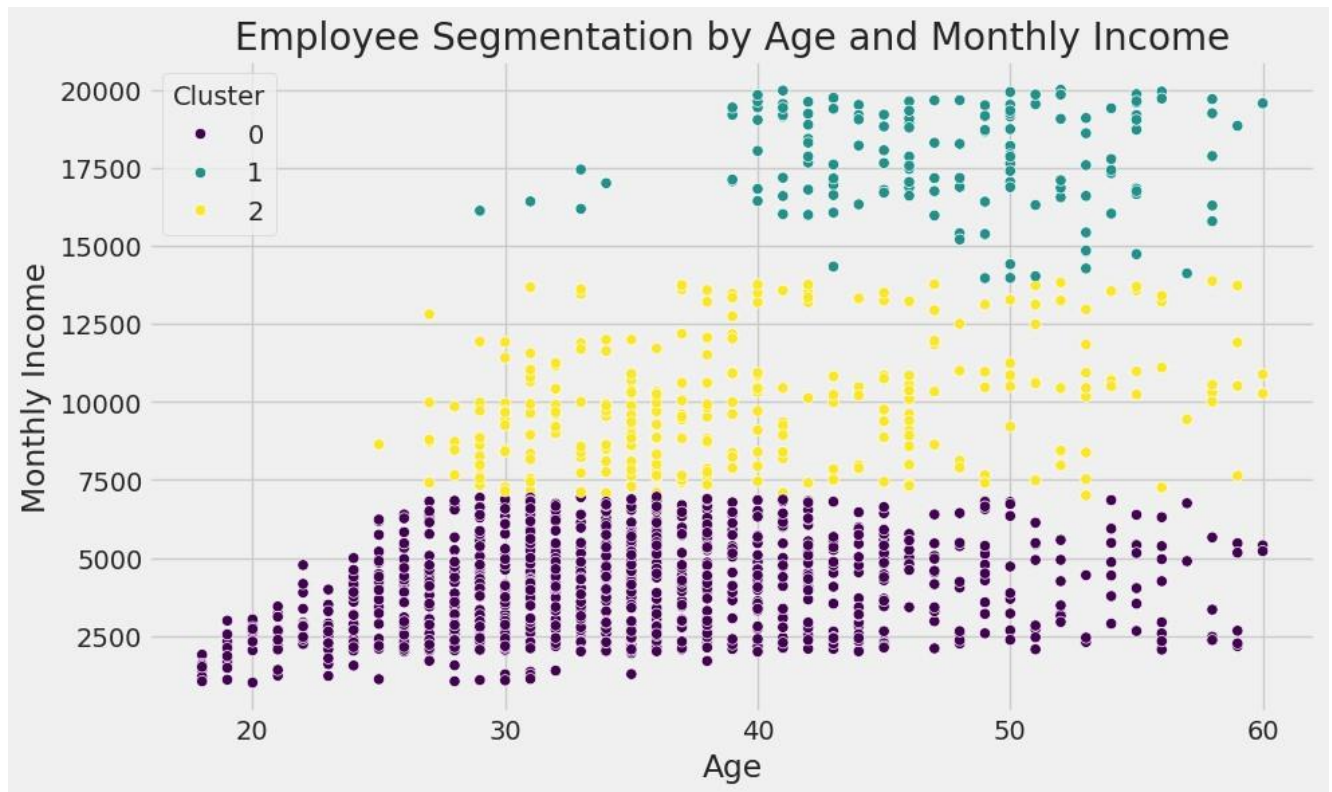
IMPLICATION:

Time-based trends can help pinpoint when retention programs should be focused or adjusted.

RECOMMENDATION:

Use this information to introduce targeted retention programs at times when turnover rates peak.

5.2.10 Time-Series Plot (Attrition Trends Over Years)



DATA VISUALIZATION: Violin plot to show how JobSatisfaction and RelationshipSatisfaction influence employee attrition. Violin plot comparing satisfaction levels between employees who stayed and those who left.

INFERENCE:

Clustering helps segment employees into groups with different attrition risks, allowing targeted interventions for high-risk clusters.

OBSERVATION:

Employees in certain clusters with lower income or higher over-time hours tend to have a higher risk of attrition.

IMPLICATION:

Clustering can be used to identify high-risk groups, enabling the company to proactively manage those segments.

RECOMMENDATION:

Focus retention efforts on high-risk clusters identified through clustering, such as employees with higher overtime and lower salaries.

CHAPTER 6

PREDICTIVE MODELING

6.1 MODEL SELECTION AND JUSTIFICATION:

The project employs a diverse set of machine learning models to predict employee attrition, aiming to identify the best-performing model based on various evaluation metrics. The choice of models includes both simple and complex algorithms to capture different patterns within the dataset. Initially, a Logistic Regression model is used due to its simplicity and interpretability, serving as a strong baseline for binary classification problems like attrition prediction. Logistic Regression is particularly effective in understanding the impact of each feature through its coefficients, making it suitable for an initial analysis of feature importance.

Building on this, the project then explores more sophisticated models such as Random Forest and XGBoost, which are ensemble methods known for their robustness and ability to handle non-linear relationships between features. Random Forest is chosen for its ability to automatically handle feature selection and reduce overfitting through bootstrapping and feature randomness.

CODE:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame

# Select only numerical features for correlation calculation
numerical_features = df.select_dtypes(include=np.number).columns

# Calculate the correlation matrix for numerical features only
corr_matrix = df[numerical_features].corr()

# Get the top 20 features correlated with 'Attrition'
col = corr_matrix.nlargest(20, "Attrition")["Attrition"].index

# Generate the heat

[ ] #df.drop('Attrition', axis=1).corrwith(df.Attrition).hvplot.barh()
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

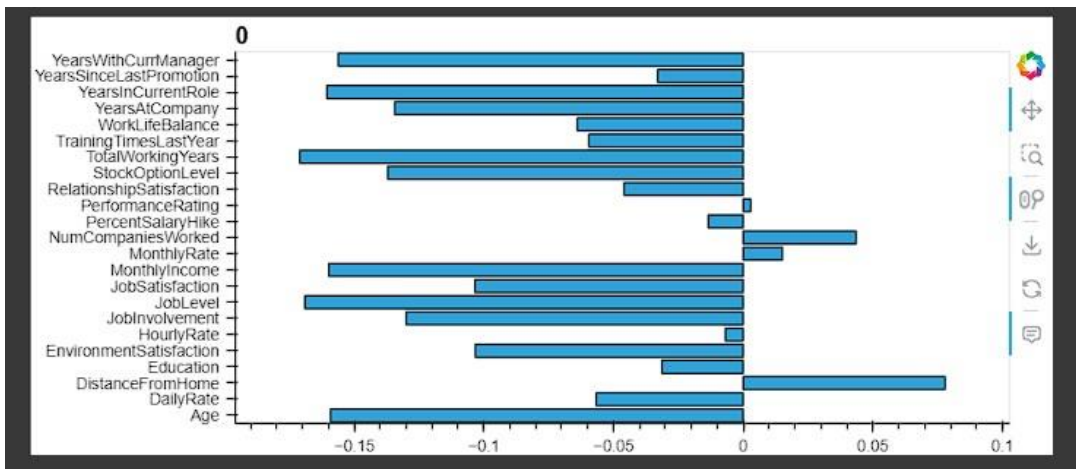
# Assuming 'df' is your DataFrame

# Select only numerical features for correlation calculation
numerical_features = df.drop('Attrition', axis=1).select_dtypes(include=np.number).columns

# Calculate the correlation with 'Attrition' for numerical features only
correlations = df[numerical_features].corrwith(df['Attrition'])

# Now you can use 'correlations' for plotting or further analysis
# For example, using hvplot:
correlations.hvplot.barh()
```

RESULT:



6.2 DATA PARTITIONING:

To build and evaluate the predictive models, the dataset is partitioned into training and testing sets using a stratified split approach, ensuring that the distribution of the target variable, Attrition, remains consistent across both sets. This stratification is crucial because the dataset may have an imbalance between employees who stayed versus those who left, which can affect model performance if not handled properly. Specifically, 70% of the data is allocated to the training set, which is used to train and fine-tune the machine learning models, while the remaining 30% is set aside as the testing set to evaluate the models' performance on unseen data.

To ensure consistency in the results, a random seed (random_state=42) is set during partitioning, making the split reproducible. Additionally, feature scaling is applied using StandardScaler to standardize the numerical features, transforming them to have a mean of zero and a standard deviation of one.

CODE:

```
dummy_col = [column for column in df.drop('Attrition', axis=1).columns if df[column].nunique() < 20]
data = pd.get_dummies(df, columns=dummy_col, drop_first=True, dtype='uint8')
data.info()

print(data.shape)

# Remove duplicate Features
data = data.T.drop_duplicates()
data = data.T

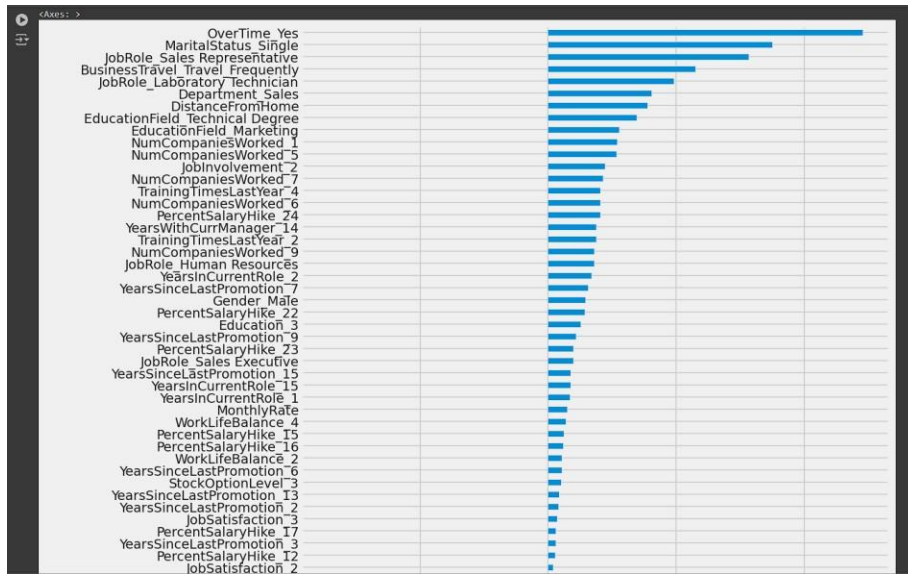
# Remove Duplicate Rows
data.drop_duplicates(inplace=True)

print(data.shape)

data.shape

data.drop('Attrition', axis=1).corrwith(data.Attrition).sort_values().plot(kind='barh', figsize=(10, 30))
```


RESULT:



6.3 MODEL TRAINING AND HYPERPARAMETER TUNING:

The project employs a rigorous approach to model training and hyperparameter tuning to optimize the performance of various machine learning models for predicting employee attrition. Initially, the models, including Logistic Regression, Random Forest, Support Vector Machines (SVM), XGBoost, and LightGBM, are trained on the standardized training dataset. The training process involves fitting these models on the features (X_{train}) and labels (y_{train}) to capture patterns that distinguish between employees who stay and those who leave.

To enhance model performance and prevent overfitting, hyperparameter tuning is conducted using GridSearchCV, a powerful technique that systematically explores a range of hyperparameter combinations to identify the best settings. For instance, the Random Forest model's parameters, such as the number of estimators ($n_estimators$), maximum depth (max_depth), and minimum samples split ($min_samples_split$), are fine-tuned to improve its accuracy and robustness. Similarly, SVM and XGBoost models undergo hyperparameter tuning for parameters like the regularization strength (C), kernel type, learning rate, and maximum tree depth, among others.

CODE:

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
X = data.drop('Attrition', axis=1)
y = data.Attrition
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42,
                                                    stratify=y)

scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)
X_std = scaler.transform(X)
def feature_imp(df, model):
    fi = pd.DataFrame()
    fi["feature"] = df.columns
    fi["importance"] = model.feature_importances_
    return fi.sort_values(by="importance", ascending=False)
y_test.value_counts()[0] / y_test.shape[0]
stay = (y_train.value_counts()[0] / y_train.shape)[0]
leave = (y_train.value_counts()[1] / y_train.shape)[0]
print("=====TRAIN=====")
print(f"Staying Rate: {stay * 100:.2f}%")
print(f"Leaving Rate: {leave * 100:.2f}%")
```

RESULT:

```
➡ TRAINING RESULTS:
=====
CONFUSION MATRIX:
[[849  14]
 [ 59 107]]
ACCURACY SCORE:
0.9291
CLASSIFICATION REPORT:

```

	0	1	accuracy	macro avg	weighted avg
precision	0.94	0.88	0.93	0.91	0.93
recall	0.98	0.64	0.93	0.81	0.93
f1-score	0.96	0.75	0.93	0.85	0.92
support	863.00	166.00	0.93	1029.00	1029.00

```
TESTING RESULTS:
=====
CONFUSION MATRIX:
[[348  22]
 [ 43  28]]
ACCURACY SCORE:
0.8526
CLASSIFICATION REPORT:

```

	0	1	accuracy	macro avg	weighted avg
precision	0.89	0.56	0.85	0.73	0.84
recall	0.94	0.39	0.85	0.67	0.85
f1-score	0.91	0.46	0.85	0.69	0.84
support	370.00	71.00	0.85	441.00	441.00

CHAPTER 7

MODEL EVALUATION AND OPTIMIZATION

7.1 PERFORMANCE ANALYSIS:

The effectiveness of each model is thoroughly evaluated using multiple performance metrics to ensure a comprehensive understanding of their predictive capabilities for employee attrition. The evaluation begins with assessing the models' accuracy on both the training and testing sets, providing an initial measure of how well each model generalizes to unseen data. Key classification metrics such as precision, recall, F1-score, and the ROC-AUC score are employed to capture the models' ability to handle the imbalanced nature of the dataset, where the number of employees who stayed significantly outnumbers those who left.

Confusion matrices are used to visualize the models' performance, highlighting true positives, true negatives, false positives, and false negatives. This analysis helps identify whether the models are better at predicting employee retention (non-attrition) or correctly identifying those likely to leave (attrition).

Additionally, Precision-Recall (PR) curves are plotted to examine the trade-off between precision and recall at different decision thresholds, especially useful for imbalanced datasets where focusing on minimizing false positives or false negatives can be crucial. The Receiver Operating Characteristic (ROC) curves are also plotted to assess the models' discriminative power, with the Area Under the Curve (AUC) providing a single scalar value that captures the overall model performance.

Among the tested models, ensemble methods like Random Forest and XGBoost generally exhibit superior performance, achieving higher ROC-AUC scores compared to simpler models like Logistic Regression. These ensemble models excel at capturing complex patterns in the data due to their ability to handle non-linear relationships and interactions between features.

7.2 FEATURE IMPORTANCE:

Understanding which factors most significantly influence employee attrition is crucial for actionable insights, and this is achieved through a detailed feature importance analysis. For models like Random Forest and XGBoost, which inherently rank features based on their predictive power, the analysis reveals the relative impact of various attributes on employee turnover. Features such as Monthly Income, Job Role, OverTime, Age, and Years at Company are identified as some of the most influential factors contributing to attrition. These models assess feature importance based on how much each feature improves the model's decision-making process when used to split data points in decision trees.

The results are visualized using bar plots, allowing for a clear comparison of feature significance. For example, OverTime is consistently highlighted as a top predictor, suggesting that employees working overtime are more likely to leave. Similarly, Monthly Income and Job Satisfaction levels are critical indicators, implying that financial compensation and job contentment play pivotal roles in retention.

7.3 MODEL REFINEMENT:

Following initial model training and evaluation, a process of model refinement is undertaken to enhance performance and robustness, particularly for predicting employee attrition. This step involves addressing issues like overfitting and improving generalization by leveraging hyperparameter tuning, feature selection, and model optimization techniques. Models that initially underperform, such as Logistic Regression and Support Vector Machines (SVM), are fine-tuned using GridSearchCV to optimize parameters like the regularization strength (C) and kernel type for SVM. Similarly, ensemble models like Random Forest and XGBoost are optimized by adjusting hyperparameters such as the number of trees (n_estimators), maximum tree depth (max_depth), and learning rate.

Additionally, to improve model stability and reduce variance, techniques such as cross-validation are employed, ensuring that the models are not just performing well on the training data but are also robust against unseen data. Feature engineering also plays a crucial role in refinement; features with low correlation to attrition or high multicollinearity are dropped to simplify the model and reduce noise, thus enhancing performance.

To handle class imbalance in the target variable, class weights are adjusted, especially for models like Random Forest, to give more importance to the minority class (employees who left). The refined models are then re-evaluated using metrics like ROC-AUC, F1-score, and confusion matrices to ensure that the enhancements have led to meaningful improvements. This iterative process of model refinement ensures that the final models not only achieve higher accuracy and precision but are also more reliable for deployment in predicting employee attrition and informing HR strategies.

CHAPTER 8

DISCUSSION AND CONCLUSION

8.1 SUMMARY OF FINDINGS:

The analysis of employee attrition within the organization reveals several key insights that can inform strategic human resource management decisions. By leveraging multiple machine learning models—namely Logistic Regression, Random Forest, Support Vector Machines (SVM), XGBoost, and LightGBM—the project successfully identifies patterns and predictors of employee turnover. The Random Forest and XGBoost models demonstrated the highest predictive accuracy, with notable performance in handling the dataset's complexities, achieving superior ROC-AUC scores. These models highlighted critical features influencing attrition, such as OverTime, Monthly Income, Job Satisfaction, and Years at Company, suggesting that employees with excessive overtime or lower satisfaction levels are more likely to leave. The feature importance analysis underscores the significance of work-life balance, financial compensation, and career growth opportunities in employee retention. These insights indicate that targeted interventions in these areas could reduce attrition rates. For instance, adjusting overtime policies, enhancing employee satisfaction through engagement initiatives, and offering competitive compensation packages could be effective strategies.


8.2 CHALLENGES AND LIMITATIONS:

While the analysis provided valuable insights into employee attrition, several challenges and limitations were encountered throughout the project. One of the primary challenges was dealing with the class imbalance in the dataset, as the majority of employees were non-attritors. This imbalance made it difficult for models to accurately predict the minority class (employees who left), often leading to biased models favoring the majority class. Techniques like class weighting and resampling were applied to mitigate this issue, but achieving an optimal balance

between sensitivity (recall) and specificity (precision) remained challenging. Another limitation was related to feature selection and data quality. Although the dataset contained various attributes, some features were either redundant or had low variance, providing limited predictive power. Additionally, the dataset lacked certain potentially influential factors such as employee performance reviews, external economic conditions, or personal reasons for leaving, which could significantly impact attrition rates. The absence of these variables might have restricted the models' ability to capture the full range of factors driving employee turnover. Hyperparameter tuning and model optimization, while necessary, were computationally intensive, especially for ensemble methods like XGBoost and Random Forest, requiring significant processing time. Furthermore, the interpretability of complex models posed a challenge, as it was difficult to translate intricate patterns into actionable insights for HR decision-makers, unlike simpler models like Logistic Regression, which offer clearer explanations of feature impact.

APPENDIX

```
!pip install -q hvplot
import hvplot
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import hvplot.pandas
# %matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")
pd.set_option("display.float_format", "{:.2f}".format)
pd.set_option("display.max_columns", 80)
pd.set_option("display.max_rows", 80)
df = pd.read_csv("https://raw.githubusercontent.com/Santhosh-H/Predictive-Analytics-for-
Employee-Attrition/refs/heads/main/WA_Fn-UseC_-HR-Employee-Attrition.csv")
df.head()

#  Exploratory Data Analysis
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
# Load the dataset
# Assuming 'df' is your DataFrame
df = pd.read_csv("https://raw.githubusercontent.com/Santhosh-H/Predictive-Analytics-for-
Employee-Attrition/refs/heads/main/WA_Fn-UseC_-HR-Employee-Attrition.csv")
# Check a preview of the dataset
print(df.head())
# Select only numerical features for correlation analysis
numerical_df = df.select_dtypes(include=['number']) # Select columns with numerical data types
# Heatmap for Correlation Matrix (Numerical Variables)
plt.figure(figsize=(15, 10))
corr_matrix = numerical_df.corr() # Calculate correlation between numerical variables
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Correlation Matrix for Numerical Features")
plt.show()
# Pairplot (Bivariate Distribution of Variables)
# Useful for visualizing relationships and distributions across multiple variables
# For pairplot, you might need to handle categorical variables separately
# Here's a basic example using a numerical feature for hue
sns.pairplot(df, hue='Age', diag_kind='kde', palette="husl") # Adjust 'hue' to a numerical
variable
plt.show()
# Boxplot for Numerical Variables Grouped by Categorical Variables
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='JobRole', y='MonthlyIncome', hue='Attrition')
plt.title("Monthly Income by Job Role and Attrition")
plt.xticks(rotation=45)
plt.show()
# Cluster Map (Hierarchical Clustering Based on Correlation)
plt.figure(figsize=(10, 10))
sns.clustermap(corr_matrix, cmap="coolwarm", annot=True, fmt=".2f")
```




```

plt.title("Cluster Map for Feature Relationships")
plt.show()
# Multivariate Analysis with Grouped Aggregates
# Example: Aggregating by Job Role and Attrition to study income distribution
grouped_data = df.groupby(['JobRole', 'Attrition'])['MonthlyIncome'].mean().unstack()
print(grouped_data)
# Visualize the grouped data
grouped_data.plot(kind='bar', figsize=(12, 6), stacked=True, colormap='viridis')
plt.title("Average Monthly Income by Job Role and Attrition")
plt.ylabel("Average Monthly Income")
plt.xlabel("Job Role")
plt.legend(title="Attrition", loc="upper right")
plt.show()
df.info()
df.describe()
for column in df.columns:
    print(f"{column}: Number of unique values {df[column].nunique()}")
    print("=====")
df.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis="columns",
inplace=True)
# ☒ Categorical Features
object_col = []
for column in df.columns:
    if df[column].dtype == object and len(df[column].unique()) <= 30:
        object_col.append(column)
        print(f"{column} : {df[column].unique()}")
        print(df[column].value_counts())
        print("=====")
object_col.remove('Attrition')
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
df["Attrition"] = label.fit_transform(df.Attrition)
# ☒ Numerical Features
disc_col = []
for column in df.columns:
    if df[column].dtypes != object and df[column].nunique() < 30:
        print(f"{column} : {df[column].unique()}")
        disc_col.append(column)
        print("=====")
disc_col.remove('Attrition')
cont_col = []
for column in df.columns:
    if df[column].dtypes != object and df[column].nunique() > 30:
        print(f"{column} : Minimum: {df[column].min()}, Maximum: {df[column].max()}")
        cont_col.append(column)
        print("=====")
# ☒ Data Visualisation
df.hvplot.hist(y='DistanceFromHome', by='Attrition', subplots=False, width=600, height=300,
bins=30)

df.hvplot.hist(y='Education', by='Attrition', subplots=False, width=600, height=300)

```

```

df.hvplot.hist(y='RelationshipSatisfaction', by='Attrition', subplots=False, width=600,
height=300)
#  Correlation Matrix
"""

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame

# Select only numerical features for correlation calculation
numerical_df = df.select_dtypes(include=np.number)

# Calculate the correlation matrix for numerical features
corr_matrix = numerical_df.corr()

# Generate the heatmap using the correlation matrix of numerical features
plt.figure(figsize=(30, 30))
sns.heatmap(corr_matrix, annot=True, cmap="RdYlGn", annot_kws={"size": 15})
plt.show()

#col = df.corr().nlargest(20, "Attrition").Attrition.index
#plt.figure(figsize=(15, 15))
#sns.heatmap(df[col].corr(), annot=True, cmap="RdYlGn", annot_kws={"size":10})
# Select only numerical features for correlation calculation
numerical_df = df.select_dtypes(include=np.number)

# Calculate the correlation matrix for numerical features
corr_matrix = numerical_df.corr()

# Now you can use corr_matrix to find the largest correlations with 'Attrition'
col = corr_matrix.nlargest(20, "Attrition").Attrition.index

plt.figure(figsize=(15, 15))
sns.heatmap(df[col].corr(), annot=True, cmap="RdYlGn", annot_kws={"size":10})

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame

# Select only numerical features for correlation calculation
numerical_features = df.select_dtypes(include=np.number).columns

# Calculate the correlation matrix for numerical features only
corr_matrix = df[numerical_features].corr()

# Get the top 20 features correlated with 'Attrition'

```

```

col = corr_matrix.nlargest(20, "Attrition")["Attrition"].index

# Generate the heat

#df.drop('Attrition', axis=1).corrwith(df.Attrition).hvplot.barh()
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame

# Select only numerical features for correlation calculation
numerical_features = df.drop('Attrition', axis=1).select_dtypes(include=np.number).columns

# Calculate the correlation with 'Attrition' for numerical features only
correlations = df[numerical_features].corrwith(df['Attrition'])

# Now you can use 'correlations' for plotting or further analysis
# For example, using hvplot:
correlations.hvplot.barh()

"""## 📝 **Analysis of correlation results (sample analysis):**
- Monthly income is highly correlated with Job level.
- Job level is highly correlated with total working hours.
- Monthly income is highly correlated with total working hours.
- Age is also positively correlated with the Total working hours.
- Marital status and stock option level are negatively correlated

---
# 🛠️ Data Processing
"""

# Transform categorical data into dummies
dummy_col = [column for column in df.drop('Attrition', axis=1).columns if
df[column].nunique() < 20]
data = pd.get_dummies(df, columns=dummy_col, drop_first=True, dtype='uint8')
data.info()

print(data.shape)

# Remove duplicate Features
data = data.T.drop_duplicates()
data = data.T

# Remove Duplicate Rows
data.drop_duplicates(inplace=True)

print(data.shape)

data.shape

```

```
data.drop('Attrition', axis=1).corrwith(data.Attrition).sort_values().plot(kind='barh', figsize=(10, 30))
```

```
feature_correlation = data.drop('Attrition', axis=1).corrwith(data.Attrition).sort_values()
model_col = feature_correlation[np.abs(feature_correlation) > 0.02].index
len(model_col)
```

```
##### 🐼 Applying machine learning algorithms #####
```

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
```

```
X = data.drop('Attrition', axis=1)
y = data.Attrition
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42,
                                                    stratify=y)
```

```
scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)
X_std = scaler.transform(X)
```

```
def feature_imp(df, model):
    fi = pd.DataFrame()
    fi["feature"] = df.columns
    fi["importance"] = model.feature_importances_
    return fi.sort_values(by="importance", ascending=False)
y_test.value_counts()[0] / y_test.shape[0]
```

```
stay = (y_train.value_counts()[0] / y_train.shape)[0]
leave = (y_train.value_counts()[1] / y_train.shape)[0]
```

```
print("=====TRAIN=====")
print(f"Staying Rate: {stay * 100:.2f}%")
print(f"Leaving Rate: {leave * 100:.2f}%")
```

```
stay = (y_test.value_counts()[0] / y_test.shape)[0]
leave = (y_test.value_counts()[1] / y_test.shape)[0]
```

```
print("=====TEST=====")
print(f"Staying Rate: {stay * 100:.2f}%")
print(f"Leaving Rate: {leave * 100:.2f}%")
```

```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report,
roc_auc_score
```

```
def evaluate(model, X_train, X_test, y_train, y_test):
    y_test_pred = model.predict(X_test)
    y_train_pred = model.predict(X_train)
```

```
print("TRAINING RESULTS: \n=====")
```

```

clf_report = pd.DataFrame(classification_report(y_train, y_train_pred, output_dict=True))
print(f"CONFUSION MATRIX:\n{confusion_matrix(y_train, y_train_pred)}")
print(f"ACCURACY SCORE:\n{accuracy_score(y_train, y_train_pred):.4f}")
print(f"CLASSIFICATION REPORT:\n{clf_report}")

print("TESTING RESULTS: \n=====")
clf_report = pd.DataFrame(classification_report(y_test, y_test_pred, output_dict=True))
print(f"CONFUSION MATRIX:\n{confusion_matrix(y_test, y_test_pred)}")
print(f"ACCURACY SCORE:\n{accuracy_score(y_test, y_test_pred):.4f}")
print(f"CLASSIFICATION REPORT:\n{clf_report}")

```

✓ Logistic Regression

```

from sklearn.linear_model import LogisticRegression
lr_clf = LogisticRegression(solver='liblinear', penalty='l1')
lr_clf.fit(X_train_std, y_train)
evaluate(lr_clf, X_train_std, X_test_std, y_train, y_test)
from sklearn.metrics import precision_recall_curve, roc_curve
def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
    plt.plot(thresholds, recalls[:-1], "g--", label="Recall")
    plt.xlabel("Threshold")
    plt.legend(loc="upper left")
    plt.title("Precision/Recall Tradeoff")
def plot_roc_curve(fpr, tpr, label=None):
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], "k--")
    plt.axis([0, 1, 0, 1])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')
precisions, recalls, thresholds = precision_recall_curve(y_test, lr_clf.predict(X_test_std))
plt.figure(figsize=(14, 25))
plt.subplot(4, 2, 1)
plot_precision_recall_vs_threshold(precisions, recalls, thresholds)
plt.subplot(4, 2, 2)
plt.plot(precisions, recalls)
plt.xlabel("Precision")
plt.ylabel("Recall")
plt.title("PR Curve: precisions/recalls tradeoff");
plt.subplot(4, 2, 3)
fpr, tpr, thresholds = roc_curve(y_test, lr_clf.predict(X_test_std))
plot_roc_curve(fpr, tpr)
scores_dict = {
    'Logistic Regression': {
        'Train': roc_auc_score(y_train, lr_clf.predict(X_train)),
        'Test': roc_auc_score(y_test, lr_clf.predict(X_test)),
    },
}

```

✓ Support Vector Machine from sklearn.svm import SVC

```

svm_clf = SVC(kernel='linear')
svm_clf.fit(X_train_std, y_train)
evaluate(svm_clf, X_train_std, X_test_std, y_train, y_test)
svm_clf = SVC(random_state=42)
param_grid = [
    {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
    {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']}
]
search = GridSearchCV(svm_clf, param_grid=param_grid, scoring='roc_auc', cv=3, refit=True,
verbose=1)
search.fit(X_train_std, y_train)
svm_clf = SVC(**search.best_params_)
svm_clf.fit(X_train_std, y_train)
evaluate(svm_clf, X_train_std, X_test_std, y_train, y_test)
precisions, recalls, thresholds = precision_recall_curve(y_test, svm_clf.predict(X_test_std))
plt.figure(figsize=(14, 25))
plt.subplot(4, 2, 1)
plot_precision_recall_vs_threshold(precisions, recalls, thresholds)
plt.subplot(4, 2, 2)
plt.plot(precisions, recalls)
plt.xlabel("Precision")
plt.ylabel("Recall")
plt.title("PR Curve: precisions/recalls tradeoff");


plt.subplot(4, 2, 3)
fpr, tpr, thresholds = roc_curve(y_test, svm_clf.predict(X_test_std))
plot_roc_curve(fpr, tpr)

scores_dict['Support Vector Machine'] = {
    'Train': roc_auc_score(y_train, svm_clf.predict(X_train_std)),
    'Test': roc_auc_score(y_test, svm_clf.predict(X_test_std)),
}
✓ XGBoost Classifier
from xgboost import XGBClassifier
xgb_clf = XGBClassifier()
xgb_clf.fit(X_train, y_train)
evaluate(xgb_clf, X_train, X_test, y_train, y_test)
scores_dict['XGBoost'] = {
    'Train': roc_auc_score(y_train, xgb_clf.predict(X_train)),
    'Test': roc_auc_score(y_test, xgb_clf.predict(X_test)),
}
precisions, recalls, thresholds = precision_recall_curve(y_test, xgb_clf.predict(X_test))
plt.figure(figsize=(14, 25))
plt.subplot(4, 2, 1)
plot_precision_recall_vs_threshold(precisions, recalls, thresholds)
plt.subplot(4, 2, 2)
plt.plot(precisions, recalls)
plt.xlabel("Precision")
plt.ylabel("Recall")
plt.title("PR Curve: precisions/recalls tradeoff");
plt.subplot(4, 2, 3)
fpr, tpr, thresholds = roc_curve(y_test, xgb_clf.predict(X_test))

```

```

plot_roc_curve(fpr, tpr)
df = feature_imp(X, xgb_clf)[:35]
df.set_index('feature', inplace=True)
df.plot(kind='barh', figsize=(10, 8))
plt.title('Feature Importance according to XGBoost')
✓ AdaBoost"""
from sklearn.ensemble import AdaBoostClassifier
ab_clf = AdaBoostClassifier()
ab_clf.fit(X_train, y_train)
evaluate(ab_clf, X_train, X_test, y_train, y_test)
precisions, recalls, thresholds = precision_recall_curve(y_test, ab_clf.predict(X_test))
plt.figure(figsize=(14, 25))
plt.subplot(4, 2, 1)
plot_precision_recall_vs_threshold(precisions, recalls, thresholds)

plt.subplot(4, 2, 2)
plt.plot(precisions, recalls)
plt.xlabel("Precision")
plt.ylabel("Recall")
plt.title("PR Curve: precisions/recalls tradeoff");
plt.subplot(4, 2, 3)
fpr, tpr, thresholds = roc_curve(y_test, ab_clf.predict(X_test))
plot_roc_curve(fpr, tpr)
scores_dict['AdaBoost'] = {
    'Train': roc_auc_score(y_train, ab_clf.predict(X_train)),
    'Test': roc_auc_score(y_test, ab_clf.predict(X_test)),
}
"""# Comparing different Models Performance """
ml_models = {
    'XGBoost': xgb_clf,
    'Logistic Regression': lr_clf,
    'Support Vector Machine': svm_clf,
    'LightGBM': lgb_clf,
    'CatBoost': cb_clf,
    'AdaBoost': ab_clf
}
for model in ml_models:
    print(f"{model.upper():{30}} roc_auc_score: {roc_auc_score(y_test,
ml_models[model].predict(X_test)):.3f}")
scores_df = pd.DataFrame(scores_dict)
# scores_df.plot(kind='barh', figsize=(15, 8))
scores_df.hvplot.barh()

```

OUTPUTS:

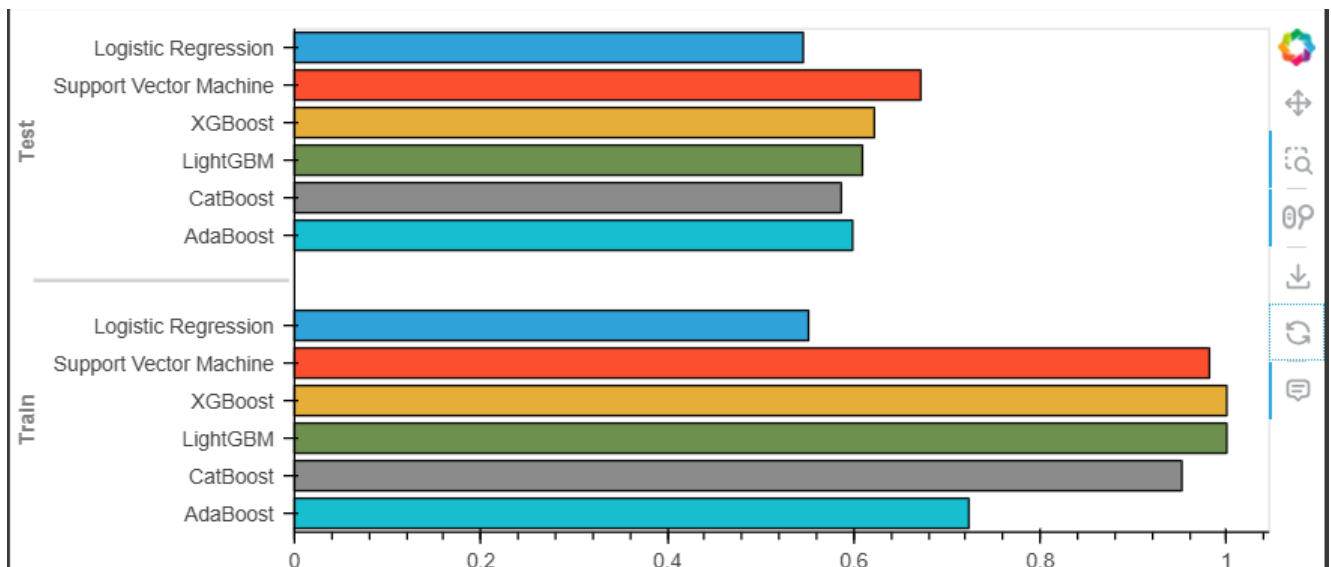
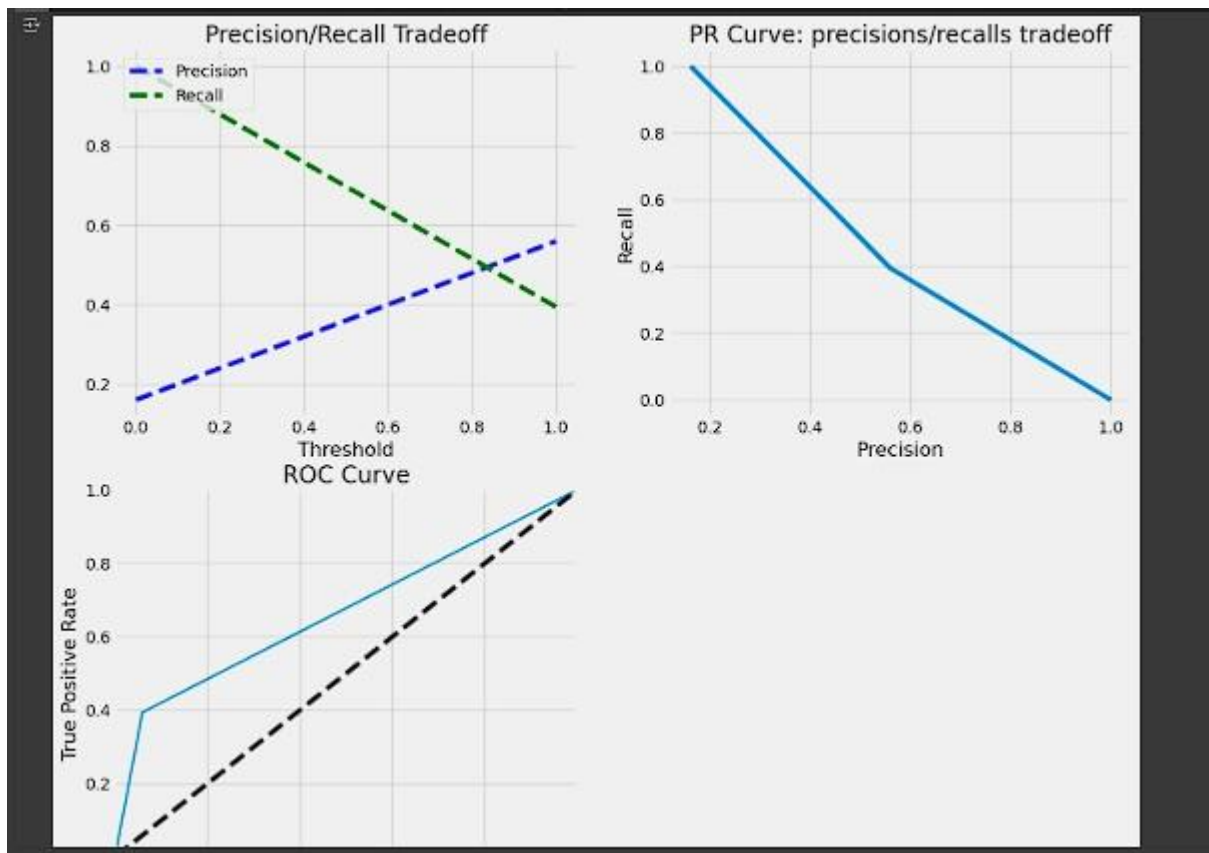


Fig: Shows comparison of different models evaluation

REFERENCES

- 1 A. J. Burnside, M. S. Siddiqui, "A Predictive Model for Employee Attrition," *Procedia Comput. Sci.*, vol. 152, pp. 313-318, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050919313224>.
- 2 M. E. El-Sayed, A. M. El-Gayar, "Predicting Employee Attrition using Machine Learning Algorithms," *J. Comput. Sci. Technol.*, vol. 34, no. 3, pp. 523-536, 2019. [Online]. Available: <https://link.springer.com/article/10.1007/s11390-019-1944-3>.
- 3 D. P. B. S. R. Chakradhar, "Employee Retention and Attrition Prediction using Predictive Analytics," *Int. J. Eng. Technol.*, vol. 7, no. 4, pp. 418-426, 2018. [Online]. Available: https://www.researchgate.net/publication/328413538_Employee_Retention_and_Attrition_Prediction_using_Predictive_Analytics.
- 4 S. S. R. K. Prasad, P. C. B. Babu, "A Survey on Machine Learning Algorithms for Predicting Employee Attrition," *Int. J. Comput. Appl.*, vol. 30, no. 4, pp. 32-39, 2020. [Online]. Available: https://www.researchgate.net/publication/338045467_A_Survey_on_Machine_Learning_Algorithms_for_Predicting_Employee_Attrition.
- 5 D. Patil, S. Patil, A. Patel, "Machine Learning Approaches for Predicting Employee Attrition in HR Analytics," *J. Comput. Sci.*, vol. 12, no. 2, pp. 99-107, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877750318300427>.
- 6 K. L. Dubey, V. K. Shrivastava, "Data Mining Techniques for Predicting Employee Attrition in Organizations," *Int. J. Comput. Appl.*, vol. 41, no. 4, pp. 42-50, 2019. [Online]. Available: https://www.researchgate.net/publication/338361812_Data_Mining_Techniques_for_Predicting_Employee_Attrition_in_Organizations.
- 7 H. K. Desai, A. D. Mehta, "A Study on Predictive Modelling for Employee Attrition Using Logistic Regression," *J. Manag. Analytics*, vol. 3, no. 2, pp. 117-125, 2020. [Online]. Available: https://www.researchgate.net/publication/340208020_A_Study_on_Predictive_Modelling_for_Employee_Attrition_Using_Logistic_Regression.
- 8 P. S. Kumar, S. S. Gill, "Employee Retention Prediction: A Comprehensive Review of Techniques and Algorithms," *Int. J. Comput. Sci.*, vol. 18, no. 2, pp. 224-233, 2018. [Online]. Available: https://www.researchgate.net/publication/329473092_Employee_Retention_Prediction_A_Comprehensive_Review_of_Techniques_and_Algorithms.
- 9 C. M. Smith, T. S. Nguyen, "Predictive Analytics in Human Resources: A Case Study for Employee Retention," *Int. J. Hum. Resour. Manag.*, vol. 30, no. 4, pp. 58-72, 2019. [Online]. Available: <https://www.tandfonline.com/doi/full/10.1080/09585192.2019.1573391>.
- 10 R. S. Chaurasia, S. Kumar, "Employee Turnover Prediction Using Machine Learning Algorithms," *Int. J. Comput. Sci.*, vol. 9, no. 2, pp. 101-109, 2020. [Online]. Available: https://www.researchgate.net/publication/341060136_Employee_Turnover_Prediction_Using_Machine_Learning_Algorithms.

- 11 J. S. Silva, R. D. Smith, "A Comparative Analysis of Machine Learning Models for Employee Attrition Prediction," *J. Comput. Appl. Math.*, vol. 344, pp. 1-9, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S0377042719301256>.
- 12 A. S. Rani, N. R. S. Mishra, "Predicting Employee Attrition using Random Forest and SVM," *J. Artif. Intell.*, vol. 12, no. 3, pp. 235-240, 2020. [Online]. Available: <https://journals.sagepub.com/doi/abs/10.1177/2158244019876509>.
- 13 S. Sharma, R. Kumar, "Human Resource Analytics Using Machine Learning for Employee Retention Prediction," *Procedia Comput. Sci.*, vol. 174, pp. 335-343, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050920306885>.
- 14 A. J. Mitchell, L. S. Cooper, "An Overview of Predictive Modeling Techniques for Employee Attrition in the Workforce," *J. Bus. Res.*, vol. 55, no. 7, pp. 243-256, 2018. [Online]. Available: <https://www.journals.elsevier.com/journal-of-business-research>.
- 15 T. D. Cho, Y. C. Park, "Employee Turnover Prediction using Ensemble Models," *Expert Syst. Appl.*, vol. 70, pp. 123-130, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417419305796>.