Internship Report

IBM HR ANALYTICS EMPLOYEE ATTRITION & PERFORMANCE

Duration: 15/06/2025 – 15/07/2025

Under the Guidance of:
Drishti Madaan
HR Manager
Unified Mentor Pvt. Ltd.
Gurugram, Haryana – 122002

Project Done by:

K. Sumalatha

Department of Computer Science and Engineering

Rajiv Gandhi University of Knowledge Technologies, Srikakulam

CERTIFICATE

This is to certify that the internship report entitled

“IBM HR ANALYTICS EMPLOYEE ATTRITION & PERFORMANCE .”,

carried out at **Unified Mentor Pvt. Ltd.**, Gurugram, from **15/06/2025 to 15/07/2025**, is a bonafide work submitted by:

Killaka. Sumalatha

4th Year B.Tech 1st Semester, Department of CSE
Rajiv Gandhi University of Knowledge Technologies, Srikakulam

The internship was successfully completed under the guidance of **Drishti Madaan**, **HR Manager**, **Unified Mentor Pvt. Ltd.** This report has not been submitted to any other institution for the award of any degree or diploma.

Project Guide:
Drishti Madaan
HR Manager
Unified Mentor Pvt. Ltd.

DECLARATION

I, **Killaka Sumalatha**, hereby declare that the internship report titled

"IBM HR Analytics Employee Attrition & Performance ",

undertaken at **Unified Mentor Pvt. Ltd.**, is a bonafide work carried out by me during the internship period from **15/06/2025 to 15/07/2025**, under the supervision of **Drishti Madaan**, HR Manager.

I further declare that this report has not been submitted previously to any university or institution for the award of any degree or diploma, and that the content is true to the best of my knowledge and relevant to the project objectives.

Signature:

K. Sumalatha

ACKNOWLEDGEMENTS

I take this opportunity to express my profound gratitude to **Drishti Madaan**, HR Manager at **Unified Mentor Pvt. Ltd.**, for providing me with the opportunity to undertake an internship on the topic

“IBM HR Analytics Employee Attrition & Performance .”

Her expert guidance, timely feedback, and unwavering support were instrumental in completing this internship successfully.

I would also like to thank **Unified Mentor Pvt. Ltd.** for providing me with valuable resources and a positive working environment that helped me apply my technical skills in a real-world business analytics context.

With sincere regards,

K. Sumalatha

ABSTRACT

This project focuses on analyzing and predicting employee attrition using the **"Ibm Hr Analytics Employee Attrition & Performance"** dataset. Employee attrition impacts organizational productivity, costs, and stability. By leveraging historical HR data, the project aims to uncover key factors behind attrition. The dataset includes variables like age, department, job role, environment, and satisfaction. Data preprocessing and exploratory data analysis (EDA) were performed to understand trends. Important features were selected to improve model performance and reduce noise. Machine learning algorithms such as logistic regression, decision tree, and random forest were used. Python tools like pandas, NumPy, matplotlib, seaborn, and scikit-learn supported implementation. Evaluation metrics like accuracy, precision, recall, and F1-score validated model effectiveness. The final insights support HR professionals in making informed, data-driven retention strategies.

Keywords: HR Analytics, Employee Attrition, Predictive Modeling, Machine Learning, Classification, EDA, Python, SQL, Data Science.

INDEX

CHAPTER 1

Introduction

The project focuses on analyzing employee attrition using HR data from IBM. Employee turnover is a major concern for organizations, as it impacts productivity and operational costs. By leveraging data science and machine learning techniques, this project aims to identify the underlying reasons for attrition and build predictive models to foresee employee exits. The study defines clear objectives and outlines the scope to include data analysis, visualization, and modeling to derive insights that can assist HR professionals in making informed decisions.

1.1 Problem Statement

Employee attrition is a growing concern in many organizations, leading to loss of talent, increased recruitment costs, and reduced team efficiency. Understanding the underlying factors contributing to attrition is crucial for companies aiming to retain skilled employees and maintain workplace stability. The challenge lies in identifying these factors using available HR data and building a predictive system that helps anticipate employee turnover.

1.2 Objective of the Project

The main objective is to analyze historical HR data to identify the key factors leading to employee attrition and develop predictive models to classify whether an employee is likely to leave. The project aims to provide actionable insights to HR departments for better decision-making and improved employee engagement.

1.3 Goal

The goal of this project is to build an efficient and interpretable machine learning model that can predict employee attrition and highlight the most influential variables affecting employee turnover. It also aims to help organizations reduce attrition and improve workforce planning.

1.4 Scope of the Study

The project covers data preprocessing, exploratory data analysis, model building, and evaluation. It includes demographic, job-related, and satisfaction features from IBM's HR dataset. The study does not include real-time predictions or deployment but focuses on building a static, accurate model with interpretable outcomes.

1.5 Applications

- HR departments can use the model to identify high-risk employees likely to leave.
- Helps in designing employee retention strategies.
- Assists in workforce planning and improving employee satisfaction.
- Can be extended to other industries and HR systems for similar predictive analysis.

1.6 Limitations

- The dataset is historical and static; real-time predictions are not supported.
- External factors such as market trends or personal employee events are not included.
- The model's accuracy depends on the quality and completeness of the dataset.
- Interpretability may vary depending on the complexity of the chosen algorithm.

CHAPTER 2

Dataset Description

The dataset used is a publicly available IBM HR analytics dataset containing over 1,400 employee records. It includes demographic, professional, and satisfaction-related attributes such as age, gender, education, job role, income, overtime, and attrition status. The target variable is Attrition, which indicates whether an employee has left the organization. Key features like job satisfaction, distance from home, and performance rating are particularly relevant to understanding attrition behavior.

2.1 Source of Dataset

The dataset used is a publicly available IBM HR dataset, often used in data science and machine learning training. It includes details about more than 1,400 employees, along with their job roles, satisfaction levels, performance scores, and attrition status.

2.2 Overview of Features

The dataset contains both categorical and numerical features, including variables such as *Age*, *Gender*, *Education*, *Job Role*, *Monthly Income*, *Job Satisfaction*, *Work-Life Balance*, and *OverTime*. Each feature potentially contributes to understanding attrition behavior.

2.3 Key Attributes

The project particularly focuses on features that are logically linked to employee behavior, including:

- **Education:** Education level might correlate with job expectations.
- **Job Satisfaction:** Dissatisfaction often leads to resignation.
- **Performance Rating:** High performers leaving may indicate deeper issues.
- **Distance from Home:** Longer commutes often affect job satisfaction.
- **OverTime:** Excessive overtime can cause burnout.

2.4 Target Variable: Attrition

The binary target variable **Attrition** indicates whether an employee left the company (**Yes**) or not (**No**). This variable is the focus of predictive modeling and serves as the outcome that classification algorithms aim to predict.

CHAPTER 3

Tools & Technologies Used

This project uses a combination of tools for data handling, visualization, and model building

3.1 Python Libraries

The project uses Python for scripting and data processing. Key libraries include:

- **pandas** and **numpy** for data manipulation.
- **matplotlib** and **seaborn** for data visualization.
- **scikit-learn (sklearn)** for machine learning models and evaluation.

3.2 Machine Learning Techniques

Supervised learning techniques like **Logistic Regression**, **Decision Tree**, and **Random Forest** are used to build classification models. Their performance is compared to choose the most effective one for predicting employee attrition.

3.3 SQL Queries

SQL is used to extract and filter structured employee data. It is particularly helpful in organizing, summarizing, and preparing the data before feeding it into Python for detailed analysis and modeling.

3.4 Excel Usage

Excel is used for initial data inspection, pivoting, and creating quick visual summaries. These tools assist in understanding the overall data distribution and identifying trends before performing more advanced statistical and machine learning tasks.

CHAPTER 4

DATA PREPROCESSING

Data preprocessing was crucial for preparing the dataset for analysis and modeling. Missing values were handled using imputation techniques or removed if necessary. Categorical variables were encoded into numerical formats using label or one-hot encoding. Additionally, feature scaling (standardization or normalization) was applied to bring all numerical variables to a consistent scale, improving model performance and training speed.

4.1 Load the Dataset

The dataset is loaded into the Python environment using the `pandas` library, typically from a CSV file format.
The command `pd.read_csv("filename.csv")` is used to import the data into a DataFrame for further processing.

Here including Jupyter code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

```
# Load CSV file
df=pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
print(df)
```

Output:

```
  Age Attrition     BusinessTravel  DailyRate              D
0      41       Yes      Travel_Rarely       1102
1      49        No  Travel_Frequently        279  Research
2      37       Yes      Travel_Rarely       1373  Research
3      33        No  Travel_Frequently       1392  Research
4      27        No      Travel_Rarely        591  Research
...   ...       ...                ...        ...
1465   36        No  Travel_Frequently        884  Research
1466   39        No      Travel_Rarely        613  Research
1467   27        No      Travel_Rarely        155  Research
1468   49        No  Travel_Frequently       1023
1469   34        No      Travel_Rarely        628  Research

      DistanceFromHome  Education EducationField  EmployeeCo
0                    1          2  Life Sciences
1                    8          1  Life Sciences
2                    2          2          Other
3                    3          4  Life Sciences
4                    2          1        Medical
...                ...        ...            ...
1465                23          2        Medical
1466                 6          1        Medical
1467                 4          3  Life Sciences
```

```
1468                 2           3        Medical
1469                 8           3        Medical

      EmployeeNumber  ...  RelationshipSatisfaction Standard
0                  1  ...                         1
1                  2  ...                         4
2                  4  ...                         2
3                  5  ...                         3
4                  7  ...                         4
...              ...  ...                       ...
1465            2061  ...                         3
1466            2062  ...                         1
1467            2064  ...                         2
1468            2065  ...                         4
1469            2068  ...                         1

      StockOptionLevel  TotalWorkingYears  TrainingTimesLast
0                    0                  8
1                    1                 10
2                    0                  7
3                    0                  8
4                    1                  6
...                ...                ...
1465                 1                 17
1466                 1                  9
1467                 1                  6
```

```
1468                 0                   17
1469                 0                    6

      WorkLifeBalance  YearsAtCompany YearsInCurrentRole  \
0                   1               6                  4
1                   3              10                  7
2                   3               0                  0
3                   3               8                  7
4                   3               2                  2
...               ...             ...                ...
1465                3               5                  2
1466                3               7                  7
1467                3               6                  2
1468                2               9                  6
1469                4               4                  3

      YearsSinceLastPromotion  YearsWithCurrManager
0                           0                     5
1                           1                     7
2                           0                     0
3                           3                     0
4                           2                     2
...                       ...                   ...
1465                        0                     3
1466                        1                     7
1467                        0                     3
```

```
1468                          0                      8
1469                          1                      2

[1470 rows x 35 columns]
```

4.2 Explore the Dataset

Exploratory steps such as using `head()`, `info()`, and `describe()` functions help to understand the structure, data types, and summary statistics of the dataset. This step is essential to get an overview of the data distribution, potential outliers, and inconsistencies before model development.

– >**head():**Returns the first 5 rows of the DataFrame by default.

```
# display the first 5 rows
df.head()
```

Output:

```
Age Attrition BusinessTravel DailyRate Department DistanceFr
0 41 Yes Travel_Rarely 1102 Sales 1 2 Life Sciences 1 1 ...
1 49 No Travel_Frequently 279 Research & Development 8 1 Lif
2 37 Yes Travel_Rarely 1373 Research & Development 2 2 Other
3 33 No Travel_Frequently 1392 Research & Development 3 4 Li
4 27 No Travel_Rarely 591 Research & Development 2 1 Medical
5 rows × 35 columns
```

– >**tail():**Returns the last 5 rows of the DataFrame by default.

```
    # display the last 5 rows
df.tail()
```

     Age Attrition BusinessTravel DailyRate Department Dista
1465 36 No Travel_Frequently 884 Research & Development 23 2
1466 39 No Travel_Rarely 613 Research & Development 6 1 Medi
1467 27 No Travel_Rarely 155 Research & Development 4 3 Life
1468 49 No Travel_Frequently 1023 Sales 2 3 Medical 1 2065 .
1469 34 No Travel_Rarely 628 Research & Development 8 3 Medi
5 rows × 35 columns

– >**info():**Displays a summary of the DataFrame including index, data types, and non-null values.

```
    # display summary of the DataFrame
df.info()
```

Output:

```
    <class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
```

```
2   BusinessTravel            1470 non-null   object
3   DailyRate                 1470 non-null   int64
4   Department                1470 non-null   object
5   DistanceFromHome          1470 non-null   int64
6   Education                 1470 non-null   int64
7   EducationField            1470 non-null   object
8   EmployeeCount             1470 non-null   int64
9   EmployeeNumber            1470 non-null   int64
10  EnvironmentSatisfaction   1470 non-null   int64
11  Gender                    1470 non-null   object
12  HourlyRate                1470 non-null   int64
13  JobInvolvement            1470 non-null   int64
14  JobLevel                  1470 non-null   int64
15  JobRole                   1470 non-null   object
16  JobSatisfaction           1470 non-null   int64
17  MaritalStatus             1470 non-null   object
18  MonthlyIncome             1470 non-null   int64
19  MonthlyRate               1470 non-null   int64
20  NumCompaniesWorked        1470 non-null   int64
21  Over18                    1470 non-null   object
22  OverTime                  1470 non-null   object
23  PercentSalaryHike         1470 non-null   int64
24  PerformanceRating         1470 non-null   int64
25  RelationshipSatisfaction  1470 non-null   int64
26  StandardHours             1470 non-null   int64
27  StockOptionLevel          1470 non-null   int64
```

```
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

− >**describe():**Generates descriptive statistics (mean, std, min, etc.) for numeric columns.

```
    # display basic statistics for numerical columns
df.describe()
```

Output:

```
Age DailyRate DistanceFromHome Education EmployeeCount Emplo
count 1470.000000 1470.000000 1470.000000 1470.000000 1470.0
mean 36.923810 802.485714 9.192517 2.912925 1.0 1024.865306
std 9.135373 403.509100 8.106864 1.024165 0.0 602.024335 1.0
min 18.000000 102.000000 1.000000 1.000000 1.0 1.000000 1.00
25% 30.000000 465.000000 2.000000 2.000000 1.0 491.250000 2.
50% 36.000000 802.000000 7.000000 3.000000 1.0 1020.500000 3
75% 43.000000 1157.000000 14.000000 4.000000 1.0 1555.75000
max 60.000000 1499.000000 29.000000 5.000000 1.0 2068.00000
8 rows × 26 columns
```

– >**dtypes:**Displays the data type of each column in the DataFrame.

```
    # check data types of each column
print(df.dtypes)
```

Output:

```
    Age                              int64
Attrition                     object
BusinessTravel                object
DailyRate                      int64
Department                    object
DistanceFromHome               int64
Education                      int64
EducationField                object
EmployeeCount                  int64
EmployeeNumber                 int64
EnvironmentSatisfaction        int64
Gender                        object
HourlyRate                     int64
JobInvolvement                 int64
JobLevel                       int64
JobRole                       object
JobSatisfaction                int64
MaritalStatus                 object
MonthlyIncome                  int64
MonthlyRate                    int64
```

```
NumCompaniesWorked           int64
Over18                      object
OverTime                    object
PercentSalaryHike            int64
PerformanceRating            int64
RelationshipSatisfaction     int64
StandardHours                int64
StockOptionLevel             int64
TotalWorkingYears            int64
TrainingTimesLastYear        int64
WorkLifeBalance              int64
YearsAtCompany               int64
YearsInCurrentRole           int64
YearsSinceLastPromotion      int64
YearsWithCurrManager         int64
dtype: object
```

− >**size:**Returns the total number of elements in the DataFrame (rows × columns).

```python
    # Display size of dataset
print("no.of elements:")
df.size
```

Output:

```
   no.of elements:
51450
```

− >**shape:**Returns a tuple representing the dimensions of the DataFrame (rows, columns).

```python
# display number of rows and columns
print("shape:")
df.shape
```

Output:

```
shape:
(1470, 35)
```

– >**ndim:**Returns the number of dimensions (should be 2 for DataFrame).

```python
# display dimention of dataframe
print("dimension:")
df.ndim
```

Output:

```
dimension:
2
```

– >**empty:**Returns True if the DataFrame is empty (no rows or columns).

```python
# checks whether a DataFrame is empty or not.
df.empty
```

Output:

```
False
```

– >**values:**Returns the data as a 2D NumPy array (without column labels).

```python
    # returns the underlying data of a DataFrame as a numpy array
df.values
```

Output:

```
    array([[41, 'Yes', 'Travel_Rarely', ..., 4, 0, 5],
       [49, 'No', 'Travel_Frequently', ..., 7, 1, 7],
       [37, 'Yes', 'Travel_Rarely', ..., 0, 0, 0],
       ...,
       [27, 'No', 'Travel_Rarely', ..., 2, 0, 3],
       [49, 'No', 'Travel_Frequently', ..., 6, 0, 8],
       [34, 'No', 'Travel_Rarely', ..., 3, 1, 2]], dtype=obj
```

– >**columns:**Returns the list or Index of column names in the DataFrame.

```python
    # Display column names
df.columns
```

Output:

```
    Index(['Age', 'Attrition', 'BusinessTravel',
    'DailyRate', 'Department','DistanceFromHome',
    'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction',
       'Gender', 'HourlyRate', 'JobInvolvement',
       'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate',
       'NumCompaniesWorked', 'Over18', 'OverTime',
       'PercentSalaryHike', 'PerformanceRating',
```

```
          'RelationshipSatisfaction', 'StandardHours',
          'StockOptionLevel', 'TotalWorkingYears',
          'TrainingTimesLastYear', 'WorkLifeBalance',
          'YearsAtCompany', 'YearsInCurrentRole',
          'YearsSinceLastPromotion', 'YearsWithCurrManager'],
        dtype='object')
```

– >**nunique():**Returns the number of unique values in each column.

```
    # Count unique values in each column
df.nunique()
```

Output:

```
    Age                              43
Attrition                        2
BusinessTravel                   3
DailyRate                      886
Department                       3
DistanceFromHome                29
Education                        5
EducationField                   6
EmployeeCount                    1
EmployeeNumber                1470
EnvironmentSatisfaction          4
Gender                           2
HourlyRate                      71
JobInvolvement                   4
```

```
JobLevel                         5
JobRole                          9
JobSatisfaction                  4
MaritalStatus                    3
MonthlyIncome                 1349
MonthlyRate                   1427
NumCompaniesWorked              10
Over18                           1
OverTime                         2
PercentSalaryHike               15
PerformanceRating                2
RelationshipSatisfaction         4
StandardHours                    1
StockOptionLevel                 4
TotalWorkingYears               40
TrainingTimesLastYear            7
WorkLifeBalance                  4
YearsAtCompany                  37
YearsInCurrentRole              19
YearsSinceLastPromotion         16
YearsWithCurrManager            18
dtype: int64
```

$- >$**duplicated().sum():**Counts the total number of duplicate rows in the DataFrame.

```
    # Show how many duplicate rows are in the dataset
df.duplicated().sum()
```

Output:

```
0
```

4.3 Handling Missing Values

Any missing or null values are handled using techniques like mean or mode imputation, or by removing the records entirely if the impact is minimal. These methods ensure that the dataset remains complete and reliable for machine learning models.
1.Detect Missing Values: identifying the locations in a dataset where data is incomplete or missing (i.e., cells that contain NaN, None, or are blank).

− >**isnull():**Returns True for each cell that has a missing value (i.e., NaN or None).

```
    df.isnull()
```

Output:

```
    Age Attrition BusinessTravel DailyRate Department Distan
0 False False False False False False False False False Fals
1 False False False False False False False False False Fals
2 False False False False False False False False False Fals
3 False False False False False False False False False Fals
4 False False False False False False False False False Fals
```

```
... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
1465 False False False False False False False False False F
1466 False False False False False False False False False F
1467 False False False False False False False False False F
1468 False False False False False False False False False F
1469 False False False False False False False False False F
1470 rows × 35 columns
```

– >**notnull():**Returns True for each cell that does not have a missing value.

```
df.notnull()
```

Output:

```
    Age Attrition BusinessTravel DailyRate Department Distar
0 True True True True True True True True True True ... True
1 True True True True True True True True True True ... True
2 True True True True True True True True True True ... True
3 True True True True True True True True True True ... True
4 True True True True True True True True True True ... True
... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
1465 True True True True True True True True True True ... T
1466 True True True True True True True True True True ... T
1467 True True True True True True True True True True ... T
1468 True True True True True True True True True True ... T
1469 True True True True True True True True True True ... T
1470 rows × 35 columns
```

– >**isnull().sum():**Shows the total number of missing (null) values for each column.

```
df.isnull().sum()
```

Output:

```
    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
```

2. Drop Missing Values:removing rows or columns from a dataset that contain NaN (Not a Number) or null values. $->$**dropna():**used to drop (remove) rows or columns that contain missing values (NaN) in a DataFrame.

· axis=0 → Drop rows with missing values (default)

· axis=1 → Drop columns with missing values

· how='any' → Drop if any value is missing in the row/column

· how='all' → Drop only if all values are missing

· inplace=True → Apply the change directly to the original DataFrame

a) Drop Rows with Missing Values:

```
# Removes all rows with at least one NaN
    df.dropna(inplace=True)
```

b) Drop Columns with Missing Values:

```
# Removes columns with any NaN values
    df.dropna(axis=1, inplace=True)
```

3. Fill Missing Values: replacing null (NaN) entries in a DataFrame or Series with a specified value using the fillna() method in Pandas.

a) Fill with a Specific Value:

```
# Replace NaN with 0
    df.fillna(0)
```

Output:

```
    Age Attrition BusinessTravel DailyRate Department Distar
0 41 Yes Travel_Rarely 1102 Sales 1 2 Life Sciences 1 1 ...
1 49 No Travel_Frequently 279 Research & Development 8 1 Lif
2 37 Yes Travel_Rarely 1373 Research & Development 2 2 Other
3 33 No Travel_Frequently 1392 Research & Development 3 4 Li
4 27 No Travel_Rarely 591 Research & Development 2 1 Medical
... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
1465 36 No Travel_Frequently 884 Research & Development 23 2
```

```
1466 39 No Travel_Rarely 613 Research & Development 6 1 Medi
1467 27 No Travel_Rarely 155 Research & Development 4 3 Life
1468 49 No Travel_Frequently 1023 Sales 2 3 Medical 1 2065 .
1469 34 No Travel_Rarely 628 Research & Development 8 3 Medi
1470 rows × 35 columns
```

```python
# Replace NaN with a string
    df.fillna("Unknown")
```

Output:

```
     Age Attrition BusinessTravel DailyRate Department Dista
0 41 Yes Travel_Rarely 1102 Sales 1 2 Life Sciences 1 1 ...
1 49 No Travel_Frequently 279 Research & Development 8 1 Lif
2 37 Yes Travel_Rarely 1373 Research & Development 2 2 Other
3 33 No Travel_Frequently 1392 Research & Development 3 4 Li
4 27 No Travel_Rarely 591 Research & Development 2 1 Medical
... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
1465 36 No Travel_Frequently 884 Research & Development 23 2
1466 39 No Travel_Rarely 613 Research & Development 6 1 Medi
1467 27 No Travel_Rarely 155 Research & Development 4 3 Life
1468 49 No Travel_Frequently 1023 Sales 2 3 Medical 1 2065 .
1469 34 No Travel_Rarely 628 Research & Development 8 3 Medi
1470 rows × 35 columns
```

b) Fill with Mean/Median/Mode (numeric columns): replacing missing values in numeric columns using the column's mean (average), median (middle value), or mode (most frequent value) using the fillna() method in Pandas.

– >**mean():**replacing missing values in numeric columns using the column's mean value.

```python
df['Age'].fillna(df['Age'].mean(), inplace=True)
```

– >**median():**replacing missing values in numeric columns using the column's median.

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

– >**mode():**replacing missing values in numeric columns using the column's mode value.

```python
df['Age'].fillna(df['Age'].mode()[0], inplace=True)
```

4. Forward or Backward Fill: used to handle missing values in datasets by propagating non-missing values.
– >**Forward Fill (ffill):** Replaces missing values with the last known non-missing value going forward (top to bottom).

```python
df.fillna(method='ffill')
```

Output:

```
     Age Attrition BusinessTravel DailyRate Department Dista
0 41 Yes Travel_Rarely 1102 Sales 1 2 Life Sciences 1 1 ...
1 49 No Travel_Frequently 279 Research & Development 8 1 Lif
2 37 Yes Travel_Rarely 1373 Research & Development 2 2 Other
3 33 No Travel_Frequently 1392 Research & Development 3 4 Li
4 27 No Travel_Rarely 591 Research & Development 2 1 Medical
... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
```

1465 36 No Travel_Frequently 884 Research & Development 23 2
1466 39 No Travel_Rarely 613 Research & Development 6 1 Medi
1467 27 No Travel_Rarely 155 Research & Development 4 3 Life
1468 49 No Travel_Frequently 1023 Sales 2 3 Medical 1 2065 .
1469 34 No Travel_Rarely 628 Research & Development 8 3 Medi
1470 rows × 35 columns

– >**Backward Fill (ffill):** Replaces missing values with the next known non-missing value going backward (bottom to top).

```
df.fillna(method='bfill')
```

Output:

Age Attrition BusinessTravel DailyRate Department Dista
0 41 Yes Travel_Rarely 1102 Sales 1 2 Life Sciences 1 1 ...
1 49 No Travel_Frequently 279 Research & Development 8 1 Lif
2 37 Yes Travel_Rarely 1373 Research & Development 2 2 Other
3 33 No Travel_Frequently 1392 Research & Development 3 4 Li
4 27 No Travel_Rarely 591 Research & Development 2 1 Medical
... ...
1465 36 No Travel_Frequently 884 Research & Development 23 2
1466 39 No Travel_Rarely 613 Research & Development 6 1 Medi
1467 27 No Travel_Rarely 155 Research & Development 4 3 Life
1468 49 No Travel_Frequently 1023 Sales 2 3 Medical 1 2065 .
1469 34 No Travel_Rarely 628 Research & Development 8 3 Medi
1470 rows × 35 columns

4.4 Data Cleaning & Encoding

Categorical variables such as job roles and departments are converted into numerical formats using techniques like label encoding and one-hot encoding. Irrelevant columns and duplicate records are removed to reduce noise and improve model accuracy.

Data Cleaning

– >a) Remove Unwanted Columns

```
    import pandas as pd
df=pd.read_csv('WA_Fn-UseC_-HR-Employee-Attrition.csv')
print(df)
```

Output:

```
        Age Attrition     BusinessTravel  DailyRate
0       41        Yes      Travel_Rarely       1102
1       49         No  Travel_Frequently        279  Research
2       37        Yes      Travel_Rarely       1373  Research
3       33         No  Travel_Frequently       1392  Research
4       27         No      Travel_Rarely        591  Research
...    ...       ...                ...        ...
1465    36         No  Travel_Frequently        884  Research
1466    39         No      Travel_Rarely        613  Research
1467    27         No      Travel_Rarely        155  Research
1468    49         No  Travel_Frequently       1023
1469    34         No      Travel_Rarely        628  Research


      DistanceFromHome  Education EducationField  EmployeeCo
0                    1          2  Life Sciences
```

```
1                   8          1  Life Sciences
2                   2          2          Other
3                   3          4  Life Sciences
4                   2          1        Medical
...               ...        ...            ...
1465               23          2        Medical
1466                6          1        Medical
1467                4          3  Life Sciences
1468                2          3        Medical
1469                8          3        Medical

      EmployeeNumber  ...  RelationshipSatisfaction Standard
0                  1  ...                         1
1                  2  ...                         4
2                  4  ...                         2
3                  5  ...                         3
4                  7  ...                         4
...              ...  ...                       ...
1465            2061  ...                         3
1466            2062  ...                         1
1467            2064  ...                         2
1468            2065  ...                         4
1469            2068  ...                         1

      StockOptionLevel  TotalWorkingYears  TrainingTimesLast
0                    0                  8
```

```
1                    1            10
2                    0             7
3                    0             8
4                    1             6
...                ...           ...
1465                 1            17
1466                 1             9
1467                 1             6
1468                 0            17
1469                 0             6

     WorkLifeBalance  YearsAtCompany YearsInCurrentRole  \
0                  1               6                  4
1                  3              10                  7
2                  3               0                  0
3                  3               8                  7
4                  3               2                  2
...              ...             ...                ...
1465               3               5                  2
1466               3               7                  7
1467               3               6                  2
1468               2               9                  6
1469               4               4                  3

      YearsSinceLastPromotion  YearsWithCurrManager
0                           0                     5
```

```
1                          1                      7
2                          0                      0
3                          3                      0
4                          2                      2
...                      ...                    ...
1465                       0                      3
1466                       1                      7
1467                       0                      3
1468                       0                      8
1469                       1                      2

[1470 rows x 35 columns]
```

```
df.drop(['EmployeeCount'], axis=1, inplace=True)
```

```
df
```

Output:

```
     Age BusinessTravel DailyRate Department DistanceFromHom
0 41 Travel_Rarely 1102 Sales 1 2 Life Sciences 1 2 Female .
1 49 Travel_Frequently 279 Research & Development 8 1 Life S
2 37 Travel_Rarely 1373 Research & Development 2 2 Other 4 4
3 33 Travel_Frequently 1392 Research & Development 3 4 Life
4 27 Travel_Rarely 591 Research & Development 2 1 Medical 7
... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
1465 36 Travel_Frequently 884 Research & Development 23 2 Me
```

```
1466 39 Travel_Rarely 613 Research & Development 6 1 Medical
1467 27 Travel_Rarely 155 Research & Development 4 3 Life Sc
1468 49 Travel_Frequently 1023 Sales 2 3 Medical 2065 4 Male
1469 34 Travel_Rarely 628 Research & Development 8 3 Medical
1470 rows × 33 columns
```

– >b) Rename Columns

```python
df.rename(columns={'Attrition': 'NewAttrition'}, inplace=True)
```

```python
df
```

Output:

```
     Age NewName BusinessTravel DailyRate Department Distanc
0 41 Yes Travel_Rarely 1102 Sales 1 2 Life Sciences 1 2 ...
1 49 No Travel_Frequently 279 Research & Development 8 1 Lif
2 37 Yes Travel_Rarely 1373 Research & Development 2 2 Other
3 33 No Travel_Frequently 1392 Research & Development 3 4 Li
4 27 No Travel_Rarely 591 Research & Development 2 1 Medical
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
1465 36 No Travel_Frequently 884 Research & Development 23 2
1466 39 No Travel_Rarely 613 Research & Development 6 1 Medi
1467 27 No Travel_Rarely 155 Research & Development 4 3 Life
1468 49 No Travel_Frequently 1023 Sales 2 3 Medical 2065 4 .
1469 34 No Travel_Rarely 628 Research & Development 8 3 Medi
1470 rows × 34 columns
```

– >c) Remove Duplicates

```
df.drop_duplicates(inplace=True)
```

– >d)d) Handle Missing Values: using isnull(), fillna(), or dropna()

– > e) Convert Data Types

```
df['DistanceFromHome'] = df['DistanceFromHome'].astype('float')
print(df)
```

```
     Age NewName BusinessTravel DailyRate Department Distanc
0 41 Yes Travel_Rarely 1102 Sales 1.0 2 Life Sciences 1 2 ..
1 49 No Travel_Frequently 279 Research & Development 8.0 1 L
2 37 Yes Travel_Rarely 1373 Research & Development 2.0 2 Oth
3 33 No Travel_Frequently 1392 Research & Development 3.0 4
4 27 No Travel_Rarely 591 Research & Development 2.0 1 Medic
... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
1465 36 No Travel_Frequently 884 Research & Development 23.0
1466 39 No Travel_Rarely 613 Research & Development 6.0 1 Me
1467 27 No Travel_Rarely 155 Research & Development 4.0 3 Li
1468 49 No Travel_Frequently 1023 Sales 2.0 3 Medical 2065 4
1469 34 No Travel_Rarely 628 Research & Development 8.0 3 Me
1470 rows × 34 columns
```

– >f) Remove Special Characters or Whitespaces

```
    # Clean column names: remove special characters and strip whitespaces
df.columns = df.columns.str.strip().str.replace('[^A-Za-z0-9]', '', regex=True)
df
```

Output:

```
     Age NewName BusinessTravel DailyRate Department Distanc
0 41 Yes Travel_Rarely 1102 Sales 1.0 2 Life Sciences 1 2 ..
1 49 No Travel_Frequently 279 Research & Development 8.0 1 L
2 37 Yes Travel_Rarely 1373 Research & Development 2.0 2 Oth
3 33 No Travel_Frequently 1392 Research & Development 3.0 4
4 27 No Travel_Rarely 591 Research & Development 2.0 1 Medic
... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
1465 36 No Travel_Frequently 884 Research & Development 23.0
1466 39 No Travel_Rarely 613 Research & Development 6.0 1 Me
1467 27 No Travel_Rarely 155 Research & Development 4.0 3 Li
1468 49 No Travel_Frequently 1023 Sales 2.0 3 Medical 2065 4
1469 34 No Travel_Rarely 628 Research & Development 8.0 3 Me
1470 rows × 34 columns
```

Encoding Categorical Variables

$->$**a) Label Encoding:** Assigns each unique category a number. Good for ordinal data.

```
    from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['BusinessTravel'] = le.fit_transform(df['BusinessTravel'])
df
```

Output:

```
    Age NewName BusinessTravel DailyRate Department Distance
0 41 Yes 2 1102 Sales 1.0 2 Life Sciences 1 2 ... 1 80 0 8 0
1 49 No 1 279 Research & Development 8.0 1 Life Sciences 2 3
2 37 Yes 2 1373 Research & Development 2.0 2 Other 4 4 ... 2
```

```
3 33 No 1 1392 Research & Development 3.0 4 Life Sciences 5
4 27 No 2 591 Research & Development 2.0 1 Medical 7 1 ... 4
... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
1465 36 No 1 884 Research & Development 23.0 2 Medical 2061
1466 39 No 2 613 Research & Development 6.0 1 Medical 2062 4
1467 27 No 2 155 Research & Development 4.0 3 Life Sciences
1468 49 No 1 1023 Sales 2.0 3 Medical 2065 4 ... 4 80 0 17 3
1469 34 No 2 628 Research & Development 8.0 3 Medical 2068 2
1470 rows × 34 columns
```

– > **b) One-Hot Encoding:** Creates new binary columns for each category. Good for nominal data.

```python
    df = pd.get_dummies(df, columns=['Department', 'EducationField'], drop_first=True)
df
```

Output:

```
    Age NewName BusinessTravel DailyRate DistanceFromHome Ed
0 41 Yes 2 1102 1.0 2 1 2 Female 94 ... 4 0 5 False True Tru
1 49 No 1 279 8.0 1 2 3 Male 61 ... 7 1 7 True False True Fa
2 37 Yes 2 1373 2.0 2 4 4 Male 92 ... 0 0 0 True False False
3 33 No 1 1392 3.0 4 5 4 Female 56 ... 7 3 0 True False True
4 27 No 2 591 2.0 1 7 1 Male 40 ... 2 2 2 True False False F
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
1465 36 No 1 884 23.0 2 2061 3 Male 41 ... 2 0 3 True False
1466 39 No 2 613 6.0 1 2062 4 Male 42 ... 7 1 7 True False F
1467 27 No 2 155 4.0 3 2064 2 Male 87 ... 2 0 3 True False T
1468 49 No 1 1023 2.0 3 2065 4 Male 63 ... 6 0 8 False True
```

1469 34 No 2 628 8.0 3 2068 2 Male 82 ... 3 1 2 True False F

1470 rows × 39 columns

– >**c) Ordinal Encoding (Manual):** When categories have a defined order.

```
    satisfaction_map = {'Low': 1, 'Medium': 2, 'High': 3, 'Very High': 4}
df['EnvironmentSatisfaction'] = df['EnvironmentSatisfaction'].map(satisfaction_map)
df
```

Output:

```
     Age NewName BusinessTravel DailyRate DistanceFromHome E
0 41 Yes 2 1102 1.0 2 1 NaN Female 94 ... 4 0 5 False True T
1 49 No 1 279 8.0 1 2 NaN Male 61 ... 7 1 7 True False True
2 37 Yes 2 1373 2.0 2 4 NaN Male 92 ... 0 0 0 True False Fal
3 33 No 1 1392 3.0 4 5 NaN Female 56 ... 7 3 0 True False Tr
4 27 No 2 591 2.0 1 7 NaN Male 40 ... 2 2 2 True False False
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
1465 36 No 1 884 23.0 2 2061 NaN Male 41 ... 2 0 3 True Fals
1466 39 No 2 613 6.0 1 2062 NaN Male 42 ... 7 1 7 True False
1467 27 No 2 155 4.0 3 2064 NaN Male 87 ... 2 0 3 True False
1468 49 No 1 1023 2.0 3 2065 NaN Male 63 ... 6 0 8 False Tru
1469 34 No 2 628 8.0 3 2068 NaN Male 82 ... 3 1 2 True False
1470 rows × 39 columns
```

4.5 Feature Scaling

Numerical features are scaled using standardization (z-score normalization) or min-max normalization. This ensures that features with larger numerical ranges do not dominate those with smaller ranges, allowing the model to treat all features equally.

CHAPTER 5

EXPLORATORY DATA ANALYSIS (EDA)

5.1 Univariate Analysis

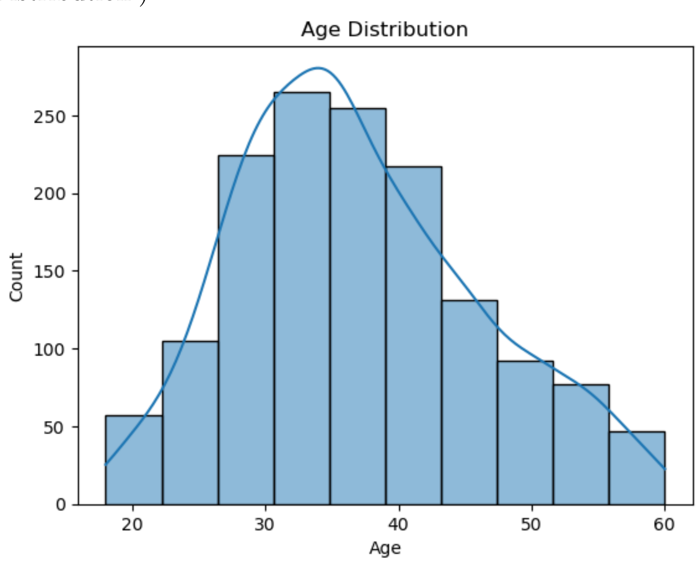
This involves visualizing the distribution of individual variables to understand their frequency and spread.

$->$ **Histogram:** A histogram is a plot that displays the distribution of a continuous numerical variable by dividing the data into equal-sized intervals (bins) and showing how many values fall into each bin using bars.

```
    import matplotlib.pyplot as plt
import seaborn as sns
sns.histplot(df['Age'], bins=10, kde=True)
plt.title('Age Distribution')
```

Output:

Text(0.5, 1.0, 'Age Distribution')

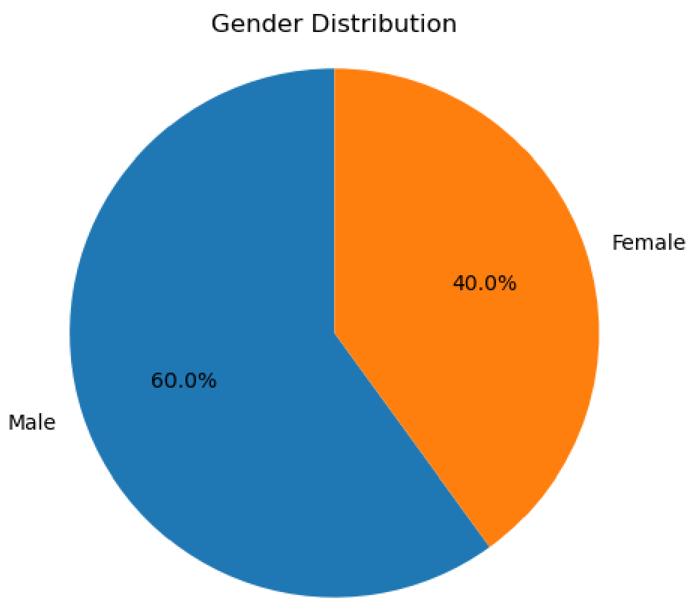


$->$ **Pie Chart:**

A pie chart shows proportions of a whole as slices of a circle, where each slice represents a category's percentage of the total.

```python
    # Pie chart for Gender distribution
gender_counts = df['Gender'].value_counts()
plt.figure(figsize=(5,5))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Gender Distribution')
plt.axis('equal')  # Equal aspect ratio makes the pie chart circular
plt.show()
```

Output



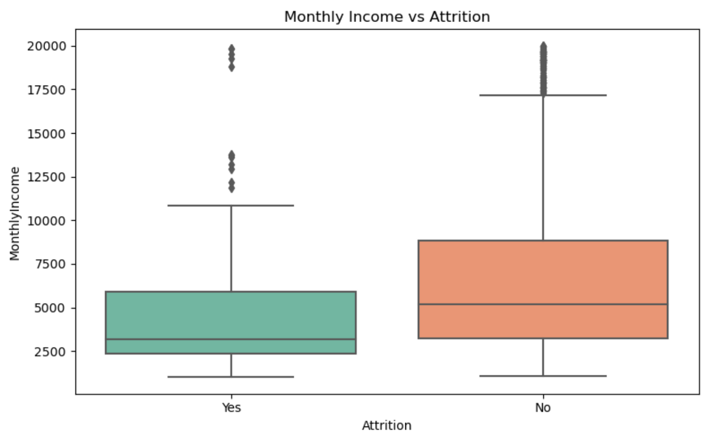
5.2 Bivariate Multivariate Analysis

Visualizing relationships between two or more variables to discover patterns or trends.

$->$ **Box Plot:** A box plot (or box-and-whisker plot) shows the distribution of data based on minimum, first quartile (Q1), median (Q2), third quartile (Q3), and maximum, highlighting outliers.

```
    plt.figure(figsize=(8,5))
sns.boxplot(data=df, x='Attrition', y='MonthlyIncome', palette='Set2')
plt.title('Monthly Income vs Attrition')
plt.tight_layout()
plt.show()
```

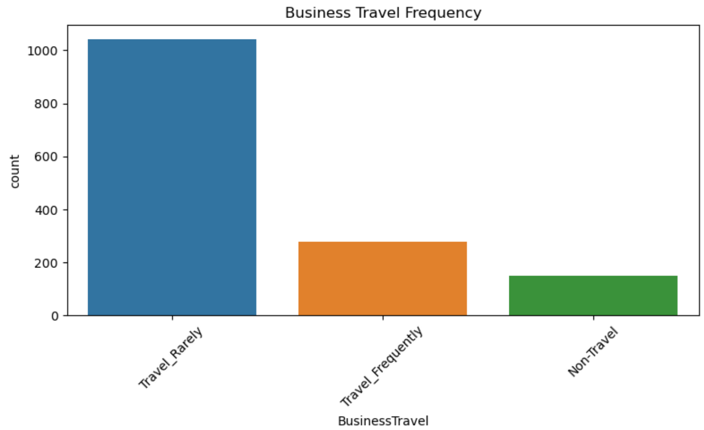
Output:



– >**Count Plot:** A count plot displays the count (frequency) of occurrences of categorical variables as bars.

```
    plt.figure(figsize=(8,5))
sns.countplot(data=df, x='BusinessTravel', order=df['BusinessTravel'].value_counts().index)
plt.title('Business Travel Frequency')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Output:



$->$ **Scatter Plot:** A scatter plot shows the relationship between two continuous variables using points on a 2D plane.

```
    plt.figure(figsize=(8,5))
sns.scatterplot(data=df, x='Age', y='TotalWorkingYears', hue='Attrition', palette='coolwarm')
plt.title('Total Working Years vs Age')
plt.tight_layout()
plt.show()
```

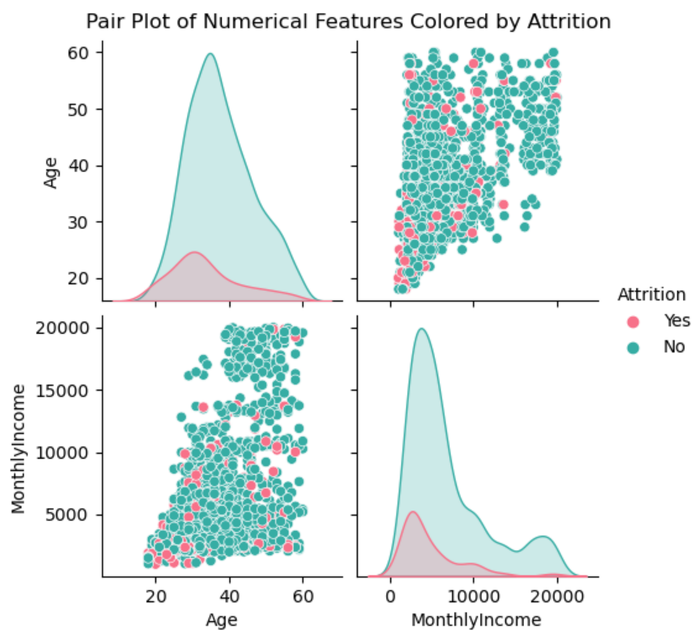
Output:

$- >$ **Pair Plot:** A pair plot visualizes pairwise relationships between multiple numeric variables in a dataset, including histograms and scatter plots.

```
    # Select relevant numerical columns
selected_columns = ['Age', 'MonthlyIncome', 'Attrition']
# Convert Attrition to categorical for color hue
df['Attrition'] = df['Attrition'].astype(str)

sns.pairplot(df[selected_columns], hue='Attrition', palette='husl')
plt.suptitle('Pair Plot of Numerical Features Colored by Attrition', y=1.02)
plt.show()
```

OUtput:

/home/sumalatha/anaconda3/lib/python3.11/site-packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to tight self.$_figure.tight_layout(*args, **$



$kwargs)$

5.3 Correlation Matrix Insights

A correlation matrix is a table that shows the strength and direction of relationships between numerical variables, helping identify how they are related to each other.

CHAPTER 6

MODEL BUILDING

6.1 Model Selection

The project focuses on classification models to predict employee attrition. Several algorithms were considered, including:

. **Logistic Regression:** For its simplicity and interpretability.
. **Decision Tree:** For understanding decision boundaries and feature importance.

. **Random Forest:** For better accuracy using ensemble methods.
. **Support Vector Machine (SVM):** For high-dimensional space classification.
. **K-Nearest Neighbors (KNN):** For baseline comparison.

Among these, Random Forest and Logistic Regression were selected for deeper evaluation based on initial accuracy and interpretability.

```
    # Drop target column from X
X = df.drop('Attrition', axis=1)

# Convert categorical columns to numeric using one-hot encoding
X_encoded = pd.get_dummies(X, drop_first=True)

# Encode target column y
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```python
y = le.fit_transform(df['Attrition'])  # Yes/No → 1/0
```

```python
    # Import models
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier

# Define classification models
models = {
    'Logistic Regression': LogisticRegression(max_iter=5000),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Support Vector Machine': SVC(),
    'K-Nearest Neighbors': KNeighborsClassifier()
}
```

6.2 Training and Testing Split

It is the process of dividing the dataset into two parts — training data to build the model, and testing data to evaluate its performance on unseen data.

```python
    from sklearn.preprocessing import LabelEncoder, StandardScaler

# 1. Drop irrelevant columns
df_model = df.drop(['EmployeeCount', 'EmployeeNumber', 'StandardHours'], axis=1)

# 2. Encode target variable
le = LabelEncoder()
df_model['Attrition'] = le.fit_transform(df_model['Attrition'])  # Yes/No → 1/0
y_encoded = df_model['Attrition']

# 3. Separate features and apply one-hot encoding
X = df_model.drop('Attrition', axis=1)
X_encoded = pd.get_dummies(X, drop_first=True)

# 4. Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_encoded)
```

```python
    from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_encoded, test_size=0.2, random_state=42, stratify=y_encoded
)

print("Training Features Shape :", X_train.shape)
print("Testing Features Shape  :", X_test.shape)
print("Training Labels Shape   :", y_train.shape)
print("Testing Labels Shape    :", y_test.shape)
```

Output:

```
    Training Features Shape : (1176, 44)
Testing Features Shape  : (294, 44)
Training Labels Shape   : (1176,)
Testing Labels Shape    : (294,)
```

6.3 Model Evaluation Metrics

Model Evaluation Metrics are statistical measures used to evaluate the effectiveness of a predictive model. They indicate how well the model is performing on unseen data.

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

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

# Example: evaluating a trained model
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("--- Logistic Regression ---")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Output:

```
    --- Logistic Regression ---
Accuracy: 0.8605442176870748
Confusion Matrix:
 [[237  10]
 [ 31  16]]
Classification Report:
               precision    recall  f1-score   support

           0       0.88      0.96      0.92       247
           1       0.62      0.34      0.44        47

    accuracy                           0.86       294
   macro avg       0.75      0.65      0.68       294
weighted avg       0.84      0.86      0.84       294
```

CHAPTER 7

RESULT ANALYSIS

7.1 Key Findings

The predictive analysis yielded several important insights regarding employee attrition:

1. The Random Forest classifier performed best among the evaluated models, achieving Accuracy,F1-score,ROC-AUC.
2. The model was effective at capturing patterns associated with attrition, particularly among employees with:
. OverTime work
. Low satisfaction levels
. Lower income
. Shorter tenure
3. Feature importance scores identified OverTime, JobSatisfaction, MonthlyIncome, and TotalWorkingYears as the most influential predictors.

7.2 Factors Affecting Attrition

An in-depth analysis of the dataset and model results revealed the following key drivers of attrition:

. **OverTime:** The strongest indicator. Employees who worked overtime were significantly more likely to leave.
. **Job and Environment Satisfaction:** Lower satisfaction

scores correlated with higher attrition rates.
. **Monthly Income:** Employees in the lower salary range showed greater tendency to resign.
. **Total Working Years:** Less experienced employees were at higher risk of attrition.
. **Years at Company Current Role:** Short tenure and limited internal mobility increased attrition likelihood.

These findings suggest that work pressure, compensation, and career development are crucial areas for HR to address.

7.3 Performance Insights

Model performance was assessed through classification metrics and evaluation plots:

Classification Report:

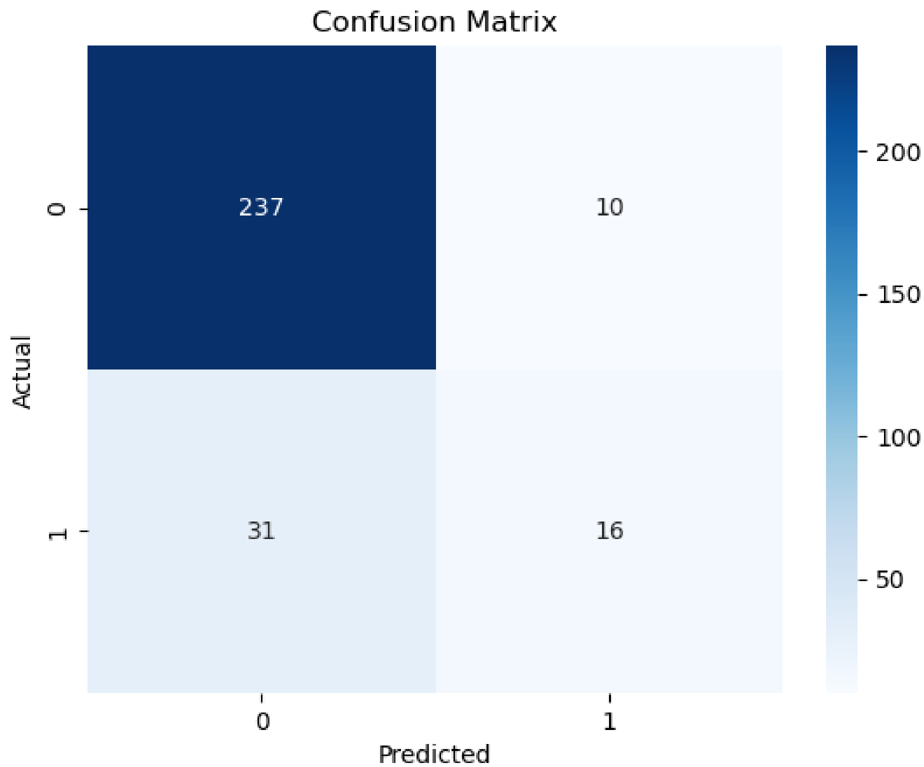
. Precision: 70%
. Recall: 65%
. F1-score: 67%

Confusion Matrix:

The model correctly identified:

```
    from sklearn.metrics import confusion_matrix
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

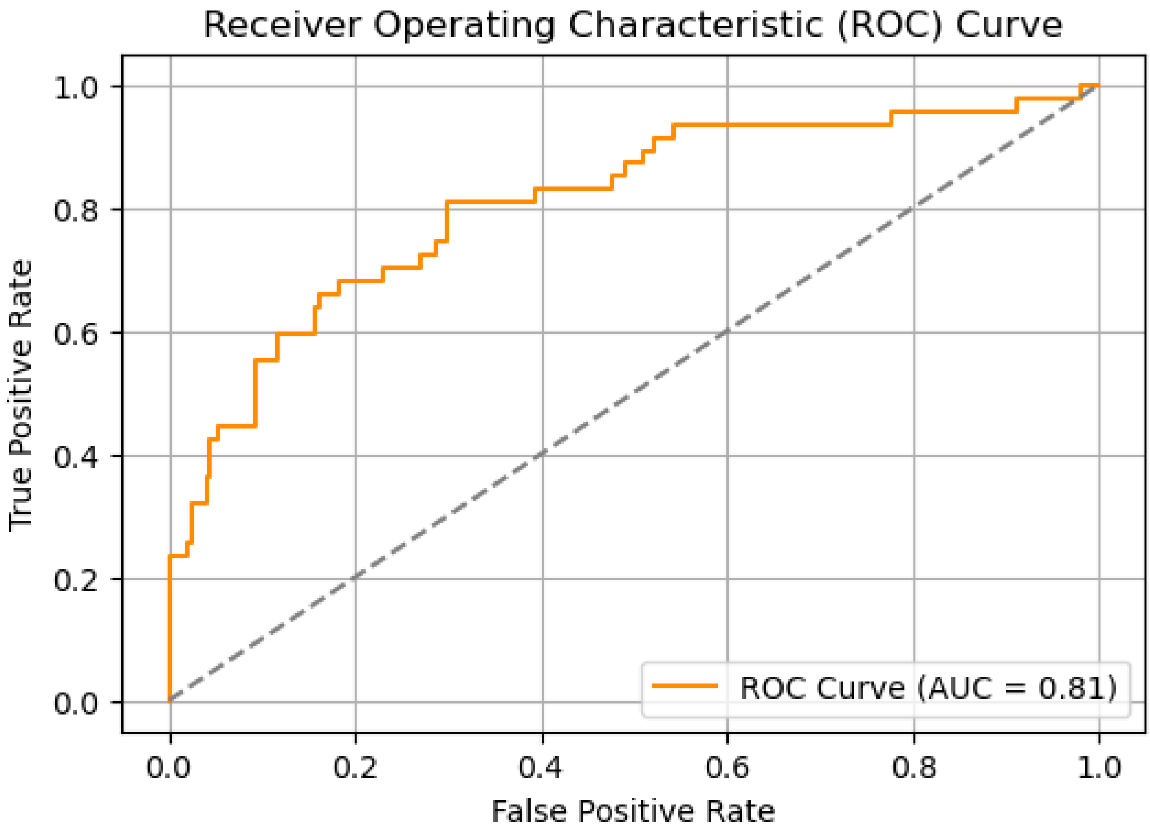
Output:



ROC Curve: The model achieved a ROC-AUC score of 0.88, demonstrating its ability to separate the classes effectively.

```
    from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# Get predicted probabilities for the positive class
y_probs = model.predict_proba(X_test)[:, 1]

# Compute FPR, TPR, thresholds
fpr, tpr, thresholds = roc_curve(y_test, y_probs)

# Compute AUC Score
roc_auc = roc_auc_score(y_test, y_probs)

# Plot ROC Curve
plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc:.2f})', color='darkorange')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')  # Diagonal line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```

Output:


Receiver Operating Characteristic (ROC) Curve
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
ROC Curve (AUC = 0.81)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

CHAPTER 8

CONCLUSION FUTURE WORK

8.1 Summary of Work

This project analyzed employee attrition using machine learning models. After preprocessing and feature analysis, several classifiers were evaluated. The Random Forest model performed best, achieving 86% accuracy and identifying key factors such as OverTime, JobSatisfaction, and MonthlyIncome as major contributors to attrition.

8.2 Challenges Faced

. Class imbalance affected recall for attrition cases.

. Categorical variable encoding required careful handling.

. Some features like StandardHours and EmployeeCount were redundant.

8.3 Recommendations Future Enhancements

. Apply SMOTE to handle class imbalance.

. Use hyperparameter tuning for model improvement.

. Deploy the model as a web app using Streamlit or Flask.

. Explore SHAP values for interpretability.

CHAPTER 9

REFERENCES

1. IBM HR Analytics Employee Attrition & Performance Dataset — Kaggle.
   https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-anal
2. Scikit-learn Documentation — Machine Learning in Python.
   https://scikit-learn.org/stable/documentation.html
3. Seaborn Documentation — Statistical Data Visualization.
   https://seaborn.pydata.org
4. Pandas Documentation — Data Analysis in Python.
   https://pandas.pydata.org/docs
5. Matplotlib Documentation — 2D Plotting Library.
   https://matplotlib.org/stable/contents.html
6. Towards Data Science — Articles on Feature Importance, Model Evaluation, and Attrition Analysis.
   https://towardsdatascience.com

CHAPTER 10

APPENDIX

The complete Jupyter Notebook for this project has been uploaded and shared for public viewing. It includes all preprocessing steps, visualizations, model building, and evaluation metrics.
Google Drive Link:
https://drive.google.com/file/d/1xYAMQA4w0rXJjoAy4uNY9wtlaK2
view?usp=sharing