

## **Predicting Employee Churn and Optimizing Workforce Productivity Using Machine Learning**

**By**

**YEONG MAN WEI**

**TP065870**

**APU3F2411CSDA**

A report submitted in partial fulfillment of the requirements for the degree of

B.Sc. (Hons) Computer Science Specialism in Data Analytics

at Asia Pacific University of Technology and Innovation.

**Supervised by Ms. Farhana Illiani Binti Hassan**

**2<sup>nd</sup> Marker: Ms. Nur Amira Binti Abdul Majid**

**2024**

**DECLARATION OF THESIS CONFIDENTIALITY**

Author's full name: **YEONG MAN WEI**

IC No./Passport No.: **030716080876**

Thesis/Project title: **PREDICTING EMPLOYEE CHURN AND OPTIMIZING WORKFORCE  
PRODUCTIVITY USING MACHINE LEARNING**

---

I declare that this thesis is classified as:

- CONFIDENTIAL
- RESTRICTED
- OPEN ACCESS

I acknowledged that Asia Pacific University of Technology & Innovation (APU) reserves the right as follows:

1. The thesis is the property of Asia Pacific University of Technology & Innovation (APU).
  2. The Library of Asia Pacific University of Technology & Innovation (APU) has the right to make copies for the purpose of research only.
  3. The Library has the right to make copies of the thesis for academic exchange.
- 

Author's Signature:



Date: 19 July 2025

Supervisor's Name: **Ms. FARHANA ILLIANI BINTI HASSAN**

Date: 19 July 2025

Signature: 

First Name: Yeong
Middle Name (only if applicable):
Last Name: Man Wei
Title of the Final Year Project / Dissertation / Thesis: Predicting Employee Churn and Optimizing Workforce Productivity Using Machine Learning
<p><b>Abstract:</b> Employee retention and workforce productivity are critical factors in determining an organization's success since a huge employee turnover lead to high recruitment costs, loss of institutional knowledge and disruption of operations. Organizations continue to struggle in identifying the factors behind the employee turnover and the elements that affect productivity levels in their workforce. This research uses CRISP-DM methodology and machine-learning techniques for analyzing employee churn and productivity. The study aims to identify key factors influencing employee attrition and productivity within the organization. The project involves data collection, preprocessing, feature engineering and model development using advanced ML algorithms such as random forest and regression. Models are evaluated based on their accuracy in predicting employee churn and analyzing productivity trends. Additionally, dashboard visualization is created for HR professionals for providing insights and actionable recommendations to improve retention strategies and workforce management. The project supports Sustainable Development Goal (SDG) 8: Decent Work and Economic Growth to maintain stable employment, enhance working environments and sustainable workforce management. Through machine learning techniques, organizations can proactively solve attrition risks, foster employee well-being which helps in contributing to economic growth and organizational resilience.</p>

A few keywords associated with the work: Employee Churn Prediction, Workforce Productivity, Machine Learning, HR Analytics, Predictive Modeling, Data Mining

General Subject: Computer Science in Data Analytics

Date of Submission: 23<sup>rd</sup> July 2025

## Acknowledgement

I would like to thank my Final Year Project (FYP) supervisor, Ms. Farhana Illiani Binti Hassan, for her ongoing guidance and encouragement throughout this project. Her feedback and recommendations have been suggestions have been very beneficial in helping me enhance my work and remain focused.

I appreciate her response immediately on Microsoft Teams where she always replied to my queries and gave clear explanations. Regardless of the issue being small or complicated, she was always ready to assist after finishing busy with work. During our physical meetings, she has shared valuable insights and ideas that helped to guide the direction of my project. She also took the time to identify and correct errors to ensure that my work was accurate and aligned with the project objectives to achieve.

Her patience and dedication had a significant influence throughout this process. She not only helped me solve the technical challenges but also provided guidance on how to structure my project effectively. She encouraged me to think critically, explore different approaches and improve my problem-solving skills. She also has provided some new idea for me. I truly appreciate her willingness to share her knowledge and guide me at every phase of the project. Her support and encouragement gave me the confidence to complete my project successfully, and I am grateful to have had the opportunity to work under her guidance.

I would also like to extend my sincere thanks to my second marker, Ms. Nur Amira Abdul Majid for her guidance and insightful recommendations regarding my FYP project particularly on my system and documentation. Her feedback has been instrumental in enhancing my project and I am truly grateful for her support.

## Abstract

Employee retention and workforce productivity are critical factors in determining an organization's success since a huge employee turnover lead to high recruitment costs, loss of institutional knowledge and disruption of operations. Organizations continue to struggle in identifying the factors behind the employee turnover and the elements that affect productivity levels in their workforce. This research uses CRISP-DM methodology and machine-learning techniques for analyzing employee churn and productivity. The study aims to identify key factors influencing employee attrition and productivity within the organization.

The project involves data collection, preprocessing, feature engineering and model development using advanced ML algorithms such as random forest and regression. Models are evaluated based on their accuracy in predicting employee churn and analyzing productivity trends. Additionally, dashboard visualization is created for HR professionals for providing insights and actionable recommendations to improve retention strategies and workforce management.

The project supports Sustainable Development Goal (SDG) 8: Decent Work and Economic Growth to maintain stable employment, enhance working environments and sustainable workforce management. Through machine learning techniques, organizations can proactively solve attrition risks, foster employee well-being which helps in contributing to economic growth and organizational resilience.

Keywords: Employee Churn Prediction, Workforce Productivity, Machine Learning, HR Analytics, Predictive Modeling, Data Mining

SDG Goal 8: Decent Work and Economic Growth

## Table of Contents

<b>Acknowledgement</b> .....	5
<b>Abstract</b> .....	6
<b>List of Figure</b> .....	15
<b>List of Tables</b> .....	25
<b>Chapter 1: Introduction</b> .....	26
<b>1.1 Introduction</b> .....	26
<b>1.2 Problem Background</b> .....	27
<b>1.2.1 Employee Churn Prediction Challenges</b> .....	28
<b>1.2.2 Impact of Employee Productivity</b> .....	28
<b>1.2.3 Lack of a Comprehensive Framework for Both Churn and Productivity</b> .....	29
<b>1.2.4 Challenges in Data Quality and Ethical Considerations</b> .....	30
<b>1.3 Project Aim</b> .....	30
<b>1.4 Objectives</b> .....	30
<b>1.5 Scope</b> .....	31
<b>1.5.1 Dataset</b> .....	31
<b>1.5.2 Deliverables</b> .....	31
<b>1.5.3 Constraint &amp; Project Boundaries</b> .....	33
<b>1.6 Potential Benefit</b> .....	34
<b>1.6.1 Tangible Benefit</b> .....	34
<b>1.6.2 Intangible Benefit</b> .....	35
<b>1.6.3 Target User</b> .....	35
<b>1.7 Overview of IR</b> .....	36
<b>1.8 Project Plan</b> .....	38
<b>1.8.1 Project Plan Sem 1</b> .....	38
<b>1.8.2 Project Plan Sem 2</b> .....	40
<b>Chapter 2: Literature Review</b> .....	42
<b>2.1 Introduction</b> .....	42
<b>2.2 Domain Research</b> .....	43
<b>2.2.1 Factors Affecting Employee Churn and Performance</b> .....	43
<b>2.2.1.1 Job Satisfaction and Career Development</b> .....	43
<b>2.2.1.2 Workplace Stress and Leadership Influence</b> .....	43

<b>2.2.1.3 Compensation, Benefits and Demographics .....</b>	44
<b>2.2.1.4 Theoretical Models Explaining Employee Churn and Performance .....</b>	44
<b>2.2.2 Impact of Employee Churn on Organizations.....</b>	44
<b>2.2.2.1 Financial and Operational Consequences.....</b>	45
<b>2.2.2.2 Cultural and Strategic Implications .....</b>	45
<b>2.2.3 Employee Retention Strategies to Sustain Workforce Productivity .....</b>	46
<b>2.2.3.1 Building a Positive Organizational Culture.....</b>	46
<b>2.2.3.2 Investing in Employee Growth and Development.....</b>	47
<b>2.2.3.3 Promoting Work-Life Balance and Employee Well-Being .....</b>	48
<b>2.2.3.4 Ensuring Long-Term Business Success Through Retention .....</b>	48
<b>2.2.4 Machine Learning Models for Employee Churn Prediction.....</b>	49
<b>2.2.4.1 Decision Tree .....</b>	49
<b>2.2.4.2 Random Forest .....</b>	50
<b>2.2.4.3 Gradient Boosting .....</b>	50
<b>2.2.4.4 Support Vector Machine .....</b>	50
<b>2.2.4.5 XGBoost .....</b>	51
<b>2.2.4.6 Deep Learning .....</b>	51
<b>2.3 Similar Search/Works .....</b>	52
<b>2.4 Technical Research .....</b>	57
<b>2.4.1 Hardware/Software for the Project .....</b>	57
<b>2.4.2 Programming Language Chosen: Python.....</b>	58
<b>2.4.3 IDE (Interactive Development Environment) chosen.....</b>	58
<b>2.4.4 Libraries/Tools Chosen.....</b>	59
<b>2.4.5 Operating System chosen .....</b>	61
<b>2.5 Summary.....</b>	61
<b>Chapter 3: Methodology.....</b>	63
<b>3.1 Introduction.....</b>	63
<b>3.2 System Development Methodology.....</b>	64
<b>3.2.1 Introduction of CRISP-DM methodology.....</b>	64
<b>3.2.2 Methodology choice and justification.....</b>	65
<b>3.2.3 Phases of CRISP-DM .....</b>	65
<b>3.3 Summary.....</b>	68

<b>Chapter 4: Design and Implementation.....</b>	69
<b>4.1 Introduction.....</b>	69
<b>4.2 Data Collection .....</b>	69
<b>4.3 Initial Data Understanding .....</b>	70
<b>4.3.1 Import Dataset .....</b>	70
<b>4.3.2 View Total Rows and Columns.....</b>	71
<b>4.3.3 View Data Types.....</b>	71
<b>4.3.4 View Column Variables.....</b>	72
<b>4.3.5 Dataset Overview .....</b>	73
<b>4.3.6 Data Dictionary .....</b>	73
<b>4.3.7 Descriptive Statistics of Employee Data.....</b>	75
<b>4.3.8 Graphical View of Variables before Data Preprocessing.....</b>	76
<b>4.3.8.1 Categorical Variables .....</b>	76
<b>4.3.8.2 Numerical Variables .....</b>	83
<b>4.3.9 Target Variable - Resigned .....</b>	95
<b>4.3.10 Correlation Heatmap of Employee Variables before Data Preprocessing .....</b>	96
<b>4.4 Data Preprocessing .....</b>	97
<b>4.4.1 Missing Values.....</b>	97
<b>4.4.2 Outliers.....</b>	98
<b>4.4.2.1 Categorical Variables .....</b>	99
<b>4.4.2.2 Numerical Variables .....</b>	106
<b>4.4.3 Data Transformation .....</b>	115
<b>4.4.4 Feature Engineering .....</b>	115
<b>4.4.4.1 Creating New Variables.....</b>	115
<b>4.4.4.2 Categorization of Continuous Variables .....</b>	118
<b>4.4.4.3 Initial Drop Columns .....</b>	120
<b>4.4.4.4 Encoding Categorical Variables .....</b>	121
<b>4.5 Data Understanding.....</b>	124
<b>4.5.1 Graphical View of Variables After Data Preprocessing.....</b>	124
<b>4.5.1.1 New Variables that are created in Feature Engineering section.....</b>	124
<b>4.5.1.2 Variables that are Encoded in Encoding section.....</b>	126
<b>4.5.2 Feature Selection .....</b>	131

<b>4.5.2.1 Numerical Variables .....</b>	132
<b>4.5.2.2 Categorical Variables .....</b>	135
<b>4.5.2.3 Top 10 Features.....</b>	137
<b>4.5.2.4 Implications for Future Analysis.....</b>	138
<b>4.5.3 Data Splitting and Resampling .....</b>	139
<b>4.5.3.1 Check Target Variable Count.....</b>	139
<b>4.5.3.2 Data Splitting (Train-Test Split).....</b>	140
<b>4.5.3.3 Applying SMOTE Techniques on Training Data .....</b>	141
<b>4.5.3.4 Output after applying SMOTE Techniques .....</b>	142
<b>4.6 Model Building.....</b>	143
<b>4.6.1 Random Forest .....</b>	144
<b>4.6.1.1 Base Model (Default Parameters).....</b>	144
<b>4.6.1.2 Tuned Model (Randomized Search CV) .....</b>	145
<b>4.6.1.3 Best Parameters and Accuracy .....</b>	146
<b>4.6.2 XGBoost.....</b>	147
<b>4.6.2.1 Base Model (Default Parameters).....</b>	147
<b>4.6.2.2 Tuned Model (Randomized Search CV) .....</b>	148
<b>4.6.2.3 Best Parameters and Accuracy .....</b>	150
<b>4.6.3 Gradient Boosting .....</b>	151
<b>4.6.3.1 Base Model (Default Parameters).....</b>	151
<b>4.6.3.2 Tuned Model (Randomized Search CV) .....</b>	152
<b>4.6.3.3 Best Parameters and Accuracy .....</b>	154
<b>4.6.4 Support Vector Machine .....</b>	155
<b>4.6.4.1 Base Model (Default Parameters).....</b>	156
<b>4.6.4.2 Tuned Model (Randomized Search CV) .....</b>	157
<b>4.6.4.3 Best Parameters and Accuracy .....</b>	159
<b>4.7 Summary.....</b>	159
<b>Chapter 5: Result and Discussion.....</b>	161
<b>    5.1 Introduction.....</b>	161
<b>    5.2 Model Evaluation and Discussions .....</b>	162
<b>5.2.1 Random Forest Base Model .....</b>	162
<b>5.2.1.1 Confusion Matrix and Classification Report.....</b>	162

<b>5.2.1.2 Learning Curve .....</b>	164
<b>5.2.1.3 ROC AUC Score.....</b>	166
<b>5.2.2 Random Forest Tuned Model .....</b>	168
<b>5.2.2.1 Confusion Matrix and Classification Report.....</b>	168
<b>    5.2.2.1.1 Comparison Results of Base and Tuned Models .....</b>	170
<b>    5.2.2.2 Learning Curve .....</b>	171
<b>    5.2.2.3 ROC and AUC.....</b>	173
<b>5.2.3 XGBoost Base Model .....</b>	175
<b>    5.2.3.1 Confusion Matrix and Classification Report.....</b>	175
<b>    5.2.3.2 Learning Curve .....</b>	177
<b>    5.2.3.3 ROC and AUC.....</b>	179
<b>5.2.4 XGBoost Tuned Model .....</b>	181
<b>    5.2.4.1 Confusion Matrix and Classification Report.....</b>	181
<b>        5.2.2.1.2 Comparison Results of Base and Tuned Models .....</b>	183
<b>        5.2.4.2 Learning Curve .....</b>	184
<b>        5.2.4.3 ROC and AUC.....</b>	186
<b>5.2.5 Gradient Boosting Base Model .....</b>	188
<b>    5.2.5.1 Confusion Matrix and Classification Report.....</b>	188
<b>    5.2.5.2 Learning Curve .....</b>	190
<b>    5.2.5.3 ROC and AUC.....</b>	192
<b>5.2.6 Gradient Boosting Tuned Model .....</b>	194
<b>    5.2.6.1 Confusion Matrix and Classification Report.....</b>	194
<b>        5.2.2.1.3 Comparison Results of Base and Tuned Models .....</b>	196
<b>        5.2.6.2 Learning Curve .....</b>	197
<b>        5.2.6.3 ROC and AUC.....</b>	199
<b>5.2.7 Support Vector Machine Base Model .....</b>	201
<b>    5.2.7.1 Confusion Matrix and Classification Report.....</b>	201
<b>    5.2.7.2 Learning Curve .....</b>	203
<b>    5.2.7.3 ROC and AUC.....</b>	205
<b>5.2.8 Support Vector Machine Tuned Model .....</b>	207
<b>    5.2.8.1 Confusion Matrix and Classification Report.....</b>	207
<b>        5.2.2.1.4 Comparison Results of Base and Tuned Models .....</b>	209

<b>5.2.8.2 Learning Curve .....</b>	210
<b>5.2.8.3 ROC and AUC.....</b>	212
<b>5.2.9 Comparison and Discussion of All Models .....</b>	214
<b>    5.2.9.1 Analysis of Base Models for Employee Churn Prediction.....</b>	214
<b>    5.2.9.2 Analysis of Tuned Models for Employee Churn Prediction .....</b>	216
<b>    5.2.9.3 Justification for Selecting the Best Model.....</b>	218
<b>5.3 Feature Importance .....</b>	221
<b>5.4 Model Deployment.....</b>	225
<b>    5.4.1 Generating Data for Dynamic Updates in Model Deployment .....</b>	225
<b>    5.4.2 Overview of The Employee Churn Analytics Dashboard System .....</b>	230
<b>    5.4.3 System Functions.....</b>	232
<b>        5.4.3.1 Functions Available for Both HR and Talent Acquisition Role.....</b>	233
<b>            5.4.3.1.1 Employee Churn Analytics System Calculation Functions.....</b>	233
<b>            5.4.3.1.2 Employee Metrics Functions.....</b>	235
<b>                5.4.3.1.2.1 Common Metrics for HR and Talent .....</b>	235
<b>                5.4.3.1.2.2 HR Role-Specific Metrics .....</b>	239
<b>            5.4.3.1.3 Login Page .....</b>	240
<b>            5.4.3.1.4 Navigation Bar .....</b>	242
<b>            5.4.3.1.5 Employee Churn Prediction Page .....</b>	244
<b>                5.4.3.1.5.1 Low Risk of Churn.....</b>	249
<b>                5.4.3.1.5.2 Moderate Risk of Churn .....</b>	251
<b>                5.4.3.1.5.3 High Risk of Churn.....</b>	253
<b>            5.4.3.2 HR Specific Functions .....</b>	255
<b>                5.4.3.2.1 HR Main Dashboard Page .....</b>	255
<b>                    5.4.3.2.1.1 View Employee Details .....</b>	257
<b>                    5.4.3.2.1.2 View Satisfaction Trends .....</b>	259
<b>                    5.4.3.2.1.3 View Employee Churn Details .....</b>	260
<b>                    5.4.3.2.1.4 Quick Actions .....</b>	261
<b>                5.4.3.2.2 Recent Activity .....</b>	262
<b>                5.4.3.2.3 Employee Performance Page .....</b>	266

<b>5.4.3.2.3.1 Employee Performance Overview .....</b>	267
<b>5.4.3.2.3.2 Quarterly Trends.....</b>	271
<b>5.4.3.2.3.3 Overtime Analysis .....</b>	273
<b>5.4.3.2.3.4 Sick Day Analysis .....</b>	275
<b>5.4.3.2.3.5 Download Summary Report .....</b>	276
<b>5.4.3.2.4 Actionable Insights for Retention and Productivity Page .....</b>	280
<b>    5.4.3.2.4.1 Compensation Insights .....</b>	284
<b>    5.4.3.2.4.2 Productivity Insights.....</b>	285
<b>    5.4.3.2.4.3 Retention Insights .....</b>	287
<b>    5.4.3.2.4.4 Engagement Insights.....</b>	288
<b>5.4.3.3 Talent Acquisition-Specific Functions.....</b>	290
<b>    5.4.3.3.1 Talent Dashboard Page.....</b>	290
<b>    5.4.3.3.2 Hiring Recommendations Page .....</b>	295
<b>        5.4.3.3.2.1 Education and Experience Distribution of Top Performers .....</b>	296
<b>        5.4.3.3.2.2 Priority Hiring Recommendations .....</b>	298
<b>5.4.4 Conclusion of Model Deployment.....</b>	302
<b>5.5 Summary.....</b>	303
<b>Chapter 6: Conclusion .....</b>	304
<b>    6.1 Critical Evaluation.....</b>	304
<b>        6.1.1 The achievement of the overall of the project .....</b>	304
<b>        6.1.2 Contribution of the project towards community/ industries .....</b>	306
<b>        6.1.3 Strength of the Project.....</b>	307
<b>    6.2 Limitations of the Project.....</b>	309
<b>    6.3 Suggestion for Future Recommendations.....</b>	310
<b>References .....</b>	312
<b>Appendices.....</b>	322
<b>    Appendix A – Project Plan Form .....</b>	322
<b>    Appendix B – Ethics Forms (Fast Track) .....</b>	323
<b>    Appendix C – Log Sheets .....</b>	327
<b>    Appendix D – Poster .....</b>	333
<b>    Appendix E – Gantt Chart.....</b>	334

<b>Appendix F – Source Codes .....</b>	<b>336</b>
<b>Appendix G – Turnitin Similarity Report .....</b>	<b>394</b>

**List of Figure**

Figure 1:Levels of Employee Retention Strategies .....	47
Figure 2:Employee churn flow analysis .....	49
Figure 3:CRISP-DM Methodology Cycle .....	64
Figure 4:Source Code of Import Dataset .....	70
Figure 5: Dataset Display.....	70
Figure 6: Dataset Display.....	71
Figure 7:Dataset Display.....	71
Figure 8: Source Code of View Total Rows and Columns.....	71
Figure 9: Source Code of View Data Types .....	71
Figure 10: Dataset Structure .....	72
Figure 11:Source Code of View Column.....	72
Figure 12:Display Columns Output.....	72
Figure 13: Source Code of Descriptive Statistics .....	75
Figure 14: Descriptive Statistics Table .....	75
Figure 15:Source Code of Categorical Variables by Using Bar Chart .....	76
Figure 16:Bar Chart for Department.....	77
Figure 17: Bar Chart for Gender .....	78
Figure 18: Bar Chart for Job_Title.....	79
Figure 19:Bar Chart for Education_Level .....	80
Figure 20:Bar Chart for Performance_Score .....	81
Figure 21:Bar Chart for Promotions .....	82
Figure 22:Source Code of Numerical Variables by using Bar Chart.....	83
Figure 23:Histogram for Age.....	84
Figure 24: Histogram for Years_At_Company.....	85
Figure 25:Histogram for Monthly_Salary .....	86
Figure 26:Histogram for Work_Hour_Per_Week .....	87
Figure 27:Histogram for Project_Handled.....	88
Figure 28:Histogram for Overtime_Hours.....	89
Figure 29:Histogram for Sick_Days .....	90
Figure 30: Histogram of Remote_Work_Frequency .....	91

Figure 31:Histogram for Team_Size .....	92
Figure 32:Histogram for Training_Hours .....	93
Figure 33:Histogram for Employee_Satisfaction_Score .....	94
Figure 34:Source Code of Target Variable .....	95
Figure 35: Pie Chart of Target Variable .....	95
Figure 36: Source Code of Correlation Heatmap .....	96
Figure 37: Correlation Heatmap .....	96
Figure 38: Source Code of Duplicate Rows .....	97
Figure 39: Count of Duplicate Rows .....	97
Figure 40:Source Code of Missing Values .....	97
Figure 41: Output of Missing Values.....	98
Figure 42: Source Code of Categorical Variables by using Bar Chart .....	99
Figure 43:Output of Outliers for Categorical Variables .....	99
Figure 44: Bar Chart for Department.....	100
Figure 45: Bar Chart for Gender .....	101
Figure 46:Bar Chart for Job_Title.....	102
Figure 47: Bar Chart for Education_Level .....	103
Figure 48: Bar Chart for Performance Score .....	104
Figure 49: Bar Chart for Promotions .....	105
Figure 50: Source Code of Numerical Variables by using Bar Chart.....	106
Figure 51: Output of Outliers for Numerical Variables.....	106
Figure 52: Source Code of Data Transformation.....	115
Figure 53: Output of Data Transformation .....	115
Figure 54:Source Code of Creating Tenure_Years.....	116
Figure 55: Output of Tenure_Years.....	117
Figure 56: Source Code of Dropping Columns .....	117
Figure 57: Source Code of Creating Health_Index.....	118
Figure 58: Source Code of Categorization of Continuous Variables .....	118
Figure 59: Source Code of Display Tenure_Years and Health_Index .....	119
Figure 60: Output of Tenure_Years and Health_Index .....	120
Figure 61: Source Code of Initial Drop Columns.....	120

Figure 62:Output of Tenure_Years and Health_Index .....	121
Figure 63:Source Code of One-Hot Encoding.....	121
Figure 64: Output of One-Hot Encoding .....	121
Figure 65:Output of One-Hot Encoding .....	121
Figure 66:Source Code of Label Encoding.....	122
Figure 67: Output of Label Encoding .....	122
Figure 68:Source Code of Converting to Ordinal Categories.....	123
Figure 69:Source Code of Encoded Data Types.....	123
Figure 70: Output of Encoded Data Types .....	124
Figure 71:Source Code of Summary Statistics .....	124
Figure 72: Source Code of Graphical Statistics for Health_Index and Tenure_Years .....	124
Figure 73:Summary Statistics of Health_Index and Tenure_Years .....	125
Figure 74: Graphical Statistics of Health_Index and Tenure_Years .....	125
Figure 75:Source Code of Graphical Statistics for Remote_Work_Levels .....	126
Figure 76: Bar Chart for Remote_Work_Levels .....	127
Figure 77: Source Code Graphical Statistics for Education_Levels.....	128
Figure 78:Bar Chart for Education_Level .....	128
Figure 79: Source Code of Graphical Statistics for Department .....	129
Figure 80: Bar Chart for Department.....	129
Figure 81:Source Code of Graphical Statistics for Job_Title .....	130
Figure 82:Bar Chart for Job_Title.....	130
Figure 83: Source Code of Feature Selection for Numerical Variables .....	132
Figure 84: Output of Mutual Information Score.....	133
Figure 85: Horizontal Bar Chart of Mutual Information Scores for Numerical Variables.....	133
Figure 86: Top 5 Variables of Mutual Information Scores for Numerical Variables.....	134
Figure 87:Source Code of Feature Selection for Categorical Variables.....	135
Figure 88: Mutual Information Scores for Categorical Variables .....	135
Figure 89: Horizontal Bar Chart of Mutual Information Scores for Categorical Variables .....	136
Figure 90:Top 5 Variables of Mutual Information Scores for Categorical Variables .....	136
Figure 91: Top 10 Features .....	137
Figure 92: Top 5 for Numerical and Categorical Features .....	138

Figure 93: Output of Combinations of Top 10 Features.....	138
Figure 94: Source Code of Checking Numbers of Employees Resigned .....	139
Figure 95: Source Code of Data Splitting.....	140
Figure 96: Output after Data Splitting .....	140
Figure 97: Source Code of Smote Techniques .....	141
Figure 98:Source Code of Smote Techniques .....	141
Figure 99: Output After Smote Techniques.....	142
Figure 100: Bar Chart for Resigned Before and After Class Imbalance .....	142
Figure 101: Source Code of Random Forest Default Tuning .....	144
Figure 102: Source Code of Random Forest Hyperparameter Tuning .....	145
Figure 103: Source Code of Random Forest Hyperparameter Tuning .....	145
Figure 104: Output of Best Parameters for Random Forest .....	146
Figure 105: Output of Random Forest's Accuracy Score .....	147
Figure 106: Source Code of XGBoost Default Tuning .....	147
Figure 107: Source Code of XGBoost Hyperparameter Tuning .....	148
Figure 108: Source Code of XGBoost Hyperparameter Tuning .....	149
Figure 109: Output of Best Parameters for XGBoost.....	150
Figure 110: Output of XGBoost's Accuracy Score.....	150
Figure 111: Source Code of Gradient Boosting Default Tuning .....	151
Figure 112: Source Code of Gradient Boosting Hyperparameter Tuning .....	152
Figure 113: Source Code of Gradient Boosting Hyperparameter Tuning .....	152
Figure 114: Output of Best Parameters for Gradient Boosting.....	154
Figure 115: Output of Gradient Boost's Accuracy Score .....	154
Figure 116: Source Code of Support Vector Machine Default Tuning .....	156
Figure 117: Source Code of Support Vector Machine Hyperparameter Tuning .....	157
Figure 118: Source Code of Support Vector Machine Hyperparameter Tuning .....	157
Figure 119: Output of Best Parameters for Support Vector Machine .....	159
Figure 120: Output of Support Vector Machine's Accuracy Score .....	159
Figure 121: Confusion Matrix for Random Forest Base Model.....	162
Figure 122: Classification Report for Random Forest Base Model.....	163
Figure 123: Source Code for Learning Curve of Random Forest Base Model .....	164

Figure 124: Learning Curve of Random Forest Base Model.....	165
Figure 125: Source Code for ROC AUC Score of Random Forest Base Model .....	166
Figure 126: ROC AUC Score of Random Forest Base Model .....	167
Figure 127: Confusion Matrix for Random Forest Tuned Model .....	168
Figure 128: Classification Report for Random Forest Tuned Model .....	169
Figure 129: Source Code for Learning Curve of Random Forest Tune Model .....	171
Figure 130: Learning Curve of Random Forest Tuned Model .....	172
Figure 131: Source Code for ROC AUC Score of Random Forest Tuned Model .....	173
Figure 132: ROC AUC Score of Random Forest Tuned Model.....	174
Figure 133: Confusion Matrix for XGBoost Base Model .....	175
Figure 134: Classification Report for XGBoost Base Model .....	176
Figure 135: Source Code for Learning Curve of XGBoost Base Model\.....	177
Figure 136: Learning Curve of XGBoost Base Model .....	178
Figure 137: Source Code for ROC AUC Score of XGBoost Base Model .....	179
Figure 138: ROC AUC Score of XGBoost Base Model.....	180
<i>Figure 139: Confusion Matrix for XGBoost Tuned Model.....</i>	181
Figure 140: Classification Report for XGBoost Tuned Model.....	182
Figure 141: Source Code for Learning Curve of XGBoost Tuned Model .....	184
Figure 142: Learning Curve of XGBoost Tuned Model.....	185
Figure 143: Source Code for ROC AUC Score of XGBoost Tuned Model .....	186
Figure 144: ROC AUC Score of XGBoost Tuned Model .....	187
Figure 145: Confusion Matrix for Gradient Boosting Base Model .....	188
Figure 146: Classification Report for Gradient Boosting Base Model.....	189
Figure 147: Source Code for Learning Curve of Gradient Boosting Base Model.....	190
Figure 148: Learning Curve of Gradient Boosting Base Model.....	191
Figure 149: Source Code for ROC AUC Score of Gradient Boosting Base Model .....	192
Figure 150: ROC AUC Score of Gradient Boosting Base Model .....	193
Figure 151: Confusion Matrix for Gradient Boosting Tuned Model.....	194
Figure 152: Classification Report for Gradient Boosting Tuned Model .....	195
Figure 153: Source Code for Learning Curve of Gradient Boosting Tuned Model .....	197
Figure 154: Learning Curve of Gradient Boosting Tuned Model .....	198

Figure 155: Source Code for ROC AUC Score of Gradient Boosting Tuned Model.....	199
Figure 156: ROC AUC Score of Gradient Boosting Tuned Model.....	200
Figure 157: Confusion Matrix for Support Vector Machine Base Model.....	201
Figure 158: Classification Report for Support Vector Machine Base Model.....	202
<i>Figure 159: Source Code for Learning Curve of Support Vector Machine Base Model .....</i>	203
<i>Figure 160: Learning Curve of Support Vector Machine Base Model .....</i>	204
Figure 161: Source Code for ROC AUC Score of Support Vector Machine Base Model.....	205
Figure 162: ROC AUC Score of Support Vector Machine Base Model .....	206
Figure 163: Confusion Matrix for Support Vector Machine Tuned Model.....	207
Figure 164: Classification Report for Support Vector Machine Tuned Model .....	208
Figure 165: Source Code for Learning Curve of Support Vector Machine Tuned Model .....	210
Figure 166: Learning Curve of Support Vector Machine Tuned Model .....	211
Figure 167: Source Code for ROC AUC Score of Support Vector Machine Tuned Model .....	212
Figure 168: ROC AUC Score of Support Vector Machine Tuned Model.....	213
Figure 169: Source Code of Feature Importance for XGBoost .....	221
Figure 170: Top 10 Importance Features (XGBoost) .....	221
Figure 171: Horizontal Bar Chart of Top 10 Importance Features (XGBoost).....	222
Figure 172: Source Code of Saving the Trained XGBoost Model .....	225
Figure 173: Source Code for Generating New Data for Dynamic Updates .....	226
Figure 174: Source Code for Generating New Data for Dynamic Updates .....	226
Figure 175: Source Code for Generating New Data for Dynamic Updates .....	227
Figure 176: Output of Generating New Data.....	228
Figure 177: Source Code for Printing the New Dataset .....	228
Figure 178: Output of Resignation_Date.....	229
Figure 179: Output for Printing Out of Existing Columns .....	229
Figure 180: Source Code of Employee Churn Analytics Dashboard System 1 .....	230
Figure 181: Interactive Toolbar for Visualizations.....	231
Figure 182: Source Code of Employee Churn Analytics Dashboard System 1 .....	232
Figure 183: Source Code of Employee Churn Analytics Dashboard System 2 .....	232
Figure 184: Source Code of Employee Churn Analytics System Calculation Functions 1.....	233
Figure 185: Source Code of Employee Churn Analytics System Calculation Functions 2.....	234

Figure 186: Source Code of Employee Churn Analytics System Calculation Functions 3.....	234
Figure 187: Source Code of Display Common Metrics for HR and Talent .....	235
Figure 188: Source Code of Display Employee Details for HR and Talent .....	236
Figure 189: Source Code of Display Churn Details for HR and Talent 1 .....	237
Figure 190: Source Code of Display Churn Details for HR and Talent 2 .....	238
Figure 191: Source Code of Display Satisfaction Trend Metrics for HR 1 .....	239
Figure 192: Source Code of Display Satisfaction Trend Metrics for HR 2 .....	239
Figure 193: Source Code of Login Page.....	240
Figure 194: Interface of Login Page for HR .....	241
Figure 195: Interface of Login Page for Talent .....	241
Figure 196: Interface of Login Page for Invalid Username or Password .....	241
Figure 197: Source Code of Navigation Bar.....	242
Figure 198: Interface of Navigation Bar for HR and Talent.....	243
Figure 199: Source Code of Employee Churn Prediction 1 .....	244
Figure 200: Source Code of Employee Churn Prediction 2 .....	244
Figure 201: Source Code of Employee Churn Prediction 3 .....	245
Figure 202: Source Code of Employee Churn Prediction 4 .....	245
Figure 203: Source Code of Employee Churn Prediction 5 .....	246
Figure 204: Source Code of Employee Churn Prediction 7 .....	246
Figure 205: Interface for Employee Churn Prediction 1 .....	248
Figure 206: Interface for Employee Churn Prediction 2 .....	248
Figure 207: Interface for Predicting Low Risk of Churn.....	249
Figure 208: Interface for Prediction Result of Predicting Low Risk of Churn.....	250
Figure 209: Interface for Predicting Moderate Risk of Churn.....	251
Figure 210: Interface for Prediction Result of Predicting Moderate Risk of Churn.....	252
Figure 211: Interface for Predicting High Risk of Churn .....	253
Figure 212: Interface for Prediction Result of Predicting High Risk of Churn .....	254
Figure 213: Source Code of HR Main Dashboard Page 1 .....	255
Figure 214: Interface of HR Analytics Dashboard .....	256
Figure 215: Interface of View Employee Details 1 .....	257
Figure 216: Interface of View Employee Details 2 .....	258

Figure 217: Interface of View Satisfaction Trend .....	259
Figure 218: Interface of View Employee Churn Details 1 .....	260
Figure 219: Interface of View Employee Churn Details 2 .....	260
Figure 220: Source Code of HR Main Dashboard Page 2 .....	261
Figure 221: Interface for Quick Actions .....	261
Figure 222: Source Code of Recent Activity 1 .....	262
Figure 223: Source Code of Recent Activity 2 .....	263
Figure 224: Interface for Overview of Recent Activity .....	264
Figure 225: Interface for Recent Activity Advanced Functions 1 .....	264
Figure 226: Interface for Recent Activity Advanced Functions 2 .....	265
Figure 227: Source Code of Employee Performance Page 1 .....	266
Figure 228: Source Code of Employee Performance Page 2 .....	266
Figure 229: Source Code of Employee Performance Page 3 .....	267
Figure 230: Interface of Employee Performance Overview .....	267
Figure 231: Interface of Average Performance Score by Department .....	268
Figure 232: Interface of Average Satisfaction by Department .....	269
Figure 233: Average Overtime Hours by Department .....	270
Figure 234: Source Code of Quarterly Trends .....	271
Figure 235: Interface of Quarterly Trends .....	272
Figure 236: Source Code of Overtime Analysis .....	273
Figure 237: Interface of Overtime Analysis .....	274
Figure 238: Source Code of Sick Day Analysis .....	275
Figure 239: Interface of Sick Day Analysis .....	275
Figure 240: Source Code of Download Summary Report 1 .....	276
Figure 241: Source Code of Download Summary Report 2 .....	277
Figure 242: Interface of Download Summary Report .....	278
Figure 243: Downloaded Summary Report .....	278
Figure 244: Employee Performance Report in Text File .....	278
Figure 245: Employee Performance Report in Excel File .....	279
Figure 246: Source Code of Actionable Insights for Retention and Productivity Page 1 .....	280
Figure 247: Source Code of Actionable Insights for Retention and Productivity Page 2 .....	281

Figure 248: Source Code of Actionable Insights for Retention and Productivity Page 3 .....	281
Figure 249: Source Code of Actionable Insights for Retention and Productivity Page 4 .....	282
Figure 250: Interface of Insight Selector .....	283
Figure 251: Option of Insight Selector .....	283
Figure 252: Interface of Compensation Insights.....	284
Figure 253: Downloaded Compensation Insight Report .....	284
Figure 254: Compensation Insights Report in text file.....	285
Figure 255: Interface of Productivity Insights .....	285
Figure 256 :Downloaded Productivity Insight Report.....	286
Figure 257: Productivity Insights Report in text file .....	286
Figure 258: Interface of Retention Insights .....	287
Figure 259: Downloaded Retention Insight Report .....	287
Figure 260: Retention Insights Report in text file .....	288
Figure 261: Interface of Engagement Insights.....	288
Figure 262: Downloaded Engagement Insight Report.....	289
Figure 263: Engagement Insights Report in text file .....	289
Figure 264: Source Code of Talent Dashboard Page 1 .....	290
Figure 265: Source Code of Talent Dashboard Page 2.....	290
Figure 266: Source Code of Talent Dashboard Page 3 .....	291
Figure 267: Interface of Source Code Talent Dashboard .....	292
Figure 268: Interface of Average Tenure Trend 1 .....	293
Figure 269: Interface of Average Tenure Trend 2 .....	294
Figure 270: Source Code of Hiring Recommendation Page.....	295
Figure 271: Interface for Hiring Recommendations .....	296
Figure 272:Pie Chart on Education Distribution for Hiring Recommendations.....	297
Figure 273: Boxplot on Experience Distribution for Hiring Recommendations .....	297
Figure 274: Source Code of Hiring Priority Recommendations.....	298
Figure 275: Interface of Priority Hiring Recommendations 1 .....	299
Figure 276: Interface of Priority Hiring Recommendations 2 .....	300
Figure 277: Interface of Generate Hiring Strategy Report .....	300
Figure 278: Downloaded Hiring Recommendations Report Summary.....	301

Figure 279: Hiring Recommendations Report Summary in Text File.....	301
Figure 280: Project Plan Form.....	322
Figure 281: Ethnics Form .....	323
Figure 282:Ethnics Form .....	324
Figure 283:Ethnics Form .....	325
Figure 284:Ethnics Form .....	326
Figure 285: Log Sheets (Meeting 1).....	327
Figure 286: Log Sheets (Meeting 2) .....	328
Figure 287: Log Sheets (Meeting 3) .....	329
Figure 288: Log Sheets (Meeting 4) .....	330
Figure 289: Log Sheets (Meeting 5) .....	331
Figure 290: Log Sheets (Meeting 6) .....	332
Figure 291: Poster .....	333
Figure 292: Gantt Chart of Semester 1 .....	334
Figure 293: Gantt Chart of Semester 2 .....	335
Figure 294: Code Implementation .....	393
Figure 295: Turnitin Similarity Report .....	394
Figure 296: Turnitin Similarity Report .....	395

**List of Tables**

Table 1: Project Plan Sem 1.....	39
Table 2: Project Plan Sem 2.....	41
<i>Table 3: Similar Works .....</i>	56
Table 4: Data Dictionary.....	74
Table 5: Outliers for Numerical Variables.....	114
Table 6: Hyperparameter Search Space for Random Forest Model .....	146
Table 7: Hyperparameter Search Space for XGBoost Model.....	149
Table 8: Hyperparameter Search Space for Gradient Boosting Model .....	153
Table 9: Hyperparameter Search Space for Support Vector Machine Model .....	158
Table 10: Comparison of Base and Tuned Model (Random Forest) .....	170
Table 11: Comparison of Base and Tuned Model (XGBoost) .....	183
Table 12: Comparison of Base and Tuned Model (Gradient Boosting) .....	196
Table 13: Comparison of Base and Tuned Model (Support Vector Machine) .....	209
Table 14: Comparison of Machine Learning Algorithms (Base Model).....	214
Table 15: Comparison of Machine Learning Algorithms (Tuned Model) .....	216
Table 16: Comparison of XGBoost Performance Metrics for Employee Attrition Prediction ..	219

## Chapter 1: Introduction

### 1.1 Introduction

In the fast-paced world of business today, retaining skilled employees and maintaining workforce productivity are essential for an organization's long-term success. Employee turnover, known as "employee churn," means an employee has resigned from a company and has been replaced by a new employee. High churn rates have severe challenges which have included the rising costs of recruitment, disruption of daily operations by the loss of critical knowledge and experience (Murcia, 2023). Hiring and training employees require companies to dedicate substantial time along with financial assets for creating essential workers in their teams. So, once an employee leaves, the company loses the investments and faces additional costs in recruiting and onboarding replacements (Francesca, 2020). Hence, understanding and reducing employee churn becomes important for achieving stability and efficiency in the workforce. According to Gallagher's 2024 Workforce Trends Report, 66% of HR executives still say retention is their biggest workforce challenge. Nowadays, there is more than 50% of operations executives list turnover as their primary obstacle which has indicating that employee retention is not just an HR issue but is a critical business concern (Finnegan et al., 2025).

Human Resources (HR) departments is responsibility to face this issue and to develop strategies related to employee retention and productivity enhancement. Traditional retention and productivity enhancement strategies often rely on historical data and generalized approaches which may fail to target the specific needs of certain employees (Jana, 2020). However, advancements in machine learning (ML) enables computers to identify patterns and make predictions on transforming how organizations analyze employee data. With machine learning techniques, HR professionals can uncover hidden trends, predict future outcomes and make data-driven decisions to improve employee retention and performance (Karimi & Viliyani, 2024).

Machine learning predicts employee churn with its applications including predicting employee behaviour based on factors such as job satisfaction, working hours and team size to predict whether an employee is likely to leave (Alamsyah & Salma, 2022). Apart from predicting employee turnover, these models also help organizations weed out some of the key factors that influence productivity typically identified as professional development opportunities, work-life balance, and workload distribution (Almeida & Duarte, 2024). With such insights, organizations can take

proactively to reduce employee turnover, improve working conditions and boost overall performance.

The goal of this project is to analyze the use of machine learning to predict employee turnover and the factors affecting workforce productivity. The prediction is done by analyzing the data of the employees to develop a model for better prediction with recommendations for turnover reduction and employee performance enhancement. These insights will allow organizations to make informed decisions, reduce disruptions and create a more stable and productivity-oriented work environment. Through predictive analytics, businesses are able to increase operational efficiency and improve their ability to retain top performers in order to ensure long-term success in a competitive market. This approach aligns with SDG 8: Decent Work and Economic Growth by fostering sustained, inclusive economic growth, full employment and decent work while promoting continuous improvement to build a resilient and sustainable workforce.

## 1.2 Problem Background

Employee turnover and poor productivity face a major challenge for organizations which are resulting in increased recruitment costs, training fees and business interruption. High employee attrition rates disrupt workforce stability which is causing knowledge loss and additional pressure on remaining employees. Similarly, low workforce productivity negatively affects business growth which is causing reduced efficiency and overall organizational performance. Although machine learning (ML) has advanced techniques for predicting employee churn and optimizing productivity, most organizations continue to employ traditional approaches and reactive decision-making approaches which often fail to capture the complex and dynamic nature of employee behavior (Verma et al., 2021; Li & Xu, 2023).

There is a growing need for integrated HR analytics solutions that combine churn prediction with productivity analysis and offer HR departments with actionable insights to improve employee retention and performance. With the help of advanced ML models, HR professionals can now gain actionable insights to enhance employee retention, increase engagement and automate performance management. This study aims to address the existing gaps by developing an ML-driven predictive dashboard that provides organizations with a data-driven framework for reducing turnover and improving workforce efficiency.

### **1.2.1 Employee Churn Prediction Challenges**

Employee attrition is a major problem for organizations that is causing recruitment and training expenses to be high as well as operational inconveniences (Kashyap & Bansal, 2022). Despite the growing availability of employee data, accurately predicting which employees are likely to leave remains a challenge. Traditional statistical methods used by many organizations often struggle to capture the complex and dynamic patterns of employee behaviour.

Recent advancements in machine learning (ML), such as neural networks and ensemble methods like techniques like random forests have proven to be more precise in employee turnover prediction from the analysis of various variables like job satisfaction, tenure, compensation and work environment (Verma et al., 2021; Li & Xu, 2023). However, the adoption of these advanced techniques in HR analytics is still limited, indicating the necessity for more extensive research and applied models to better facilitate churn prediction. Therefore, it is essential to overcome these limitations to ensure workforce stability and reducing operational inefficiencies.

One of the major challenges is poor quality employee data such as missing, unstructured, or incomplete HR records minimize the accuracy of churn predictions (Mehrabi et al., 2021). In addition, using employee data for predictive modeling is raising ethical and privacy concerns and is generating transparency, bias and regulatory compliance challenges in relation to regulations such as GDPR (Raghavan et al., 2020). Another is the limited HR expertise in ML implementation, as majority of HR departments lack the technical knowledge required to build, interpret, and deploy ML-driven churn prediction models (Dastin, 2022). To solve these issues, organizations need comprehensive and ethical ML-driven solutions that can accurately identify churn risks while maintaining fairness and compliance.

### **1.2.2 Impact of Employee Productivity**

Workforce productivity is another key challenge that directly affects an organization's competitiveness and long-term growth. Various factors such as remote work, employee engagement, training and job satisfaction all have a critical impact on productivity (Pereira et al., 2020). Organizations that fail to monitor and enhance productivity often experience reduced performance, lower employee morale and increased burnout rates. This absence of actionable insights often leads to missed opportunities for improving productivity and solving problems.

Despite the availability of employee performance data, the majority of businesses are operating without real-time analytics software to enable proactive productivity management (Jiang et al., 2022). Companies typically use annual performance assessments for HR evaluation, but such processes deliver information only after the fact with no direct direction for future workforce efficiency improvements. Furthermore, the absence of AI-driven productivity monitoring systems results in missed opportunities to identify performance trends and recognize high-performing employees.

Implementing advanced ML models can solve these limitations by analyzing real-time employee data such as work hours, collaboration patterns and training effectiveness to predict and enhance productivity (Chien et al., 2021). Additionally, these models can identify employee engagement trends to allow the HR teams to customize personalized interventions to improve performance and well-being. Moreover, they can detect early signs of burnout through behavioral and workload analysis to ensure proactive measures are taken to sustain productivity levels.

### **1.2.3 Lack of a Comprehensive Framework for Both Churn and Productivity**

Expert research has improved individual studies into employee turnover prediction and performance evaluations separately yet lacks shared models which tackle these challenges concurrently (Chien et al., 2021). Research has mainly examined churn prediction and productivity analysis without taking into consideration how these processes work dynamically with each other. For example, employees who are at risk of leaving may also be those with lower productivity levels. Conversely, low productivity could increase the likelihood of turnover, but these factors are often analysed in isolation. Integrative research conducted by Chen et al. (2021) validates how combining churn prediction models with performance analytics delivers better results, but there are few have successfully implemented this combined approach.

Studies suggest that predictive models combining churn and performance analysis can significantly improve workforce management strategies (Li & Xu, 2023). An integrated approach would allow organizations to simultaneously track employee churn risk and productivity patterns which can give a comprehensive view of workforce stability. By identifying correlations between low productivity and turnover risk, HR teams can intervene early and implement targeted retention strategies. Therefore, developing such an integrated ML-powered HR analytics system can help

organizations retain employees and enhance productivity while supporting long-term economic growth.

#### **1.2.4 Challenges in Data Quality and Ethical Considerations**

A major barrier to implementing ML in HR analytics is data quality and ethical concerns. Employee data shows frequent inconsistencies and incompleteness that leads to unsatisfactory predictive model accuracy according to Kashyap & Bansal (2022). Many organizations store HR data across multiple disconnected systems and make it challenging to consolidate information for effective analysis.

Additionally, ML models face critical security threats due to their biased operations. Historical HR data may reflect unconscious biases related to gender, age, or race and this leads to unfair or discriminatory predictions (Mehrabi et al., 2021). Without proper bias mitigation techniques, AI-driven HR models can reinforce existing inequalities, creating ethical and legal challenges (Raghavan et al., 2020). To ensure fair and transparent ML implementation in HR analytics, organizations must adopt strong data management strategies to enhance data accuracy and consistency. Organizations will be able to realize the complete potential of ML-driven workforce analytics and maintain fair practices and accurate results by solving these issues.

### **1.3 Project Aim**

The aim of the project is to enhance employee retention and productivity within organizations by developing a predictive machine learning model. This model will identify employees at risk of leaving and offer insights into productivity patterns with helping organizations to reduce turnover costs and improve operational efficiency.

### **1.4 Objectives**

1. To develop machine learning model for predicting employee churn based on key factors.
2. To analyse the relationship between productivity metrics and employee performance.
3. To design dashboard that provides HR professionals with model predictions on employee churn and insights into factors affecting productivity.
4. To evaluate, compare and select the best machine learning models based on performance metrics to provide actionable insights for employee retention and productivity optimization.

## 1.5 Scope

### 1.5.1 Dataset

This is a dataset consisting of 100,000 rows providing key data regarding employees' performance, productivity, and demographics within a corporate environment. It was obtained from Kaggle.com and includes information related to employees' jobs, working habits, levels of education, performance, salary information and levels of satisfaction. The dataset is ideal for predicting employee churn and optimizing workforce productivity through machine learning. It contains both numerical and categorical data that could be used to analyze trends in employee behavior, identify factors influencing turnover, factors in productivity and create predictive models to enhance organizational performance.

### 1.5.2 Deliverables

Project deliverables adopt a formal methodology from data understanding and preprocessing to machine learning model building, evaluation, and deployment. Every step in the project aims to optimize the predictive capability of employee churn as well as productivity and yield actionable intelligence for HR decision-making.

#### 1. Data Understanding and Analysis

The initial process involves understanding data structures which are including exploring variables, the distributions and the distributions' understandability of predicting employee churn and productivity workforce optimization. Exploratory Data Analysis (EDA) serves an essential function in discovering data relationships which influence employee retention patterns.

Various research works recognize EDA as an essential component of HR analytical methods. Suresh et al. (2022) analyzed employee attrition data through exploration to determine salary and job satisfaction and work-life balance were the main drivers of employee turnover. Machine learning models benefit from existing evidence demonstrating the requirement to study important HR indicators beforehand. The research by Jain, Tomar and Jana (2023) showed an improvement in model accuracy by 11% through their correlation analysis and feature selection modeling technique for employee churn prediction.

By applying EDA techniques such as correlation analysis, feature selection and data visualization, organizations can identify key patterns influencing employee turnover and optimize workforce productivity. This initial analysis ensures that predictive models are developed using relevant, high-impact features which helps resulting in more accurate retention strategies.

## **2. Data Preprocessing and Feature Engineering**

After gaining insights into the dataset, the next step involves preprocessing and feature engineering. The data cleaning process includes handling missing values and inconsistent data while performing normalization on numerical features and suitable encoding of categorical variables. The creation of new features serves to make model predictive stronger to enhance its strength. The procedure involves data aggregation techniques and variable transformation processes or attribute combination to have a better capture of the factors influencing churn and productivity. The result will be a refined dataset optimized for training machine learning models.

## **3. Model Development and Training**

The project will focus on the development and training of machine learning models. Various algorithms like classification models for predicting churn and regression models for optimizing productivity will be evaluated. Next, the models would be trained on the prepared dataset and their performance would be compared based on standard metrics like accuracy, precision, recall and F1 score. The model with the highest accuracy will be selected based on its predictive capabilities and ability to generalize across different scenarios.

## **4. Model Evaluation and Interpretation**

Once the models are trained, the next deliverable will involve evaluating their performance and interpreting the results. The evaluation phase will concentrate on both employee churn prediction accuracy and productivity-associated factors detection. Model accuracy stands as the main evaluation criterion to select the top performing model which will also be evaluated using confusion matrices together with ROC curves and precision-recall curves. True Positive (TP) and True Negative (TN) and False Positive (FP) and False Negative (FN) are the four predictive categories that the confusion matrix uses to evaluate classification models. The

assessment values enable measurements for accuracy and precision together with recall and F1 score to evaluate model performance. The confusion matrix shows where the model is making errors such as over-predicting churn (False Positives) or failing to predict employee churn (False Negatives). Through the results examination HR teams gain detailed understanding of the model's performance which becomes important for developing retention strategies that advance productivity outcomes. Furthermore, feature importance analysis will be conducted to identify which attributes are most influential in the predictions and offer additional insights into the primary drivers of employee churn and productivity.

## 5. Model Deployment and Reporting

The final deliverable will involve deploying the trained machine learning models into an interactive dashboard for real-time employee churn prediction and workforce productivity insights. The dashboard will allow HR professionals and management to input new employee data and receive predictions on churn risk and productivity levels. The dashboard will feature dynamic visualizations such as heatmaps, trend analysis charts and risk categorization tables to allow HR teams to easily interpret model predictions. The designed visualizations enables HR professionals to take anticipatory decisions which help preserve employees while maximizing workforce potential. The dashboard needs to conduct regular evaluations of new employee data for preserving accurate and relevant analytical insights.

Although the dashboard will not provide real-time updates, it will regularly refresh and update the predictions to ensure that they reflect the latest data. The system will have built-in monitoring features to measure prediction accuracy while workforce trends develop. Through implementation of this dashboard system, HR teams will have an user-friendly interface that generates data-based decisions and supports retention strategy refinement by using current information.

### 1.5.3 Constraint & Project Boundaries

#### Constraints:

1. The data used in this project will be sourced from publicly available datasets on Kaggle in compliance with privacy regulations and anonymization requirements.

2. The project will rely on pre-existing machine learning models to avoid the development of new algorithms from scratch.
3. The system will rely on the accuracy of historical data and any errors in the dataset may affect prediction quality and have no real-time adaptability unless manually updated or retrained.

#### **What will be done as part of the project:**

1. Data preprocessing will be performed to clean and remove noise from the dataset.
2. A machine learning model will be developed to predict employee churn.
3. An interactive dashboard will be created to visualize employee data and churn predictions.
4. Model evaluation will be conducted using metrics like the confusion matrix to assess accuracy.
5. Actionable insights will be provided to help HR teams interpret patterns and make data-driven decisions to improve employee retention and productivity.

#### **What will not be done as part of the project:**

1. Real-time data integration or live updates will not be implemented and updates will require manual data uploads.
2. Employee-specific intervention plans or automated decision-making processes will not be generated by the system.
3. Continuous monitoring and automatic retraining of the machine learning model will not be included.

## **1.6 Potential Benefit**

### **1.6.1 Tangible Benefit**

Tangible benefits are measurable and quantifiable outcomes that can be tracked or calculated directly. This project will deliver the following tangible benefits:

1. Reduced Employee Turnover Costs: The organization can implement targeted retention strategies, reducing recruitment and training expenses by predicting employee churn.

2. Improved Decision-Making: The dashboard provides real-time insights which enable HR teams to make decisions to improve employee satisfaction and productivity.
3. Enhanced Workforce Productivity: Organizations can achieve measurable improvements in output and effectiveness through optimizing employee performance and efficiently allocating resources based on key productivity drives.
4. Operational Efficiency: HR able to save time and increase accuracy in HR reporting by automating the analysis of employee data in reducing manual work.
5. Cost Savings: Identifying potential employees who intend to leave early enables prevention of productivity losses and the associated costs of turnover.

### **1.6.2 Intangible Benefit**

Intangible benefits are non-measurable outcomes that positively influence organizational culture, employee engagement and long-term strategies. This project will deliver the following intangible benefits:

1. Improved Employee Satisfaction: Solving factors leading to dissatisfaction creates better employee morale and fosters a positive work environment.
2. Better Workforce Planning: The model's insights support long-term workforce planning and strategic decision-making.
3. Proactive HR Strategies: HR departments can take early action and implement specific retention initiatives by identifying churn risks.
4. Enhanced Organizational Reputation: Lower turnover and better employee engagement contribute to a stronger employer brand and improved public reputation.
5. Increased Data-Driven Culture: Implementing machine learning fosters a culture of using advanced analytics to inform HR policies and business strategies.

### **1.6.3 Target User**

1. **HR Teams:** HR teams will be the primary users of the dashboard since they are responsible for managing the workforce and solving issues related to employee retention and churn.

They will use the system to analyse employee churn risk factors with identify trends and take proactive measures for workforce retention.

2. **Company Executives:** Senior leaders such as the CEO, CFO or other stakeholders will use the insights from the model to make data-driven decisions. They will examine on the reports, key metrics and overall workforce productivity insights to develop strategic decisions regarding staffing, resource allocation and workforce planning.
3. **Talent Acquisition Teams:** Talent Acquisition teams gain insights from the system by understanding employee churn patterns and predicting potential future attrition. It will help them refine their recruitment strategies and focus on hiring for positions that may experience higher turnover or require additional support.

## 1.7 Overview of IR

The first chapter, Introduction, is important as it provides an overview of the project which has its background, objectives, scope and expected outcomes. It begins by introducing the project and explaining the problem it aims to address. This chapter highlights the project's aim and objectives to guide the overall direction of the work. The project scope definition includes selection of relevant dataset as well as essential deliverables together with all existing limitations and constraints. The discussion explains how tangible and intangible aspects can help projects enhance organizational processes and decision-making. The conclusion of this chapter defines its target audience and includes project timeline details alongside important timeline indicators.

The second chapter, Literature Review, analyzes multiple research projects along with similar works related to the project. The review of literature starts with an explanation of its role in advancing project development before continuing to its analysis. This chapter examines prior studies in the domain to identify key findings and explores technical methodologies relevant to the project. By reviewing similar works, it offers insights into best practices and potential challenges. The chapter concludes with a summary about how literature findings connect to project methodology development to ensure a strong theoretical foundation.

The third chapter, Methodology, shows the systematic approach used to conduct the project. It describes the overall research strategy and explains the selecting methodology. The steps involved in completing the project include data collection, data analysis and model development. The paper

includes a section explaining the methods used to process data for accuracy and reliability. This chapter presents these procedures to establish a comprehensive comprehension of project execution starting from its initiation through completion.

The final chapter, Conclusion, summarizes the overall findings and outcomes of the project. It reflects on how the project's objectives were achieved and evaluates the effectiveness of the methodology used. The chapter delivers an overview of practical uses for project outcomes together with implementation recommendations. The research faces certain project limitations which the final chapter highlights while proposing enhancement strategies for the future investigation. The conclusion provides a thorough review of the project's contributions and emphasizes its value in solving the problem statement.

## 1.8 Project Plan

### 1.8.1 Project Plan Sem 1

Task Name	Duration	Start Date	Finish Date	Status
<b>Project Proposal Form</b>	<b>16 days</b>	<b>Mon 16/12/24</b>	<b>Mon 6/1/25</b>	<b>Completed</b>
<b>CHAPTER 1: INTRODUCTION</b>	<b>4 days</b>	<b>Thu 13/2/25</b>	<b>Tue 18/2/25</b>	<b>Completed</b>
1.1 Introduction	1 day	Mon 13/2/25	Mon 13/2/25	Completed
1.2 Problem Background	2 days	Thu 13/2/25	Fri 14/2/25	Completed
1.3 Project Aim	1 day	Fri 14/2/25	Fri 14/2/25	Completed
1.4 Objectives	1 day	Fri 14/2/25	Fri 14/2/25	Completed
1.5 Scope	1 day	Fri 14/2/25	Fri 14/2/25	Completed
1.6 Potential Benefit	1 day	Sat 15/2/25	Sat 15/2/25	Completed
1.6.1 Tangible Benefit	1 day	Sat 15/2/25	Sat 15/2/25	Completed
1.6.2 Intangible Benefit	1 day	Sat 15/2/25	Sat 15/2/25	Completed
1.6.3 Target User	1 day	Sun 16/2/25	Sun 16/2/25	Completed
1.7 Overview of IR	1 day	Mon 17/2/25	Mon 17/2/25	Completed
1.8 Project Plan	1 day	Tue 18/2/25	Tue 18/2/25	Completed
<b>CHAPTER 2: LITERATURE REVIEW</b>	<b>8 days</b>	<b>Wed 19/2/25</b>	<b>Fri 28/2/25</b>	<b>Completed</b>
2.1 Introduction	1 day	Wed 19/2/25	Wed 19/2/25	Completed
2.2 Domain Research	4 days	Thu 20/2/25	Tue 25/2/25	Completed
2.3 Similar Systems/Works	3 days	Wed 26/2/25	Fri 28/2/25	Completed
2.5 Summary	1 day	Fri 28/2/25	Fri 28/2/25	Completed
<b>CHAPTER 3: METHODOLOGY</b>	<b>7 days</b>	<b>Sat 1/3/25</b>	<b>Mon 10/3/25</b>	<b>Completed</b>
3.1 Introduction	1 day	Sat 1/3/25	Sat 1/3/25	Completed
3.2 Methodology	1 day	Sat 1/3/25	Sat 1/3/25	Completed
3.3 Data Collection	1 day	Sun 2/3/25	Sun 2/3/25	Completed
3.4 Initial Data Understanding	4 days	Sun 2/3/25	Wed 5/3/25	Completed
3.5 Data Preprocessing	4 days	Wed 5/3/25	Sun 9/3/25	Completed
3.6 Data Understanding	2 days	Sun 9/3/25	Mon 10/3/25	Completed
3.7 Summary	1 day	Mon 10/3/25	Mon 10/3/25	Completed

<b>CHAPTER 4: CONCLUSION</b>	<b>2 days</b>	<b>Tue 11/3/25</b>	<b>Wed12/3/25</b>	<b>Completed</b>
4.1 Achievements of the First Part of the Project	1 day	Tue 11/3/25	Wed12/3/25	Completed
4.2 Research Justification and SDG Alignment	1 day	Tue 11/3/25	Wed12/3/25	Completed
4.3 Research Gaps and Future Improvements	1 day	Tue 11/3/25	Wed12/3/25	Completed
<b>Reference</b>	<b>1 day</b>	<b>Tue 11/3/25</b>	<b>Wed12/3/25</b>	<b>Completed</b>
<b>Appendices</b>	<b>1 day</b>	<b>Tue 11/3/25</b>	<b>Wed12/3/25</b>	<b>Completed</b>

*Table 1: Project Plan Sem I*

## 1.8.2 Project Plan Sem 2

Task Name	Duration	Start Date	Finish Date	Status
<b>CHAPTER 1: INTRODUCTION</b>	<b>8 days</b>	<b>Fri 16/5/25</b>	<b>Fri 23/5/25</b>	<b>Completed</b>
1.1 Introduction	1 day	Fri 16/5/25	Fri 16/5/25	Completed
1.2 Problem Background	3 days	Sat 17/5/25	Mon 19/5/25	Completed
1.3 Project Aim	1 day	Tue 20/5/25	Tue 20/5/25	Completed
1.4 Objectives	1 day	Tue 20/5/25	Tue 20/5/25	Completed
1.5 Scope	1 day	Tue 20/5/25	Tue 20/5/25	Completed
1.6 Potential Benefit	1 day	Wed 21/5/25	Wed 21/5/25	Completed
1.6.1 Tangible Benefit	1 day	Wed 21/5/25	Wed 21/5/25	Completed
1.6.2 Intangible Benefit	1 day	Wed 21/5/25	Wed 21/5/25	Completed
1.6.3 Target User	1 day	Thu 22/5/25	Thu 22/5/25	Completed
1.7 Overview of IR	1 day	Thu 22/5/25	Thu 22/5/25	Completed
1.8 Project Plan	1 day	Fri 23/5/25	Fri 23/5/25	Completed
<b>CHAPTER 2: LITERATURE REVIEW</b>	<b>10 days</b>	<b>Fri 23/5/25</b>	<b>Sun 1/6/25</b>	<b>Completed</b>
2.1 Introduction	1 day	Fri 23/5/25	Fri 23/5/25	Completed
2.2 Domain Research	4 days	Sat 24/5/25	Tue 27/5/25	Completed
2.3 Similar Systems/Works	4 days	Wed 28/5/25	Mon 1/6/25	Completed
2.5 Summary	1 day	Sun 1/6/25	Sun 1/6/25	Completed
<b>CHAPTER 3: METHODOLOGY</b>	<b>5 days</b>	<b>Mon 2/6/25</b>	<b>Mon 7/6/25</b>	<b>Completed</b>
3.1 Introduction	1 day	Mon 2/6/25	Sat 2/6/25	Completed
3.2 Methodology	4 days	Tue 3/6/25	Fri 6/6/25	Completed
3.3 Summary	1 day	Sat 7/6/25	Sat 7/6/25	Completed
<b>CHAPTER 4: DESIGN AND IMPLEMENTATION</b>	<b>29 days</b>	<b>Sat 7/6/25</b>	<b>Tue 8/7/25</b>	<b>Completed</b>
4.1 Introduction	1 day	Sat 7/6/25	Sat 7/6/25	Completed
4.2 Data Collection	1 day	Sun 8/6/25	Sun 8/6/25	Completed
4.3 Initial Data Understanding	4 days	Mon 9/6/25	Thu 12/6/25	Completed
4.4 Data Preprocessing	4 days	Thu 12/6/25	Sun 15/6/25	Completed

4.5 Data Understanding	6 days	Mon 16/6/25	Sat 21/6/25	Completed
4.6 Model Building	16 days	Sun 22/6/25	Tue 8/7/25	Completed
4.7 Summary	1 day	Tue 8/7/25	Tue 8/7/25	Completed
<b>CHAPTER 5: RESULT AND DISCUSSIONS</b>	<b>13 days</b>	<b>Tue 8/7/25</b>	<b>Sun 20/7/25</b>	<b>Completed</b>
5.1 Introduction	1 day	Tue 8/7/25	Tue 8/7/25	Completed
5.2 Model Evaluation and Discussions	3 days	Wed 9/7/25	Fri 11/7/25	Completed
5.3 Feature Importance	1 day	Sat 12/7/25	Sat 12/7/25	Completed
5.4 Model Deployment	7 days	Sun 13/7/25	Sat 19/7/25	Completed
5.5 Summary	1 day	Sun 20/7/25	Sun 20/7/25	Completed
<b>CHAPTER 6: CONCLUSION</b>	<b>2 days</b>	<b>Tue 21/7/25</b>	<b>Mon 21/7/25</b>	<b>Completed</b>
6.1 Critical Evaluation	2 days	Sun 21/7/25	Mon 21/3/25	Completed
6.2 Limitations	1 day	Tue 22/7/25	Tue 22/7/25	Completed
6.3 Recommendations	1 day	Tue 22/7/25	Tue 22/7/25	Completed
<b>Reference</b>	<b>1 day</b>	<b>Tue 22/7/25</b>	<b>Tue 22/7/25</b>	<b>Completed</b>
<b>Appendices</b>	<b>1 day</b>	<b>Tue 22/7/25</b>	<b>Tue 22/7/25</b>	<b>Completed</b>

*Table 2: Project Plan Sem 2*

## Chapter 2: Literature Review

### 2.1 Introduction

This chapter explores employee churn prediction with workforce productivity analysis through machine learning techniques for human resource decision-making. Additionally, the chapter covers the technical requirements for data preprocessing and model building.

The chapter will start with domain research that evaluates main workforce productivity and employee turnover issues. This section acknowledges machine learning uses along with retention optimization in identifying factors influencing employee churn such as performance and salary. Furthermore, previous research concerning ML-based models focus on both employee retention strategies and productivity optimization to analyze their effectiveness and accuracy.

Additionally, an analysis of similar systems and related works is conducted which focuses on the techniques used in previous studies to obtain the outcomes and limitations. The reviewed helps identify weaknesses which guide improvements to the proposed method. This review helped to identify gaps in existing approaches and provides insights into enhancing the proposed solution. The analysis explores the ways these systems function together with fundamental organizational objectives which include economic development together with sustainable workforce maintenance.

The chapter also includes technical research which explains the tools and technologies required for this project. The implementation involves Python-based ML models in Visual Studio Code, which uses applicable libraries and frameworks for data evaluation and model training and assessment tasks. Project requirements lead to the development of hardware selection justification and database system along with operating environment choices.

The chapter summary provides essential information that supports the following sections about methodology together with implementation.

## 2.2 Domain Research

### 2.2.1 Factors Affecting Employee Churn and Performance

Organizational performance and stability suffer major effects from employee turnover rates known as employee churn. Understanding the factors influencing both churn and employee performance is important for developing effective retention strategies. The main factors which affect employee retention include job satisfaction with career development opportunities, work-life balance, leadership behaviour and compensation.

#### 2.2.1.1 Job Satisfaction and Career Development

According to Hosen (2022) job dissatisfaction leads to elevated turnover rates in multinational companies that operate in Malaysia. When employees become dissatisfied about their work duties or compensation levels or career prospects they will likely search for positions in other organizations. Research conducted by Lin and Huang (2021) proved organizations can boost job satisfaction through effective learning culture which results in enhanced job performance with lower staff turnover. When employees notice organizations putting major effort into their career development, they feel more valued along with enhance performance and stronger organizational commitment.

#### 2.2.1.2 Workplace Stress and Leadership Influence

Work-related stress is another critical factor that impact both employee turnover intentions and job performance. Putro et al. (2020) presented a study that demonstrated workplace stress boosted turnover and decreased productivity. Workplace stress develops from deadlines that are unattainable and unclear job responsibilities and heavy workloads and the absence of proper leadership support. The combination of chronic stress causes employees to leave their workplaces and simultaneously decreases their productivity as well as their work engagement.

Additionally, leadership behavior and organizational culture will affect employee retention and performance. The study conducted by Bangura and Lourens (2024) found that ineffective leadership styles, combined with a stressful work environment, accelerate turnover rates and adversely affect productivity. Conversely, Gunawan and Andani (2020) discovered that transformational leadership creates beneficial workplace conditions leading to job satisfaction coupled with enhanced performance while it decreases employee turnover. Motivated leaders who

additionally provide clear direction and positive feedback enhance performance and foster long-term employee loyalty.

#### **2.2.1.3 Compensation, Benefits and Demographics**

Compensation and benefits play a fundamental role in driving employee performance at the same time they retain valuable employees. Studies by Al-Suraihi et al. (2021) emphasized that competitive salary packages, performance-based incentives and clear career progression pathways lead to improved job engagement and reduced turnover rates. Employees with satisfaction from their compensation are more productive and motivated while working which lead to less likely to leave their organizations.

Demographic and job-related attributes also influence decisions to stay with a company and affect work performance. Nilhar Apriani et al. (2023) found that employees with longer tenure and competitive salaries tend to be highly committed to their work, maintain stability, and are less prone to leaving. Additionally, work-family conflict directly affects job performance because workers who try to juggle work and family life suffer from exhaustion as well as decreased work productivity.

#### **2.2.1.4 Theoretical Models Explaining Employee Churn and Performance**

Several theoretical models help explain the elements which influence these factors. According to Herzberg's Two-Factor Theory (1959), employees derive satisfaction from career growth and recognition, while poor working conditions and inadequate salaries result in dissatisfaction. Combining the improvement of salary conditions with the establishment of strong work relationships generates exceptional turnover reduction (Al-Suraihi et al., 2021). Furthermore, Maslow's Hierarchy of Needs posits that employees have five levels of needs starting from biological needs and ending with self-actualization demands. When employers fulfill these needs, employees are more likely to remain satisfied and committed. The extent of job-team and social community connections for employees determines their job embeddedness which shows direct correlation to reduced intentions for turnover according to Job Embeddedness Theory.

### **2.2.2 Impact of Employee Churn on Organizations**

Employee turnover at high rates impacts organizations in three major ways by affecting financial outcomes and operational performance while damaging sustainability. Higher employee turnover

costs organizations additional money on recruitment without proper workflow coordination and thus reduces workplace productivity. Additionally, frequent departures negatively influence workplace morale and increase remaining staff duties leading to damaged organizational commitment.

### **2.2.2.1 Financial and Operational Consequences**

In Southeast Asia, Malaysia has experienced a few employees' turnover rates. Hosen (2022) noted that employee turnover can lead to service delivery inefficiencies, increased burdens on remaining employees and reduced organizational loyalty. Similarly, Al-Suraihi et al. (2021) analyse those high levels of employee turnover affecting an organization's productivity, competitiveness and long-term sustainability. Operations become unstable due to staffing interruptions that reduce service quality until customer trust and business reputation are weakened.

Moreover, a shortage of skilled employees can lead organizations to experience managerial difficulties that then negatively impact productivity, profitability and service quality. The unstable employee retention pattern causes organizations to depend on replacement staff who decrease operational effectiveness and hinder skills from developing (Putro et al., 2020). Additionally, high attrition rates can also affect interpersonal relationships among employees thereby weakening team unity and reduced workplace spirit. The psychological strain on remaining employees with increased job responsibilities can escalate stress levels and reduce overall job satisfaction.

From a financial perspective, the cost of employee turnover includes substantial expenses required for recruiting and selecting employee candidates along with their initial training (Al-Suraihi et al., 2021). Organizations must invest substantial resources in onboarding to ensure new hires meet performance expectations. If high turnover results in the departure of skilled employees, maintaining service quality and business continuity becomes increasingly challenging. The loss of experienced workers affects immediate productivity and long-term strategic planning and growth.

### **2.2.2.2 Cultural and Strategic Implications**

Beyond financial and operational consequences, churn has a significant impact on workplace culture and employee engagement. Employees who depart frequently from the team can lead to instability within teams which is reducing trust and collaboration among employees. A workplace with high employee turnover may be challenging to build a strong team spirit and loyalty and this

may lead to even more people leaving. Conversely, organizations that cultivate a learning-oriented culture and provide clear career advancement opportunities tend to experience lower turnover and higher employee satisfaction (Lin & Huang, 2021). According to the researchers the support and recognition of professional growth by the organization drives employees to maintain their commitment toward the organization.

To reduce the negative impact of employee turnover, organizations are adopting predictive modeling along with human resource analytics to forecast potential departures and implement proactive retention strategies. By identifying key factors such as job satisfaction, compensation trends and workload distribution, companies can analyze their impact on employee turnover and apply machine learning models to establish predictive relationships. These insights enable organizations to develop targeted interventions that enhance workforce stability and better retention with strengthening team cohesion which leads to long-term business success.

### **2.2.3 Employee Retention Strategies to Sustain Workforce Productivity**

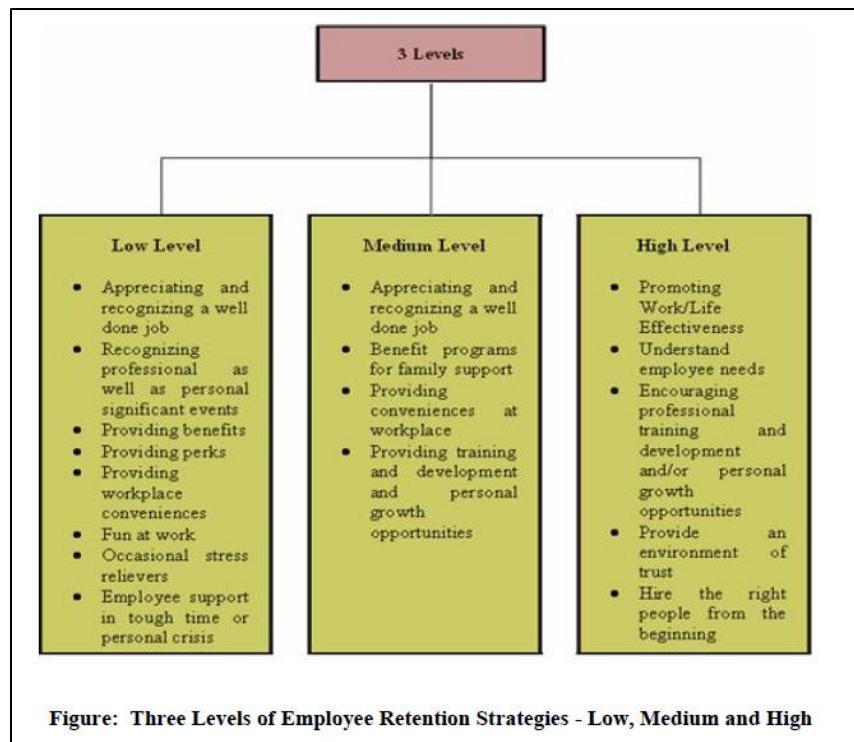
In today's competitive corporate environment, organizations face significant challenges in developing and keeping talented employees. Employee retention is important in ensuring sustainable business growth, as high turnover rates can lead to productivity losses, increased hiring costs and disruptions in workflow. Competitive companies that keep their valuable employees produce an established work team with a dedicated talent base and high motivation levels which automatically leads to production excellence combined with satisfied customers overall market competitiveness (Andrew et al., 2021).

#### **2.2.3.1 Building a Positive Organizational Culture**

A strong organizational culture that prioritizes employee well-being along with inclusivity and purpose-driven work significantly improves retention rates. Smith & Martinez (2020) found that companies with a positive work environment experience lower turnover and higher employee engagement, both of which are directly linked to workforce productivity. Employees who feel valued and with their organization's mission tend to be more committed, motivated and efficient in their roles.

Organizations that emphasize sustainability and ethical workplace practices also attract high-performing employees especially among younger generations because these individuals value

corporate responsibility and workplace values. According to Anand et al. (2023) highlights that organizations with strong retention strategies not only reduce turnover costs but also maintain a high-quality workforce which leads to a greater operational efficiency and innovation. When employees remain in an organization for longer periods, they develop deeper expertise and work more effectively which saves time and resources by reducing the need for constant staff recruitment and training. Figure 1 below shows the research conducted by Andrew et al. (2021) follows three distinct organizational retention levels starting from low to medium and high.



*Figure 1: Levels of Employee Retention Strategies*

### 2.2.3.2 Investing in Employee Growth and Development

A key employee retention tactic that builds workforce productivity involves offering ongoing learning access paired with skills education alongside leadership advancement programs. Organizations that invest in career growth initiatives allows their employees to perform at higher levels helps in reducing inefficiencies and boosting overall output (Johnson & Patel, 2021). Additionally, the organization benefits alongside employees through mentoring and coaching programs which enable staff members to fulfill their potential. Over time, a well-trained workforce contributes to higher efficiency with reduced errors and improved business performance.

Employees are also more likely to stay in organizations that provide clear career advancement opportunities. Companies should ensure that employees have access to professional development programs, promotions and new challenges that align with their personal and professional growth goals.

#### **2.2.3.3 Promoting Work-Life Balance and Employee Well-Being**

Work-life balance maintains significant role in reducing burnout experience in employees and increasing workforce productivity. Employees who experience flexible work arrangements, mental health support and wellness programs are more likely to remain engaged and motivated in their roles (Lee & Kim, 2022). Studies have shown that organizations that implement employee-friendly policies see reductions in absenteeism and higher work output thus has making retention strategies an essential pathway toward long-term productivity success (Andrew et al. ,2021).

Organizations can enhance employee retention by fostering a positive workplace environment that values and supports its workforce. Recognizing and appreciating employees for their contributions helps boost morale and job satisfaction. There are two key aspects to ensure continuous development and motivation which are ample career development opportunities and personal growth options (Walid Abdullah et al., 2021). The work environment when built around friendliness combined with cooperation fosters better employee engagement and connection to an organization which leads them to maintain commitment.

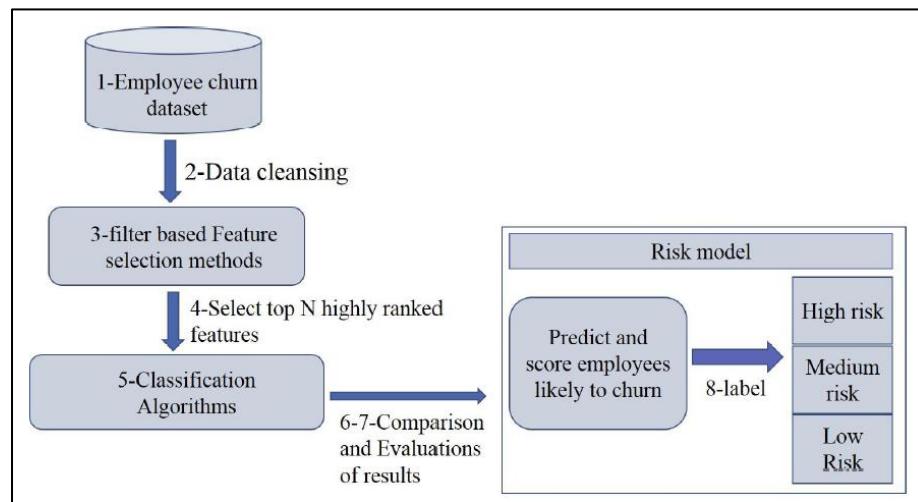
#### **2.2.3.4 Ensuring Long-Term Business Success Through Retention**

Effective employee retention strategies not only reduce turnover costs but also sustain workforce productivity by fostering a stable, engaged and skilled workforce. When employees remain with an organization for longer periods, they develop deeper expertise and work more effectively which reduces the time and resources spent on frequent recruitment and training (Andrew et al. ,2021).

Organizations that invest in a positive workplace culture, career development and employee well-being create an environment where employees can stay longer, perform better and leads to business success. As companies continue to adapt to evolving workforce trends, retaining top talent will remain essential for maintaining a competitive edge in the global market.

## 2.2.4 Machine Learning Models for Employee Churn Prediction

Machine learning (ML) has become a powerful tool for predicting employee churn therefore it is enabling organizations to identify at-risk employees and take proactive retention measures. Research by Komal et al. (2022) demonstrates a structured approach to predicting employee churn. The process begins with data cleansing and followed by feature selection using filter-based methods to choose the top N ranked features. These features are then passed through classification algorithms to build the model. Then, the models predict and classify employees into high, medium or low risk categories to ensure an accurate prediction by identifying key factors.



*Figure 2:Employee churn flow analysis*

Some of the models of machine learning such as Decision Trees, Random Forest, Gradient Boosting, Support Vector Machines (SVM) and Deep Learning have been used with large-scale research on their potential in employee churn prediction. Each models possess unique features that determine which one should be selected for a particular dataset and business operational requirements

### 2.2.4.1 Decision Tree

Decision Tree (DT) is one of the most fundamental machine learning algorithms used for employee churn prediction. It splits the dataset into subsets recursively with the help of feature values and creates a tree-based model to give prediction. The model offers interpretable advantages since its determination rules enable HR professionals to understand the outcomes easily. Research by Jain, Tomar, and Jana (2023) proves that Decision Trees produce effective churn classification results

which match or surpass complex models. However, the main drawback of Decision Trees occurs when they overfit the data especially when the model becomes complex because of its high depth. To solve this, ensemble techniques such as Random Forest and Gradient Boosting are often preferred.

#### **2.2.4.2 Random Forest**

Random Forest (RF) is an ensemble learning method that enhances the predictive performance of individual Decision Trees by combining multiple trees. It works by randomly selecting subsets of data and features, training multiple decision trees and aggregating their predictions to reduce variance and improve accuracy. Past research has established that Random Forest consistently performs better than one Decision Tree in churn classification. For example, Karimi and Viliyani (2024) found that Random Forest provided better prediction accuracy compared to traditional regression models. Additionally, a study by Md Sumon Gazi (2024) reported that Random Forest achieved the highest accuracy (86.05%) in predicting employee churn and is surpassing Decision Trees and Gradient Boosting. Its effectiveness against overfitting and its ability to handle large datasets with high-dimensional features make it a popular choice for churn prediction.

#### **2.2.4.3 Gradient Boosting**

Gradient Boosting Machines (GBM) are another ensemble learning technique that iteratively improves model performance by correcting the errors of previous trees. The output of Random Forest remains independent whereas Gradient Boosting Machine builds trees one after the other so each next tree aims to reduce remaining error from previous trees. The performance of Gradient Boosted Machine (GBM) for modelling employee churn has been proven in a number of studies. A Gradient Boosting classifiers analysis performed by Ajit (2022) reached 97% accuracy in employee at-risk identification according to the research. While GBM offers high predictive power, it is computationally more expensive and requires careful tuning of hyperparameters to avoid overfitting.

#### **2.2.4.4 Support Vector Machine**

Support Vector Machine (SVM) is an effective supervised learning algorithm for identifying binary classification tasks. SVM works by finding the optimal hyperplane that maximizes the margin between different classes to enhance the model's ability which is allowing improved generalization to unseen data. One of SVM's key advantages is its ability to handle high-

dimensional datasets and simultaneously prevent overfitting issues from occurring. Research conducted by Karimi and Viliyani (2024) found that SVM performed competitively in employee churn prediction compared to logistics regression during employee churn forecasting. The analysis conducted by Yao (2024) showed that SVM utilized with particle swarm optimization reached high accuracy levels in employee turnover prediction. The performance optimization of SVM becomes challenging because selecting appropriate kernel functions along with its computational resource needs increase when dealing with extensive datasets. Despite these challenges, its strong generalization capability makes it a valuable tool for employee churn prediction.

#### **2.2.4.5 XGBoost**

Extreme Gradient Boosting (XGBoost) is a highly efficient and scalable ensemble learning method that has shown strong performance in employee churn prediction. It improves upon traditional Gradient Boosting by using second-order optimization, regularization and parallel processing to make it suitable for large and complex HR datasets. Nan and Zhang (2023) demonstrated that an enhanced XGBoost model achieved 99.5% accuracy in predicting churn by applying variance coding and noise smoothing techniques. Muthugala et al. (2024) also reported 91% accuracy using gain-based feature selection which has highlighting XGBoost's effectiveness in handling class imbalance and ranking key predictors such as job satisfaction and overtime. Similarly, Down (2023) found that XGBoost outperformed logistic regression and support vector machines on the IBM HR dataset by achieving an AUC of 0.86. Although XGBoost requires careful hyperparameter tuning and significant computational resources, it handles missing data well and provides strong generalization.

#### **2.2.4.6 Deep Learning**

The prediction of employee churn benefits greatly from deep learning technology through the implementation of neural networks. By applying multiple layers of artificial neurons, deep learning models can discover complex patterns in large datasets that traditional ML models may struggle to detect. Research from Murcia (2023) found that deep learning models demonstrated the best results among all ML approaches used for churn prediction. However, despite their superior performance, deep learning models require substantial computational resources, extensive training data and are often criticized for their lack of interpretability. The lack of explainability makes deep learning models impractical for human resource analytics due to essential decision-making needs.

## 2.3 Similar Search/Works

Table 2 below provides an overview of existing machine learning algorithms and models for predicting employee churn, identifying their strengths and weaknesses and presenting case studies of successful implementations.

<b>Research Topic</b>	<b>Author(s)</b>	<b>Description</b>	<b>Dataset Used</b>	<b>Analysis Techniques</b>	<b>Evaluation Techniques</b>	<b>Result / Outcomes</b>	<b>Limitations</b>
Churn Prediction of Employees Using Machine Learning Techniques	Nilasha Bandyopadhyay* and Anil Jadhav (2021)	This paper aims to discuss a structured way for predicting employee churn using various classification techniques.	Focused on IT professionals aged 20 to 39, the group with the highest turnover. Data was collected through surveys.	Support Vector Machine, Naive Bayes Classifier, Random Forest Classifier	Accuracy, Recall, Specificity, Precision, F-Measure, Area Under Curve (AUC)	The Random Forest classifier proved to be the best model for IT employee churn prediction with highest accuracy and recall value.	The study used a limited dataset of 158 entries. More data points and features could improve model accuracy.
Employee Attrition Prediction in the USA: A Machine Learning Approach for HR	Gazi, Md Sumon, et al. (2024)	The objective is to explore the application of machine learning in forecasting employee attrition and its impact on HR analytics and	From the IBM Human Resource workforce attrition survey dataset.	Random Forest, Gradient Boosting, Ada-Boost Algorithm, XGBoost, Decision	Confusion Matrix, Algorithm Performance Evaluation like accuracy, precision,	The Random Forest algorithm achieved the highest accuracy (86.05%), followed by Gradient Boosting and Decision Tree.	The model's accuracy depends on high-quality employee data, and incomplete or biased data may lead to unreliable predictions. Additionally, privacy

Analytics and Talent Retention Strategies		talent retention strategies.		Tree Algorithm	recall and F1-score.		and ethical concerns arise because of using machine learning for employee churn prediction may raise issues related to data privacy, security and potential bias in decision-making.
Predicting Employee Attrition Using Machine Learning Techniques	Francesca et al. (2020)	This study analyzes the impact of objective factors on employee attrition to identify key drivers influencing an employee's decision to leave and predict their likelihood of resignation.	From IBM Analytics, which includes 35 features and approximately 1500 samples	Decision Tree, Bayesian Method, Logistics Regression, Support Vector Machine	Accuracy, Precision, Recall, F1-score	The Gaussian Naïve Bayes classifier achieved the best performance, with the highest recall rate (0.54) and a false negative rate of just 4.5% of total observations.	Further research could explore sensor-based data integration for real-time behavioral insights to enhance prediction accuracy and corporate decision-making.

Employee Turnover Prediction A: Comparative Study of Supervised Machine Learning Models	Suvoj Reddy Kovvuri, Lydia Sri Divya Dommeti (2022)	This research work compares supervised machine learning methods to predict an employee's future in a company	Dataset is taken from Kaggle.com with 4,653 rows 9 columns with 8 features and 1 target label.	Logistic Regression, Naive Bayes Classifier, Random Forest Classifier, XGBoost	Accuracy, Precision, Recall, F1-score, ROC AUC Score,	The XGBoost model achieved the highest accuracy of 85.3% followed by Random Forest at 83%. The XGBoost also led in precision (0.88) and F1 score (0.77).	A limitation of the analysis is the lack of clear feature importance and is suggesting a need for deeper exploration and tuning.
Using Machine Learning to Predict Employee Turnover	Luísa Pereira (2024)	This study examines the factors influencing employee turnover at Willis Towers Watson (WTW) and evaluates machine learning models for predicting turnover.	A sample of employees who work or have previously worked in this department.	Logistic Regression, K-Nearest Neighbor (KNN), Decision Tree, Random Forest, XGBoost	Confusion Matrix, AUC-ROC, Evaluation Metrics (Accuracy, Precision)	Based on multiple models, the results indicate that Random Forest achieved the highest accuracy, reaching 77% across various evaluation criteria.	The study was limited by a small sample size, a skewed age distribution due to WTW's hiring trends and the inability to collect job satisfaction data from former employees. Additionally, the number of explanatory variables was

							restricted.served as another limitation.
Employee Attrition Prediction Using Machine Learning Algorithms	Lekan Akinode & Olufunke Bada. (2022)	This research work compares supervised machine learning methods forecast an employee's position within an organization	IBM analytics unbalanced dataset which includes 35 attributes for 1470 employees	Logistic Regression, Naive Bayes Classifier, Random Forest Classifier, XGBoost	Accuracy, Specificity, Precision, Sensitivity, ROC AUC Score, F1-score	The XGBoost model achieved the highest accuracy with 85.4% on the imbalanced dataset and 84.48% on the synthetic balanced dataset. Random Forest followed with 83.6% imbalanced dataset and 82.08% balanced dataset.	The dataset's imbalance required synthetic data generation and future work could address additional factors like working conditions and demographic aspects.
Study and Prediction Analysis of the Employee Turnover using Machine	Raj Chakraborty and Krishna Mridha.	This study examines employee turnover in HR Analytics which highlighting its financial and operational costs. It	Data from the HR department of the company available at Kaggle were used to estimate the	Gradient Booster, Support Vector Machine, Logistic Regression,	Precision, recall, and F1-scores.	Among the tested models, Random Forest demonstrated the highest accuracy at 90.20%, while Naïve Bayes had	The study does not assess turnover factors across additional variables such as gender, education, skill levels and performance.

Learning Approaches		explores key factors like job position, overtime, and work level using machine learning techniques.	employee's turnover. The dataset includes 10 different attributes of 1470 personnel.	K-Nearest Neighbors, Naïve Bayes, Random Forest		the lowest at 80.20%.	Moreover, the findings are specific to India's sales industry and may not be widely generalizable.
Predictive model of employee attrition based on stacking ensemble learning	Chung, D. et al. (2023)	The aim of this study is to predict employee attrition so that measures can be taken for talent management.	IBM HR Analytics Employee Attrition & Performance data which consists of 1,470 records	Logistic Regression, Random Forest, XGBoost, SVM, Artificial Neural Network	Accuracy, Precision, Recall, F1-score, AUC	The top results are Random Forest with 0.947 accuracy and XGBoost with 0.925 accuracy which shows strong precision, F1 score and AUC scores.	The dataset may not include all psychological and subjective factors influencing attrition. The model's accuracy could improve by discovering new factors through qualitative analysis.

Table 3: Similar Works

Overall, the studies show the effectiveness of machine learning algorithms in predicting employee churn across various industries. There are several techniques including Random Forest, XGBoost, Support Vector Machines and Gradient Boosting have been widely adopted for their ability to identify key factors influencing attrition. Among these, Random Forest consistently proved itself as the top-performing model because it delivered high accuracy and maintained stability in different datasets.

Although XGBoost may not be the most widely used model, it has consistently delivered impressive results and achieving high accuracy along with strong precision, recall and F1-scores in multiple studies. XGBoost excels in handling imbalanced datasets and has outperformed models like Logistic Regression and Naive Bayes in predicting employee attrition.

Besides, common evaluation metrics such as accuracy, precision, recall, F1-score and Area Under Curve (AUC) are frequently employed to assess the performance of these algorithms. Studies focus on accuracy as their main evaluation metric but also explain recall and F1-score as they help to better understand the model's ability to correctly identify employees at risk of leaving.

Multiple studies have identified common challenges regarding the quality and size of datasets with some relying on limited samples that may affect their ability to provide widely applicable findings. Additionally, ethical concerns related to privacy, data bias and fairness remain critical considerations when implementing machine learning models in HR analytics.

## **2.4 Technical Research**

### **2.4.1 Hardware/Software for the Project**

The hardware used for this project is a Lenovo IdeaPad Slim 5 and it consists of an Intel Core Ultra 5 125H processor with a speed of 1.20 GHz. It is a 64-bit operating system with an x64-based processor for efficient execution of data analysis and machine learning algorithms. The laptop has 16GB LPDDR5 RAM and 512GB SSD for ample memory and storage capacity to handle large datasets running multiple applications smoothly. It also includes integrated Intel Iris Xe Graphics, which supports visualization and graphical computations. Additionally, the device is equipped with a 14-inch Full HD (1920x1080) IPS display with an anti-glare coating to ensure a clear and comfortable viewing experience. With Wi-Fi 6 and Bluetooth 5.1 connectivity, the laptop offers fast and stable internet connectivity. It also comes with multiple ports which include USB-C,

USB-A, HDMI, 3.5mm audio jack and an SD card reader which helps to offer flexibility in connecting external devices. One of its key advantages is its battery life of up to 10 hours and this is making it a suitable choice for extended work sessions.

#### **2.4.2 Programming Language Chosen: Python**

Python has become the preferred programming language for data science and machine learning due to its simplicity, flexibility and extensive ecosystem of libraries. Its clear syntax and ease of use make it accessible for both beginners and experts for facilitating rapid development and implementation of complex algorithms (Van Rossum & Drake, 2019).

One of the advantages for Python is its broad collection of libraries that support data manipulation, statistical analysis and machine learning. Libraries such as Pandas and NumPy enable efficient data handling and computation, while Scikit-learn provides various of machine learning algorithms (Pedregosa et al., 2011). Visualization tools like Matplotlib and Seaborn further enhance Python's capabilities by enabling insightful data representation (Hunter, 2007).

Python's adaptability extends to deep learning and high-performance computing, with frameworks such as TensorFlow and PyTorch optimized for large-scale machine learning applications (Abadi et al., 2016; Paszke et al., 2019). Additionally, its compatibility with cloud computing and big data technologies ensures scalability for enterprise applications (Zaharia et al., 2020).

The combination of extensive library support and adaptability and active community support makes Python the best option for implementing machine learning models in HR analytics and predictive modeling work.

#### **2.4.3 IDE (Interactive Development Environment) chosen**

By using the Integrated Development Environment (IDE) Jupyter Notebook in Visual Studio Code, it enhances code development and management. Jupyter Notebook is a widely adopted interactive computing environment that facilitates literate programming by combining live code, explanatory text, and execution results in a single document. According to Pimentel et al. (2019), Jupyter Notebooks are extensively used across scientific and industrial domains due to their ability to support interactive programming and data analysis.

Jupyter Notebooks provide a powerful framework for data analysis and visualization by enabling users to gather, execute, and display results seamlessly (Fangoehr et al., 2019). Their flexibility makes them an ideal choice for machine learning and data science applications, allowing for easy documentation, sharing, and reproducibility of analysis.

Additionally, Visual Studio Code (VS Code) is used alongside Jupyter Notebook to enhance the development experience. VS Code offers robust features such as syntax highlighting, debugging, and Git integration, which streamline the coding workflow. Its built-in support for Jupyter Notebooks allows users to execute and manage Jupyter cells within the VS Code interface, providing a cohesive environment for efficient and productive programming.

#### 2.4.4 Libraries/Tools Chosen

There are several libraries and tools are used to facilitate data preprocessing, analysis and visualization in this project.

##### Data Preprocessing and Analysis

- **Pandas:** A Python library for efficient data manipulation using DataFrames and Series to enable for filtering, aggregation and transformation of large datasets.
- **NumPy:** A core library for numerical computing to provide multi-dimensional arrays, matrices, and optimized mathematical functions.
- **Scikit-learn:** A machine learning library offering supervised and unsupervised algorithms, model selection, preprocessing and evaluation tools.
- **Matplotlib:** A versatile plotting library for visualizations like line plots and bar charts. Fangohr et al. (2019) highlight its grid-based structure for clear and structured figures.
- **Seaborn:** Built on Matplotlib, it simplifies statistical visualizations like heatmaps and violin plots for exploratory data analysis.
- **Tabulate:** A library used for generating formatted tables in Python and useful for displaying structured data in a visually appealing way.
- **OS:** A built-in library for interacting with the operating system such as handling file paths and directories.
- **Mutual\_info\_classif:** This tool is used for selecting relevant features based on mutual information from Scikit-learn.

- **OneHotEncoder, LabelEncoder:** These tools are used for converting categorical variables into numeric values to prepare data for machine learning models from Scikit-learn.
- **Train\_test\_split:** A tool from Scikit-learn used to split data into training and testing datasets.
- **SMOTE (Synthetic Minority Over-sampling Technique):** A method from imblearn used to handle imbalanced datasets by oversampling the minority class.

## **Model Building**

- **LinearSegmentedColormap:** It is used for creating custom color maps and useful for visualization purposes in machine learning tasks from Matplotlib.
- **Confusion\_matrix, classification\_report, accuracy\_score:** Evaluation tools from Scikit-learn to assess model performance.
- **RandomForestClassifier:** A machine learning algorithm from Scikit-learn for classification tasks using an ensemble of decision trees.
- **Learning\_curve, StratifiedKFold:** Tools from Scikit-learn to evaluate model performance and ensure it is reliable across different subsets of the dataset.
- **ROC\_curve, roc\_auc\_score:** Metrics from Scikit-learn for evaluating model performance for classification problems.
- **RandomizedSearchCV:** A model selection tool from Scikit-learn used for hyperparameter optimization through random sampling.
- **GradientBoostingClassifier:** A machine learning model from Scikit-learn that builds an ensemble of trees, each correcting the errors of the previous one.
- **SVC (Support Vector Classifier):** A classification algorithm from Scikit-learn for binary and multi-class classification tasks.
- **StandardScaler:** A tool from Scikit-learn for scaling data features to standardize them (mean = 0, variance = 1).

- **XGBoost:** This high-performance gradient boosting algorithm is ideal for classification and regression tasks and is known for speed and accuracy.
- **Randint, uniform:** These functions are used for generating random integers and continuous uniform distributions and used for hyperparameter tuning from Scipy.stats,

### **Deployment**

- **Joblib:** A library used for serializing Python objects and is commonly used for saving trained machine learning models to disk for later use.
- **Faker:** A Python library used to generate fake data such as names, addresses, phone numbers and emails. It's commonly used for testing, data simulation or populating databases with realistic-looking data for development and testing purposes.

### **2.4.5 Operating System chosen**

The operating system used for this project is Windows 11 (64-bit). It provides a stable and user-friendly environment for development to ensure its compatibility with various machine learning tools, libraries and visualization software. Windows 11 supports efficient multitasking, enhanced security features and smooth integration with hardware components which is making it suitable for data analysis and model development.

### **2.5 Summary**

This chapter provides a literature review of employee churn prediction that covers domain research, similar research, and technical research. The domain research explores key topics such as the factors affecting employee churn and performance and the impact of employee churn on organizations and employee retention strategies to sustain workforce productivity. It also discusses employee retention strategies aimed at sustaining workforce productivity, which is directly tied to the goal of decent work and economic growth (SDG 8). By solving these factors, the research contributes to improving workforce stability, reducing turnover and fostering better working conditions.

Additionally, it discusses various machine learning models used for employee churn prediction and analyzing their effectiveness and challenges. It also discusses previous work and relevant

research by comparing various approaches used in employee churn prediction. This chapter discusses the strengths and limitations of current research which helps to provide insights into best practices and research gaps which could inform future efforts at the alignment of human resource strategies with SDG 8. The literature consistently showed that Random Forest achieved the highest accuracy among models tested followed by Gradient Boosting and Decision Trees. These findings guided the selection of models in this study which included Random Forest, XGBoost, SVM and Logistic Regression. A common challenge identified in past studies was class imbalance which had significantly reduced the ability to detect minority of churned cases. To solve this, SMOTE (Synthetic Minority Oversampling Technique) was often applied to enhance prediction performance and reduce false negatives.

This chapter discusses the strengths and limitations of current research which helps to provide insights into best practices and research gaps which could inform future efforts at the alignment of human resource strategies with SDG 8. Further, the technical research chapter covers implementation considerations which include Python as the primary programming language. The chapter discusses the selection of tools such as Jupyter Notebook and libraries such as Pandas, NumPy, Scikit-learn and TensorFlow that facilitate data processing, model development and model assessment.

In summary, this chapter explores employee churn prediction, machine learning applications, feature extraction techniques and model evaluation. It shows the importance of data-driven approaches in understanding and solving employee churn. This chapter highlights the importance of data-driven approaches in understanding and mitigating employee churn, which directly contributes to achieving SDG 8: Decent Work and Economic Growth. The chapter concludes with a comparative analysis of previous studies and similar systems, assessing their effectiveness in predicting and mitigating employee churn. By reviewing these past studies, the model and technique chosen in this research correspond with established standards to allow the proposed system to build upon proven research practices. This literature review provides a foundation for the following chapters, integrating theoretical insights with technical approaches to develop an effective employee churn prediction model.

## Chapter 3: Methodology

### 3.1 Introduction

This chapter introduces the methodology used to guide the development of the employee churn prediction project. A project needs an appropriate methodology to implement systematic methods toward reaching its intended objectives. This chapter will discuss the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology, explaining its justification and the various phases involved.

Jaggia, S., Kelly, A., Lertwachara, K., & Chen, L. (2020) report that education about Business Analytics primarily teaches the modeling phase to students while neglecting the comprehensive analytics process. To solve this, CRISP-DM can provide a comprehensive approach that covers all phases of the analytics lifecycle and offers a better understanding of the analytics process. Not only that, CRISP-DM offers a structured approach to handle complex data analytics projects through its iterative and adaptable nature. The method proves appropriate for the project because of its emphasis on business understanding, data preparation and model evaluation. Given the need to predict employee churn, CRISP-DM supports the project's objective of transforming raw data into actionable business insights (Wirth & Hipp, 2020). Through a detailed explanation of each stage, this chapter ensures clarity on how the project is structured from data collection to model deployment to ensure reliability and accuracy in the outcomes.

## 3.2 System Development Methodology

### 3.2.1 Introduction of CRISP-DM methodology

CRISP-DM, which stands for Cross-Industry Process for Data Mining, is a widely adopted methodology for data mining and data analysis projects. The methodology provides an organized structure that maintains both comprehensive and efficient execution of the complete process. The methodology is divided into six major phases which are business understanding, data understanding, data preparation, modeling, evaluation and deployment. Each major stage highlights related tasks and deliverables. It guides the data mining process from understanding business objectives to deploying the final model. The figure below shows the CRISP-DM reference model with their corresponding phases, tasks and outputs throughout the data mining life cycle.

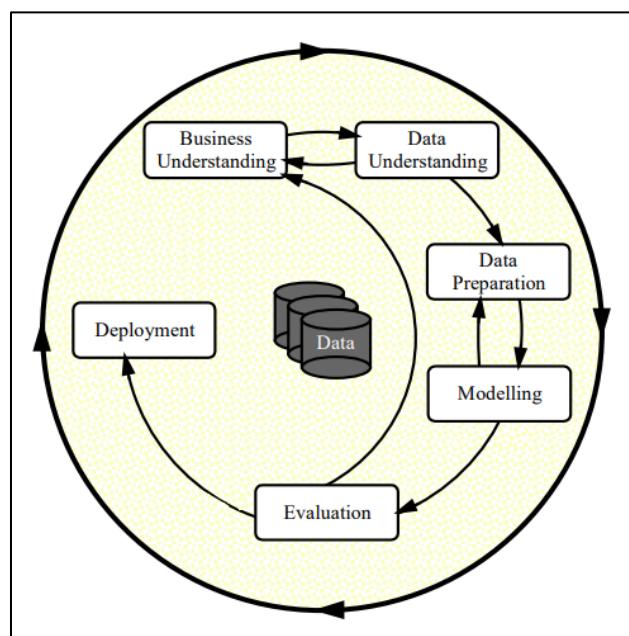


Figure 3:CRISP-DM Methodology Cycle

### 3.2.2 Methodology choice and justification

The decision to use CRISP-DM for this project is based on the structured design and flexible execution together with complete coverage of data analysis steps. CRISP-DM is particularly suited for this project due to its ability to address the entire range of project requirements from goal definitions through deployment of data mining models. Through iterative operation CRISP-DM allows our project to enhance model accuracy by refining models through knowledge acquired across each step. Making regular model refinements becomes essential for employee churn prediction tasks because data changes over time which requires improved accuracy.

CRISP-DM stands apart from the methodologies SEMMA and KDD because it places a greater emphasis on understanding the business context and ensuring that the final model meets the business objectives. This is essential for the employee churn prediction project because the final outcomes will directly inform business strategies related to employee retention and organizational planning. Additionally, CRISP-DM's focus on data preparation and model evaluation makes it ideal for handling noise data. Data cleaning along with data transformation are essential for generating dependable predictions in the employee churn datasets.

In summary, CRISP-DM offers a structured approach that covers all stages of the data analysis lifecycle which makes the best fit for the employee churn prediction project.

### 3.2.3 Phases of CRISP-DM

#### Phase 1: Business Understanding

This initial phase focuses on understanding the project's goals and requirements from a business perspective and converting them into a data mining problem. A successful data mining project involves business situation assessment followed by identifying available resources and goal definition. So, a clearly **defined business problem** ensures alignment between business requirements and analytical approaches which forms the strong foundation for complete data mining activities.

In this phase, the main objectives are to **understand the issues surrounding employee churn** and **workforce productivity** as well as to define key business goals such as reducing turnover rates and improving employee performance. This understanding will guide the data mining process,

ensuring that the insights provided are directly relevant to decision-making and actionable for HR management.

### **Phase 2: Data Understanding**

This second phase consists of **data collection through exploration** and description to gain insight and identify potential issues. The tasks during this phase consist of **statistical analysis** coupled with **checking data quality** and **understanding data variables with their relationships**. This phase is closely related to business understanding as a thorough understanding of the data helps clarify project objectives and refine the approach to analysis.

In this phase, historical datasets are found and analyzed. It is important to understand how variables such as job satisfaction, tenure and compensation are interrelated, as these relationships provide a **comprehensive view of the factors** influencing employee churn and productivity. Identifying data quality issues and relationships between variables will refine the approach to modeling and ensure that the data is reliable and meaningful.

### **Phase 3: Data Preparation**

The data preparation phase is to transform raw data into a suitable format for modeling applications. The required steps involve selecting appropriate data, cleaning inconsistencies, handling missing values and creating new variables based on the chosen modeling techniques. In this phase, numerical variables such as salary, performance scores and tenure are normalized while categorical data like department or job role may be encoded for modeling purposes. Additionally, new features are created to capture relationships that influence both churn and productivity. The iterative nature of this phase ensures the dataset continually refined and ready for modeling.

### **Phase 4: Modeling**

Modeling techniques will be selected and applied to the prepared data in this phase. The process involves **building models, adjusting parameters** and testing various approaches to identify the best fit for the business problem. The data is divided into three sections which **are training, testing and validation sets** to ensure the model's accuracy and generalizability. Then, the choice of model is chosen depends on the data characteristics and business objectives. Parameters are adjusted to optimize model performance and ensure that they meet the business objectives. This phase is

important in selecting the right model and ensuring its effectiveness in making accurate predictions for employee churn based on historical and current data.

### **Phase 5: Evaluation**

The evaluation phase focuses on assessing the **model's performance against the business objectives**. The evaluation activities consist of multiple steps such as result interpretation followed by **accuracy validation to verify model performance against project goals**. This phase also involves a thorough review to identify any overlooked business concerns. The model's ability to predict churn accurately and identify factors influencing productivity is assessed to ensure that the insights it provides are actionable for HR teams. For example, evaluating whether the model can successfully identify employees at risk of leaving or whether it can highlight areas where productivity can be improved is critical. The final evaluation ensures that the model delivers the desired business outcomes before proceeding to deployment.

### **Phase 6: Deployment**

In this phase, it involves implementing the developed model to deliver actionable insights and support business decision-making. This includes integrating the model into operational systems or HR dashboards where HR professionals can access predictions and insights. The deployment process also includes creating a plan for ongoing system integration and establishing protocols for model maintenance and updates. Continuous monitoring ensures that the model remains relevant as business needs evolve such as changes in employee behavior, company policies or market conditions. By providing regular updates and maintaining the model's accuracy, HR teams can make data-driven decisions that optimize employee retention and productivity. This final phase ensures that the model continues to deliver value which helps in supporting HR professionals in their strategic planning and day-to-day operations.

### 3.3 Summary

Chapter 3 provides a detailed description of the methodology employed in this research which focuses on predicting employee churn and optimizing workforce productivity by using machine learning techniques. The chapter follows the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology and is chosen for its structured and iterative approach which aligns well with the goals of this project. This methodology guides the entire process from data collection and preprocessing to model building and evaluation.

The chapter explains each of the six phases of CRISP-DM in detail. It begins with business understanding, where key project goals such as reducing turnover and improving workforce productivity are defined. The data understanding phase involves identifying the most relevant variables and assessing data quality. In the data preparation phase, raw data is cleaned, transformed, and encoded to ensure compatibility with machine learning algorithms. The modelling phase focuses on selecting and fine-tuning classification models to predict churn effectively. This is followed by the evaluation phase, where model performance is assessed to ensure it meets business objectives. Finally, the deployment phase discusses how the trained model can be integrated into HR operations to support real-time decision-making.

Overall, this methodology provides each of the main steps from problem definition to model implementation. With this approach, organizations can develop a system that provides insights from data to reduce employee churn and maintain a better workforce.

## Chapter 4: Design and Implementation

### 4.1 Introduction

This chapter presents the design and implementation process of the proposed employee resignation prediction system. It shows the key steps involved in transforming raw data into actionable insights beginning with data collection and preliminary data understanding. The chapter provides detailed steps of how the dataset was explored, cleaned and prepared for modeling such as missing values, handling outliers and engineering new features to enhance model performance.

Graphical visualizations are used throughout the chapter to facilitate better comprehension of variable distributions and relationships. A thorough feature selection process is conducted to identify the most relevant predictors for employee resignation. Furthermore, the dataset is split and resampled using techniques such as SMOTE to ensure balanced representation in the training phase.

Subsequently, several machine learning models have been built which has Random Forest, XGBoost, Gradient Boosting and Support Vector Machine. For each model, both baseline and hyperparameter-tuned versions are developed and compared in terms of performance metrics. The goal is to identify the best-performing model that can be applied to predict potential employee resignations with high accuracy. This comprehensive approach ensures the reliability of the final prediction model.

### 4.2 Data Collection

The data was collected from Kaggle.com. Below is the Employee Churn Productivity Datasets in a CSV file.

Dataset Name	Source (from Kaggle.com)
Extended_Employee_Performance_and_Productivity_Data	Employee Performance and Productivity Dataset

## 4.3 Initial Data Understanding

### 4.3.1 Import Dataset

```

import os
import pandas as pd
from tabulate import tabulate

# Define the file path
file_path = "Extended_Employee_Performance_and_Productivity_Data.csv"

# Check if the file exists in the current directory
if os.path.isfile(file_path):
    df = pd.read_csv(file_path)
    print("Dataset loaded successfully!")

    # Display the first five rows of the dataset
    print("\nFirst five rows of the dataset:")
    print(tabulate(df.head(), headers='keys', tablefmt='pretty', showindex=False))
else:
    print(f"Error: The file '{file_path}' was not found!")

```

*Figure 4: Source Code of Import Dataset*

The dataset is loaded into a Pandas DataFrame. If successful, it prints the confirmation message "Dataset loaded successfully!". To verify that the data has been loaded correctly, the script displays the first five rows. If the file is not found, an error message is shown.

```

Dataset loaded successfully!

First five rows of the dataset:
+---+---+---+---+---+---+---+
| Employee_ID | Department | Gender | Age | Job_Title | Hire_Date | Years_At_Company | Education_Level |
+---+---+---+---+---+---+---+
| 1 | IT | Male | 55 | Specialist | 2022-01-19 08:03:05.556036 | 2 | High School |
| 2 | Finance | Male | 29 | Developer | 2024-04-18 08:03:05.556036 | 0 | High School |
| 3 | Finance | Male | 55 | Specialist | 2015-10-26 08:03:05.556036 | 8 | High School |
| 4 | Customer Support | Female | 48 | Analyst | 2016-10-22 08:03:05.556036 | 7 | Bachelor |
| 5 | Engineering | Female | 36 | Analyst | 2021-07-23 08:03:05.556036 | 3 | Bachelor |
+---+---+---+---+---+---+---+

```

*Figure 5: Dataset Display*

Performance_Score	Monthly_Salary	Work_Hours_Per_Week	Projects_Handled	Overtime_Hours	Sick_Days
5	6750	33	32	22	2
5	7500	34	34	13	14
2	4800	52	10	28	12
2	4800	38	11	29	13
3	7800	46	31	8	0

*Figure 6: Dataset Display*

Remote_Work_Frequency	Team_Size	Training_Hours	Promotions	Employee_Satisfaction_Score	Resigned
57	14	66	0	2.63	False
40	12	61	2	1.72	False
68	10	0	1	1.86	True
58	15	9	1	1.25	True
80	15	95	0	2.77	False

*Figure 7:Dataset Display*

Figures above display the first five rows output of the datasets to make sure that the variables are imported successfully.

#### 4.3.2 View Total Rows and Columns

```
# (Rows, Columns)
print("(Rows, Columns)")      (Rows, Columns)
print(df.shape , "\n")        (100000, 20)
```

*Figure 8: Source Code of View Total Rows and Columns*

The dataset contains 100,000 rows with 20 columns.

#### 4.3.3 View Data Types

```
# View the data types and missing values
print(df.info(), "\n")
```

*Figure 9: Source Code of View Data Types*

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Employee_ID      100000 non-null   int64  
 1   Department       100000 non-null   object  
 2   Gender            100000 non-null   object  
 3   Age               100000 non-null   int64  
 4   Job_Title         100000 non-null   object  
 5   Hire_Date         100000 non-null   object  
 6   Years_At_Company 100000 non-null   int64  
 7   Education_Level   100000 non-null   object  
 8   Performance_Score 100000 non-null   int64  
 9   Monthly_Salary    100000 non-null   int64  
 10  Work_Hours_Per_Week 100000 non-null   int64  
 11  Projects_Handled  100000 non-null   int64  
 12  Overtime_Hours    100000 non-null   int64  
 13  Sick_Days          100000 non-null   int64  
 14  Remote_Work_Frequency 100000 non-null   int64  
 15  Team_Size          100000 non-null   int64  
 16  Training_Hours     100000 non-null   int64  
 17  Promotions          100000 non-null   int64  
 18  Employee_Satisfaction_Score 100000 non-null   float64 
 19  Resigned            100000 non-null   bool    
dtypes: bool(1), float64(1), int64(13), object(5)
memory usage: 14.6+ MB
None

```

*Figure 10: Dataset Structure*

The figure above shows the table of the dataset structure. The dataset contains 100,000 entries across 20 variables, with no missing values. Each attribute's data type is displayed and is showing a mix of integers, floats, objects (strings) and a Boolean column.

#### 4.3.4 View Column Variables

The dataset contains the following column variables:

```

# check columns
print("[Columns]")
print(df.columns, "\n")

```

*Figure 11: Source Code of View Column*

```

[Columns]
Index(['Employee_ID', 'Department', 'Gender', 'Age', 'Job_Title', 'Hire_Date',
       'Years_At_Company', 'Education_Level', 'Performance_Score',
       'Monthly_Salary', 'Work_Hours_Per_Week', 'Projects_Handled',
       'Overtime_Hours', 'Sick_Days', 'Remote_Work_Frequency', 'Team_Size',
       'Training_Hours', 'Promotions', 'Employee_Satisfaction_Score',
       'Resigned'],
      dtype='object')

```

*Figure 12: Display Columns Output*

### 4.3.5 Dataset Overview

This dataset consists of 100,000 rows which include the key aspects of employee performance, productivity and demographics in a corporate environment. It provides a comprehensive view of various employee variables including job-related details, work habits, education, performance and satisfaction levels. The dataset is valuable for multiple HR analytics applications, such as:

1. Employee churn prediction – Identifying employees at risk of leaving.
2. Productivity analysis – Understanding factors influencing employee efficiency.
3. Performance evaluation – Assessing employee contributions based on key metrics.

By analyzing the dataset, organizations can gain insights into employee behavior, job satisfaction, and performance trends to support strategic HR decision-making.

### 4.3.6 Data Dictionary

Column Name	Description	Data Type	Type
Employee_ID	Unique identifier for each employee	Integer	Numerical
Department	The department in which the employee is working (e.g., Sales, HR, IT)	String	Categorical
Gender	The gender of the employee (Male, Female, Other)	String	Categorical
Age	The age of the employee (between 22 to 60 years)	Integer	Numerical
Job_Title	The employee's role or job title (e.g., Manager, Analyst, Developer)	String	Categorical
Hire_Date	The date when the employee was hired	DateTime	Categorical (Time-Based)
Years_At_Company	The number of years of the employee has worked for the company	Integer	Numerical

Education_Level	The highest level of education achieved by the employees (e.g., High School, Bachelor)	String	Categorical
Performance_Score	The employee's performance rating (1 to 5 scale)	Integer	Categorical (Ordinal)
Monthly_Salary	The monthly salary of the employee in USD	Float	Numerical
Work_Hours_Per_Week	The number of hours the employee works per week	Integer	Numerical
Projects_Handled	The total number of projects handled by the employee	Integer	Numerical
Overtime_Hours	Total overtime hours worked in the last year	Integer	Numerical
Sick_Days	The number of sick days taken by the employee	Integer	Numerical
Remote_Work_Frequency	The percentage of time worked remotely	Float	Categorical
Team_Size	The number of people in the employee's team	Integer	Numerical
Training_Hours	The number of hours spent in training by the employee	Integer	Numerical
Promotions	The total number of promotions the employee has received	Integer	Categorical (Ordinal)
Employee_Satisfaction_Score	The Employee satisfaction rating (1.0 to 5.0 scale)	Float	Categorical (Ordinal)
Resigned	Whether the employee has resigned (Yes = 1, No = 0)	Boolean	Categorical (Binary)

Table 4: Data Dictionary

### 4.3.7 Descriptive Statistics of Employee Data

```

import pandas as pd
import matplotlib.pyplot as plt

# Summary Statistics
summary = df.describe().transpose()
fig, ax = plt.subplots(figsize=(14, 6))
ax.axis("tight")
ax.axis("off")

# Create Table
table = ax.table(cellText=summary.round(2).values,
                  colLabels=summary.columns,
                  rowLabels=summary.index,
                  cellLoc="center",
                  loc="center")

# Adjust table properties
table.auto_set_font_size(False)
table.set_fontsize(12)

# Column width
for i in range(len(summary.columns)):
    table.auto_set_column_width([i])

for key, cell in table.get_celld().items():
    cell.set_height(0.05)
    cell.set_width(0.1)

plt.show()

```

*Figure 13: Source Code of Descriptive Statistics*

The code generates descriptive statistics of the dataset using Pandas and Matplotlib. It uses `df.describe()` to generate the summary statistics for all numerical columns in the dataset such as count, mean, standard deviation, minimum, maximum and quartiles. Besides, `transpose()` is used to swap the rows and columns which make each statistic such as mean will be in rows and each variable such as age will be in columns.

	count	mean	std	min	25%	50%	75%	max
Employee_ID	100000.0	50111.73	28830.16	1.0	25177.75	50184.5	75033.25	100000.0
Age	100000.0	41.02	11.25	22.0	31.0	41.0	51.0	60.0
Years_At_Company	100000.0	4.48	2.87	0.0	2.0	4.0	7.0	10.0
Performance_Score	100000.0	2.81	1.43	1.0	2.0	3.0	4.0	5.0
Monthly_Salary	100000.0	6299.68	1365.24	3850.0	5250.0	6300.0	7200.0	9000.0
Work_Hours_Per_Week	100000.0	45.06	8.95	30.0	37.0	45.0	53.0	60.0
Projects_Handled	100000.0	24.42	14.47	0.0	12.0	24.0	37.0	49.0
Overtime_Hours	100000.0	14.62	8.67	0.0	7.0	15.0	22.0	29.0
Sick_Days	100000.0	7.0	4.33	0.0	3.0	7.0	11.0	14.0
Remote_Work_Frequency	100000.0	55.8	12.82	30.0	46.0	56.0	66.0	80.0
Team_Size	100000.0	10.01	5.49	1.0	5.0	10.0	15.0	19.0
Training_Hours	100000.0	49.46	28.86	0.0	25.0	49.0	75.0	99.0
Promotions	100000.0	0.93	0.83	0.0	0.0	1.0	2.0	2.0
Employee_Satisfaction_Score	100000.0	2.99	1.15	1.0	2.0	2.98	3.98	5.0

*Figure 14: Descriptive Statistics Table*

The table below presents a summary of the dataset's variables for key statistical metrics such as count, mean, standard deviation, minimum, maximum and quartiles (25%, 50%, and 75%). The mean values are particularly useful for quickly analyzing and understanding trends in the data. These statistics offer valuable insights into the distribution of numerical variables. However, not all variables in the dataset are purely numerical. For example, Employee\_ID serves as a unique identifier so its mean and other statistical values are not relevant for analysis. However, for the numerical variables such as “Monthly\_Salary” and “Work\_Hours\_Per\_Week” can be analyzed for meaningful insights. The table provides a clear overview of these distributions which helps us to quickly understand the trends , ranges and central tendencies in the dataset.

#### 4.3.8 Graphical View of Variables before Data Preprocessing

Exploratory Data Analysis (EDA) for each attribute will be performed on the raw data by using graphical methods before data preprocessing. Both categorical and numerical variables will be analyzed to uncover patterns in the data.

##### 4.3.8.1 Categorical Variables

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

categorical_cols = ['Department', 'Gender', 'Job_Title', 'Education_Level', 'Performance_Score', 'Promotions']

sns.set_style("white")

for col in categorical_cols:
    plt.figure(figsize=(12, 6))
    ax = sns.countplot(data=df, x=col, edgecolor="black", alpha=0.7, color="#D291BC")

    # Add count labels on top of bars
    for p in ax.patches:
        ax.annotate(f'{int(p.get_height())}', 
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', fontsize=10, color='black')

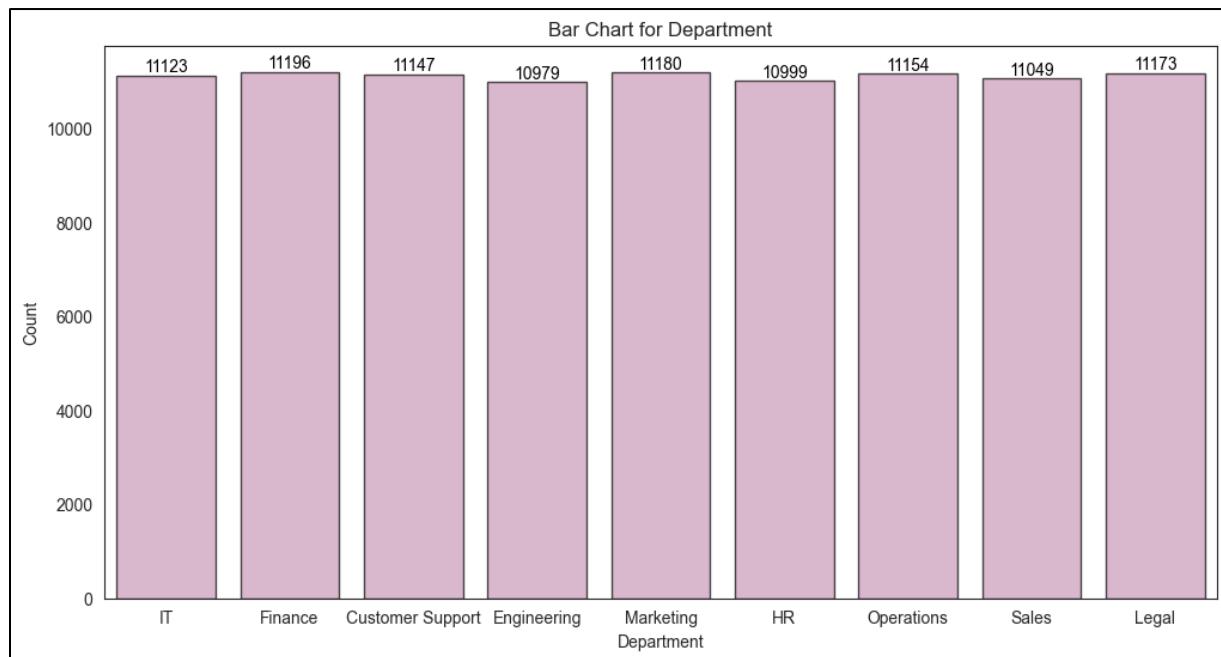
    plt.title(f"Bar Chart of {col}")
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.grid(False)
    plt.show()
```

*Figure 15:Source Code of Categorical Variables by Using Bar Chart*

Each categorical variable will be using bar charts to analyze. The sns.countplot() function is used to create bar plots for each categorical column to show the count of occurrences for each unique

category. The grid is disabled to maintain a neat appearance and each plot is displayed using plt.show() to ensure that each categorical variable is visualized separately for better analysis.

## **Department**



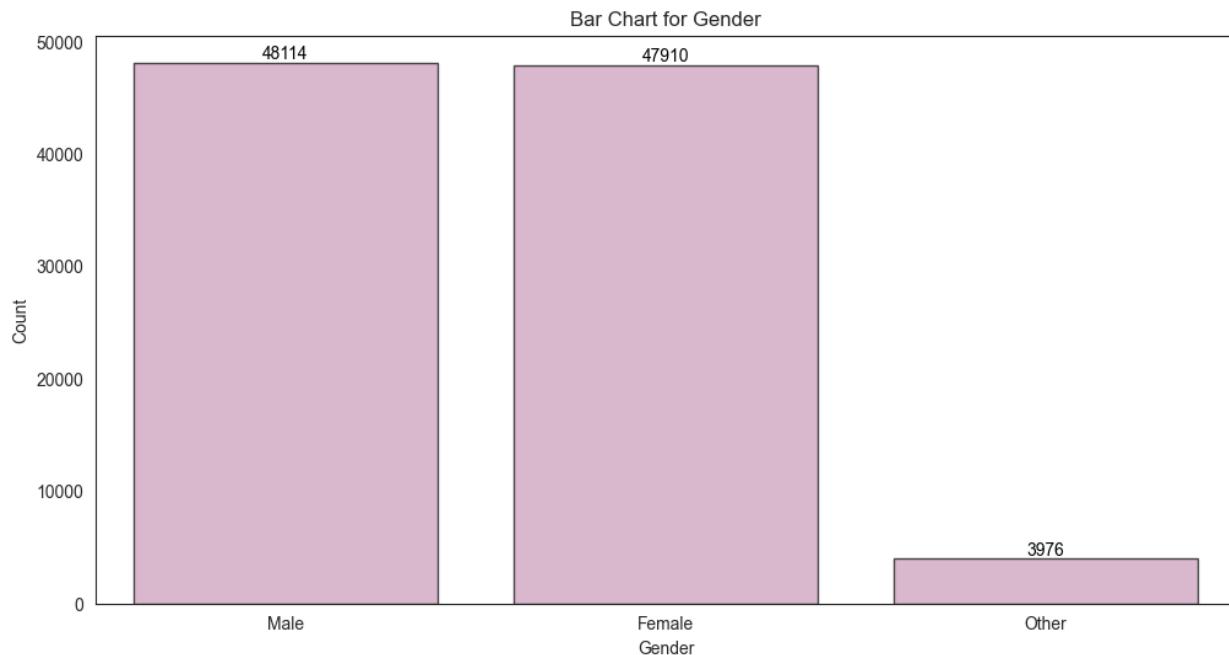
*Figure 16:Bar Chart for Department*

The bar chart visualizes the distribution of "Department" in the dataset. Each bar represents a different department with the height of the bar showing the number of employees in each category. The distribution appears to be fairly balanced as each department having a count of between 10,979 and 11,196 employees.

The Engineering and HR departments have slightly fewer employees compared to others, while Marketing and IT departments have slightly higher employee counts. However, these differences are minimal and show that the workforce is evenly distributed across departments.

The company maintains a structured workforce allocation based on its numerical employee distribution between departments which indicates a comparable number of employees. The balanced workforce distribution enables efficient planning of resources and HR decision making during data preprocessing and further analysis.

## Gender

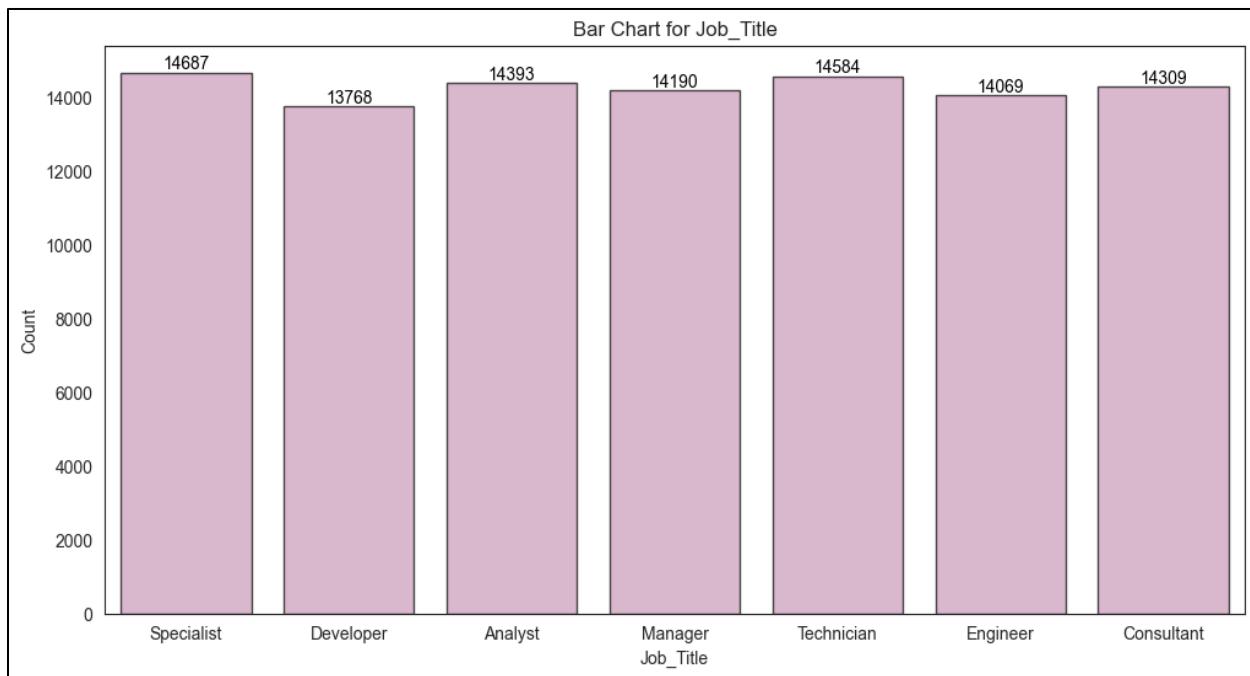


*Figure 17: Bar Chart for Gender*

The bar chart shows "Department" variables with each bar represents the majority of employees are Male (48,114) and Female (47,910). However, the "Other" gender category has a significantly lower count (3,976) and suggests that the dataset is highly imbalanced in terms of gender diversity, with Male and Female employees dominating the workforce while employees identifying as "Other" are underrepresented.

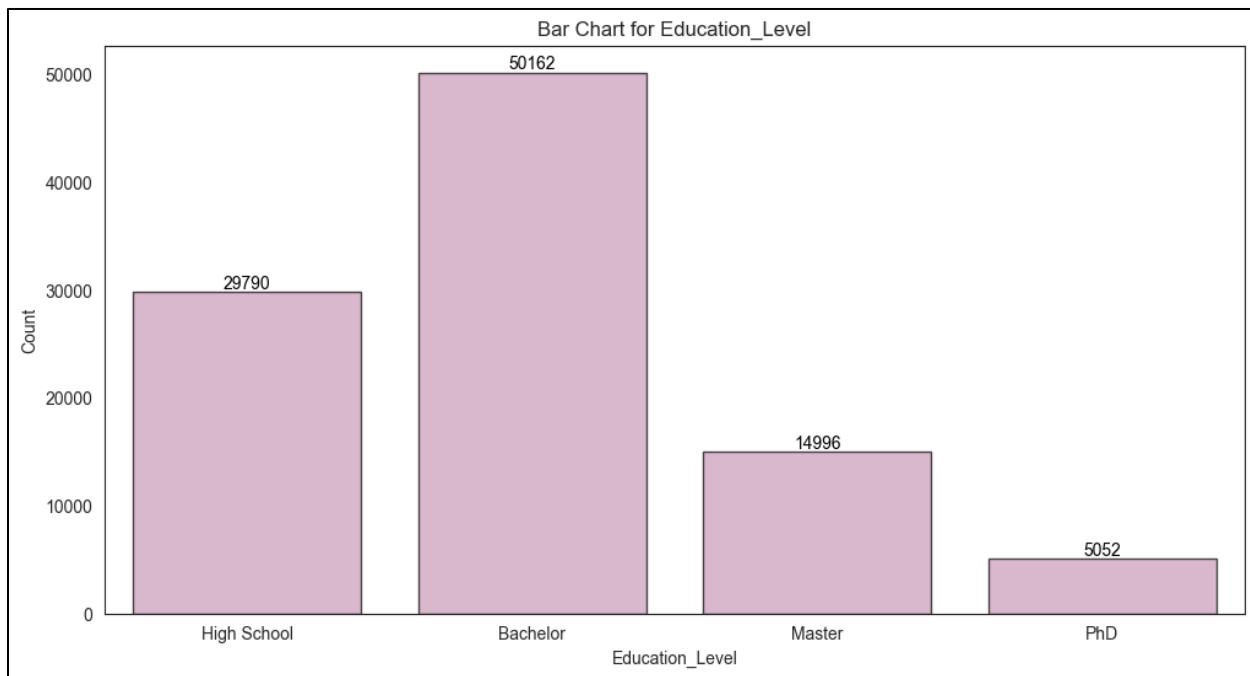
From an Exploratory Data Analysis (EDA) perspective, this imbalance should be considered during data preprocessing to ensure fairness in gender-based analysis. If gender is used in predictive modeling or workforce analysis, techniques like oversampling the minority class or applying weighting strategies might be necessary to prevent biases in machine learning models and HR-related decision-making.

## Job Title



*Figure 18: Bar Chart for Job\_Title*

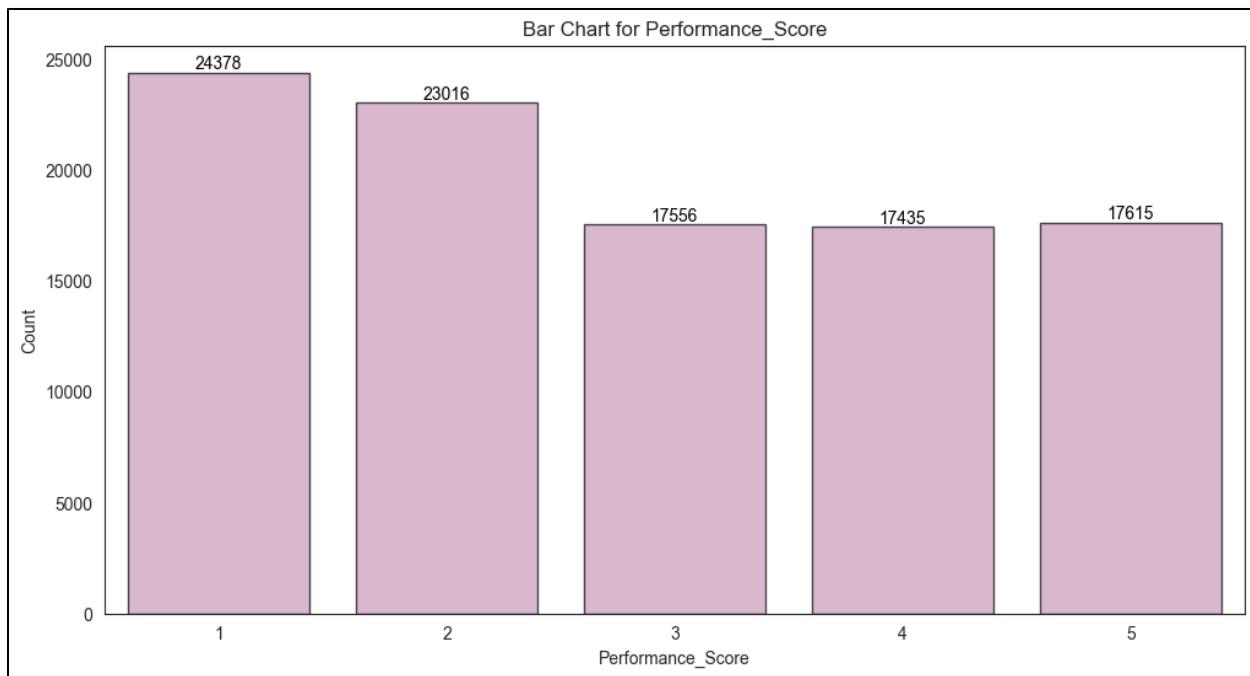
The bar chart displays the distribution of different job titles which include Specialist, Developer, Analyst, Manager, Technician, Engineer and Consultant. The count for each job title is consistent with values ranging from 13,768 to 14,687 which show a balanced distribution across the categories. The highest count is for Specialist with a count of 14,687, while the lowest count is for Consultant with count of 13,768. This uniformity suggests that there are no significant imbalances or skewness in the data regarding job titles. This characteristic can help ensure that the dataset is relatively stable for further analysis and modeling without any need for special preprocessing due to uneven class distributions.

**Education Level**

*Figure 19:Bar Chart for Education\_Level*

The bar chart shows the distribution of “Education\_Level” variable which showing that most employees hold a Bachelor's degree with 50,162 numbers in this category. High School graduates follow with 29,790 employees and are at the second most common education level. There is a noticeable drop in the number of employees with higher education with 14,996 holding a Master's degree and only 5,052 employees having a PhD. This indicates that the dataset is predominantly made up of employees with a Bachelor's degree while fewer people hold advanced degrees such as a Master's or PhD which shows that higher levels of education are less common in this dataset.

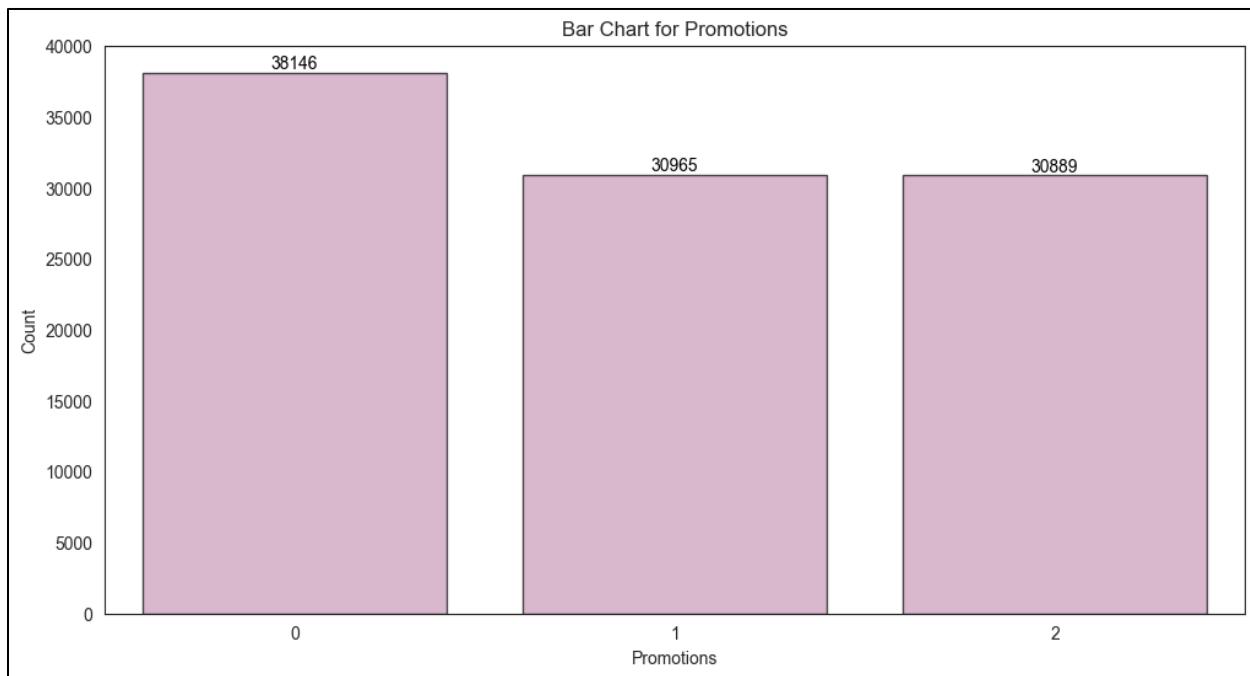
## **Performance Score**



*Figure 20:Bar Chart for Performance\_Score*

The bar chart displays the distribution of "Performance\_Score" ranging from 1 to 5. A score of "1," representing low performance has the highest count at 24,378 while a score of "5," representing high performance has the lowest count at 17,435. The counts for performance scores 2, 3, and 4 fall between 17,556 and 23,016. This indicates a slightly left-skewed distribution where lower performance scores are more common in the dataset.

## Promotions



*Figure 21:Bar Chart for Promotions*

The bar chart displays the distribution of "Promotions" across three categories which are 0, 1, and 2 promotions. The counts for each category range from 30,889 to 38,146. The category with the highest count is for employees with 1 promotion, meaning they received a promotion during their tenure with a total of 30,965 employees. The count for 0 promotions is 38,146 which indicates that these employees did not receive any promotions during their tenure, while the count for employees with 2 promotions is 30,889. This indicates a balanced distribution of promotions across the dataset with minimal variation between the categories and shows that promotions are evenly distributed among employees. However, the data also shows that a larger number of employees did not receive any promotion compared to those who received one or more.

#### 4.3.8.2 Numerical Variables

```

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

numerical_cols = [
    'Age', 'Years_At_Company', 'Monthly_Salary', 'Work_Hours_Per_Week',
    'Projects_Handled', 'Overtime_Hours', 'Sick_Days', 'Remote_Work_Frequency',
    'Team_Size', 'Training_Hours', 'Employee_Satisfaction_Score'
]

sns.set_style("whitegrid")

for col in numerical_cols:
    plt.figure(figsize=(12, 7))

    counts, bins, patches = plt.hist(df[col], bins=10, edgecolor="black", alpha=0.7, color="#D291BC")
    bin_centers = (bins[:-1] + bins[1:]) / 2
    plt.plot(bin_centers, counts, marker="o", linestyle="--", color="#C21E56", label=f"{col} Distribution", linewidth=2)

    # Counts & percentages
    total = counts.sum()
    for count, patch in zip(counts, patches):
        height = patch.get_height()
        if height > 0:
            plt.text(patch.get_x() + patch.get_width()/2, height + 100,
                     f"{int(count)}\n({count/total:.1%})",
                     ha="center", va="bottom", fontsize=10, color="black")

    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.title(f"Histogram of {col}")
    plt.legend()
    plt.ylim(0, max(counts) * 1.2)
    plt.grid(False)
    plt.show()

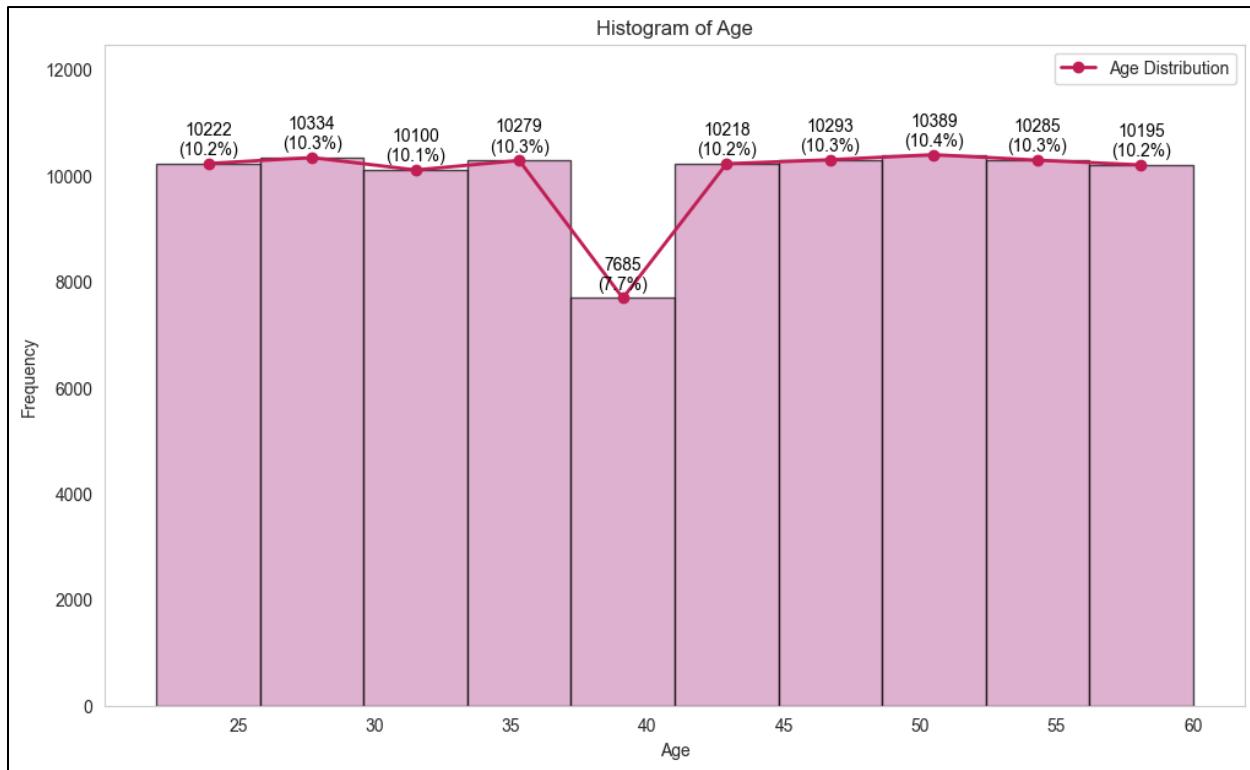
```

*Figure 22:Source Code of Numerical Variables by using Bar Chart*

Each numerical variable will be using histogram to analyze. The plt.hist() function is used to plot the distribution of each numerical column by dividing the data into 10 bins. This binning choice allows for a more detailed view of the data distribution although there may be slight overlap in the positions of adjacent bins due to the continuous nature of the data. The plt.plot() function overlays a line on the histogram to provide a smoother overview of the distribution. Besides , count values and percentages are displayed on the bars with plt.text() to provide insights into the exact frequency and proportion of data points in each bin.

Red circle markers are added to specific bins using plt.scatter() to highlight key data points, trends or outliers . This helps identify important ranges in the distribution. By analyzing the histograms, we can gain insights into the distribution of each numerical variable, detect skewness or normality, and identify the presence of outliers or clusters. By using a histogram for analysis, it provides insights into the distribution, skewness and normality which is helping to identify outliers and inform further steps such as data preprocessing.

## Age



*Figure 23: Histogram for Age*

The histogram displays the distribution of "Age" which performs a uniform spread across most age groups with a slight drop around 40 years old. The x-axis represents the age group while y-axis shows the frequency of employees. The highest frequency occurs in the age range of 50-55 years old with 10,389 employees (10.4% of the total) while the lowest frequency is observed to be around 40 years old with 7,685 employees (7.7% of the total). Most age groups are around 10% of the total population and this indicates a balanced distribution overall. There are no significant outliers or skewness across the distribution age ranges.

## Years At Company

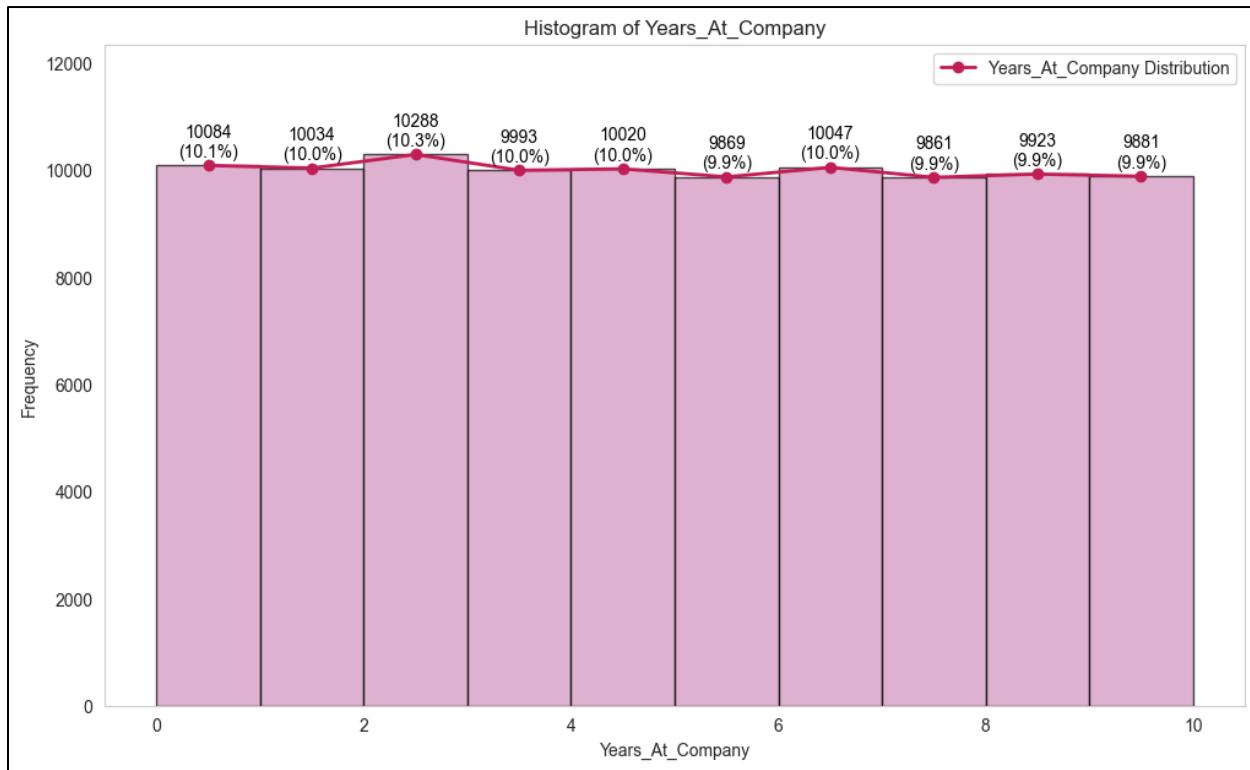
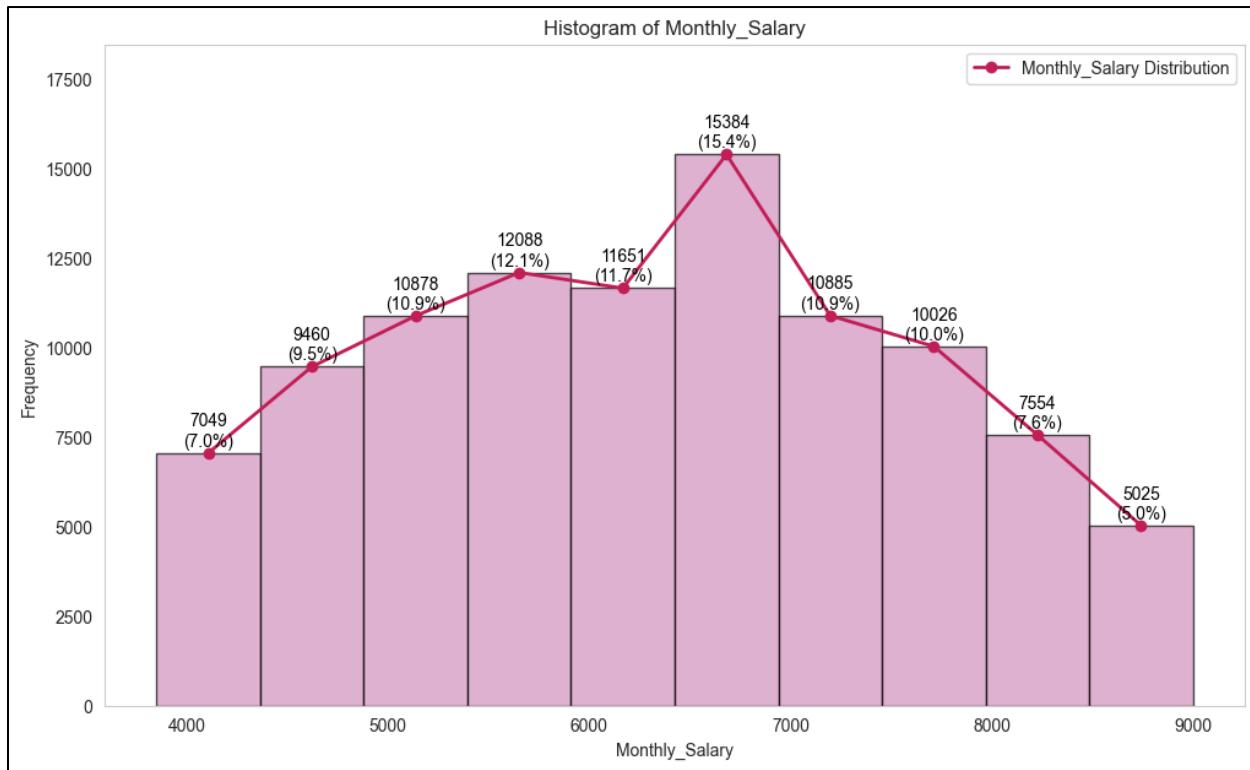


Figure 24: Histogram for Years\_At\_Company

The histogram displays the distribution of "Years\_At\_Company" which represents the number of years for employees who have spent at the company ranging from 0 to 10 years. The distribution is consistent with frequencies from 9,861 to 10,288. The highest count is for employees who have worked for 2 years with 10,288 employees and is making up 10.3% of the total. The lowest count is for employees with 10 years at the company which is 9,861 with 9.8% of the total. The frequencies for other years range from 9,869 to 10,084. This indicates that there is an even distribution with only minimal variations in the number of employees across different years of tenure at the company.

## Monthly Salary



*Figure 25: Histogram for Monthly\_Salary*

The histogram of “Monthly\_Salary” shows the distribution of salaries among employees. The highest frequency occurs in the 6,500-7,000 salary range with approximately 15,384 employees (15.4% of the total) and this indicates that most employees earn salaries within this range. The frequency gradually decreases as the salary increases or decreases, with the lowest frequency observed in the 8,000-8,500 range, where only 5,025 employees (5.0% of the total) fall. The distribution appears to be relatively symmetric with a slight right skew because the frequency decreases more sharply for higher salary ranges. According to this distribution, the majority of employees have monthly salaries around the middle range with a smaller percentage of employees earning lower or higher salaries.

## Work Hour Per Week

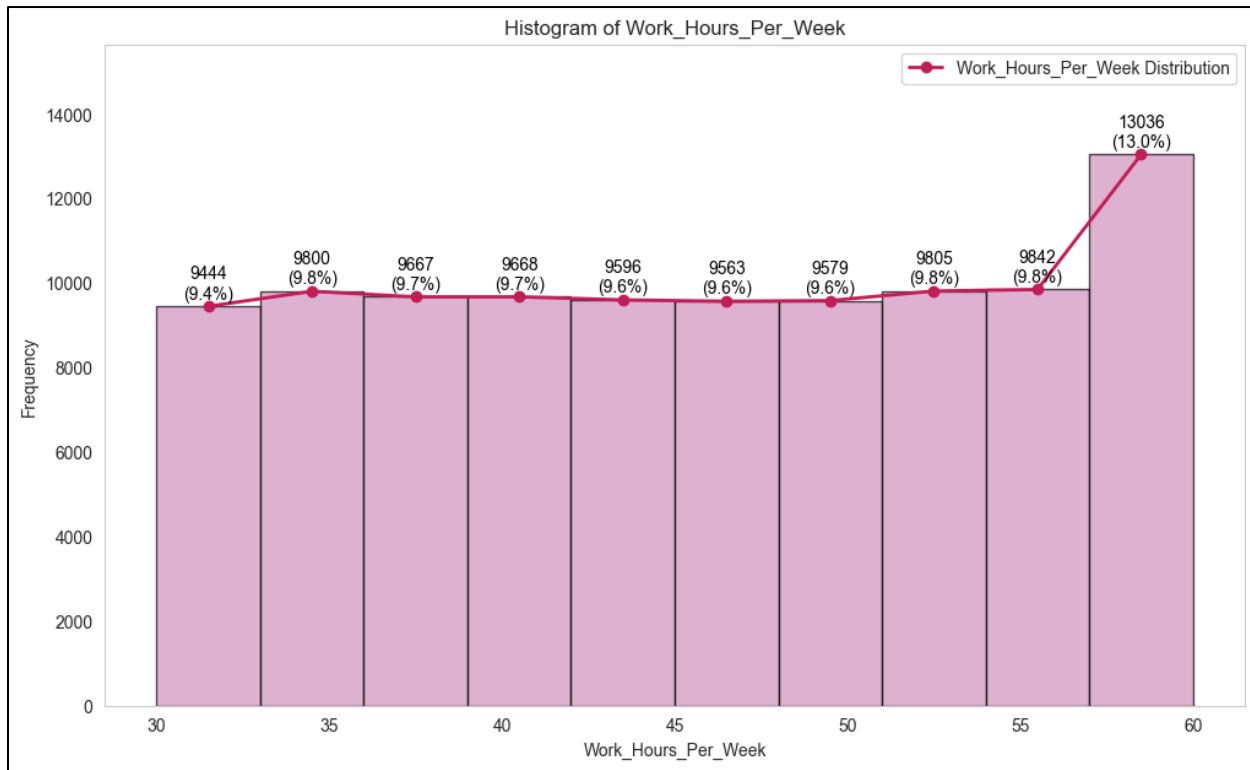


Figure 26: Histogram for Work\_Hour\_Per\_Week

The histogram of "Work\_Hours\_Per\_Week" shows the distribution of work hours among employees. The x-axis represents the work hours per week, ranging from 30 to 60 hours, while the y-axis indicates the frequency of employees within each respective range. The highest frequency occurs in the 60-hour range, with 13,036 employees (13%) and indicates a significant proportion of employees working longer hours compared to other ranges. The frequencies for the 30-55 hour ranges remain relatively consistent, with about 9,444 to 9,842 employees per range (9.4% to 9.8%) of the total workforce. This suggests that most employees work a balanced number of hours. However, the histogram also shows a right-skewed distribution, with the 60-hour category standing out as a major peak. The data indicates that multiple employees exceed their standard work week by considerable hours. The overall distribution provides insights into the typical work hours where the concentration of employees in the 30-55 hours range and the spike at 60 hours.

## Project Handled

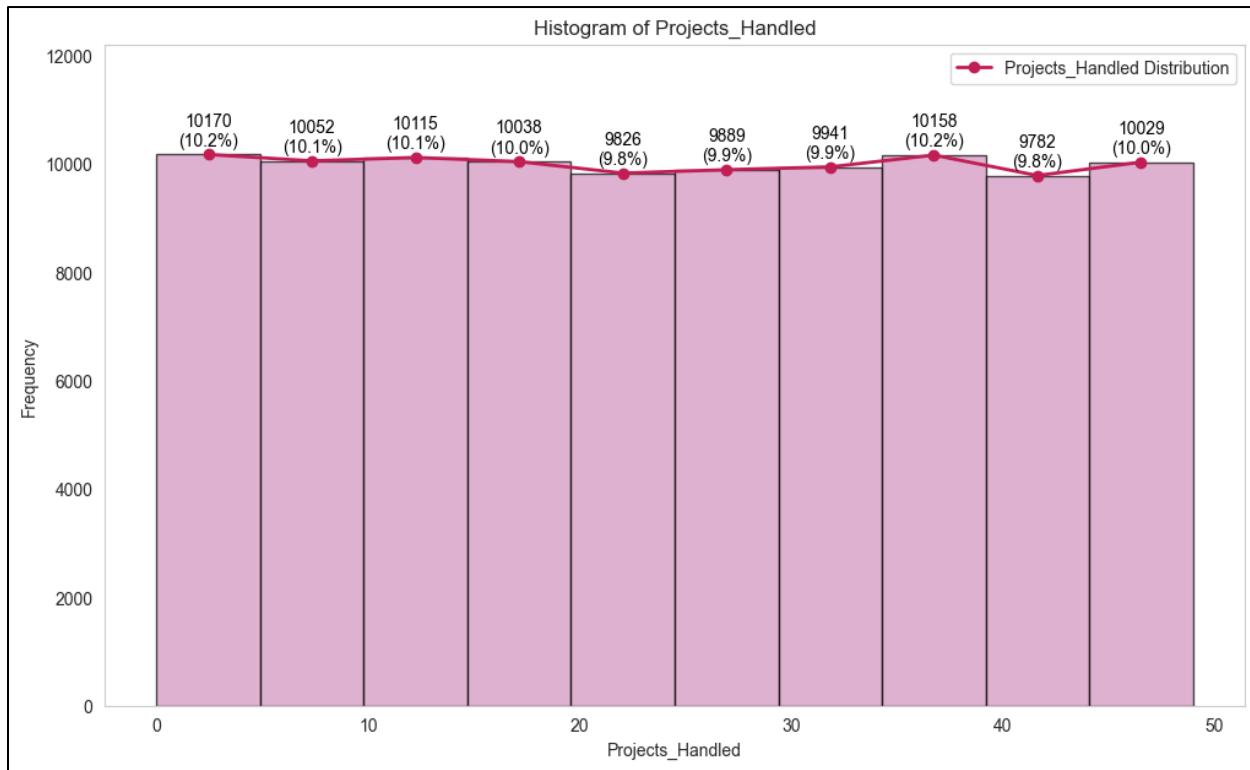


Figure 27: Histogram for Project\_Handled

The histogram of "Projects\_Handled" displays a consistent distribution of the number of projects handled by employees. The x-axis represents the number of projects handled from 0 to 50, while the y-axis shows the frequency of employees in each respective range. The distribution shows a uniform distribution by maintaining frequencies at approximately 10,000 employees for each range in number of projects handled. The frequency remains stable throughout the distribution, indicating that most employees handle a similar number of projects with little fluctuation across the different ranges. The overall distribution appears even and shows that there is no significant bias toward employees managing fewer or more projects. The histogram shows no significant outliers and performs a uniform distribution without extreme values or anomalies in the number of projects assigned to employees.

## Overtime Hours

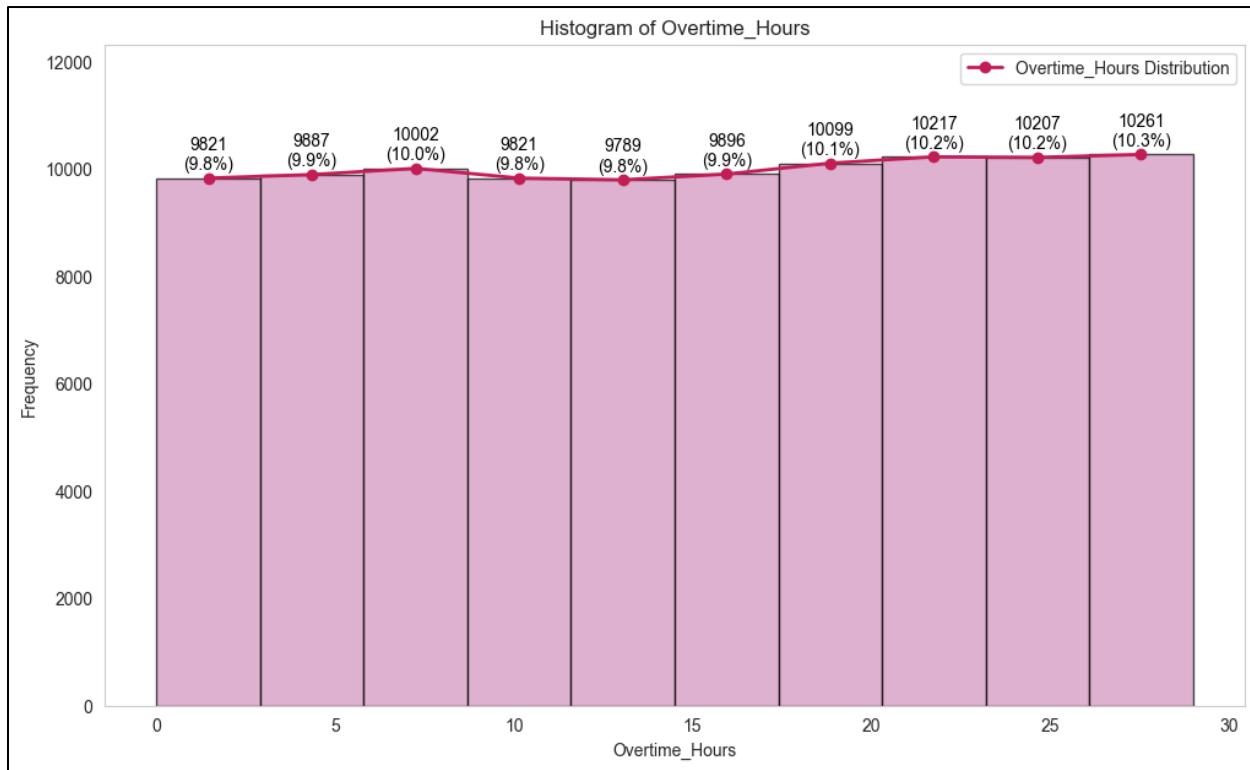


Figure 28: Histogram for Overtime\_Hours

The histogram of “Overtime\_Hours” displays a consistent distribution across different overtime ranges from 0 to 30 hours per year. The x-axis represents the number of overtime hours per year, while the y-axis shows the frequency of employees in each overtime range. The distribution is nearly uniform with each bin containing around 10,000 employees and the percentages above the bars hovering around 10% for most ranges. The highest frequency occurs in the 0-hour range, with 10,217 employees indicating a significant portion of employees who do not work overtime. The lowest frequency is around 20 hours overtime work with 9,789 employees. This distribution suggests that the number of employees working overtime is spread out evenly across the different overtime hours ranges with no clear bias toward employees working minimal or excessive overtime. The histogram reveals no significant outliers or skewness.

## Sick Days

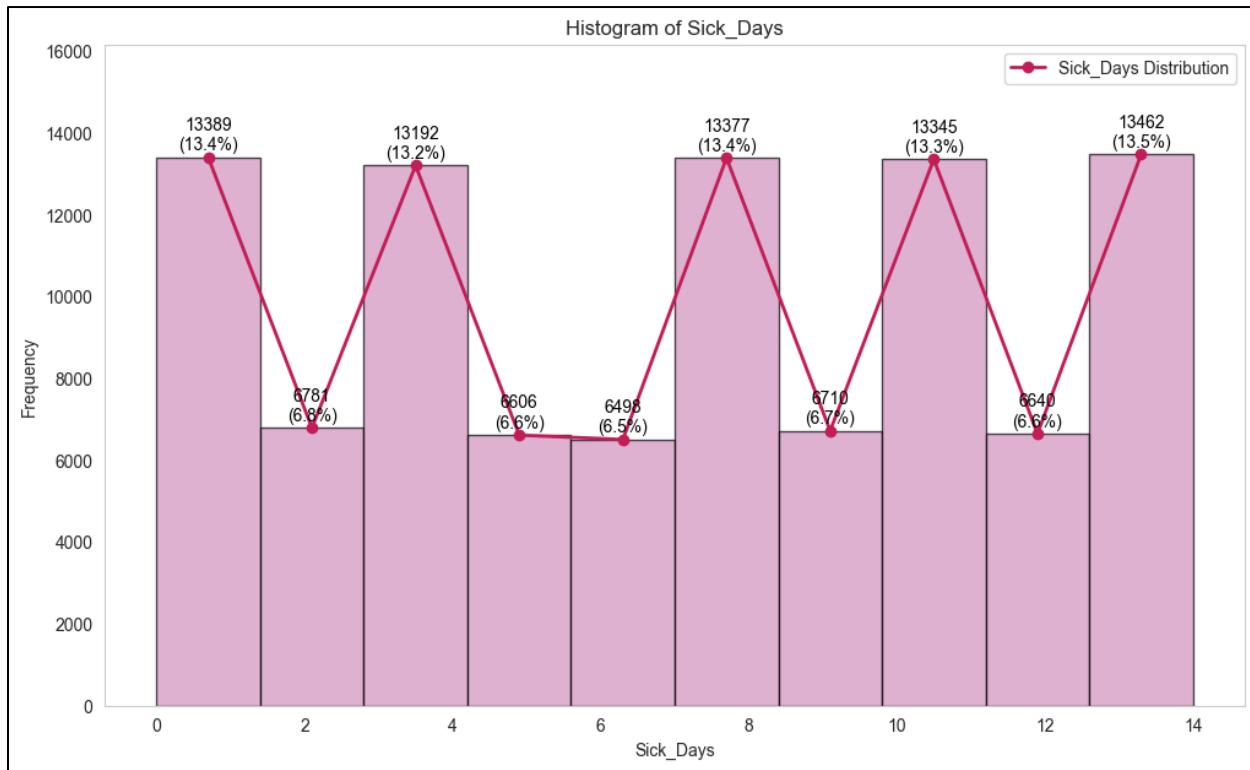
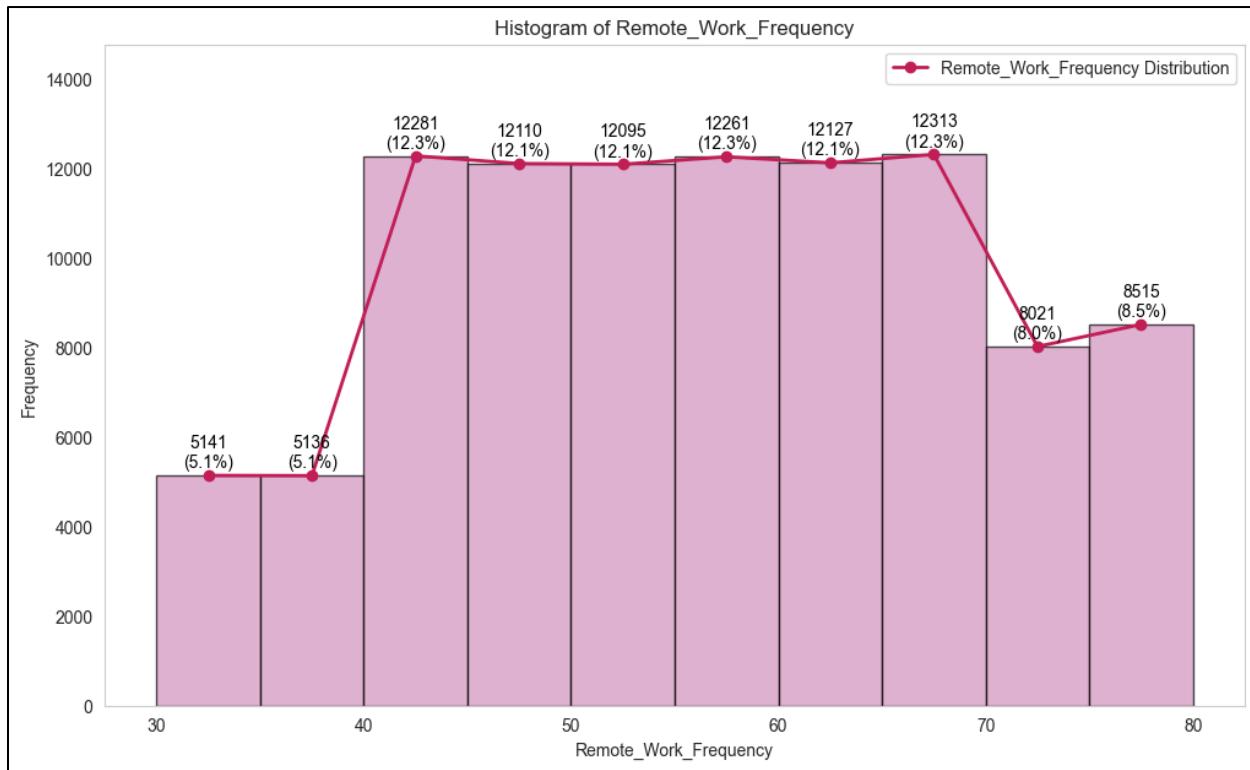


Figure 29: Histogram for Sick\_Days

The histogram of "Sick\_Days" displays a consistent distribution of sick days taken by employees which ranges 0 to 14 sick days per year. The highest frequency occurs in the 12-14 sick days range with 13,462 employees (13.5%) while the lowest frequencies are observed in some bins such as 2 and 6 sick days, with 6.6% of employees in each. Overall, the distribution shows that while most employees fall within the 0-4 sick days range, there is still a notable portion of employees who report taking 12 to 14 sick days. The histogram reveals no significant outlier with slight fluctuations in the frequency of employees across the different sick day ranges.

## Remote Work Frequency

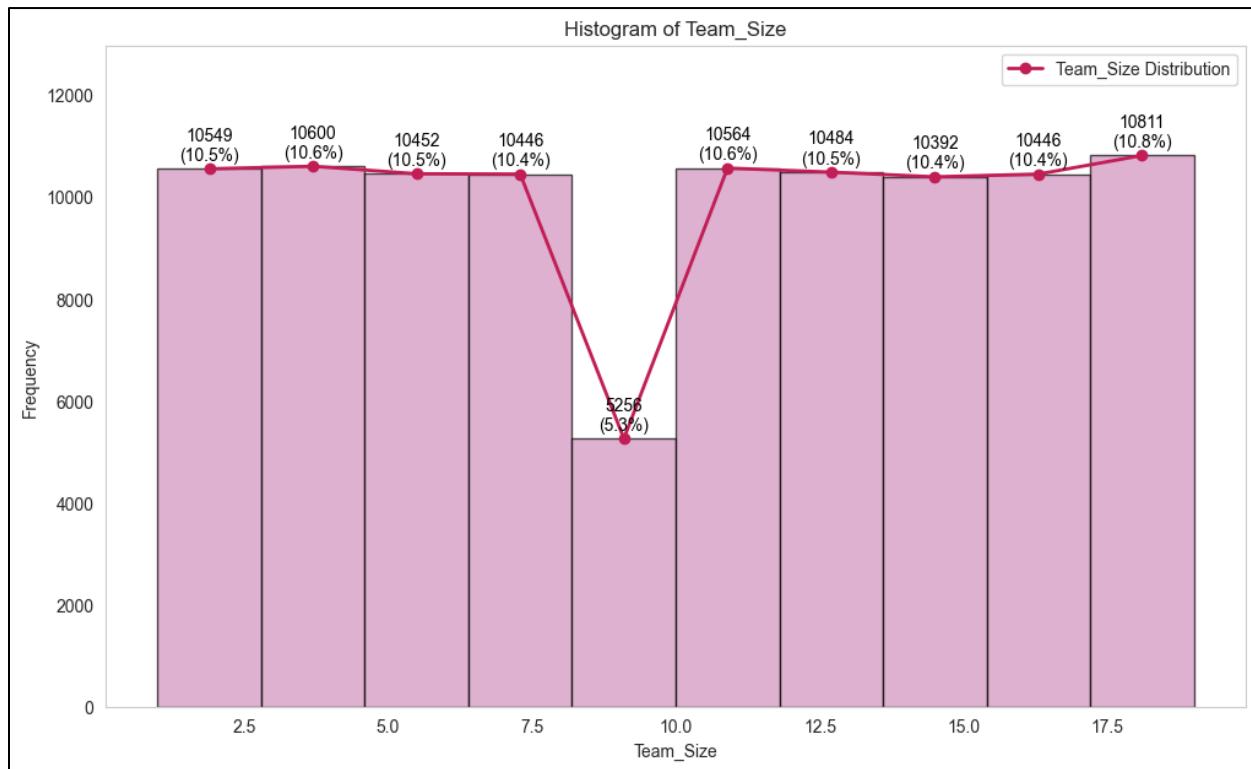


*Figure 30: Histogram of Remote\_Work\_Frequency*

The histogram shows the distribution of employees' Remote\_Work\_Frequency percentage which ranging from 30% to 80%. The highest frequencies are seen in the 40%–45%, 55%–60%, and 65%–70% bins with each making up about 12.3% of the total. This suggests that many employees tend to have remote work schedules around those ranges and may possibly reflect common hybrid work arrangements such as 2 to 3 days remote work per week. Besides, the lowest frequencies are at the 30% and 35% bins with each at around 5.1%, indicating fewer employees work remotely at these lower levels. This might suggest that very minimal remote work is either not preferred, not offered or not practical for certain job roles.

Meanwhile, the drop at 70%–75% (8.0%) and slight increase again at 75%–80% (8.5%) could imply that fewer employees have very high remote frequency possibly due to job requirements needing physical presence or company policies limiting full-remote option ranges.

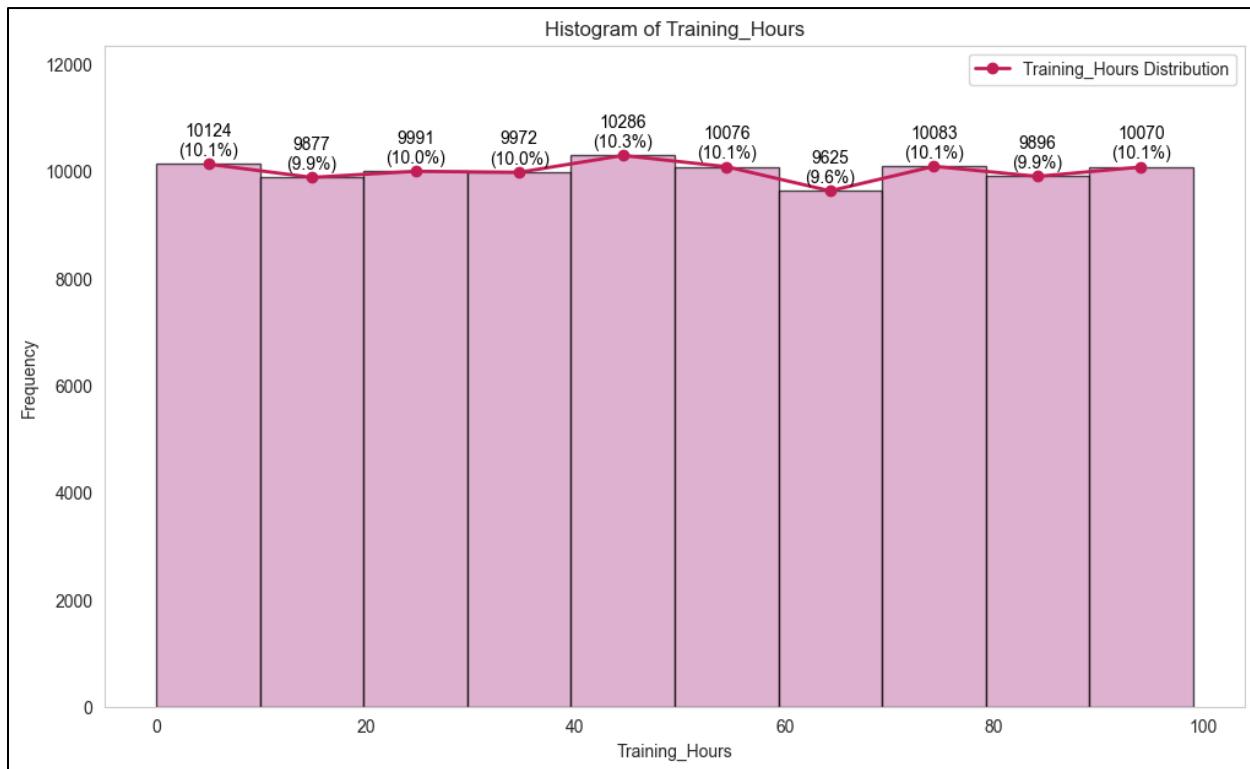
## Team Size



*Figure 31: Histogram for Team\_Size*

The histogram of "Team\_Size" displays the distribution of the number of people in employee teams. From the graph can observe that there is a sharp decrease around team size 9, where only 5.3% of the total employees fall into this category. After this drop, there is a slight recovery in the frequencies for team sizes 11, 13 and 15. However, the overall distribution of team sizes doesn't seem to follow a clear pattern. This could be due to the different projects or departments having different needs. For example, while 5-member teams might be more common, larger teams with 19 members may also have a similar number of employees because it depends on the nature of the work. This suggests that the team sizes are likely to be influenced by departmental requirements or project complexity, where some roles or departments may not need as many people to function effectively. Therefore, the distribution may be influenced by organisational demands rather than just reflecting a broad trend.

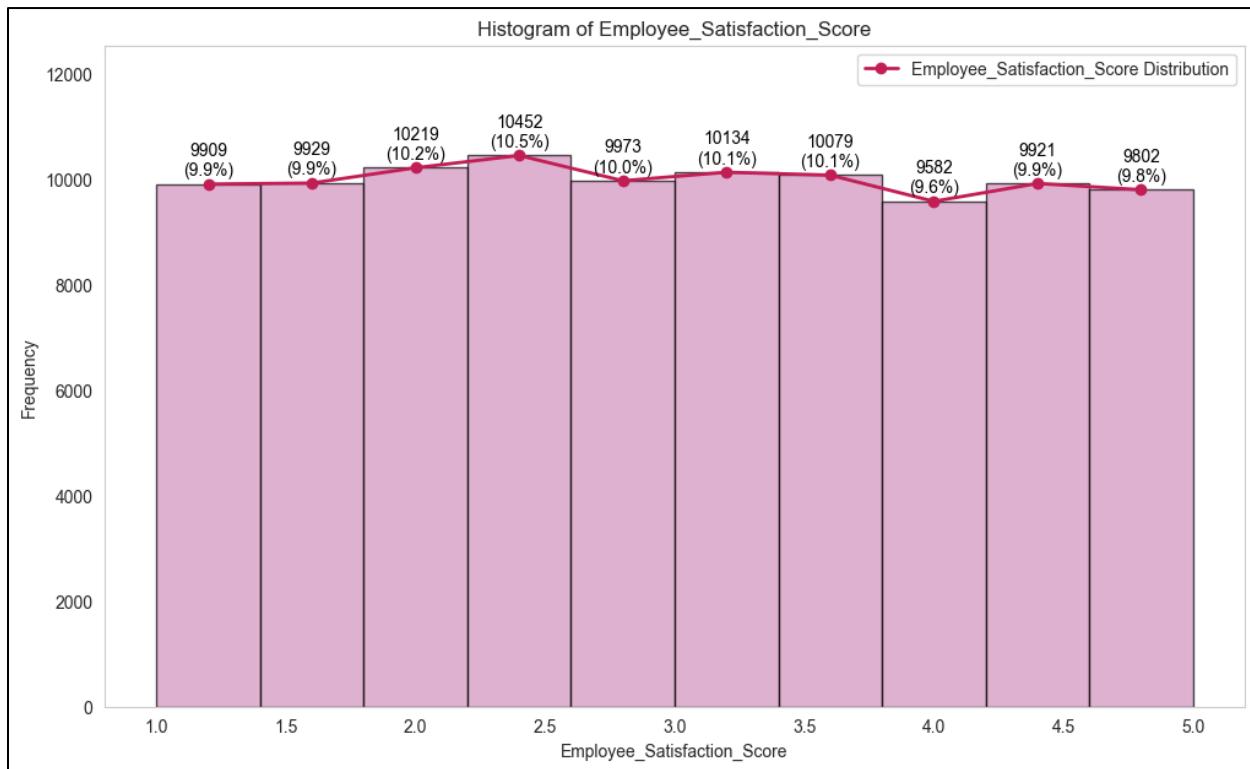
## Training Hours



*Figure 32: Histogram for Training\_Hours*

The histogram of "Training\_Hours" shows uniform distribution of training hours across employees. The x-axis represents the number of trainings hours from 0 to 100, while the y-axis displays the frequency of employees who fall into each respective training hour range. The frequency remains relatively stable across the different ranges of training hours, with each range consistently having around 9,625 to 10,286 employees. The distribution of training hours is uniform, with frequencies consistent across different ranges and percentages ranging from 9.6% to 10.3%. This suggests that the number of training hours assigned to employees is consistent without significant bias toward either lower or higher amounts. Overall, the histogram indicates that employees receive similar training hours with no extreme values or outliers.

## Employee Satisfaction Score



*Figure 33: Histogram for Employee\_Satisfaction\_Score*

The histogram of "Employee\_Satisfaction\_Score" shows a consistent distribution with frequencies remaining consistent across different score ranges which ranging from 9.6% to 10.5% with each range showing a count of around 9,582 to 10,452 employees. This uniformity suggests that the employee satisfaction scores are evenly distributed. The consistency shows that different satisfaction scores indicate that employees' satisfaction is spread out relatively equally. The histogram shows a balanced distribution of satisfaction levels within the organization. Employees display both lower and higher satisfaction, indicating a variety of satisfaction levels within the workforce.

### 4.3.9 Target Variable - Resigned

The “Resigned” variable is set as the target variable to predict the employee churn.

```

import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("Extended_Employee_Performance_and_Productivity_Data.csv")

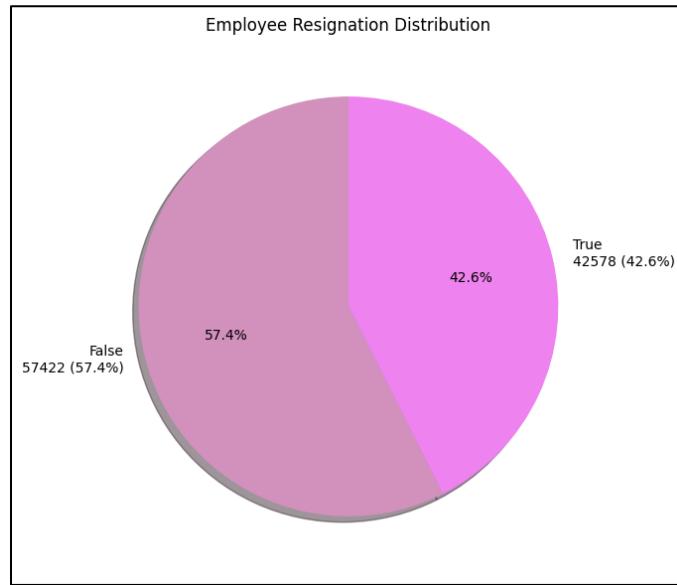
# Count the number of employees who resigned and did not resign
resigned_counts = df["Resigned"].value_counts()

# Counts and Percentages
labels = [f"{category}\n{count} ({count/sum(resigned_counts)*100:.1f}%)"
          for category, count in resigned_counts.items()]
plt.figure(figsize=(6, 7))
plt.pie(resigned_counts, labels=labels, autopct='%1.1f%%',
        colors=["#D291BC", "#EE82EE"], startangle=90, shadow=True)
plt.title("Employee Resignation Distribution")
plt.axis('equal')

print(resigned_counts.to_string(index=True))
print(f"Total Employees: {resigned_counts.sum()}")
plt.show()

```

*Figure 34: Source Code of Target Variable*



*Figure 35: Pie Chart of Target Variable*

Figure above displays the pie chart of employee resignation distribution which shows a significant class imbalance. There are 90% of employees (89,990) who have not resigned, whereas only 10% (10,010) have resigned. This imbalance in the dataset will be addressed during the data preprocessing phase to ensure accurate predictive modeling.

### 4.3.10 Correlation Heatmap of Employee Variables before Data Preprocessing

```
# Convert categorical to numerical
df["Resigned"] = df["Resigned"].astype(int)

numeric_df = df.select_dtypes(include=[np.number])
correlation_matrix = numeric_df.corr()
plt.figure(figsize=(12, 8)) # Set figure size
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title("Correlation Heatmap of Employee Attributes")
plt.show()
```

Figure 36: Source Code of Correlation Heatmap

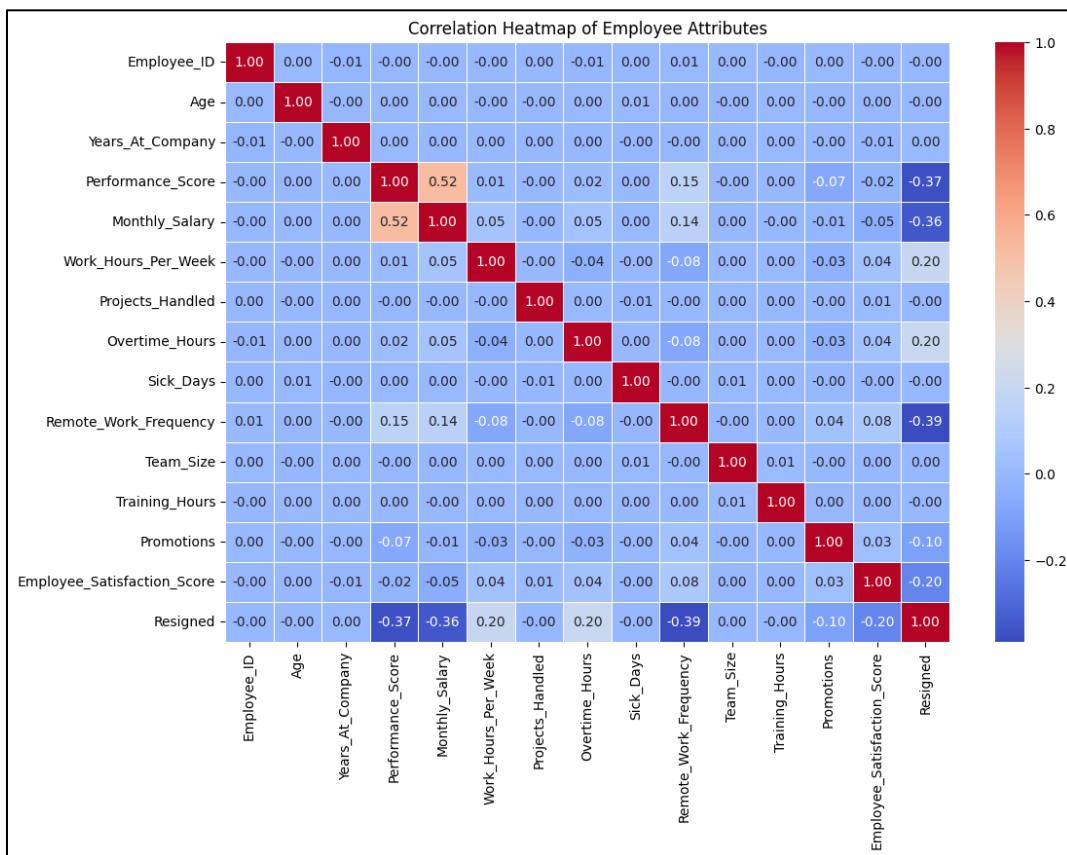


Figure 37: Correlation Heatmap

The heatmap above shows the correlation between various employee attributes. A moderate positive correlation (0.52) is observed between Performance\_Score and Monthly\_Salary which is suggesting that employees with better performance scores tend to receive higher salaries. Notably, Remote\_Work\_Frequency has a negative correlation with Resigned (-0.39) which has implying

that employees who work remotely more frequently are less likely to resign. A similar negative correlation exists between Employee\_Satisfaction\_Score and Resigned (-0.20), which correspond with the expectation that more satisfied employees are less likely to leave the company.

There are other correlations such as between Work\_Hours\_Per\_Week and Resigned (0.20) showing a slight trend where longer work hours might be linked to higher resignation rates. However, most variables in the dataset show only weak or negligible correlations which have suggested limited direct relationships.

Overall, the heatmap highlights a few moderate relationships between performance, salary, remote work and resignation while most other variables remain largely independent.

## 4.4 Data Preprocessing

### 4.4.1 Missing Values

```
# Count of duplicate rows
print("[Count of duplicate rows]")
print(df.duplicated().sum(), "\n")
```

*Figure 38: Source Code of Duplicate Rows*

```
[Count of duplicate rows]
0
```

*Figure 39: Count of Duplicate Rows*

The figure below presents the duplicate rows check which proves that the dataset contains 0 duplicate rows, and this indicates that all entries are unique.

```
missing_values = df.isnull().sum()
print(missing_values)
```

*Figure 40: Source Code of Missing Values*

Employee_ID	0
Department	0
Gender	0
Age	0
Job_Title	0
Hire_Date	0
Years_At_Company	0
Education_Level	0
Performance_Score	0
Monthly_Salary	0
Work_Hours_Per_Week	0
Projects_Handled	0
Overtime_Hours	0
Sick_Days	0
Remote_Work_Frequency	0
Team_Size	0
Training_Hours	0
Promotions	0
Employee_Satisfaction_Score	0
Resigned	0
Years_At_Company_Cat	0
<b>dtype:</b>	<b>int64</b>

*Figure 41: Output of Missing Values*

Additionally, the missing values check confirms that the dataset has no missing values to ensure that the data completeness and reliability for further analysis.

#### 4.4.2 Outliers

To detect potential outliers, different visualization techniques are used for numerical and categorical variables:

- Histograms are used for categorical variables to analyze their frequency distribution.
- Boxplots are used to identify outliers in numerical variables because it provides a clear representation of data distribution, median, quartiles and potential extreme values.

#### 4.4.2.1 Categorical Variables

```

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

categorical_cols = ['Department', 'Gender', 'Job_Title', 'Education_Level', 'Performance_Score', 'Promotions']

sns.set_style("white")

# Define outliers
threshold = 5

# Identify outliers
for col in categorical_cols:
    category_counts = df[col].value_counts()
    rare_categories = category_counts[category_counts < threshold].index.tolist()

    if rare_categories:
        print(f"Outliers in {col}: {rare_categories}")
    else:
        print(f"Outliers in {col}: None")

for col in categorical_cols:
    plt.figure(figsize=(12, 6))
    ax = sns.countplot(data=df, x=col, edgecolor="black", alpha=0.7, color="#D291BC")

    for p in ax.patches:
        ax.annotate(f'{int(p.get_height())}', 
                    (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', fontsize=10, color='black')

    plt.title(f"Bar Chart for {col}")
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.grid(False)
    plt.show()

```

*Figure 42: Source Code of Categorical Variables by using Bar Chart*

The source code is used to identify and analyze outliers in categorical variables such as Department, Gender, Job\_Title and Education\_Level. It creates bar charts for each variable to display the frequently of each category to allow visualizing the potential outliers in the data. All categorical variables are analyzed using bar charts.

```

Outliers in Department: None
Outliers in Gender: None
Outliers in Job_Title: None
Outliers in Education_Level: None
Outliers in Performance_Score: None
Outliers in Promotions: None

```

*Figure 43:Output of Outliers for Categorical Variables*

There are no outliers found for the categorical variables. Below will show the detailed analysis for each variable by using a bar chart to further prove that there are no outliers.

## Department

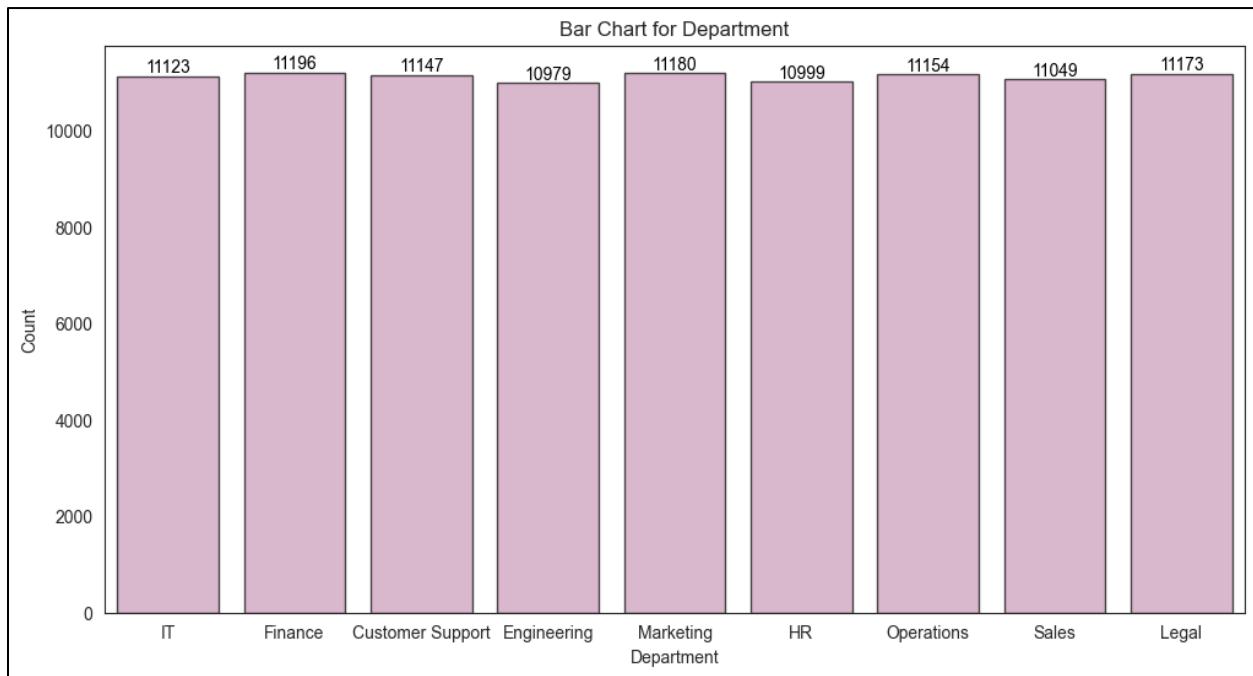
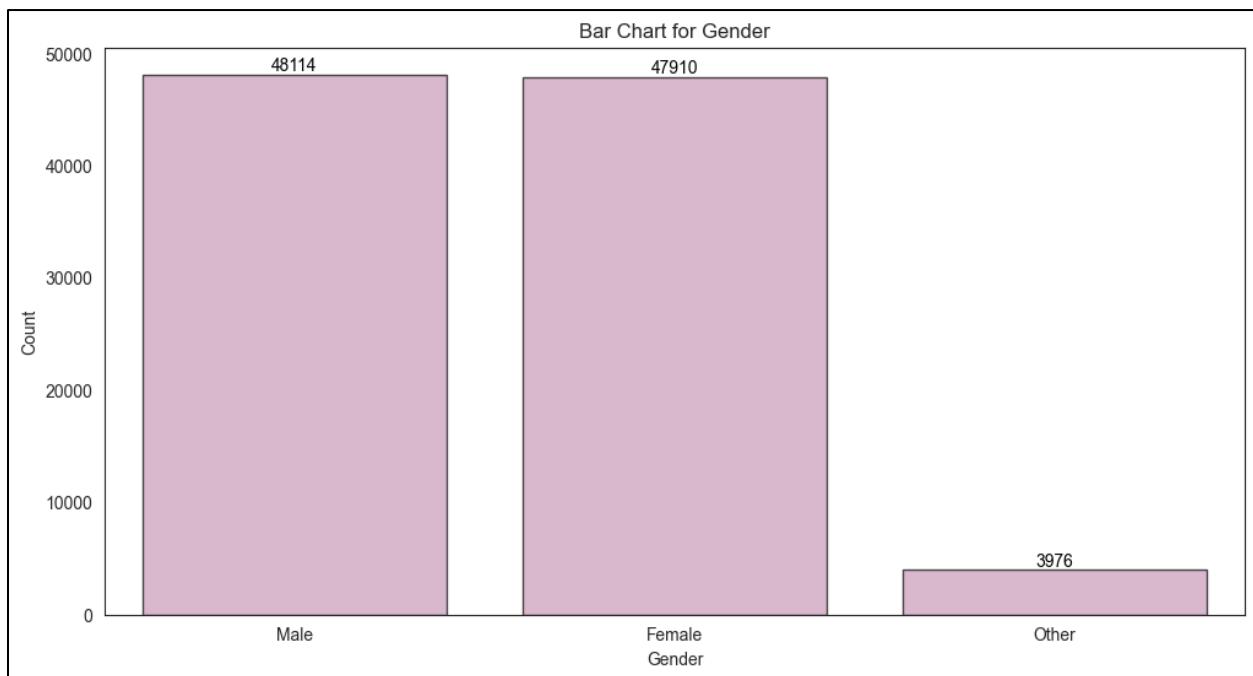


Figure 44: Bar Chart for Department

The bar chart for the “Department” variable clearly shows that the number of employees in each department is relatively consistent and there is no department showing an unusually high or low count. This supports the observation that there are no outliers and skewness are found for this variable because the distribution of employees across different departments is quite uniform.

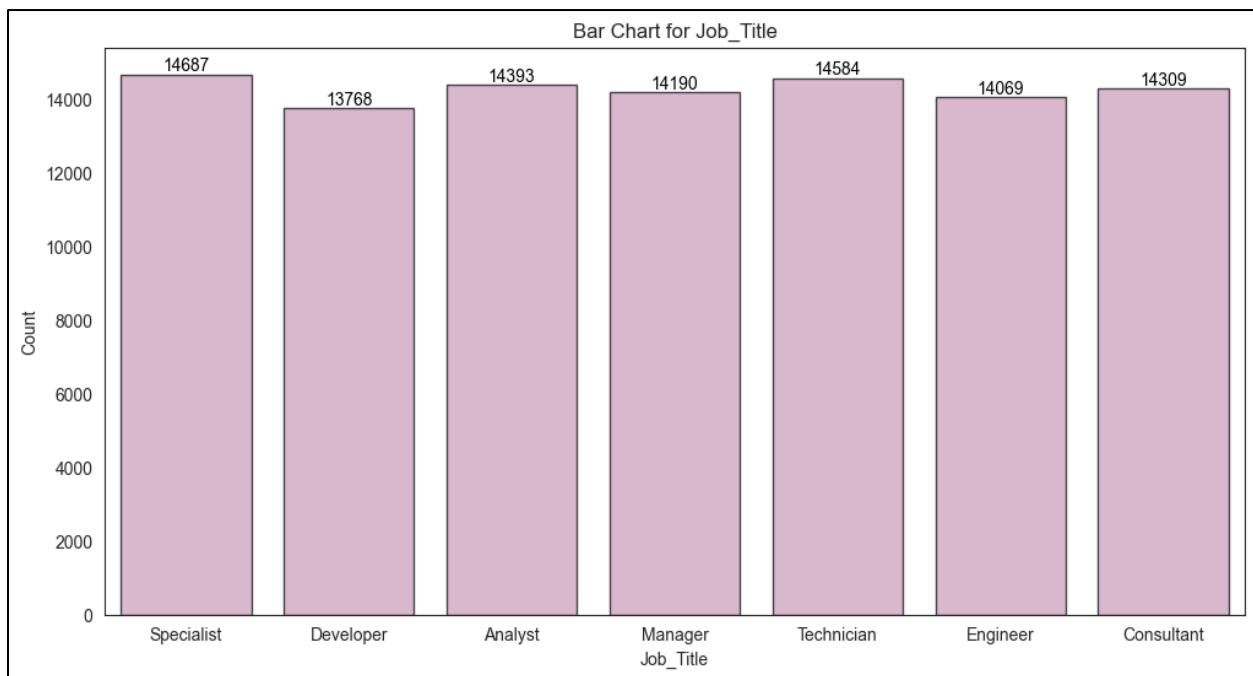
## Gender



*Figure 45: Bar Chart for Gender*

The bar chart for the “Gender” variable shows that the distribution is dominated by Male and Female categories with the counts of around 48,000 each. However, there is also a much smaller category “Other” with only about 3,976 counts. This indicates that the "Other" category is relatively rare compared to the Male and Female categories. While it could be considered a rare category, it is important to note that removing it might cause bias as it may represent valid gender identities. Therefore, the "Other" category will keep maintaining inclusivity and avoid potential misrepresentation in the dataset.

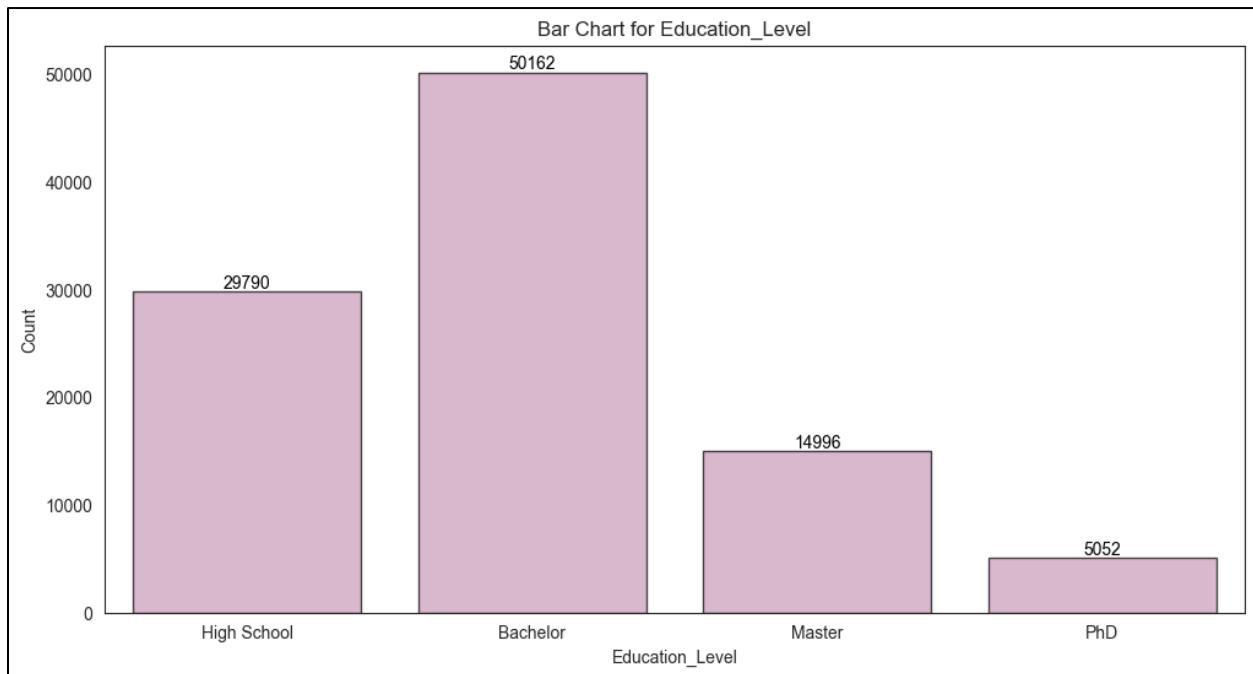
## **Job Title**



*Figure 46:Bar Chart for Job\_Title*

The bar chart above for “Job\_Title” displays the frequency distribution of different job titles in the dataset. Each category has a similar count with around 14,000 and it is clearly shown that no job title is significantly underrepresented. All job titles are well-distributed and have no unusual job roles present in this dataset. Therefore, no outliers and skewness are found.

## Education Level

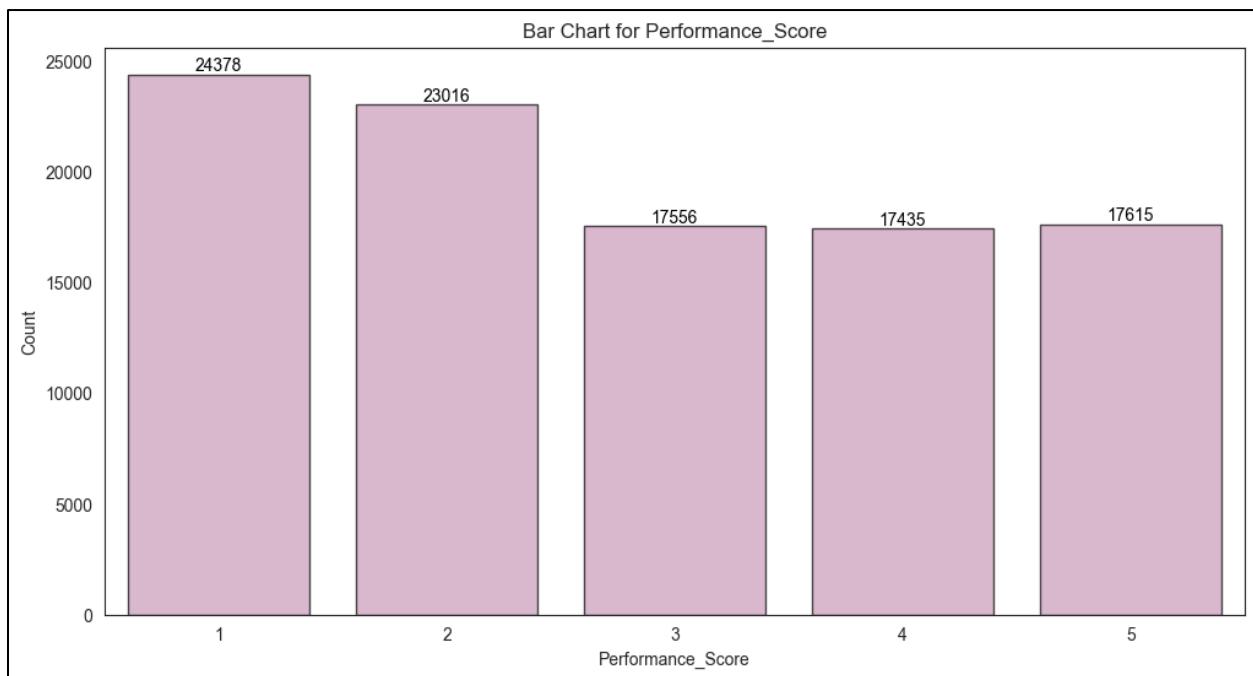


*Figure 47: Bar Chart for Education\_Level*

The bar chart for “Education\_Level” shows the frequency distribution of different education levels in the dataset. As can be seen from the bar chart, the highest is bachelor with 50,162 while PhD has the lowest count with 5052 which makes it a potential outlier. The significant drop in numbers from Bachelor's to Master's and PhD indicates that higher education levels are less common in this dataset. The distribution of employees by education level is positively skewed with most employees holding a Bachelor's degree. There are fewer employees with Master's degrees and PhDs and showing that most of the employees fall within the Bachelor's category.

However, it is important to note that the PhD category represents employees with advanced education and removing it could lead to a loss of valuable insights especially if education level plays a role in predicting employee performance, satisfaction or turnover. Therefore, since the PhD category reflects a valid and distinct group within the dataset, it should be retained to maintain the completeness and accuracy of the analysis to avoid misrepresenting the distribution of education levels within the workforce.

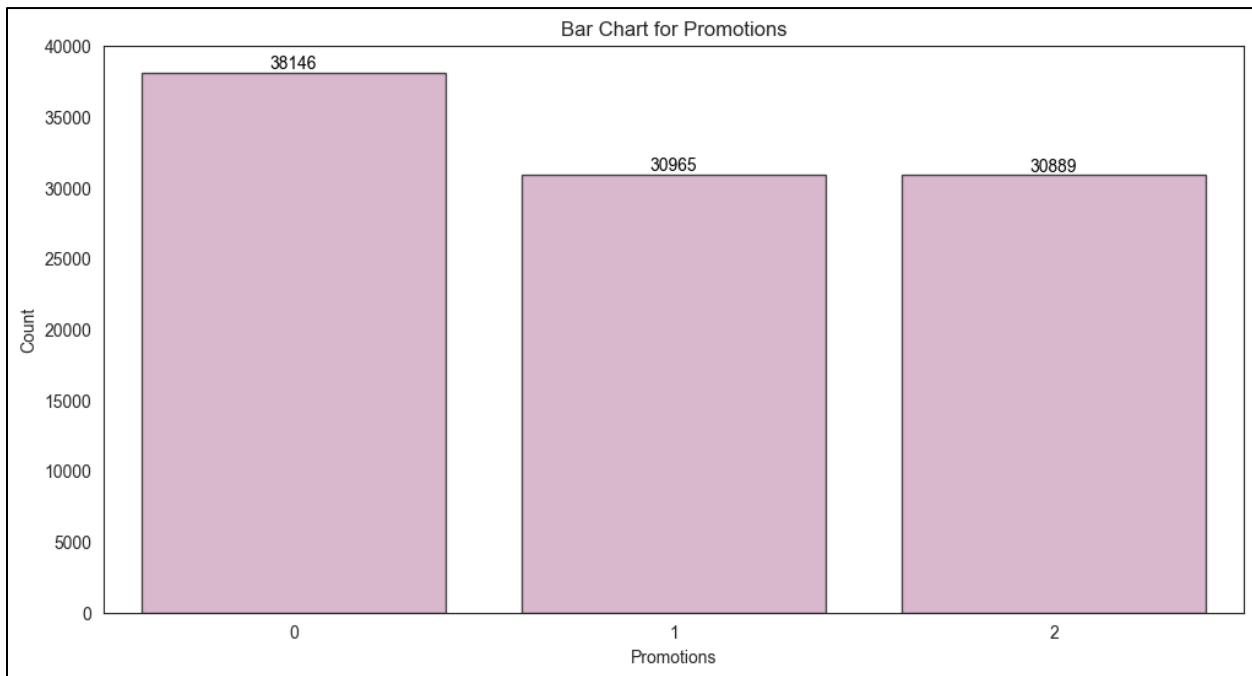
## Performance Score



*Figure 48: Bar Chart for Performance Score*

The bar chart above for “Performance\_Score” shows that scores range from 1 to 5. For the performance score 1 (24,378) and 2 (23,016) have noticeably higher counts compared to scores 3, 4, and 5, which range around 17,400–17,600. This indicates a slightly left-skewed distribution where lower performance scores are more common in the dataset. However, there are no extreme outliers as all categories are reasonably represented. Therefore, although the data is not perfectly even, it does not contain any significant anomalies.

## Promotions



*Figure 49: Bar Chart for Promotions*

The bar chart above for “Promotions” show the distribution of employees based on the number of promotions received. The dataset is nearly evenly split between employees with 0, 1 and 2 promotions with each category having a count close to 33,000. Since there is no category is underrepresented, the “Promotions” variable does not contain outliers. The balanced distribution indicates that employees have received promotions at a relatively uniform rate.

The bar chart above for “Promotions” show the distribution of employees based on the number of promotions received. It clearly shows that the highest number of employees (38,146) did not receive any promotions while the counts for those with 1 (30,965) and 2 (30,889) promotions are slightly lower and nearly identical. This suggests that the data is slightly imbalanced with a larger proportion of employees having no promotion history. However, the difference is not extreme enough to be considered an outlier. It is because the "Promotions" variable does not display any severe skewness or extreme irregularities.

#### 4.4.2.2 Numerical Variables

```

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

df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], errors='coerce')

numerical_columns = ['Age', 'Years_At_Company', 'Monthly_Salary',
                     'Work_Hours_Per_Week', 'Projects_Handled',
                     'Overtime_Hours', 'Sick_Days', 'Remote_Work_Frequency',
                     'Team_Size', 'Training_Hours', 'Employee_Satisfaction_Score']

# IQR
def detect_outliers(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return data[(data < lower_bound) | (data > upper_bound)]
outlier_results = []

for col in numerical_columns:
    outliers = detect_outliers(df[col])
    if not outliers.empty:
        outlier_results.append(f"Outliers in {col}: {outliers.shape[0]}")
    else:
        outlier_results.append(f"Outliers in {col}: None")

    # Boxplot
    plt.figure(figsize=(8, 6))
    sns.boxplot(y=df[col], color="#D291BC", orient='h')
    plt.title(f'Boxplot for {col}')
    plt.show()

for result in outlier_results:
    print(result)

```

Figure 50: Source Code of Numerical Variables by using Bar Chart

```

Outliers in Age: None
Outliers in Years_At_Company: None
Outliers in Monthly_Salary: None
Outliers in Work_Hours_Per_Week: None
Outliers in Projects_Handled: None
Outliers in Overtime_Hours: None
Outliers in Sick_Days: None
Outliers in Team_Size: None
Outliers in Training_Hours: None
Outliers in Employee_Satisfaction_Score: None

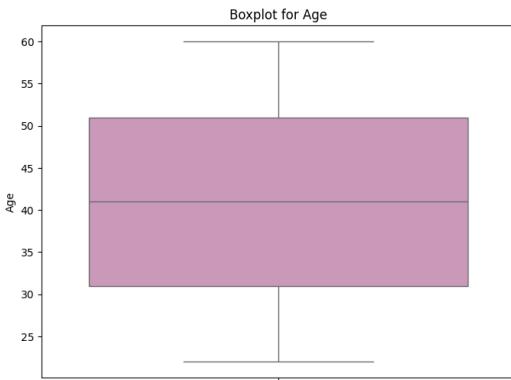
```

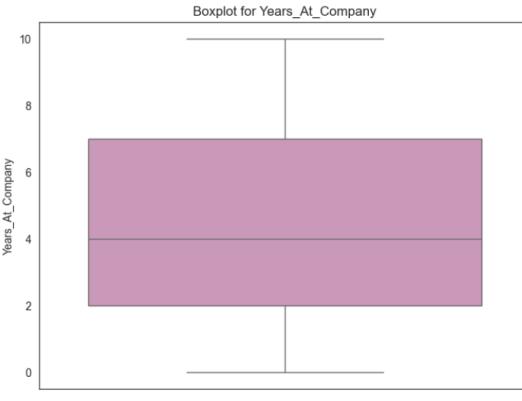
Figure 51: Output of Outliers for Numerical Variables

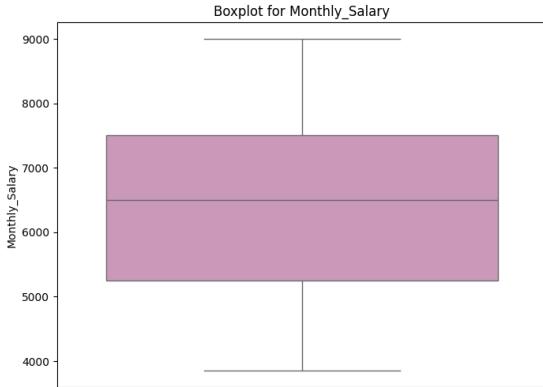
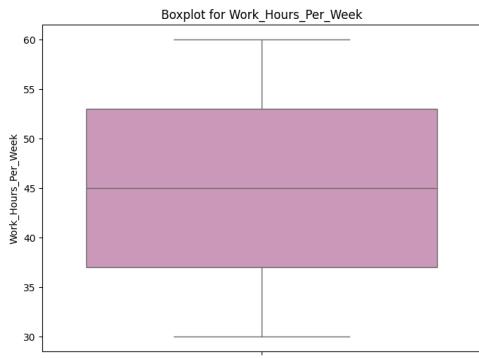
The code analyzes numerical variables in the dataset by generating boxplots to detect outliers. First, the dataset is loaded, and the `Hire_Date` column is converted into a datetime format to ensure proper data handling. Next, a list of numerical variables is defined such as `Age`, `Years at Company` and `Monthly Salary`. The code then iterates through each numerical column and generates a boxplot using the `seaborn` library.

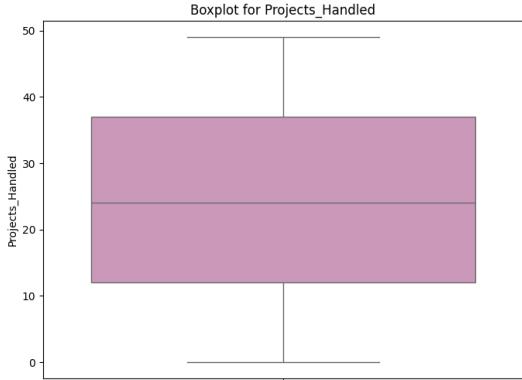
Boxplots help visualize the distribution of data and identify potential outliers, which appear as points outside the whiskers of the plot. After applying the outlier detection logic using the Interquartile Range (IQR), the results show that there are no outliers in any of the numerical columns. Each boxplot is displayed individually to allow for a detailed examination of any extreme values in the dataset to ensure that there is no outliers that are hidden. Key components of a boxplot:

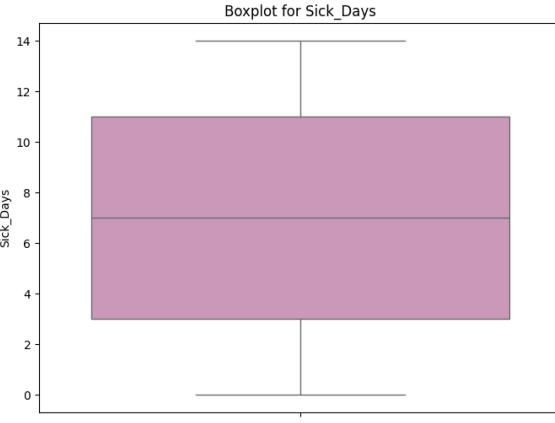
1. Median - Central Line in Box
2. Interquartile Range - Box
3. Whiskers - Lines Extending from the Box
4. Skewness - Measure of Asymmetry
5. Outliers – Fall beyond the whiskers

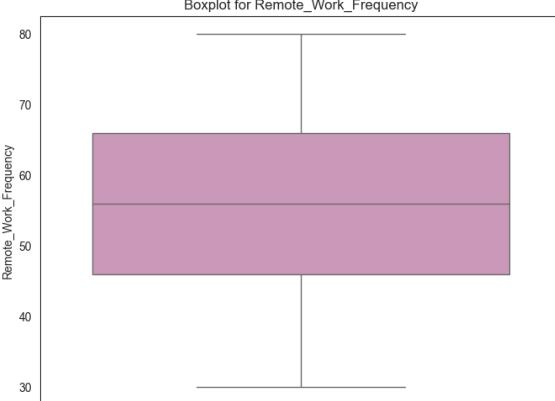
Variables	Boxplot	Observations
Age	<p>Boxplot for Age</p> 	<p><b>Median:</b> The median age is around 40.</p> <p><b>Interquartile Range:</b> The middle 50% of the data is between approximately 30 and 50.</p> <p><b>Whiskers:</b> The minimum and maximum values</p>

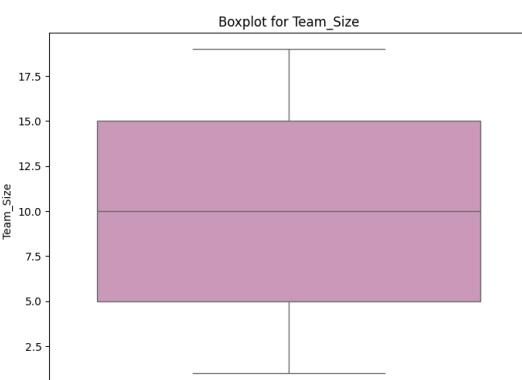
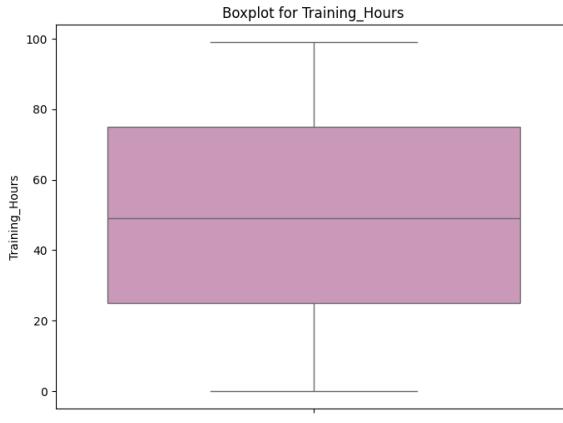
		<p>within the range appear to be around 22 and 60.</p> <p><b>Skewness:</b> No skewness found.</p> <p><b>Outliers:</b> No outliers found.</p>
<b>Years_At_Company</b>		<p><b>Median:</b> The median value is around 4.</p> <p><b>Interquartile Range (IQR):</b> The middle 50% of the data lies between 2 years and 7 years.</p> <p><b>Whiskers:</b> The whiskers extend from around 0 to 10.0.</p> <p><b>Skewness:</b> No skewness found.</p> <p><b>Outliers:</b> No outliers found, which means the data is well-behaved and there are no extreme values.</p>

<p><b>Monthly_Salary</b></p>	 <p>Boxplot for Monthly_Salary</p> <p>Y-axis: Monthly_Salary (4000 to 9000)</p> <p>Median: Around 6500</p> <p>Quartiles: Q1 ~5000, Q3 ~7500</p> <p>Whiskers: Min ~4000, Max ~9000</p>	<p><b>Median:</b> The median value is around 6500.</p> <p><b>Interquartile Range (IQR):</b> The middle 50% of the data lies between 5000 and 7500.</p> <p><b>Whiskers:</b> The whiskers extend from around 4000 to 9000.</p> <p><b>Skewness:</b> No skewness found.</p> <p><b>Outliers:</b> No outliers found.</p>
<p><b>Work_Hours_Per_Week</b></p>	 <p>Boxplot for Work_Hours_Per_Week</p> <p>Y-axis: Work_Hours_Per_Week (30 to 60)</p> <p>Median: Around 45</p> <p>Quartiles: Q1 ~38, Q3 ~50</p> <p>Whiskers: Min ~30, Max ~60</p>	<p><b>Median:</b> Around 45 hours per week.</p> <p><b>Interquartile Range (IQR):</b> The middle 50% of the data lies between approximately 40 and 50 hours per week.</p> <p><b>Whiskers:</b> The whiskers extend from</p>

		<p>around 30 to 58 hours per week.</p> <p><b>Outliers:</b> No outliers found.</p>
<b>Project_Handled</b>	 <p>A boxplot titled "Boxplot for Projects_Handled" showing the distribution of handled projects. The y-axis is labeled "Projects_Handled" and ranges from 0 to 50. The box represents the interquartile range (IQR) from approximately 10 to 40. The median is at 25. Whiskers extend from 0 to 50. There are no outliers.</p>	<p><b>Median:</b> Around 25 projects.</p> <p><b>Interquartile Range (IQR):</b> The middle 50% of the data lies between approximately 10 and 40 projects.</p> <p><b>Whiskers:</b> The whiskers extend from around 0 to 50 projects.</p> <p><b>Skewness:</b> No skewness found.</p> <p><b>Outliers:</b> No outliers found.</p>

	 <p><b>Overtime_Hours</b></p>	<p><b>Median:</b> Around 15 hours.</p> <p><b>Interquartile Range (IQR):</b> The middle 50% of the data lies between approximately 7 and 22 hours.</p> <p><b>Whiskers:</b> The whiskers extend from around 0 to 28 hours.</p> <p><b>Skewness:</b> No skewness found.</p> <p><b>Outliers:</b> No outliers found.</p>
	 <p><b>Sick_Days</b></p>	<p><b>Median:</b> Around 7 days.</p> <p><b>Interquartile Range (IQR):</b> The middle 50% of the data lies between approximately 4 and 11 days.</p>

		<p><b>Whiskers:</b> The whiskers extend from around 0 to 14 days.</p> <p><b>Skewness:</b> No skewness found.</p> <p><b>Outliers:</b> No outliers found.</p>
<b>Remote_Work_Frequency</b>	 <p>Boxplot for Remote_Work_Frequency</p> <p>Y-axis: Remote_Work_Frequency (30, 40, 50, 60, 70, 80)</p> <p>Median: Around 55%.</p> <p>Interquartile Range (IQR): The middle 50% of the data lies between approximately 45% and 70%.</p> <p>Whiskers: The whiskers extend from around 30% to 80%.</p> <p>Skewness: No skewness found.</p> <p>Outliers: No outliers found.</p>	<p><b>Median:</b> Around 55%.</p> <p><b>Interquartile Range (IQR):</b> The middle 50% of the data lies between approximately 45% and 70%.</p> <p><b>Whiskers:</b> The whiskers extend from around 30% to 80%.</p> <p><b>Skewness:</b> No skewness found.</p> <p><b>Outliers:</b> No outliers found.</p>

		<b>Median:</b> Around 10.
		<b>Interquartile Range (IQR):</b> The middle 50% of the data lies between approximately 5 and 15.
<b>Team_Size</b>		<b>Whiskers:</b> The whiskers extend from around 2.5 to 17.5.
		<b>Skewness:</b> No skewness found.
		<b>Outliers:</b> No outliers found.
<b>Training_Hours</b>		<b>Median:</b> Around 50 hours.
		<b>Interquartile Range (IQR):</b> The middle 50% of the data lies between approximately 25 and 75 hours.
		<b>Whiskers:</b> The values range from

		around 0 to 100 hours.												
	<b>Skewness:</b> No skewness found.  <b>Outliers:</b> No outliers found.													
<b>Employee_Satisfaction_Score</b>	<p><b>Median:</b> Around 3.</p> <p><b>Interquartile Range (IQR):</b> The middle 50% of the data lies between approximately 2.5 and 3.5 score rates.</p> <p><b>Whiskers:</b> The values range from 1.0 to 5.0.</p>  <table border="1"> <caption>Data for Employee_Satisfaction_Score Boxplot</caption> <thead> <tr> <th>Statistic</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>Minimum</td> <td>1.0</td> </tr> <tr> <td>Q1 (First Quartile)</td> <td>2.0</td> </tr> <tr> <td>Median</td> <td>3.0</td> </tr> <tr> <td>Q3 (Third Quartile)</td> <td>4.0</td> </tr> <tr> <td>Maximum</td> <td>5.0</td> </tr> </tbody> </table>	Statistic	Value	Minimum	1.0	Q1 (First Quartile)	2.0	Median	3.0	Q3 (Third Quartile)	4.0	Maximum	5.0	<b>Skewness:</b> No skewness found.  <b>Outliers:</b> No outliers found.
Statistic	Value													
Minimum	1.0													
Q1 (First Quartile)	2.0													
Median	3.0													
Q3 (Third Quartile)	4.0													
Maximum	5.0													

Table 5: Outliers for Numerical Variables

After using boxplots to analyze each numerical variables, there are no outliers and skewness were detected. This shows that the dataset is consistent without extreme deviations. All variable values are across all variables fall within expected ranges showing a balanced and well-distributed data without significant anomalies.

### 4.4.3 Data Transformation

```
df["Resigned"] = df["Resigned"].astype(int)
random_rows = df[["Employee_ID", "Resigned"]].sample(n=10)

print("\n [Resigned] - 10 random rows printed\n")
print(tabulate(random_rows, headers="keys", tablefmt="pretty"))
```

*Figure 52: Source Code of Data Transformation*

The target variable “Resigned” was transformed from a TRUE/FALSE format to a 0/1 format. This transformation makes the dataset more suitable for machine learning, as most algorithms require numerical values rather than categorical ones. By changing TRUE to 1 and FALSE to 0, the data is now better prepared for model training and prediction to ensure its compatibility with various machine learning models. Below is the output showing 10 random rows to make sure that the transformation was successfully applied.

[Resigned] - 10 random rows printed		
	Employee_ID	Resigned
9527	9528	0
53956	53957	0
3969	3970	0
50504	50505	0
45494	45495	1
75061	75062	1
39237	39238	0
29876	29877	0
52415	52416	0
9813	9814	0

*Figure 53: Output of Data Transformation*

### 4.4.4 Feature Engineering

#### 4.4.4.1 Creating New Variables

Two new variables have been created as part of the feature engineering process to explore potential relationships between them which is “Tenure\_Years” and “Health\_Index”.

- The “Tenure\_Years” variable is calculated based on the number of years an employee has worked at the company and is determined by the real-time day counts.
- The “Health\_Index” variable is to assess an employee’s health by calculated using the number of sick days taken in relation to their work hours per week.

## 1. Tenure\_Years

```

import pandas as pd
from tabulate import tabulate
from dateutil.relativedelta import relativedelta

df = pd.read_csv("Extended_Employee_Performance_and_Productivity_Data.csv")

df['Hire_Date'] = pd.to_datetime(df['Hire_Date']).dt.date
df['Hire_Year'] = pd.to_datetime(df['Hire_Date']).dt.year
df['Hire_Month'] = pd.to_datetime(df['Hire_Date']).dt.month

# Calculate Tenure in years from today's date
df['Tenure_Years'] = df['Hire_Date'].apply(lambda d: round(
    relativedelta(pd.to_datetime('today').date(), d).years +
    relativedelta(pd.to_datetime('today').date(), d).months / 12, 2
))

random_rows = df[['Employee_ID', 'Hire_Date', 'Hire_Year', 'Hire_Month', 'Tenure_Years']].sample(n=10)
random_rows_reset = random_rows.reset_index(drop=True)

print(tabulate(random_rows_reset, headers="keys", tablefmt="pretty"))

```

Figure 54:Source Code of Creating Tenure\_Years

“Tenure\_Years” is created by calculating the difference between the current date and the employee's hire date. This variable offers a better result than using the “Years\_At\_Company” column from the dataset for several reasons. Since the calculation is based on the current date, it ensures up-to-date information and provide an accurate reflection of an employee's tenure. The “Years\_At\_Company” column may not always reflect recent changes such as new hires or dataset updates.

Besides, the calculation of “Tenure\_Years” is dynamic and adapts in real-time, which is particularly beneficial for continuous analysis or when dealing with data that is updated often. For example, if an employee has recently joined, the “Tenure\_Years” calculation will immediately capture their time at the company whereas the “Years\_At\_Company” column might not be updated as quickly. To better demonstrate the process, the output below shows the “Hire\_Date”, “Hire\_Year”, “Hire\_Month” and the “Tenure\_Years” for a random selection of employees to have a better understanding.

	Employee_ID	Hire_Date	Hire_Year	Hire_Month	Tenure_Years
0	50598	2019-01-26	2019	1	6.17
1	32197	2021-07-21	2021	7	3.67
2	79530	2015-04-24	2015	4	9.92
3	84130	2023-12-01	2023	12	1.25
4	52752	2023-09-26	2023	9	1.5
5	60317	2018-10-04	2018	10	6.42
6	56099	2017-01-27	2017	1	8.17
7	62076	2018-04-22	2018	4	6.92
8	72241	2023-09-13	2023	9	1.5
9	89386	2016-02-12	2016	2	9.08

*Figure 55: Output of Tenure\_Years*

For example, the first employee with Employee\_ID 50,598 was hired on “2019-01-26”. The calculation of their Tenure\_Years from the hire date to the current date “2025-03-08” results in **6.17 years** of tenure.

- **Hire Date: 2019-01-26**
- **Current Date: 2025-03-08**

#### **Calculation:**

##### **1. From January 2019 to January 2025:**

6 full years

##### **2. From January 2025 to February 2025:**

1 month = 1 / 12 = 0.0833 years

##### **3. From February 2025 to March 2025:**

10 / 365 = 0.0274 years

##### **4. Total Tenure in Years:**

6 years + 0.0833 years + 0.0274 years = 6.17 years

```
df = df.drop(columns=['Hire_Date', 'Years_At_Company'], errors='ignore')
```

*Figure 56: Source Code of Dropping Columns*

After creating the new variable 'Tenure\_Years', the 'Hire\_Date' and 'Years\_At\_Company' variables will be dropped. This is because they convey the same information with 'Tenure\_Years' combining both variables to calculate the real-time difference in years.

## 2. Health\_Index

```
# Health_Index
max_lost_hours = (df['Sick_Days'] * (df['Work_Hours_Per_Week'] / 5)).max()
df['Health_Index'] = round((df['Sick_Days'] * (df['Work_Hours_Per_Week'] / 5)) / max_lost_hours, 2)
```

*Figure 57: Source Code of Creating Health\_Index*

“Work\_Hours\_Per\_Week” is divided into 5 to assume the employee is working 5 days per week so by dividing can get the number of hours they would work per day. Next , “Sick\_Days” calculates the total number of hours the employee was absent from work based on their sick days.

The Health\_Index calculates the total lost hours for each employee based on their sick days and weekly working hours. This value is then divided by the max\_lost\_hours, which represents the highest number of lost hours across all employees in the dataset. This normalization allows the Health\_Index to reflect how "healthy" or "sick" an employee is relative to the worst-case scenario—the employee with the highest number of lost hours.

The Health\_Index can be interpreted as follows:

- **A value close to 1** means the employee has lost nearly as many hours as the most absent employee and is indicating significant **sickness**.
- **A value close to 0** means the employee has taken few sick days which suggest that they have better **health** and fewer absences.

### 4.4.4.2 Categorization of Continuous Variables

```
# Remote Work Level Categorization
df['Remote_Work_Level'] = pd.cut(df['Remote_Work_Frequency'],
                                bins=[29, 47, 64, 80],
                                labels=['Low', 'Medium', 'High'],
                                include_lowest=True)
```

*Figure 58: Source Code of Categorization of Continuous Variables*

"Remote\_Work\_Frequency" was categorized into three distinct levels based on the percentage of time employees work remotely:

- **Low:** 29% to 47%
- **Medium:** 47% to 64%
- **High:** 64% to 80%

This categorization simplifies the interpretation of a continuous variable by grouping employees into meaningful segments. The cut-off points were calculated by dividing the observed range of values (29% to 80%) into three approximately equal-width intervals. Specifically, the **total range (80 - 29 = 51)** was **divided by 3** which has resulting in intervals of roughly **17 units each**.

### **Define Bin Boundaries:**

- **Low:** 29% to  $(29 + 17 \text{ units}) = 29\% - 46\%$
- **Medium:** 46% to  $(46 + 17 \text{ units}) = 47\% - 63\%$
- **High:** 63% to 80% =  $64\% - 80\%$

By transforming remote work percentages into categorical levels, it becomes easier to analyse patterns, compare employee behaviour and gain insights across different remote work groups. This approach also supports clearer visualizations and can improve the performance and interpretability of certain statistical or machine learning models.

```
df = df.round(2)
print(tabulate(df[['Tenure_Years', 'Health_Index']].head(), headers="keys", tablefmt="pretty"))
print("\nAll columns in the DataFrame after Feature Engineering:\n")
print(df.columns)
```

Figure 59: Source Code of Display Tenure\_Years and Health\_Index

	Tenure_Years	Health_Index
0	3.42	0.08
1	1.17	0.57
2	9.67	0.74
3	8.67	0.59
4	3.92	0.0

Figure 60: Output of Tenure\_Years and Health\_Index

The first five rows of “Tenure\_Years” and “Health\_Index” will be displayed to confirm the changes.

#### 4.4.4.3 Initial Drop Columns

```
# Drop unnecessary columns
columns_to_drop = ['Remote_Work_Frequency', 'Gender']
df = df.drop(columns=[col for col in columns_to_drop if col in df.columns], errors='ignore')
```

Figure 61: Source Code of Initial Drop Columns

There are two columns that were dropped from the dataset which is “Remote\_Work\_Frequency” and “Gender” in this section.

The “Remote\_Work\_Frequency” variable was removed because it had already been transformed into a new categorical variable, Remote\_Work\_Level, which grouped employees into Low, Medium and High remote work categories. So, keeping both the original and the derived variable would create redundancy and could potentially introduce multicollinearity in the model.

Meanwhile, the “Gender” column was excluded to avoid potential bias in the prediction of employee churn. It is because including gender as a feature might lead the model to associate resignation patterns with gender differences, which could raise ethical concerns and result in biased predictions. Therefore, removing this variable supports the development of a fairer and more responsible model focused on job-related attributes rather than personal demographic factors.

```

    Columns in the DataFrame after creating new variables:

Index(['Employee_ID', 'Department', 'Gender', 'Age', 'Job_Title',
       'Education_Level', 'Performance_Score', 'Monthly_Salary',
       'Work_Hours_Per_Week', 'Projects_Handled', 'Overtime_Hours',
       'Sick_Days', 'Team_Size', 'Training_Hours', 'Promotions',
       'Employee_Satisfaction_Score', 'Resigned', 'Tenure_Years',
       'Health_Index', 'Remote_Work_Level'],
      dtype='object')
```

*Figure 62:Output of Tenure\_Years and Health\_Index*

The current columns in the DataFrame are listed to ensure that the new variables have been added and the intended columns have been successfully removed.

#### 4.4.4.4 Encoding Categorical Variables

Categorical data in machine learning cannot be directly processed by most algorithms because they are expected in numerical inputs. Therefore, encoding categorical variables is necessary to convert them into a numerical format. There are two common encoding techniques, which are One-Hot Encoding and Label Encoding.

```
# One-Hot Encoding for Department and Job_Title
df_encoded = pd.get_dummies(df, columns=['Department', 'Job_Title'], drop_first=True)

# One-Hot Encoded Department and Job_Title
print("\nOne-Hot Encoded Department and Job_Title (First 5 rows):")
print(tabulate(df_encoded[df_encoded.columns[df_encoded.columns.str.startswith('Department') |
                                             df_encoded.columns.str.startswith('Job_Title')]].head(), headers="keys", tablefmt="pretty"))
```

*Figure 63:Source Code of One-Hot Encoding*

One-Hot Encoded Department and Job_Title (First 5 rows):							
	Department_Engineering	Department_Finance	Department_HR	Department_IT	Department_Legal	Department_Marketing	Department_Operations
0	False	False	False	True	False	False	False
1	False	True	False	False	False	False	False
2	False	True	False	False	False	False	False
3	False	False	False	False	False	False	False
4	True	False	False	False	False	False	False

*Figure 64: Output of One-Hot Encoding*

	Department_Sales	Job_Title_Consultant	Job_Title_Developer	Job_Title_Engineer	Job_Title_Manager	Job_Title_Specialist	Job_Title_Technician
1	False	False	False	False	False	True	False
2	False	False	True	False	False	False	False
3	False	False	False	False	False	True	False
4	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False

*Figure 65:Output of One-Hot Encoding*

One-hot encoding is used for nominal variables such as "Department" and "Job\_Title" in the datasets. It creates a new binary column for each category in the original variable and assigned a "1" if the input belongs to that category and a "0" otherwise. For example, if the Department variable includes categories such as "Engineering", "HR" and "Finance", one-hot encoding will create separate columns for each department. The input from the "Engineering" department would have "1" in the "Department\_Engineering" column and "0" in the others. This method ensures that no category is treated as having a higher or lower value than others and maintains the categorical nature of the variable.

```
# Ordinal Encoding for Education_Level
education_order = ['High School', 'Bachelor', 'Master', 'PhD']
df_encoded['Education_Level'] = pd.Categorical(df_encoded['Education_Level'], categories=education_order, ordered=True)
df_encoded['Education_Level'] = df_encoded['Education_Level'].cat.codes

# Ordinal Encoding for Remote_Work_Level
df_encoded['Remote_Work_Level'] = pd.Categorical(df_encoded['Remote_Work_Level'], categories=['Low', 'Medium', 'High'], ordered=True)
df_encoded['Remote_Work_Level'] = df_encoded['Remote_Work_Level'].cat.codes

# Treat 'Promotions' and 'Performance_Score' as categorical
df_encoded['Promotions'] = df_encoded['Promotions'].astype('category')
df_encoded['Performance_Score'] = df_encoded['Performance_Score'].astype('category')

# Select columns to display
selected_columns = ['Employee_ID', 'Education_Level', 'Remote_Work_Level']

print("Employee_ID, Education_Level, Remote_Work_Level (First 5 rows):")
print(tabulate(df_encoded[selected_columns].head(), headers="keys", tablefmt="pretty"))
```

Figure 66: Source Code of Label Encoding

Employee_ID, Education_Level, Remote_Work_Level (First 5 rows):			
	Employee_ID	Education_Level	Remote_Work_Level
0	1	0	1
1	2	0	0
2	4	1	2
3	5	1	1
4	6	0	2

Figure 67: Output of Label Encoding

Besides, label encoding is used for ordinal variables, such as "Education\_Level" and "Remote\_Work\_Level". This technique assigns an integer to each category based on the order specified by the user. For example, Education\_Level could be encoded as follows:

- High School – 0
- Bachelor – 1
- Master – 2
- PhD – 3

This encoding method is suitable for variables where the categories have a meaningful rank or hierarchy as the algorithm can then interpret the numbers which may reflect the inherent order.

Both encoding techniques transform categorical variables into a form that machine learning algorithms can work with to allow them to analyze patterns in the data effectively. One-Hot Encoding is preferred when the categories do not have an order, while Label Encoding is more suitable when there is a natural ranking or order to the categories.

```
df_encoded['Promotions'] = df_encoded['Promotions'].astype('category')
df_encoded['Performance_Score'] = df_encoded['Performance_Score'].astype('category')
```

*Figure 68:Source Code of Converting to Ordinal Categories*

The “Promotions” and “Performance\_Score” variables have converted to ordinal categories because they have a meaningful order. “Promotions” is the number of promotions an employee has received (0,1,2) where the values have an inherent ranking ( $0 < 1 < 2$ ). Similarly, “Performance\_Score” ranges from 1 to 5, where each number represents a level of performance with higher values will have a better performance. So, converting these features to ordinal categories ensures that the model accurately understands the order of the values and treats them appropriately, preserving the natural ranking and preventing any misinterpretation of the relationship between the categories. This step is important for predicting employee churn more accurately based on these factors.

Below shown the datatype of variables after encoding:

```
print(df_encoded.dtypes)
```

*Figure 69:Source Code of Encoded Data Types*

Education_Level	int8
Performance_Score	category
Monthly_Salary	float64
Work_Hours_Per_Week	int64
Projects_Handled	int64
Overtime_Hours	int64
Sick_Days	int64
Team_Size	int64
Training_Hours	int64
Promotions	category
Employee_Satisfaction_Score	float64
Resigned	int64
Tenure_Years	float64
Health_Index	float64
Remote_Work_Level	int8

Figure 70: Output of Encoded Data Types

After converting the “Promotions” and “Performance\_Score” variables into ordinal categories, they are now correctly represented with the category datatype.

## 4.5 Data Understanding

### 4.5.1 Graphical View of Variables After Data Preprocessing

In this section will be discuss the variables that have been pre-processed. For the variables that have not undergone preprocessing, there are no changes will be made and can be referred to in [Section 4.3.8](#) for the analysis.

#### 4.5.1.1 New Variables that are created in Feature Engineering section

```
df[['Health_Index', 'Tenure_Years']].describe()
```

Figure 71: Source Code of Summary Statistics

```
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# Health_Index
sns.histplot(df['Health_Index'], kde=True, bins=30, ax=axes[0], alpha=0.7, color="#D291BC")
axes[0].set_title('Distribution of Health Index')

# Tenure_Years
sns.histplot(df['Tenure_Years'], kde=True, bins=30, ax=axes[1], alpha=0.7, color="#D291BC")
axes[1].set_title('Distribution of Tenure Years')

plt.show()
```

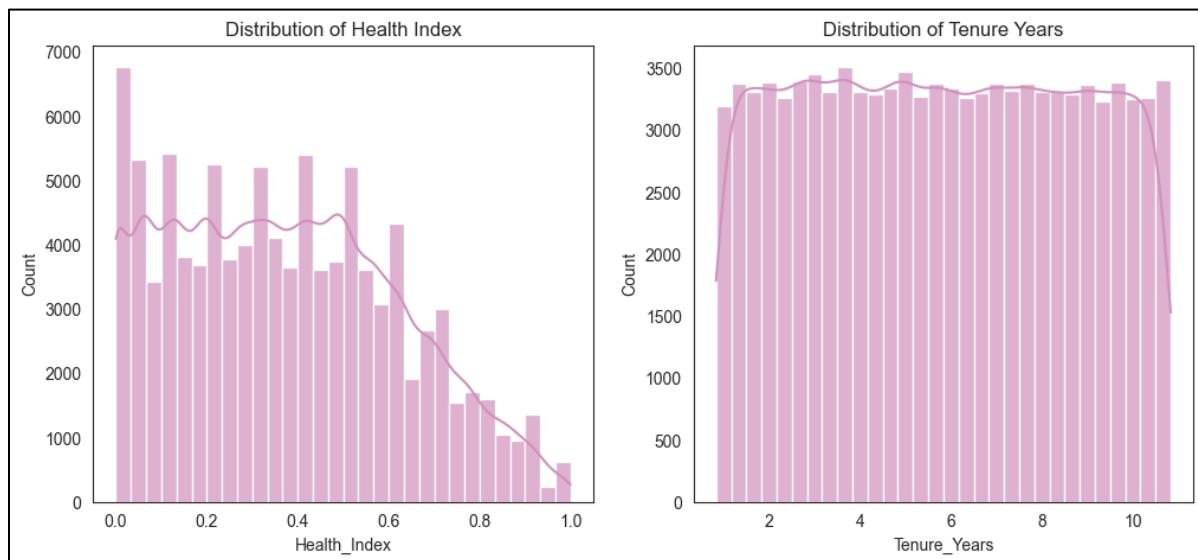
Figure 72: Source Code of Graphical Statistics for Health\_Index and Tenure\_Years

The summary statistics and distribution visualizations provide a quick overview of the newly created variables, "Health\_Index" and "Tenure\_Years." These steps help analyze the data by showing key measures like the mean, median, standard deviation and ranges along with a visual distribution of the data. Visual distribution allows to analyze the general pattern in the data and to have a better understanding of the characteristics of the newly created variables.

	Health_Index	Tenure_Years
count	100000.000000	100000.000000
mean	0.375632	5.784808
std	0.247847	2.881986
min	0.000000	0.830000
25%	0.170000	3.250000
50%	0.360000	5.750000
75%	0.560000	8.250000
max	1.000000	10.830000

*Figure 73:Summary Statistics of Health\_Index and Tenure\_Years*

The table above shows the descriptive statistics for "Health\_Index" and "Tenure\_Years". For Health\_Index, the average value is 0.375 with a range from 0 to 1 and this shows that there are varying health levels among employees. For Tenure\_Years, the average is 5.78 years with employees' tenure ranging from 0.83 to 10.83 years.



*Figure 74: Graphical Statistics of Health\_Index and Tenure\_Years*

## Health Index

The distribution shows that most employees have a health index closer to 0, with fewer employees experiencing higher levels of sickness. The rightward skew indicates a larger proportion of employees have taken fewer sick days compared to others and this shows that there is a better health among most of the employees.

## Tenure Years

This graph shows that the distribution of employee tenure is relatively even across the years with no significant peaks. Employees have a uniform distribution of tenure years and show that the company has employees with varied lengths of service which can provide insights into the workforce's stability.

### 4.5.1.2 Variables that are Encoded in Encoding section

#### 1. Remote\_Work\_Levels

```

sns.set(style="whitegrid")
bar_color = "#D291BC"

# Remote Work Levels
plt.figure(figsize=(9, 7))
ax = sns.countplot(x='Remote_Work_Level', data=df_encoded, color=bar_color, edgecolor="black")
remote_counts = df_encoded['Remote_Work_Level'].value_counts()
total_remote = remote_counts.sum()

for p in ax.patches:
    x = p.get_x() + p.get_width() / 2
    y = p.get_height()
    percentage = (y / total_remote) * 100
    ax.annotate(f'{int(y)}\n({percentage:.2f}%)', (x, y + 50), ha='center', va='bottom', fontsize=10, color='black')

plt.title('Bar Chart of Remote Work Levels')
plt.ylabel('Count')
plt.grid(False)
plt.show()

```

Figure 75:Source Code of Graphical Statistics for Remote\_Work\_Levels

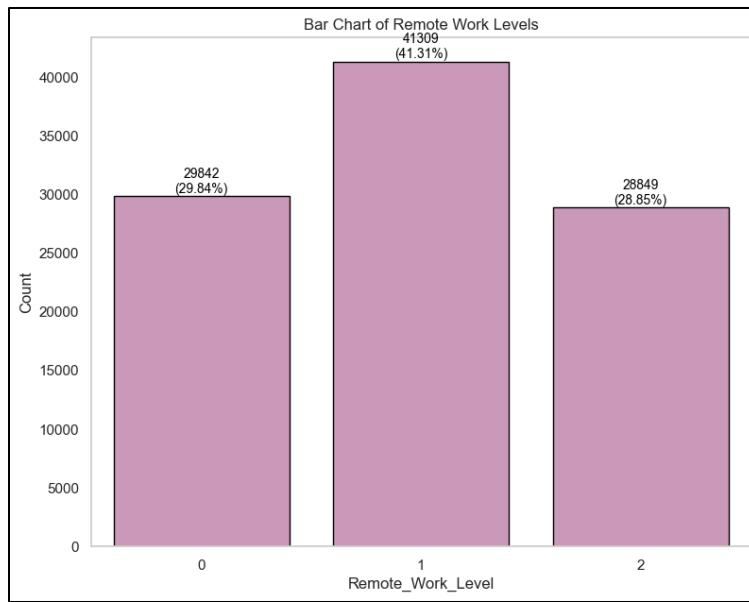


Figure 76: Bar Chart for Remote\_Work\_Levels

Figure above displays the bar chart of “Remote\_Work\_Level” after preprocessing. After encoding, it is transformed into three distinct values: **0**, **1** and **2**, which represent the following categories:

- **0**: Low remote work level
- **1**: Medium remote work level
- **2**: High remote work level

The bar chart shows the distribution of employees across the “Remote\_Work\_Level” which showing that 29.84% employees are in Level 0, 41.31% in Level 1, and 28.85% in Level 2.

## 2. Education\_Levels

```
# Education Levels
plt.figure(figsize=(9, 7))
ax = sns.countplot(x='Education_Level', data=df_encoded, color=bar_color, edgecolor="black")
education_counts = df_encoded['Education_Level'].value_counts()
total_education = education_counts.sum()

for p in ax.patches:
    x = p.get_x() + p.get_width() / 2
    y = p.get_height()
    percentage = (y / total_education) * 100
    ax.annotate(f'{int(y)}\n({percentage:.2f}%)', (x, y + 50), ha='center', va='bottom', fontsize=10, color='black')

plt.title('Bar Chart of Education Levels')
plt.ylabel('Count')
plt.grid(False)
plt.show()
```

Figure 77: Source Code Graphical Statistics for Education\_Levels

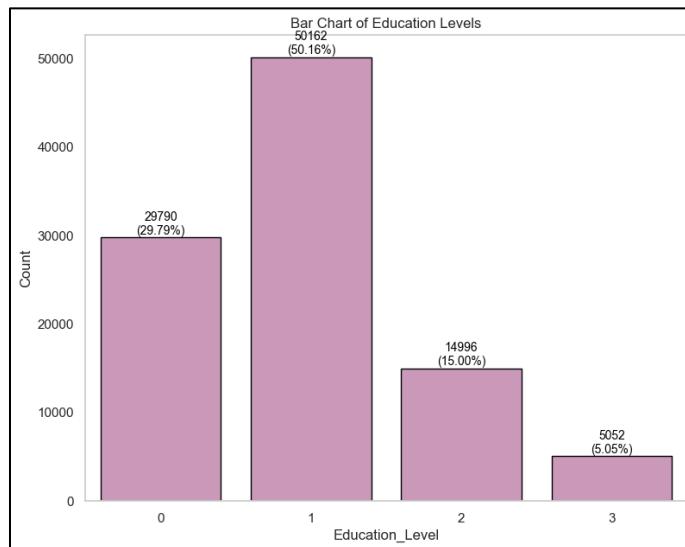


Figure 78: Bar Chart for Education\_Level

Figure above displays the bar chart of “Education\_Level” after preprocessing. The 'Education\_Level' variable has been encoded into four distinct values: 0, 1, 2, and 3, which represent:

- 0: High School
- 1: Bachelor
- 2: Master
- 3: PhD

The graph shows the distribution of employees at these educational levels with 50.16% of the employees having a Bachelor's degree, 29.79% having a High School education, 15% having a Master's degree and only 5.05% having a PhD. This clearly shows the view of the “Education\_Level” variable had after encoding.

### 3. Department

```
# Department
plt.figure(figsize=(10, 8))
department_columns = [col for col in df_encoded.columns if col.startswith('Department')]
department_counts = df_encoded[department_columns].sum()
total_department = department_counts.sum()
ax = department_counts.plot(kind='bar', color=bar_color, edgecolor="black")

for i, count in enumerate(department_counts):
    percentage = (count / total_department) * 100
    plt.text(i, count + 0.05, f'{count}\n({percentage:.2f}%)', ha='center', va='bottom', fontsize=10, color='black')

plt.title('Bar Chart of Department Categories')
plt.ylabel('Count')
plt.xticks(rotation=45, ha="right")
plt.grid(False)
plt.show()
```

Figure 79: Source Code of Graphical Statistics for Department

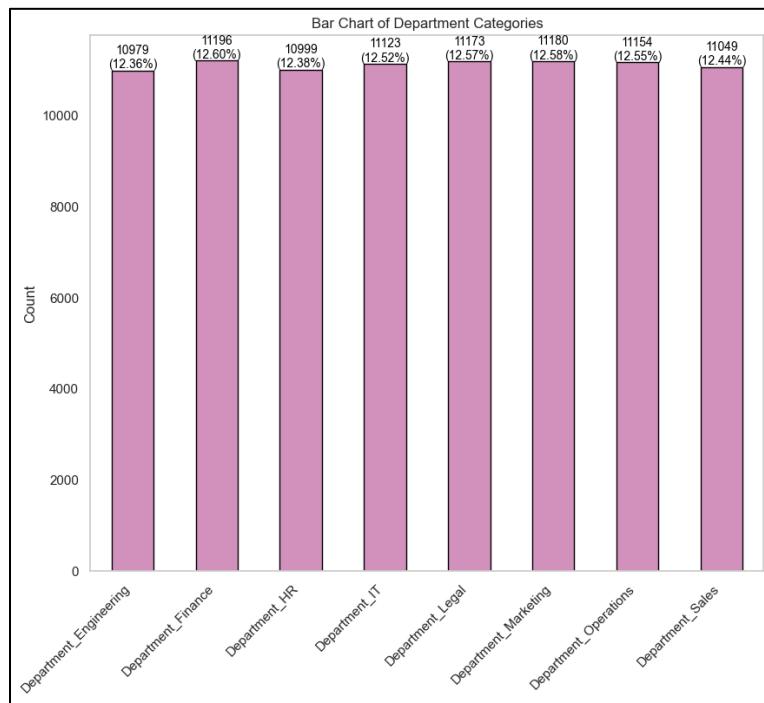


Figure 80: Bar Chart for Department

The figure above shown the bar chart of “Department” categories after encoding. The Department variable has been encoded into distinct categories for each department. The chart shows the

distribution of employees across these departments, where the departments are represented by the encoded category names such as Department\_Engineering, Department\_Finance and Department\_HR.

The graph demonstrates departmental employee counts through bars and includes segment percentage numbers above the bars. The bar chart shows that 12.36% of employees are in the Engineering department while 12.58% are in the Marketing department. This distribution provides an overview of how employees distribute across departments allocation after encoding.

#### 4. Job\_Title

```
# Job Title
plt.figure(figsize=(10, 8))
job_title_columns = [col for col in df_encoded.columns if col.startswith('Job_Title')]
job_title_counts = df_encoded[job_title_columns].sum()
total_job_title = job_title_counts.sum()
ax = job_title_counts.plot(kind='bar', color=bar_color, edgecolor="black")

for i, count in enumerate(job_title_counts):
    percentage = (count / total_job_title) * 100
    plt.text(i, count + 0.05, f'{count}\n({percentage:.2f}%)', ha='center', va='bottom', fontsize=10, color='black')

plt.title('Bar Chart of Job Title Categories')
plt.ylabel('Count')
plt.xticks(rotation=45, ha="right")
plt.grid(False)
plt.show()
```

Figure 81: Source Code of Graphical Statistics for Job\_Title

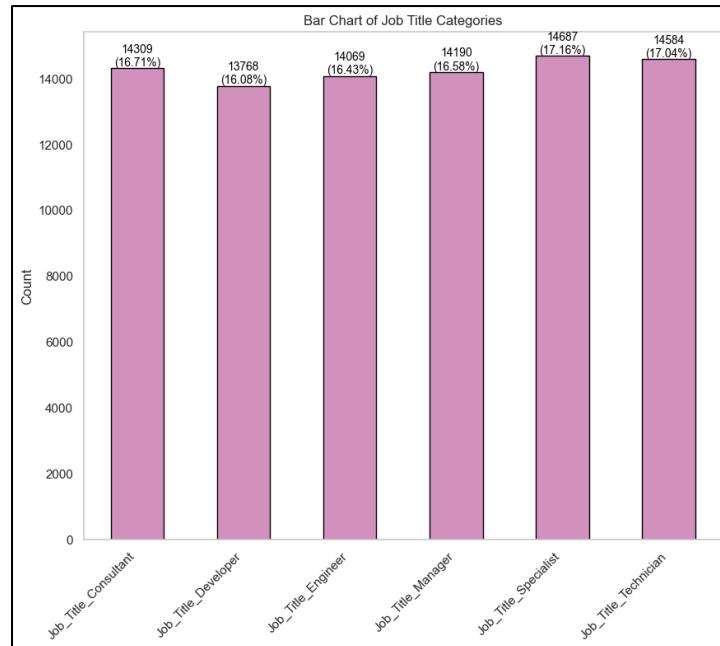


Figure 82: Bar Chart for Job\_Title

The figure above shown the bar chart of “Job\_Title” categories after encoding. The Job Title variable has been encoded into distinct categories for each job title such as Job\_Title\_Constant, Job\_Title\_Developer, Job\_Title\_Engineer and others. The chart shows the distribution of employees across these job titles with the count and percentage of employees in each category.

For example, the 'Job\_Title\_Constant' category has 16.71% of employees while 'Job\_Title\_Manager' has 16.58%. This distribution provides an overview of how employees are allocated across different job titles after the encoding process.

#### 4.5.2 Feature Selection

In this section, Mutual Information (ML) scores were used to assess the relationship between variables and the target variable (Resigned) to determine which variables contribute most to predicting employee resignation. The reason of choosing mutual information because the correlation between the target variable (Resigned) and the variables was found to be very weak. Since correlation only captures linear relationships and it fails to identify non-linear dependencies between the variables and the target.

So, Mutual Information (ML) can detect both linear and non-linear relationships for identifying informative features, regardless of the type of relationship. By using mutual information, variables can capture more complex patterns that could influence employee resignation when simple linear correlation is not be visible.

Numerical and categorical variables are separated during feature selection because they require different methods for mutual information calculation. Numerical variables represent continuous data, while categorical variables represent discrete categories. Therefore, treating them separately ensures that the appropriate method is applied to each feature type for more accurate results.

To further enhance the feature selection process, a threshold for mutual information scores will be established. Variables identified as having a weak relationship with the target variable will be removed and only the top 10 features will be selected for finalizing the feature set used in model training.

#### 4.5.2.1 Numerical Variables

```

numerical_columns = [col for col in df.select_dtypes(include=['int64', 'float64']).columns if col not in ['Promotions', 'Performance_Score', 'Employee_ID', 'Resigned']]

# Define X as features and y as target variable
X_mi = df[numerical_columns]
y = df['Resigned']

# Calculate mutual information
mi_numerical = mutual_info_classif(X_mi, y, random_state=42)

# Create DataFrame to store MI scores
mi_df = pd.DataFrame({'Variable': X_mi.columns, 'MI Score': mi_numerical}).sort_values(by='MI Score', ascending=False)
print(mi_df)

plt.figure(figsize=(12, 6))
plt.barh(mi_df['Variable'], mi_df['MI Score'], color="#D291BC")
plt.xlabel('Mutual Information Score')
plt.title('Mutual Information Scores for Numerical Variables')
plt.grid(False)
plt.gca().set_facecolor('white')
plt.show()

# Different thresholds to see the impact on selected features
thresholds = [0.001, 0.005]
for threshold in thresholds:
    selected_variable = mi_df[mi_df['MI Score'] > threshold]['Variable'].tolist()
    print(f"Top 5 Variables with threshold {threshold}: {selected_variable}")

```

*Figure 83: Source Code of Feature Selection for Numerical Variables*

The mutual information scores were calculated for all the numerical features in the dataset. These scores provide insight into the relationship between each feature and the target variable, Resigned. A higher mutual information score indicates a stronger relationship with the target, while a lower score suggests a weaker or no relationship.

There are different thresholds (0.001 & 0.005) were tested to determine which features have the most significant relationship with the target variable, “Resigned”. A lower threshold allows for the inclusion of features with weaker relationships while a higher threshold helps to focus on the most strongly correlated features. By employing this method, it can help in identifying the features that generate the most value for predicting employee resignation. Below shown the MI score with the horizontal bar chart which shows the mutual information score for numerical variables.

Numerical Features - Mutual Information:		
	Variable	MI Score
1	Monthly_Salary	0.115301
2	Employee_Satisfaction_Score	0.030293
3	Work_Hours_Per_Week	0.027497
4	Overtime_Hours	0.027337
5	Health_Index	0.007738
6	Projects_Handled	0.003041
7	Education_Level	0.001542
8	Team_Size	0.001087
9	Tenure_Years	0.000780
10	Training_Hours	0.000550
11	Sick_Days	0.000311
12	Age	0.000000

Figure 84: Output of Mutual Information Score

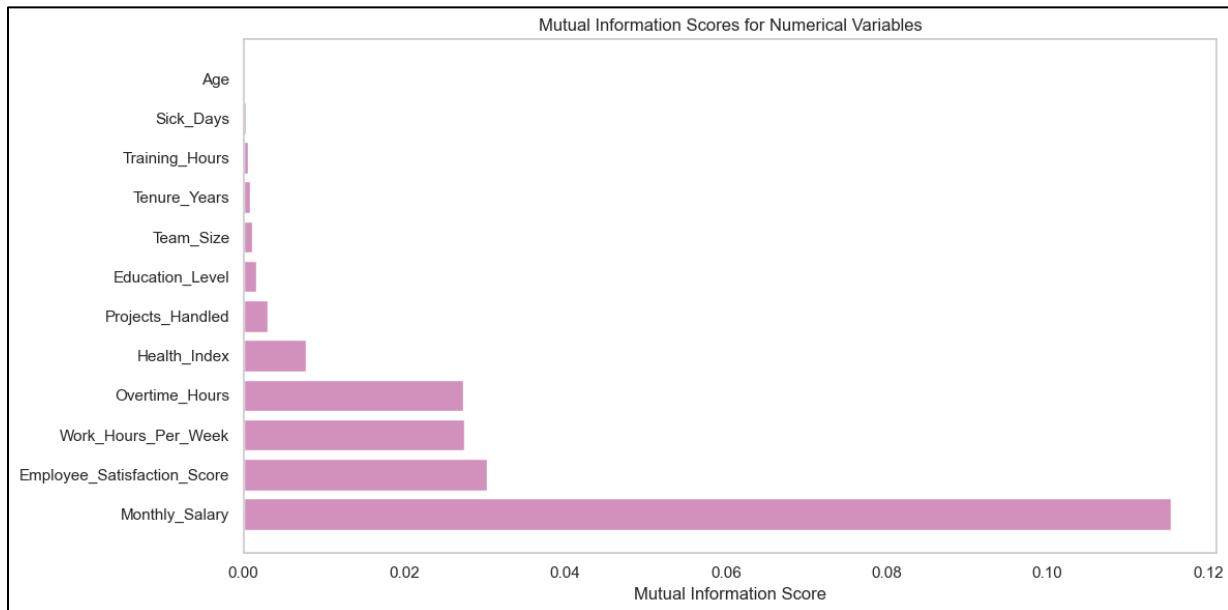


Figure 85: Horizontal Bar Chart of Mutual Information Scores for Numerical Variables

The Mutual Information (MI) scores for numerical variables show several important insights regarding their relationship with the target variable, “Resigned”.

Among all features, “Monthly\_Salary” has the highest MI score of 0.115301 and has indicated the strongest relationship with employee resignation. This suggests that salary levels play a significant role in an employee's decision to resign. Next, “Employee\_Satisfaction\_Score” (0.030293), “Work\_Hours\_Per\_Week” (0.027497) and “Overtime\_Hours” (0.027337) show moderate MI scores which is implying a reasonable influence on resignation behavior. “Health\_Index”

(0.007738) and “Projects\_Handled” (0.003041) demonstrate weaker but still meaningful relationships.

In contrast, features such as “Education\_Level”, “Team\_Size”, “Tenure\_Years”, “Training\_Hours”, “Sick\_Days” and “Age” show very low or even zero MI scores which suggesting minimal contribution to predicting resignation.

Based on the selected thresholds of 0.001 and 0.005, the results are shown below:

```
Top 5 Variables with threshold 0.001: ['Monthly_Salary', 'Employee_Satisfaction_Score', 'Work_Hours_Per_Week', 'Overtime_Hours', 'Health_Index', 'Projects_Handled', 'Education_Level', 'Team_Size']
Top 5 Variables with threshold 0.005: ['Monthly_Salary', 'Employee_Satisfaction_Score', 'Work_Hours_Per_Week', 'Overtime_Hours', 'Health_Index']
```

*Figure 86: Top 5 Variables of Mutual Information Scores for Numerical Variables*

The variables with threshold 0.001 are:

- Monthly\_Salary
- Employee\_Satisfaction\_Score
- Work\_Hours\_Per\_Week
- Overtime\_Hours
- Health\_Index
- Project\_Handled
- Education\_Level
- Team\_Size

These variables show a stronger connection to the target variable, "Resigned." However, when applying a higher threshold of 0.005, the list is narrowed down to:

- Monthly\_Salary
- Employee\_Satisfaction\_Score
- Work\_Hours\_Per\_Week
- Overtime\_Hours
- Health\_Index

This indicates that only the most strongly related variables remain, while those with low MI scores will likely be excluded to improve the model's efficiency and accuracy.

#### 4.5.2.2 Categorical Variables

```
# Mutual Information for Categorical Features that encoded
mutual_info_cat = mutual_info_classif(X_encoded, y, random_state=42)

# DataFrame for Categorical Mutual Information
mutual_info_cat_df = pd.DataFrame({
    'Feature': X_encoded.columns,
    'MI Score': mutual_info_cat
}).sort_values(by='MI Score', ascending=False)

print("\nCategorical Features - Mutual Information:\n")
print(mutual_info_cat_df)

# Top 5
top_n_cat_features = mutual_info_cat_df.head(5)['Feature'].tolist()

print("\nTop 5 Categorical Features:")
print(top_n_cat_features)

plt.figure(figsize=(12, 6))
plt.barh(mutual_info_cat_df['Feature'], mutual_info_cat_df['MI Score'], color="#D291BC")
plt.xlabel('Mutual Information Score')
plt.title('Mutual Information Scores for Categorical Features')
plt.grid(False)
plt.show()
```

Figure 87: Source Code of Feature Selection for Categorical Variables

Categorical Features - Mutual Information:		
	Feature	MI Score
1	Performance_Score	0.089355
2	Remote_Work_Level	0.058328
3	Promotions	0.014693
4	Job_Title_Technician	0.010779
5	Job_Title_Manager	0.004670
6	Job_Title_Specialist	0.004398
7	Department_Marketing	0.003238
8	Department_HR	0.002818
9	Department_Operations	0.002378
10	Job_Title_Engineer	0.001727
11	Department_Legal	0.001212
12	Education_Level	0.001127
13	Department_Finance	0.000859
14	Job_Title_Developer	0.000827
15	Job_Title_Consultant	0.000495
16	Department_IT	0.000087
17	Department_Engineering	0.000000
18	Department_Sales	0.000000

Figure 88: Mutual Information Scores for Categorical Variables

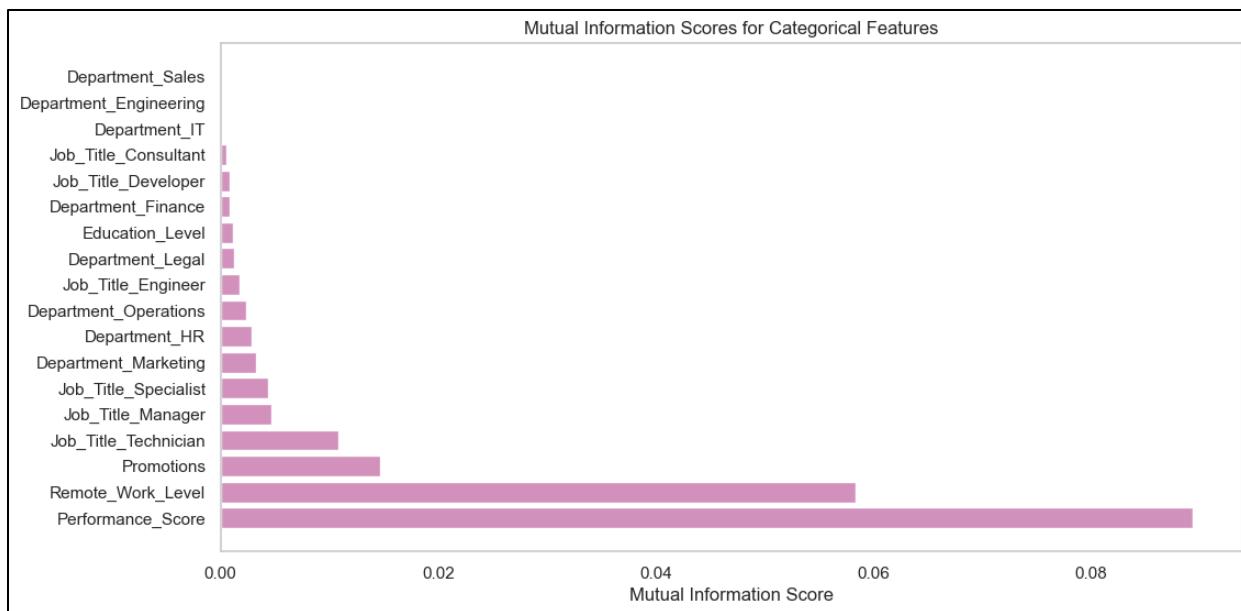


Figure 89: Horizontal Bar Chart of Mutual Information Scores for Categorical Variables

The mutual information (MI) scores for categorical variables show the strength of their relationship with the target variable, **Resigned**.

Among these features, “Performance\_Score” has the highest MI score at 0.089355 which has indicating it is the most influential categorical predictor of employee resignation. This is followed by “Remote\_Work\_Level” (0.058328) which suggests that remote work arrangements play a meaningful role in resignation decisions. The other features such as “Promotions” (0.014693), “Job\_Title\_Technician” (0.010779) and “Job\_Title\_Manager” (0.004670) also contribute to the prediction although to a lesser extent.

Conversely, variables such as “Department\_Sales”, “Department\_Engineering”, and “Department\_IT” show near-zero or zero MI scores which has indicating little to no predictive value.

```
Top 5 Categorical Features:
['Performance_Score', 'Remote_Work_Level', 'Promotions', 'Job_Title_Technician', 'Job_Title_Manager']
```

Figure 90: Top 5 Variables of Mutual Information Scores for Categorical Variables

The Top 5 Categorical Features based on the Mutual Information (MI) scores are:

- Performance\_Score
- Remote\_Work\_Level
- Promotions
- Job\_Title\_Technician
- Job\_Title\_Manager

These features have the highest MI scores which indicating a strong relationship with the target variable, “Resigned”.

#### 4.5.2.3 Top 10 Features

```
# Top 5 numerical features by MI score
top_n_num_features = mi_df.sort_values(by='MI Score', ascending=False).head(5)['Variable'].tolist()

# Top 5 categorical features by MI score
top_n_cat_features = mutual_info_cat_df.sort_values(by='MI Score', ascending=False).head(5)['Feature'].tolist()

# Combine them into Top 10 Features
top_10_features = top_n_num_features + top_n_cat_features

print("Top 5 Numerical Features (by MI):")
for i, feat in enumerate(top_n_num_features, start=1):
    print(f"{i}. {feat}")

print("\nTop 5 Categorical Features (by MI):")
for i, feat in enumerate(top_n_cat_features, start=1):
    print(f"{i}. {feat}")

print("\nCombination of Top 10 Features:")
for i, feat in enumerate(top_10_features, start=1):
    print(f"{i}. {feat}")
```

*Figure 91: Top 10 Features*

The top 10 most relevant features for predicting employee churn were selected using mutual information (MI) scores. Since numerical and categorical variables were processed and evaluated separately, there was a risk of bias if all top features were selected from only one type. To solve this, the top five features were chosen from each category which is numerical and categorical based on their respective MI scores. Then, these were combined into a final list of 10 features. This balanced approach ensures that both types of variables are fairly represented in the model to

improve their ability to learn meaningful patterns from different data types and contribute to more accurate and unbiased predictions.

Top 5 Numerical Features (by MI):	Top 5 Categorical Features (by MI):
1. Monthly_Salary 2. Employee_Satisfaction_Score 3. Work_Hours_Per_Week 4. Overtime_Hours 5. Health_Index	1. Performance_Score 2. Remote_Work_Level 3. Promotions 4. Job_Title_Technician 5. Job_Title_Manager

Figure 92: Top 5 for Numerical and Categorical Features

Combination of Top 10 Features:
1. Monthly_Salary 2. Employee_Satisfaction_Score 3. Work_Hours_Per_Week 4. Overtime_Hours 5. Health_Index 6. Performance_Score 7. Remote_Work_Level 8. Promotions 9. Job_Title_Technician 10. Job_Title_Manager

Figure 93: Output of Combinations of Top 10 Features

The figures above show the combined top 10 features selected by taking the top 5 from both numerical and categorical features based on mutual information scores.

#### 4.5.2.4 Implications for Future Analysis

In this stage, Mutual Information (MI) scores were computed for both numerical and categorical variables to assess their relationship with the target variable, "Resigned." The analysis determines the key variables with stronger relationships, such as "Monthly\_Salary" and "Employee\_Satisfaction\_Score" for numerical variables, and "Remote\_Work\_Level" for categorical variables. These variables demonstrated stronger relationships with employee resignation and offer meaningful insights into the factors influencing churn.

While several features exhibited minimal or no correlation with the target, the top five features from each category which is numerical and categorical were selected to form a final set of ten most

informative predictors. This method ensures a balanced and comprehensive feature set that fairly represents both data types and reduces the risk of bias or overfitting.

By excluding weak and redundant variables, the model is better positioned to focus on relevant patterns by leading to more efficient training and improved predictive performance. This feature selection strategy provides a strong foundation for future modeling efforts by ensuring clarity, fairness and data-driven decision-making.

### 4.5.3 Data Splitting and Resampling

#### 4.5.3.1 Check Target Variable Count

```
# Check Target Variable
print(df['Resigned'].value_counts())
✓ 0s
Resigned
0    57422
1    42578
Name: count, dtype: int64
```

Figure 94: Source Code of Checking Numbers of Employees Resigned

The target variables “Resigned” exhibits a significant class imbalance. As seen from the figure above, there are 57,422 employees who did not resign and 42,578 employees who resigned. This imbalance is more visually represented in the pie chart in [Section 3.4.9](#). This imbalance could cause the model to be biased toward predicting the majority class which is non-resigned employees.

To solve this problem, the **SMOTE (Synthetic Minority Over-sampling Technique)** will be applied to generate synthetic samples for the minority class which is the employees who resigned. This technique helps balance the dataset and allows the model to learn effectively from both classes and make more accurate and fair predictions.

#### 4.5.3.2 Data Splitting (Train-Test Split)

```

from sklearn.model_selection import train_test_split

X_cat_top = X_combined[top_n_cat_features].reset_index(drop=True)
X_num_top = df[top_n_num_features].reset_index(drop=True)
X = pd.concat([X_cat_top, X_num_top], axis=1)

y = df['Resigned']

# Split into train (80%) and test (20%)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# Train set
print("[Train] class distribution:")
train_counts = y_train.value_counts()
print(train_counts)
print("Total:", train_counts.sum(), "\n")

# Test set
print("[Test] class distribution:")
test_counts = y_test.value_counts()
print(test_counts)
print("Total:", test_counts.sum())

```

*Figure 95: Source Code of Data Splitting*

This code performs a train-test split for a churn prediction model. It first combines the top selected categorical and numerical features into a single dataset X and sets y as the target variable (Resigned). Then, it splits the data into 80% training and 20% testing using stratified sampling to maintain the class distribution. Finally, it prints the class distribution for both sets to verify that the split is balanced.

```

[Train] class distribution:
Resigned
0    45938
1    34062
Name: count, dtype: int64
Total: 80000

[Test] class distribution:
Resigned
0    11484
1     8516
Name: count, dtype: int64
Total: 20000

```

*Figure 96: Output after Data Splitting*

This output above shows the class distribution after splitting the data into training and testing sets. There are out of 80,000 training samples, 34,062 (42.63%) are labelled as "Resigned" (True) while

45,938 (57.77%) are labelled as "Not Resigned" (False). In the test set of 20,000 samples, 8,516 (42.58%) are "Resigned" while 11,484 (57.42%) are "Not Resigned".

#### 4.5.3.3 Applying SMOTE Techniques on Training Data

```
from imblearn.over_sampling import SMOTE

# Apply SMOTE only to training set
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

print("Class distribution before SMOTE:\n", y_train.value_counts())
print("\nClass distribution after SMOTE:\n", pd.Series(y_train_resampled).value_counts())
```

*Figure 97: Source Code of Smote Techniques*

```
# Before SMOTE
ax = y_train.value_counts().plot(kind='bar', title='Class Distribution Before SMOTE', color="#D291BC")
plt.xticks([0, 1], ['Not Resigned', 'Resigned'], rotation=0)
plt.ylabel('Count')

for p in ax.patches:
    ax.annotate(f'{p.get_height()}', 
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', fontsize=10, color='black', xytext=(2, 10), textcoords='offset points')
plt.ylim(0, max(y_train.value_counts()) + 10000)
plt.grid(False)
plt.tick_params(axis="x", bottom=False)
plt.show()

# After SMOTE
ax = y_train_resampled.value_counts().plot(kind='bar', title='Class Distribution After SMOTE', color="#D291BC")
plt.xticks([0, 1], ['Not Resigned', 'Resigned'], rotation=0)
plt.ylabel('Count')

for p in ax.patches:
    ax.annotate(f'{p.get_height()}', 
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', fontsize=10, color='black', xytext=(0, 10), textcoords='offset points')
plt.ylim(0, max(y_train.value_counts()) + 10000)
plt.grid(False)
plt.tick_params(axis="x", bottom=False)
plt.show()
```

*Figure 98:Source Code of Smote Techniques*

The feature set X is constructed by combining the top selected categorical and numerical features, while the target variable y is set as the “Resigned” column, which indicates whether an employee has resigned. Then, the dataset is split into training and testing sets with 80% used for training and 20% for testing. Stratified sampling is applied to maintain the original class distribution in both sets. After splitting, SMOTE (Synthetic Minority Over-sampling Technique) is applied only to the

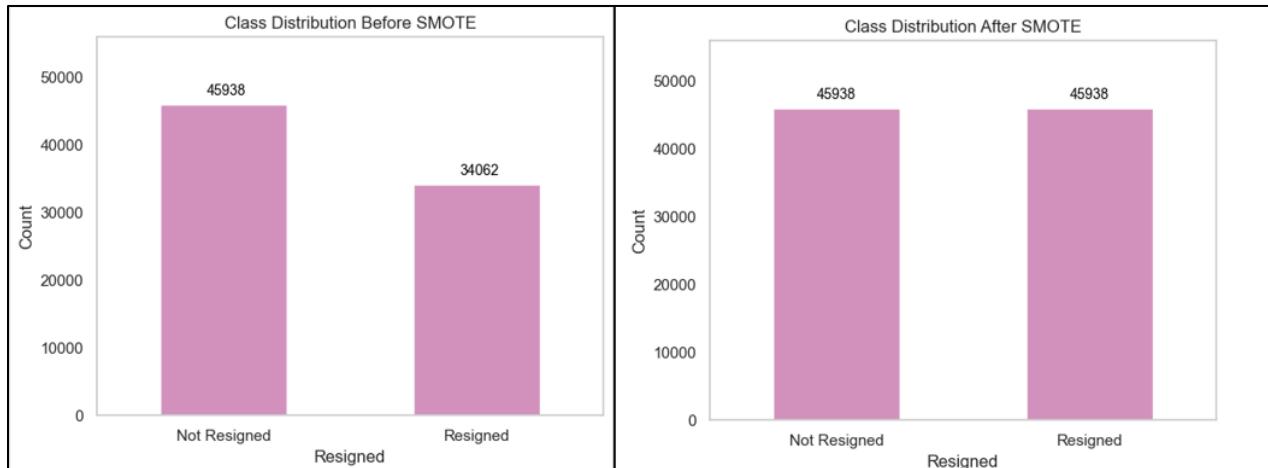
training set to solve class imbalance. SMOTE generates synthetic samples for the minority class (resigned employees) to balance the dataset to ensure that the model can learn patterns from both classes more effectively without bias toward the majority (non-resigned) class.

#### 4.5.3.4 Output after applying SMOTE Techniques

```
Class distribution before SMOTE:
Resigned
0    45938
1    34062
Name: count, dtype: int64

Class distribution after SMOTE:
Resigned
0    45938
1    45938
Name: count, dtype: int64
```

*Figure 99: Output After Smote Techniques*



*Figure 100: Bar Chart for Resigned Before and After Class Imbalance*

Figure above shows the distribution of the target variable "Resigned" for both before and after applying the SMOTE to handle the class imbalance.

#### Before SMOTE

There was a significant class imbalance with 45,938 employees who did not resign (class "0") and only 34,062 employees who resigned (class "1"). This imbalance can lead to a model that is biased

toward predicting the majority class, "Not Resigned," since it is more heavily represented in the dataset. In this situation, the model may fail to recognize patterns related to the minority class which could negatively affect its ability to predict employee resignation.

### **After SMOTE**

The class distribution was balanced with both classes ("Not Resigned" and "Resigned") having 45,938 instances each. SMOTE works by generating synthetic examples for the minority class (Resigned) to ensure that both classes are equally represented in the training data. This helps to reduce bias in the model and ensures that it learns patterns from both the majority and minority classes.

Therefore, balancing the dataset is important to check before model building because an imbalanced dataset could skew the model's predictions. After SMOTE, the model is trained on a more balanced dataset which improves its ability to predict both the "Resigned" and "Not Resigned" classes effectively to enhance the overall performance and fairness of the model.

With the class distribution balanced, the dataset is ready for model building and hyperparameter tuning to develop a predictive model that can effectively classify both resigned and non-resigned employees.

## **4.6 Model Building**

In this 4.6 section, four machine learning classification models which is Random Forest, XGBoost, Gradient Boosting and Support Vector Machine (SVM) were developed and evaluated to predict employee churn. These models were selected based on the findings from the literature review in Chapter 2, where similar works have widely adopted these algorithms due to their proven effectiveness in classification tasks particularly in human resource analytics and churn prediction. By reflecting with previous studies, this research aims to ensure a comparison and build upon established machine learning approaches for more stronger performance evaluation.

All models in this study were trained using a resampled training dataset that had been previously balanced using the Synthetic Minority Over-sampling Technique (SMOTE) to solve class imbalance. To ensure methodological consistency, the base models with default parameters and the tuned models have applied Stratified K-Fold cross-validation. Although the dataset was already

balanced by SMOTE, stratification was still implemented to maintain proportional class distribution across folds and to prevent sampling bias during model validation.

For hyperparameter tuning, each model was optimized using RandomizedSearchCV across a comprehensive and model-specific parameter space which are selected based on best practices and recent research. This approach allowed for effective exploration of key hyperparameters such as tree depth, learning rate, regularization terms and kernel-specific settings. The base models were also evaluated using learning curves to assess performance scalability and detect potential underfitting or overfitting. These strategies help to ensure a fair, strong and generalizable comparisons across all machine learning.

## 4.6.1 Random Forest

### 4.6.1.1 Base Model (Default Parameters)

```

import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.ensemble import RandomForestClassifier

# Train on resampled data
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train_resampled, y_train_resampled)

# Predict on test set
y_pred = rf_model.predict(X_test)

# Evaluation
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D91BC" ])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('Random Forest - Confusion Matrix (Base)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'])
plt.yticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

*Figure 101: Source Code of Random Forest Default Tuning*

The figure above shows the implementation of the Random Forest base classification model which was built and evaluated for predicting employee churn. This model was developed using the default parameters of the RandomForestClassifier from the Scikit-learn library and was trained on a training dataset that had been balanced using the SMOTE technique to solve class imbalance. After training, predictions were generated on the original test dataset to assess generalization performance. The model's effectiveness was evaluated using a confusion matrix, classification

report and accuracy score. Further analysis and interpretation of this model's performance are provided in Chapter 5.

#### 4.6.1.2 Tuned Model (Randomized Search CV)

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from matplotlib.colors import LinearSegmentedColormap
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
```

Figure 102: Source Code of Random Forest Hyperparameter Tuning

```
# Parameter space
param_dist = {
    'n_estimators': [100, 200, 300, 400, 500],
    'max_depth': [10, 20, 30, 40],
    'min_samples_split': [5, 10, 15],
    'min_samples_leaf': [2, 4, 6],
    'max_features': ['sqrt', 'log2'],
    'class_weight': ['balanced']
}

# Stratified CV
stratified_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

rf = RandomForestClassifier(random_state=42)
random_search = RandomizedSearchCV(
    estimator=rf,
    param_distributions=param_dist,
    n_iter=360,
    cv=stratified_cv,
    scoring='f1',
    verbose=1,
    random_state=42,
    n_jobs=-1
)
random_search.fit(X_train_resampled, y_train_resampled)

# Evaluate best model
best_randomforest = random_search.best_estimator_
y_pred_best = best_randomforest.predict(X_test)

# Best parameters
best_params = random_search.best_params_
print("\n--- Best Hyperparameters for Random Forest ---")
print(f"Number of Trees (n_estimators): {best_params['n_estimators']}")
print(f"Maximum Depth (max_depth): {best_params['max_depth']}")
print(f"Minimum Samples to Split (min_samples_split): {best_params['min_samples_split']}")
print(f"Minimum Samples at Leaf (min_samples_leaf): {best_params['min_samples_leaf']}")
print(f"Max Features per Split (max_features): {best_params['max_features']}")
print(f"Class Weight: {best_params['class_weight']}")

# Confusion matrix
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_best))
print("\nClassification Report:\n", classification_report(y_test, y_pred_best))
print("Accuracy Score:", accuracy_score(y_test, y_pred_best))

# Confusion matrix plot
cm = confusion_matrix(y_test, y_pred_best)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D9EBCF" ])

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('Random Forest - Confusion Matrix (Tuned)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'])
plt.yticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()
```

Figure 103: Source Code of Random Forest Hyperparameter Tuning

In this section, hyperparameter tuning was applied to the Random Forest model to improve its predictive performance in employee churn classification. The table below summarizes the key hyperparameters selected for optimization based on machine learning best practices. These parameters were tuned by using RandomizedSearchCV with a 5-fold stratified cross-validation strategy and the model was evaluated by using the F1-score as the scoring metric.

Hyperparameter	Explanation	Values/Range
n_estimators	Number of trees in the forest	[100, 200, 300, 400, 500]
max_depth	Maximum depth of each tree	[10, 20, 30, 40]
min_samples_split	Minimum samples required to split a node	[5, 10, 15]
min_samples_leaf	Minimum samples required to be at a leaf node	[2, 4, 6]
max_features	Number of features considered when splitting a node	['sqrt', 'log2']
class_weight	Weight balancing for handling imbalanced classes	['balanced']

Table 6: Hyperparameter Search Space for Random Forest Model

The selected hyperparameter ranges were guided by recent research findings. According to Karimi and Viliyani (2024), tuning Random Forest on HR datasets using similar parameters significantly improved performance to achieve an accuracy of 98.2% in churn classification. Another study by Gazi (2024) also showed that Random Forest achieved superior performance over traditional models when optimized across these parameters. Therefore, these studies support the importance of selecting hyperparameters to ensure better model generalization and more accurate churn predictions.

#### 4.6.1.3 Best Parameters and Accuracy

```
Fitting 5 folds for each of 360 candidates, totalling 1800 fits

--- Best Hyperparameters for Random Forest ---
Number of Trees (n_estimators): 100
Maximum Depth (max_depth): 10
Minimum Samples to Split (min_samples_split): 10
Minimum Samples at Leaf (min_samples_leaf): 2
Max Features per Split (max_features): sqrt
Class Weight: balanced
```

Figure 104: Output of Best Parameters for Random Forest

Accuracy Score: 0.84995

*Figure 105: Output of Random Forest's Accuracy Score*

Figures above display the best hyperparameters and accuracy score for the Random Forest model.

The optimal configuration identified through RandomizedSearchCV is as follows:

- Number of Trees (n\_estimators): 100
- Maximum Depth (max\_depth): 10
- Minimum Samples to Split (min\_samples\_split): 10
- Minimum Samples at Leaf (min\_samples\_leaf): 2
- Max Features per Split (max\_features): sqrt
- Class Weight: balanced

The model achieved an accuracy score of 85% on the test dataset. A detailed analysis of its performance will be further discussed in Chapter 5.

## 4.6.2 XGBoost

### 4.6.2.1 Base Model (Default Parameters)

```

import xgboost as xgb
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap

xgb_model = xgb.XGBClassifier()

xgb_model.fit(X_train_resampled, y_train_resampled)

# Predict on the test set
y_pred = xgb_model.predict(X_test)

# Evaluate performance
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Plot confusion matrix
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D91BC" ])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('XGBoost - Confusion Matrix (Base)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'])
plt.yticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

*Figure 106: Source Code of XGBoost Default Tuning*

The figure above shows the implementation of the XGBoost base classification model for employee churn prediction. The model was built using the default parameters of the XGBClassifier from the XGBoost library and was trained on a dataset balanced using the SMOTE technique to solve class imbalance. After training, the model was used to generate predictions on the test dataset. The model's performance was evaluated using a confusion matrix, classification report and accuracy score which are all imported from Scikit-learn. Further analysis and interpretation of this model's performance will be discussed in Chapter 5.

#### 4.6.2.2 Tuned Model (Randomized Search CV)

```

import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from scipy.stats import randint, uniform
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap
import numpy as np

# StratifiedKFold
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# XGBoost model
xgb_model = xgb.XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    random_state=42
)

# Hyperparameter space
param_dist = {
    'n_estimators': randint(100, 500),
    'max_depth': randint(3, 15),
    'learning_rate': uniform(0.01, 0.1),
    'subsample': uniform(0.6, 0.3),
    'colsample_bytree': uniform(0.6, 0.3),
    'gamma': uniform(0, 5),
    'reg_lambda': uniform(0.1, 1),
    'reg_alpha': uniform(0, 1)
}

# Randomized Search
random_search = RandomizedSearchCV(
    estimator=xgb_model,
    param_distributions=param_dist,
    n_iter=500,
    scoring='f1',
    cv=cv,
    verbose=2,
    random_state=42,
    n_jobs=-1
)
random_search.fit(X_train_resampled, y_train_resampled)

# Best estimator
best_xgboost = random_search.best_estimator_
y_pred = best_xgboost.predict(X_test)

```

Figure 107: Source Code of XGBoost Hyperparameter Tuning

```

# Best hyperparameter
best_params = random_search.best_params_
print("\n--- Best Hyperparameters for XGBoost ---")
print(f"Number of Trees (n_estimators): {best_params['n_estimators']}")
print(f"Maximum Tree Depth (max_depth): {best_params['max_depth']}")
print(f"Learning Rate: {best_params['learning_rate']:.4f}")
print(f"Subsample Ratio: {best_params['subsample']:.4f}")
print(f"Column Subsample (colsample_bytree): {best_params['colsample_bytree']:.4f}")
print(f"Gamma (Minimum Loss Reduction): {best_params['gamma']:.4f}")
print(f"L2 Regularization (reg_lambda): {best_params['reg_lambda']:.4f}")
print(f"L1 Regularization (reg_alpha): {best_params['reg_alpha']:.4f}")

# Evaluation
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Confusion Matrix Plot
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#FFFFFF", "#D91B0C" ])

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('XGBoost - Confusion Matrix (Tuned)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks([0, 1], ['Not Resigned', 'Resigned'])
plt.yticks([0, 1], ['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

*Figure 108: Source Code of XGBoost Hyperparameter Tuning*

In this section, hyperparameter tuning was applied to the XGBoost model to improve its predictive performance in employee churn classification. The table below summarizes the key hyperparameters selected for optimization based on established best practices in machine learning. These parameters were tuned using RandomizedSearchCV with a 5-fold stratified cross-validation strategy, and the model was evaluated using the F1-score as the scoring metric.

Hyperparameter	Explanation	Values / Range
n_estimators	Number of boosting rounds or trees	randint (100, 500)
max_depth	Maximum tree depth to control complexity	randint (3, 15)
learning_rate	Shrinkage rate for leaf weights	uniform (0.01, 0.1)
subsample	Fraction of training data per tree to control overfitting	uniform (0.6, 0.3)
colsample_bytree	Fraction of features sampled per tree	uniform (0.6, 0.3)
gamma	Minimum loss reduction required for a split	uniform (0, 5)
reg_lambda	L2 regularization term on weights	uniform (0.1, 1)
reg_alpha	L1 regularization term on weights	uniform (0, 1)

*Table 7: Hyperparameter Search Space for XGBoost Model*

The selected hyperparameter ranges were guided by recent research. A comparative study by Imani (2023) highlighted tuning parameters such as max\_depth, learning\_rate, n\_estimators and regularization terms to enhance XGBoost's performance in employee churn prediction. There has

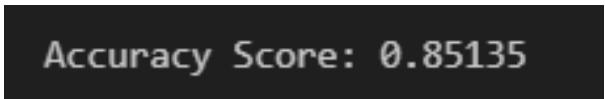
another study by Nan and Zhang (2023) which showed that XGBoost outperformed logistic regression and decision trees when key hyperparameters like subsample, colsample\_bytree and gamma were fine-tuned. These findings support the importance of hyperparameter optimization in building robust and generalizable XGBoost models for churn prediction.

#### 4.6.2.3 Best Parameters and Accuracy

```
Fitting 5 folds for each of 500 candidates, totalling 2500 fits

--- Best Hyperparameters for XGBoost ---
Number of Trees (n_estimators): 305
Maximum Tree Depth (max_depth): 5
Learning Rate: 0.0353
Subsample Ratio: 0.8999
Column Subsample (colsample_bytree): 0.8688
Gamma (Minimum Loss Reduction): 4.0901
L2 Regularization (reg_lambda): 0.4370
L1 Regularization (reg_alpha): 0.1254
```

*Figure 109: Output of Best Parameters for XGBoost*



**Accuracy Score: 0.85135**

*Figure 110: Output of XGBoost's Accuracy Score*

The figures above display the best hyperparameters and accuracy score for the XGBoost model. The optimal configuration identified through tuning includes the following:

- Number of Trees (n\_estimators): 305
- Maximum Tree Depth (max\_depth): 5
- Learning Rate: 0.0353
- Subsample Ratio: 0.8999
- Column Subsample (colsample\_bytree): 0.8688
- Gamma (Minimum Loss Reduction): 4.0901
- L2 Regularization (reg\_lambda): 0.4370
- L1 Regularization (reg\_alpha): 0.1254

The model achieved an accuracy score of 85.14% when tested on the test dataset. Further analysis and interpretation of the model's performance will be discussed in Chapter 5.

## 4.6.3 Gradient Boosting

### 4.6.3.1 Base Model (Default Parameters)

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

gb_model = GradientBoostingClassifier()
gb_model.fit(X_train_resampled, y_train_resampled)

# Predict on test set
y_pred = gb_model.predict(X_test)

# Evaluate
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Plot confusion matrix
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D91BC" ])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('Gradient Boosting - Confusion Matrix (Base)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks([0.5, 1.5], ['Not Resigned', 'Resigned'])
plt.yticks([0.5, 1.5], ['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

*Figure 111: Source Code of Gradient Boosting Default Tuning*

The figure above shows the implementation of the Gradient Boosting base classification model for employee churn prediction. The model was built using the GradientBoostingClassifier with default parameters and trained on a dataset balanced using the SMOTE technique to solve class imbalance. After training, the model was used to predict outcomes on the test dataset. The model's performance was evaluated using standard classification metrics such as the confusion matrix, classification report and accuracy score. These evaluation metrics were imported from the Scikit-learn library. Further analysis and interpretation of this model's performance will be presented in Chapter 5.

#### 4.6.3.2 Tuned Model (Randomized Search CV)

```

from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from scipy.stats import randint, uniform
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap
import numpy as np

# Hyperparameter space
param_dist = {
    'n_estimators': randint(50, 150),
    'learning_rate': uniform(0.01, 0.05),
    'max_depth': randint(3, 5),
    'min_samples_split': randint(10, 30),
    'min_samples_leaf': randint(5, 15),
    'subsample': uniform(0.7, 0.3)
}

# Cross-validation strategy
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Initialize model
gb = GradientBoostingClassifier(random_state=42)

# Randomized Search CV
random_search = RandomizedSearchCV(
    estimator=gb,
    param_distributions=param_dist,
    n_iter=500,
    scoring='f1',
    cv=cv,
    verbose=2,
    n_jobs=-1,
    random_state=42
)
random_search.fit(X_train_resampled, y_train_resampled)

# Best model
best_gradientboost = random_search.best_estimator_
y_pred = best_gradientboost.predict(X_test)

```

Figure 112: Source Code of Gradient Boosting Hyperparameter Tuning

```

# Best hyperparameter
best_params = random_search.best_params_
print("\n--- Best Hyperparameters for Gradient Boosting ---")
print(f"Number of Trees (n_estimators): {best_params['n_estimators']}")
print(f"Learning Rate: {best_params['learning_rate']:.4f}")
print(f"Maximum Tree Depth (max_depth): {best_params['max_depth']}")
print(f"Min Samples to Split (min_samples_split): {best_params['min_samples_split']}")
print(f"Min Samples at Leaf (min_samples_leaf): {best_params['min_samples_leaf']}")
print(f"Subsample Ratio: {best_params['subsample']:.4f}")

# Evaluation
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D9EBCF" ])

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('Gradient Boosting - Confusion Matrix (Tuned)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'])
plt.yticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

Figure 113: Source Code of Gradient Boosting Hyperparameter Tuning

In this section, hyperparameter tuning was applied to the Gradient Boosting Classifier to enhance its predictive performance in employee churn forecasting. The table below summarizes the key hyperparameters selected for optimization based on machine learning best practices. These were tuned using RandomizedSearchCV with 5-fold stratified cross-validation and performance was evaluated using the F1-score.

Hyperparameter	Explanation	Values / Range
n_estimators	Number of boosting rounds (trees)	randint (50, 150)
learning_rate	Shrinkage rate to control learning pace	uniform (0.01, 0.05)
max_depth	Maximum depth of each tree to manage complexity	randint (3, 5)
min_samples_split	Minimum samples required to split a node	randint (10, 30)
min_samples_leaf	Minimum samples required to be at a leaf node	randint (5, 15)
subsample	Fraction of samples used per tree to prevent overfitting	uniform (0.7, 0.3)

*Table 8: Hyperparameter Search Space for Gradient Boosting Model*

The selected hyperparameter ranges were guided by recent research findings that show the importance of tuning tree-specific parameters to improve model generalization and performance. A 2023 study by Sharma and Kumar found that adjusting max\_depth, learning\_rate and n\_estimators helps to enhance Gradient Boosting performance in churn prediction when handling with class imbalance. Similarly, research conducted by Dimas et al. (2021) emphasized the role of subsample and min\_samples\_leaf in preventing overfitting and improving model stability across various HR datasets. These studies collectively support that careful optimization of these parameters is essential to effectively capture the complex patterns associated with employee attrition. The chosen hyperparameter space in this study ensures a balanced exploration of depth, learning rate, regularization and sampling control leads to a more robust and interpretable Gradient Boosting model for churn classification.

#### 4.6.3.3 Best Parameters and Accuracy

```
Fitting 5 folds for each of 500 candidates, totalling 2500 fits

--- Best Hyperparameters for Gradient Boosting ---
Number of Trees (n_estimators): 136
Learning Rate: 0.0591
Maximum Tree Depth (max_depth): 4
Min Samples to Split (min_samples_split): 12
Min Samples at Leaf (min_samples_leaf): 9
Subsample Ratio: 0.7594
```

Figure 114: Output of Best Parameters for Gradient Boosting

**Accuracy Score: 0.8514**

Figure 115: Output of Gradient Boost's Accuracy Score

The figures above display the best hyperparameters and accuracy score for the Gradient Boosting model. The optimal configuration identified includes the following:

- Number of Trees (n\_estimators): 136
- Learning Rate: 0.0591
- Maximum Tree Depth (max\_depth): 4
- Min Samples to Split (min\_samples\_split): 12
- Min Samples at Leaf (min\_samples\_leaf): 9
- Subsample Ratio: 0.7594

The model achieved an accuracy score of 85.14% when tested on the test dataset. Further analysis and interpretation of the model's performance will be discussed in Chapter 5.

#### 4.6.4 Support Vector Machine

It is important to note that for the Support Vector Machine (SVM) model, there is only 50% of the training dataset which total of 45,938 samples was used for both training and hyperparameter tuning. This approach was adopted due to the high computational complexity of SVM when using the Radial Basis Function (RBF) kernel. SVMs require significant computational power when performing hyperparameter tuning with RandomizedSearchCV to optimize parameters such as C and gamma.

Initially, an attempt was made to tune the model using the full training dataset which consists of 91,876 samples. However, due to the high computational complexity of SVM and the limitations of available computational resources, this configuration failed to complete within a reasonable time frame despite extended training attempts. As a result, a representative subset of 45,938 samples was selected for training the SVM. This subset provided a reasonable balance between model performance and computational efficiency to ensure that the model could generalize effectively.

Although other models such as Random Forest, XGBoost, and Gradient Boosting were trained using the full dataset, the SVM model was tuned and trained on a smaller subset due to its higher computational demands. Despite using only 50% of the training data, the SVM model demonstrated strong performance. All models were evaluated on the same test dataset to ensure consistency and fairness in the comparison.

#### 4.6.4.1 Base Model (Default Parameters)

```

from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap
import numpy as np

# Take 50% sample subset from the resampled training data
subset_indices = np.random.choice(len(X_train_resampled), size=45938, replace=False)
X_train_subset = X_train_resampled.iloc[subset_indices]
y_train_subset = y_train_resampled.iloc[subset_indices]

svm_model_subset = SVC()
svm_model_subset.fit(X_train_subset, y_train_subset)

y_pred = svm_model_subset.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D291BC"])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('SVM - Confusion Matrix (Base)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks([0.5, 1.5], ['Not Resigned', 'Resigned'])
plt.yticks([0.5, 1.5], ['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

*Figure 116: Source Code of Support Vector Machine Default Tuning*

The figure above shows the implementation of the Support Vector Machine (SVM) base classification model for employee churn prediction. The model was constructed using the SVC class from the Scikit-learn library with all default parameter settings. This base model was trained on the resampled training dataset which had been previously balanced using the SMOTE technique to solve class imbalance. After training, the model was used to generate predictions on the test dataset. The model's performance was evaluated using standard classification metrics such as the confusion matrix, classification report and accuracy score. Further analysis and interpretation of this model's performance will be presented in Chapter 5.

#### 4.6.4.2 Tuned Model (Randomized Search CV)

```

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from scipy.stats import uniform

# Define hyperparameter space
param_dist = {
    'svm_C': uniform(0.1, 10),
    'svm_gamma': uniform(0.001, 1),
    'svm_kernel': ['rbf']
}

# Build SVM pipeline
svm_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('svm', SVC(probability=True, random_state=42))
])

# Take 50% sample subset from resampled training data
subset_indices = np.random.choice(X_train_resampled.shape[0], size=45938, replace=False)
X_train_sampled = X_train_resampled.iloc[subset_indices]
y_train_sampled = y_train_resampled.iloc[subset_indices]

# Stratified 5-Fold Cross-Validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Perform RandomizedSearchCV
random_search = RandomizedSearchCV(
    estimator=svm_pipeline,
    param_distributions=param_dist,
    n_iter=20,
    scoring='f1',
    cv=cv,
    verbose=2,
    n_jobs=-1,
    random_state=42
)
random_search.fit(X_train_sampled, y_train_sampled)

# Best model and predictions
best_svm_model = random_search.best_estimator_
y_pred = best_svm_model.predict(X_test)

```

Figure 117: Source Code of Support Vector Machine Hyperparameter Tuning

```

# Best parameters
best_params = random_search.best_params_
print("\n--- Best Hyperparameters for Support Vector Machine ---")
print(f"Regularization Parameter (C): {best_params['svm_C']:.4f}")
print(f"Kernel Coefficient (gamma): {best_params['svm_gamma']:.4f}")
print(f"Kernel Type: {best_params['svm_kernel']}")

# Evaluation
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D291BC"])

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('Support Vector Machine - Confusion Matrix (Tuned)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'])
plt.yticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

Figure 118: Source Code of Support Vector Machine Hyperparameter Tuning

In this section, hyperparameter tuning was applied to the Support Vector Machine (SVM) model using RandomizedSearchCV to enhance predictive accuracy in employee churn classification. The model was built through a pipeline that included feature standardization with StandardScaler and classification using SVC from Scikit-learn. The tuning was performed using a stratified 5-fold cross-validation strategy and evaluated using the F1-score due to class imbalance.

The table below summarizes the key hyperparameters selected for SVM optimization based on machine learning best practices.

Hyperparameter	Explanation	Values / Range
svm_C	Regularization parameter that controls the trade-off between training error and margin maximization	uniform (0.1, 10)
svm_gamma	Kernel coefficient for non-linear hyperplanes to define influence of a single training example	uniform (0.001, 1)
svm_kernel	Specifies the kernel type to be used in the algorithm	[‘rbf’]

*Table 9: Hyperparameter Search Space for Support Vector Machine Model*

The selected hyperparameter ranges were guided by recent research findings that show the importance of tuning tree-specific parameters to improve model generalization and performance.

There are several recent studies have highlighted the significance of tuning these hyperparameters. Research by Vijaya Saradhi (2025) emphasized that the application of RandomizedSearchCV in tuning C and gamma improves model performance when computational resources are constrained. Similarly, Guido et al. (2023) demonstrated that fine-tuning C and gamma enhances the F1-score in churn prediction tasks when dealing with imbalanced datasets. Villalobos-Arias et al. (2020) also showed that the use of the RBF kernel in combination with properly tuned C and gamma resulted in high predictive accuracy in HR analytics.

Moreover, Wainer and Fonseca (2020) found that a randomized search over continuous ranges of C and gamma using an RBF kernel could approximate optimal performance without the computational burden of an exhaustive grid search. These findings collectively support the decision to focus on C, gamma and the RBF kernel in this study. The chosen tuning ranges allow for capturing complex and nonlinear relationships in the employee churn dataset while maintaining computational efficiency.

#### 4.6.4.3 Best Parameters and Accuracy

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
--- Best Hyperparameters for Support Vector Machine ---
Regularization Parameter (C): 6.1754
Kernel Coefficient (gamma): 0.1715
Kernel Type: rbf

Confusion Matrix:
[[9201 2283]
 [1311 7205]]
```

Figure 119: Output of Best Parameters for Support Vector Machine

```
Accuracy Score: 0.8203
```

Figure 120: Output of Support Vector Machine's Accuracy Score

The figure above displays the best hyperparameters and corresponding evaluation results for the Support Vector Machine (SVM) model. The optimal configuration identified includes the following:

- Regularization Parameter (C): 6.1754
- Kernel Coefficient (gamma): 0.1715
- Kernel Type: rbf

The model achieved an accuracy score of 82.03% when tested on the test dataset. Further analysis and interpretation of the model's performance will be discussed in Chapter 5.

#### 4.7 Summary

This chapter detailed the complete design and implementation process undertaken to develop a predictive model for employee churn. It began with data collection and initial data understanding, where the dataset was imported, its structure reviewed, and exploratory analyses conducted. This included viewing data types, generating descriptive statistics with visualizing both categorical and numerical variables to understand their distributions and relationships. The target variable, “Resigned,” was also analysed and found the class imbalance issue.

During the data preprocessing phase, outliers were solved and data transformations such as encoding categorical features and feature engineering were applied to improve the dataset's suitability for modeling. Mutual Information (MI) scores were used for feature selection and the top 10 features which are five categorical and five numerical were selected to ensure fair representation and reduce bias. Then, the dataset was split into training (80%) and test (20%) sets with SMOTE techniques only applied to the training set to correct class imbalance and prevent biased learning.

The second part of the chapter focused on model building. Four machine learning classification models which has included Random Forest, XGBoost, Gradient Boosting and Support Vector Machine (SVM) were built and assessed. Each model was first implemented with default (base) parameters to establish a baseline and followed by hyperparameter tuning using RandomizedSearchCV.

All models were trained on the SMOTE-balanced training dataset. Both base and tuned models employed Stratified K-Fold cross-validation to preserve class distribution in each fold and reduce sampling bias. Hyperparameter tuning was performed using model-specific parameter spaces informed by machine learning best practices and supported by recent research.

One important aspect to highlight is the handling of the Support Vector Machine (SVM) model. Due to its computational intensity, hyperparameter tuning was conducted on a representative 45,938-sample subset of the full 91,876-sample training set. This configuration achieved better generalization performance and faster runtime compared to larger subsets. Meanwhile, the other models were tuned using the full SMOTE-balanced dataset.

Each model's performance was evaluated using standard classification metrics including accuracy, confusion matrix, and classification report. This chapter established a strong data pipeline which was from preprocessing and feature engineering to balanced resampling and model optimization. The consistent methodology and careful tuning across all models ensure fair, generalizable and reliable predictive results. A detailed evaluation and comparison for each of the model performances will be discussed in Chapter 5.

## Chapter 5: Result and Discussion

### 5.1 Introduction

This chapter focuses on the evaluation and discussion of the machine learning models developed to predict employee churn. The four classification algorithms which are Random Forest, XGBoost, Gradient Boosting and Support Vector Machine (SVM) are assessed to determine their effectiveness in identifying employees who are likely to resign. The evaluation compares both base and tuned models to observe the impact of hyperparameter tuning on prediction accuracy and generalization.

Each model is evaluated by using standard classification metrics such as accuracy, F1-score, confusion matrix and Area Under the ROC Curve (AUC-ROC). These metrics provide insights into the reliability of the model predictions. Besides, learning curves are analysed to identify underfitting or overfitting patterns while ROC curves are used to visualize the trade-off between true positive and false positive rates.

The evaluation of each algorithm includes three key components: the classification report and confusion matrix, the learning curve, and the ROC-AUC analysis. These results form the basis for a comparative discussion to identify the best-performing model in terms of both predictive performance and computational efficiency. Feature importance analysis is also conducted for tree-based models to understand the most influential factors driving churn. Lastly, the chapter briefly outlines model deployment considerations.

Overall, this chapter aims to provide a comprehensive discussion of model performance, supported by both quantitative results and visual analyses, to determine the most suitable model for employee churn prediction and workforce decision-making.

## 5.2 Model Evaluation and Discussions

### 5.2.1 Random Forest Base Model

#### 5.2.1.1 Confusion Matrix and Classification Report

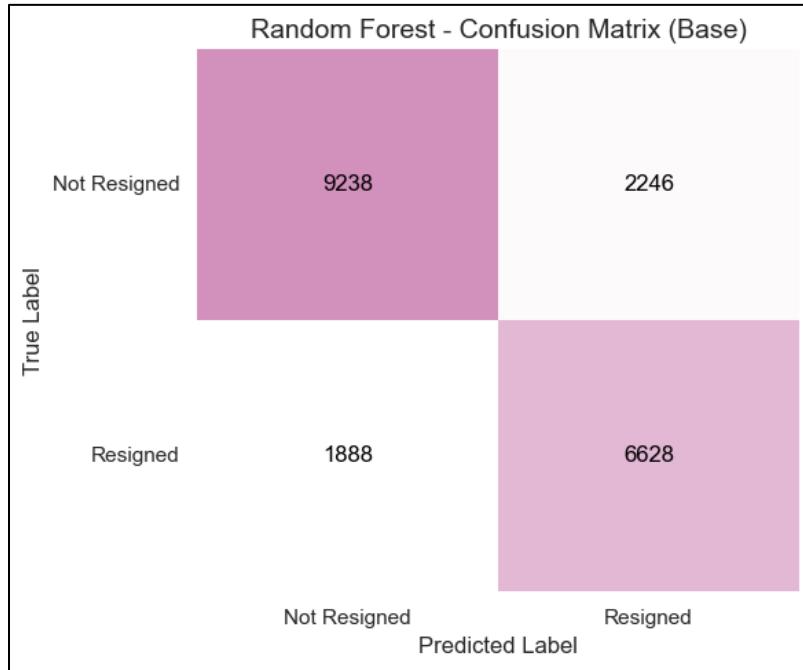


Figure 121: Confusion Matrix for Random Forest Base Model

Figure above shows the **confusion matrix** for the **Random Forest Base Model** used in predicting employee churn. The model effectively differentiates between employees who have resigned and non-resigned. Specifically, it correctly identified 9,238 employees who did not resign (True Negatives) and 6,628 employees who did resign (True Positives). However, 2,246 employees were misclassified as resigned despite not resigning (False Positives) while 1,888 actual resignations were incorrectly predicted as not resigned (False Negatives).

Confusion Matrix:				
[[9238 2246] [1888 6628]]				
Classification Report:				
	precision	recall	f1-score	support
0	0.83	0.80	0.82	11484
1	0.75	0.78	0.76	8516
accuracy			0.79	20000
macro avg	0.79	0.79	0.79	20000
weighted avg	0.79	0.79	0.79	20000
Accuracy Score: 0.7933				

*Figure 122: Classification Report for Random Forest Base Model*

The **classification report** provides a more detailed evaluation of the **Random Forest Base Model**. For employees who did not resign (class 0), the model achieved a precision of 83%, recall of 80% and an F1-score of 82%. For resigned employees (class 1), the precision was 75%, recall was 78% and F1-score was 76%. The overall accuracy of the model is 79.3%.

Overall, the model performs well in identifying employee churn and retention with a balanced trade-off between precision and recall. However, improvements could be made in reducing false positives and false negatives to enhance its reliability for decision-making in human resource planning.

### 5.2.1.2 Learning Curve

```

from sklearn.model_selection import learning_curve, StratifiedKFold
import numpy as np

# Stratified Cross-Validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Learning curve
train_sizes, train_scores, test_scores = learning_curve(
    estimator=rf_model,
    X=X_train_resampled,
    y=y_train_resampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

# Mean and Std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Learning curve
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label='Training Score', color='blue')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color='blue')
plt.plot(train_sizes, test_mean, label='Cross-validation Score', color='red')
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color='red')
plt.title('Learning Curve - Random Forest (Base)', fontsize=14)
plt.xlabel('Training Set Size')
plt.ylabel('F1 Score')
plt.legend(loc='best')
plt.grid(True)
plt.tight_layout()
plt.show()

```

*Figure 123: Source Code for Learning Curve of Random Forest Base Model*

The figure above shows the source code for generating a learning curve of the Random Forest Base Model using stratified 5-fold cross-validation. The “learning\_curve()” function from “sklearn.model\_selection” is used to evaluate the model’s performance across varying training set sizes. The F1 score is chosen as the performance metric as it is suitable for imbalanced classification tasks such as employee churn prediction.

The code calculates the mean and standard deviation of both training and cross-validation scores which are plotted using Matplotlib. The blue curve represents the training score while the red curve shows the cross-validation score with shaded areas indicating one standard deviation from the mean.

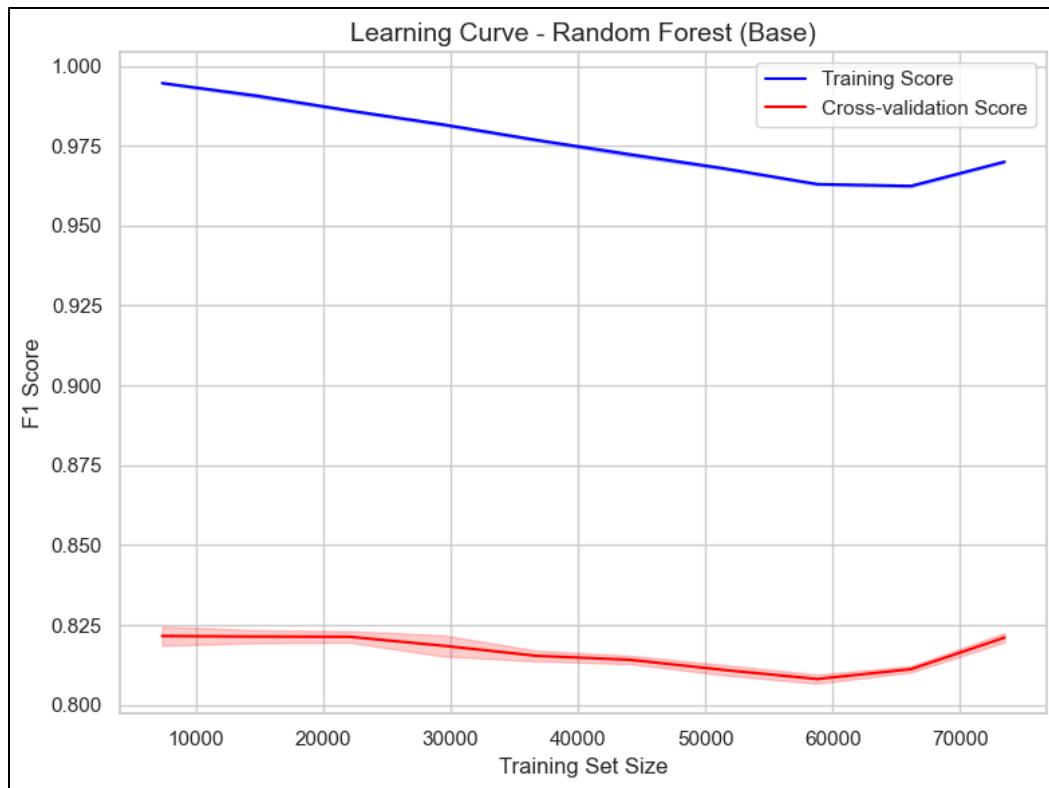


Figure 124: Learning Curve of Random Forest Base Model

The **learning curve for the Random Forest Base Model** indicates an overfitting. The training F1 score remains consistently high, which is above 0.97 across all training sizes. This shows that the model performs exceptionally well on seen data. However, the cross-validation F1 score stops improving and remains steady around 0.82 of training set size and even slightly decreases before slightly rising again. This consistent gap between training and validation performance suggests that the model has high variance as it fits the training data very well but struggles to generalize to unseen data.

This issue will be solved during the model tuning phase. The techniques such as hyperparameter tuning, regularization or model ensembling could help mitigate overfitting and improve generalization.

### 5.2.1.3 ROC AUC Score

```

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

y_probs = rf_model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_probs)
auc_score = roc_auc_score(y_test, y_probs)

plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})')
plt.plot([0,1], [0,1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Random Forest (Base)', fontsize=14)
plt.legend()
plt.grid()
plt.show()

```

*Figure 125: Source Code for ROC AUC Score of Random Forest Base Model*

The figure above shows the source code of the ROC (Receiver Operating Characteristic) curve for the Random Forest Base Model. It begins by importing a few functions from “sklearn.metrics”, “roc\_curve” to compute the false positive rate (FPR) and true positive rate (TPR) and “roc\_auc\_score” to calculate the AUC (Area Under the Curve) score. The predicted probabilities of the positive class (class 1) are obtained from the model using “predict\_proba”.

Then, The ROC curve is plotted using “matplotlib.pyplot”, where the x-axis represents the false positive rate and the y-axis represents the true positive rate. Additionally, the AUC score is displayed on the graph to provide a numerical summary of the model's performance.

The plot includes a grid for better readability and the dashed line represents the baseline for random classification. The model's performance is assessed by comparing how far the ROC curve deviates from the diagonal line with a greater deviation indicating better performance.

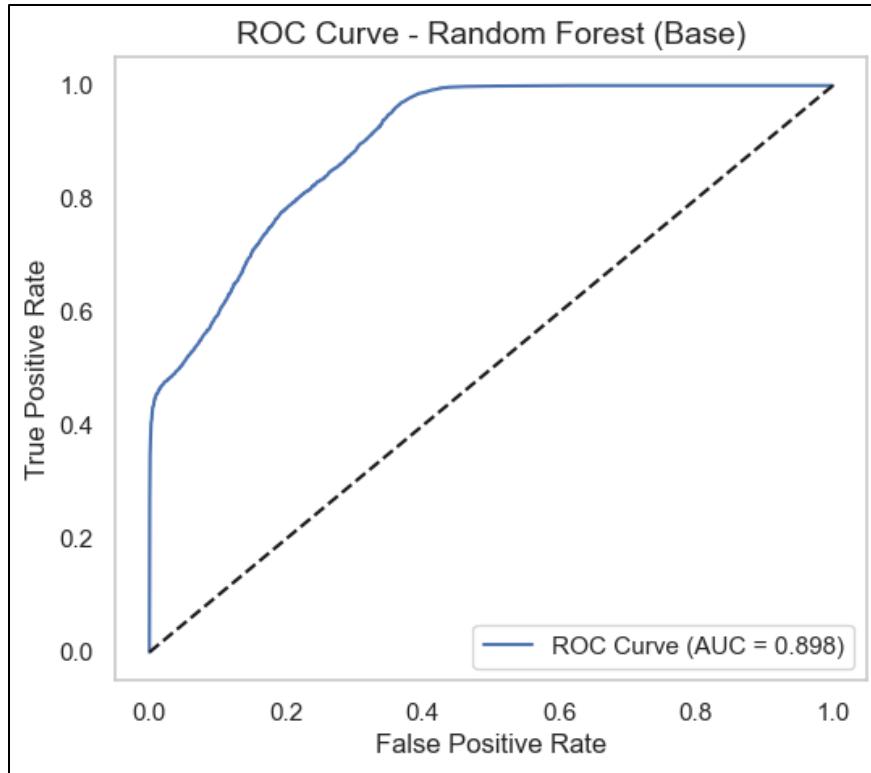


Figure 126: ROC AUC Score of Random Forest Base Model

The **ROC curve for the Random Forest Base Model** shows a strong classification performance. The curve bows significantly toward the top-left corner which has indicating a high true positive rate and a low false positive rate across various threshold values. The Area Under the Curve (**AUC**) is **0.898** which suggests that the model is highly capable of distinguishing between resigned and non-resigned employees. A perfect model would have an AUC of 1.0, while a purely random model would score 0.5. Therefore, an AUC of 0.898 reflects a well-performing model with a good balance between sensitivity and specificity but its performance could be further improved during the tuning phase.

## 5.2.2 Random Forest Tuned Model

### 5.2.2.1 Confusion Matrix and Classification Report

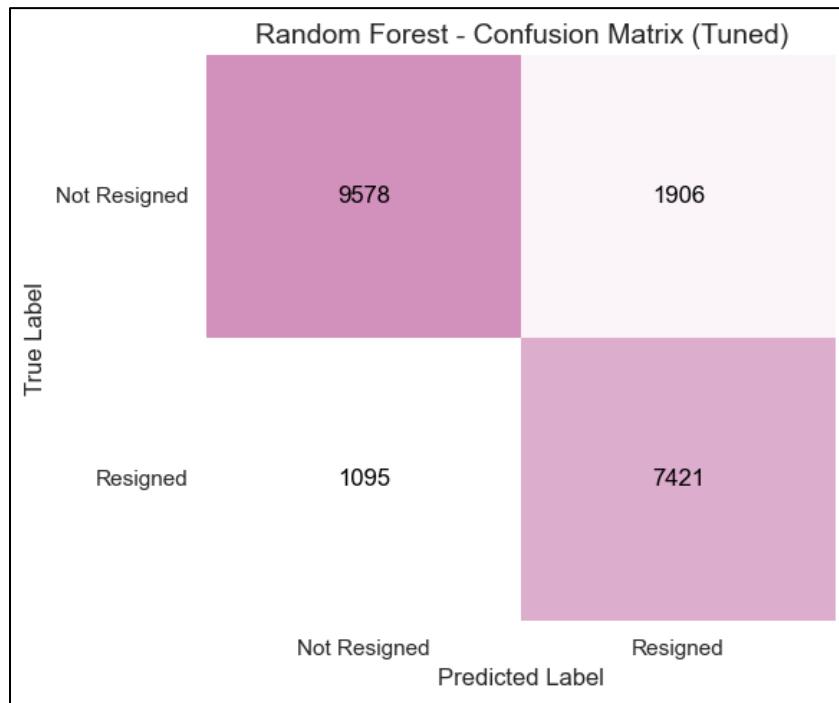


Figure 127: Confusion Matrix for Random Forest Tuned Model

The figure above shows the **confusion matrix for the Random Forest Tuned Model** used in predicting employee churn. The model effectively differentiates between employees who have resigned and non-resigned. Specifically, it correctly identified 9,578 employees who did not resign (True Negatives) and 7,421 employees who did resign (True Positives). However, 1,906 employees were misclassified as resigned despite not resigning (False Positives) while 1,095 actual resignations were incorrectly predicted as not resigned (False Negatives).

Confusion Matrix:				
[[9578 1906] [1095 7421]]				
Classification Report:				
	precision	recall	f1-score	support
0	0.90	0.83	0.86	11484
1	0.80	0.87	0.83	8516
accuracy			0.85	20000
macro avg	0.85	0.85	0.85	20000
weighted avg	0.85	0.85	0.85	20000
Accuracy Score: 0.84995				

*Figure 128: Classification Report for Random Forest Tuned Model*

The **classification report** provides a more detailed evaluation of the **Random Forest Tuned Model**. For employees who did not resign (class 0), the model achieved a precision of 90%, a recall of 83% and an F1-score of 86% which indicated strong and improved performance in identifying those likely to stay. For resigned employees (class 1), the precision was 80%, recall was 87% and F1-score was 83% which reflected an improvement in detecting actual churn cases. The overall accuracy of the model is 84.99%, which rounded off to 85%.

This tuned model demonstrates enhanced predictive performance in minimizing the misclassification of resigned employees. The improvement in recall for class 1 allows HR teams to better identify employees at risk of leaving to enable more timely retention strategies. Additionally, the increased precision for class 0 reduces the likelihood of non-resigned employees being incorrectly flagged for resignation to help HR teams avoid unnecessary interventions.

Overall, the tuned model demonstrates enhanced predictive performance compared to the base model, especially in minimizing misclassification of resigned employees. This improvement makes it more reliable and effective for supporting strategic human resource decisions.

### 5.2.2.1.1 Comparison Results of Base and Tuned Models

Model Type	Accuracy	Precision (Class 0)	Precision (Class 1)	Recall (Class 0)	Recall (Class 1)	F1-Score (Macro Average)
<b>Random Forest (Base Model)</b>	0.79	0.83	0.75	0.80	0.78	0.79
<b>Random Forest (Tuned Model)</b>	0.85	0.90	0.80	0.83	0.87	0.85

*Table 10: Comparison of Base and Tuned Model (Random Forest)*

The table above shows a **comparison between the Base Model and the Tuned Model** which has highlighted improvements especially in the recall for resigned employees (class 1).

All performance metrics have increased in the Tuned Model. For example, recall for class 1 from 78% in the Base Model to 87% in the Tuned Model. This reflects the model's enhanced ability to correctly identify employees at risk of resigning and reducing the number of false negatives which is the employees who resigned but were incorrectly predicted to stay. This improvement in recall is significant because it reduces the number of false negative employees who resigned but were incorrectly predicted to stay. By reducing false negatives, the Tuned Model enables HR teams to take timely action and intervene with employees who are likely to leave which helps in improving retention strategies and workforce planning.

Additionally, the overall accuracy of the Tuned Model increased to 85% compared to 79% for the Base Model and emphasized that the tuned model's improved ability to predict both non-resigned and resigned employees. This higher accuracy emphasizes the Tuned Model's better ability to predict both non-resigned and resigned employees to give HR teams a more reliable tool for decision-making. The precision for both classes also improved with class 1 rising from 75% to 80% and showing that the tuned model is more reliable in predicting resigned employees. The improved precision for both classes ensures better resource allocation for retention efforts.

The improved precision for both classes ensures better resource allocation for retention efforts. The improvements in recall for class 1 (resigned employees) and the increase in precision help to make the Tuned Model more balanced and effective. It not only provides more accurate predictions

but also enhances HR teams' ability to intervene early to make informed decisions and allocate resources efficiently.

The Tuned Model is better than the Base Model because tuning enhanced its ability to identify employees at risk of resigning by improving both precision and recall for predicting both non-resigned and resigned employees. The increase in recall for resigned employees (from 78% to 87%) shows that the Tuned Model is much more capable of identifying employees who are likely to leave, reducing false negatives and allowing HR teams to intervene more effectively. Additionally, the improvement in accuracy from 79% to 85% and the increase in precision for both classes ensure that the Tuned Model provides more reliable predictions and reduces both false positive and false negatives. This makes the Tuned Model a more balanced and effective solution for HR teams, as it improves workforce planning, retention efforts and resource allocation by identifying the right employees to target for intervention. The Tuned Model is better optimized to the dataset and offers a more dependable tool for data-driven HR decisions.

### 5.2.2.2 Learning Curve

```

from sklearn.model_selection import learning_curve
import numpy as np
import matplotlib.pyplot as plt

# Learning curve
train_sizes, train_scores, test_scores = learning_curve(
    estimator=best_randomforest,
    X=X_train_resampled,
    y=y_train_resampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

# Means and Std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

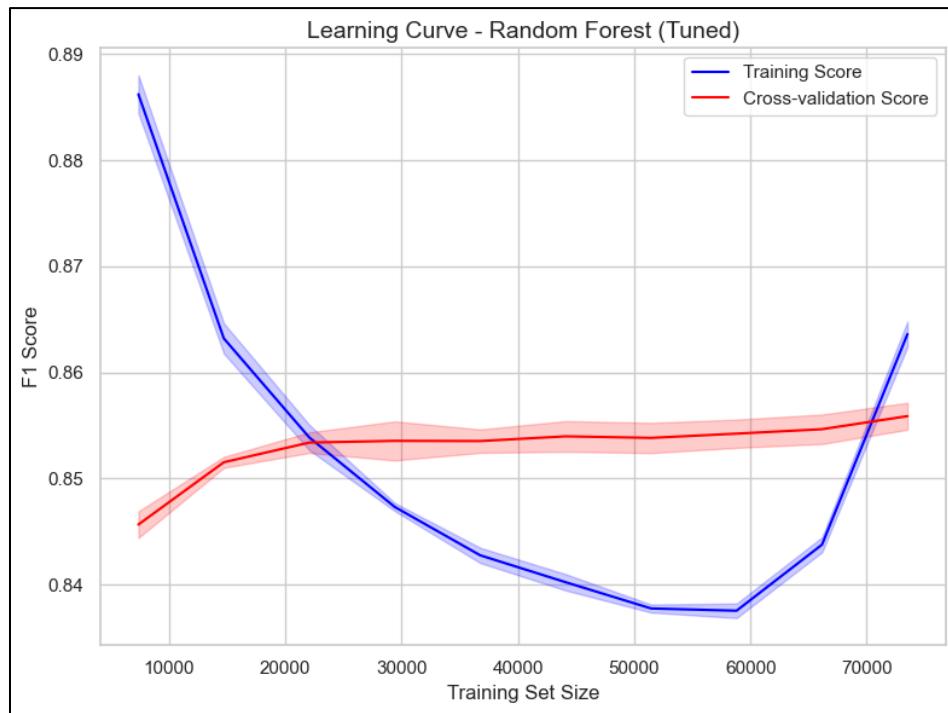
# Learning curve
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label='Training Score', color='blue')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color='blue')
plt.plot(train_sizes, test_mean, label='Cross-validation Score', color='red')
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color='red')
plt.title('Learning Curve - Random Forest (Tuned)', fontsize=14)
plt.xlabel('Training Set Size')
plt.ylabel('F1 Score')
plt.legend(loc='best')
plt.grid(True)
plt.tight_layout()
plt.show()

```

*Figure 129: Source Code for Learning Curve of Random Forest Tune Model*

The figure above shows the source code for generating a learning curve of the Random Forest Tuned Model using stratified 5-fold cross-validation. The “learning\_curve()” function from “sklearn.model\_selection” is used to evaluate the model’s performance across varying training set sizes. The F1 score is chosen as the performance metric as it is suitable for imbalanced classification tasks such as employee churn prediction.

The code calculates the mean and standard deviation of both training and cross-validation scores which are plotted using Matplotlib. The blue curve represents the training score while the red curve shows the cross-validation score with shaded areas indicating one standard deviation from the mean.



*Figure 130: Learning Curve of Random Forest Tuned Model*

The figure above shows the **learning curve for the Random Forest Tuned Model** which illustrates on the model’s performance evolves with an increasing training set size. Initially, both the training score (in blue) and the cross-validation score (in red) are high. However, as the training set size increases, the training score sharply declines around 20,000. This indicates that the model is becoming less overfitted as it receives more data which allows it to generalize better and avoid merely memorizing the training data.

Conversely, the cross-validation score starts lower but gradually improves as more data is added which suggests that the model’s ability to generalize to unseen data is strengthening. As the training set size continues to grow, the gap between the training score and cross-validation score narrows. This indicates that the model is reaching a point of stability where it performs well on both the training and cross-validation data.

This learning curve demonstrates that the Tuned Model benefits from larger datasets which result in improved performance and better generalization as the amount of data increases. The learning curve for the Tuned Model also shows better stability and generalization compared to the Base Model with the advantages of tuning in handling larger datasets effectively.

### 5.2.2.3 ROC and AUC

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

y_probs = best_randomforest_overfitting.predict_proba(x_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_probs)
auc_score = roc_auc_score(y_test, y_probs)

plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})')
plt.plot([0,1], [0,1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Random Forest (Tuned)', fontsize=14)
plt.legend()
plt.grid()
plt.show()
```

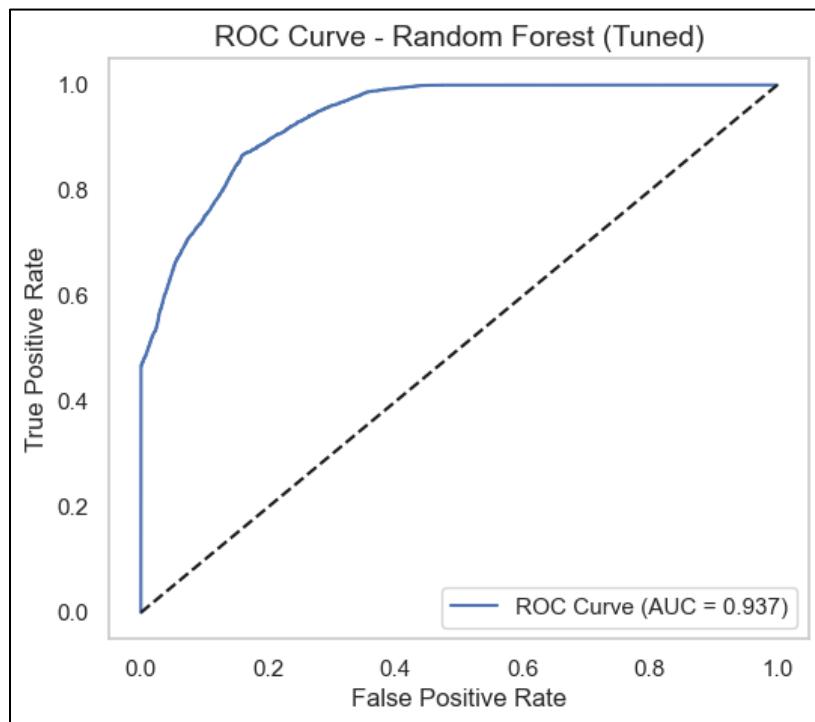
*Figure 131: Source Code for ROC AUC Score of Random Forest Tuned Model*

The figure above shows the source code of the ROC (Receiver Operating Characteristic) curve for the Random Forest Tuned Model. It begins by importing a few functions from “sklearn.metrics”, “roc\_curve” to compute the false positive rate (FPR) and true positive rate (TPR) and

“roc\_auc\_score” to calculate the AUC (Area Under the Curve) score. The predicted probabilities of the positive class (class 1) are obtained from the model using “predict\_proba”.

Then, The ROC curve is plotted using “matplotlib.pyplot”, where the x-axis represents the false positive rate and the y-axis represents the true positive rate. Additionally, the AUC score is displayed on the graph to provide a numerical summary of the model's performance.

The plot includes a grid for better readability and the dashed line represents the baseline for random classification. The model's performance is assessed by comparing how far the ROC curve deviates from the diagonal line with a greater deviation indicating better performance.



*Figure 132: ROC AUC Score of Random Forest Tuned Model*

Figure above displays the **ROC curve for the Tuned Random Forest Model** with significant improvement in the model's ability to discriminate between resigned (class 1) and non-resigned (class 0) employees. The **AUC** score of **0.937** indicates a stronger performance compared to the base model as it shows a higher ability to correctly classify employees. The curve signifies an improved true positive rate (TPR) and a lower false positive rate (FPR). The dashed diagonal line represents random guessing and the Tuned Random Forest curve deviates sharply from this line is

showing its superior performance. This improvement in AUC score from 0.898 (Base Model) to 0.937 (Tuned Model) demonstrates that tuning has significantly enhanced the model's predictive capability to make it more effective for predicting employee churn.

### 5.2.3 XGBoost Base Model

#### 5.2.3.1 Confusion Matrix and Classification Report

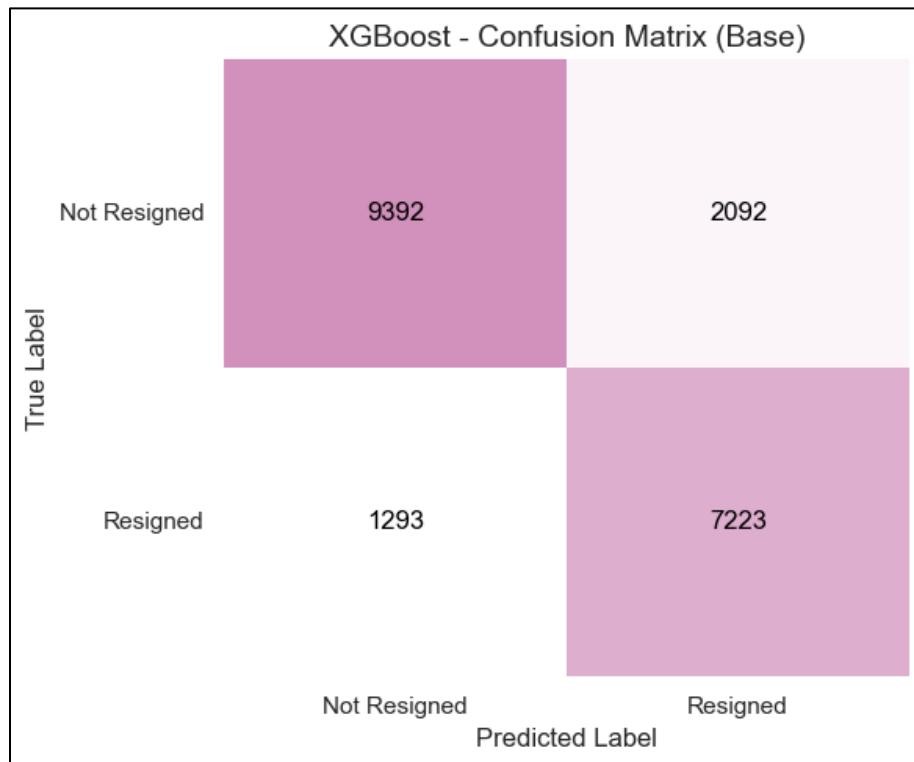


Figure 133: Confusion Matrix for XGBoost Base Model

The **confusion matrix for the XGBoost Base Model** used in predicting employee churn is shown above. The model successfully differentiates between employees who have resigned and those who have not. Specifically, it correctly identified 9,392 employees who did not resign (True Negatives) and 7,223 employees who did resign (True Positives). However, the model misclassified 2,092 non-resigned employees as resigned (False Positives), while 1,293 actual resignations were incorrectly predicted as non-resigned (False Negatives).

Confusion Matrix:				
[[9392 2092] [1293 7223]]				
Classification Report:				
	precision	recall	f1-score	support
0	0.88	0.82	0.85	11484
1	0.78	0.85	0.81	8516
accuracy			0.83	20000
macro avg	0.83	0.83	0.83	20000
weighted avg	0.83	0.83	0.83	20000
Accuracy Score: 0.83075				

*Figure 134: Classification Report for XGBoost Base Model*

The **classification report** above shows the performance of the **XGBoost Base Model**. For non-resigned employees (class 0), the model achieved a precision of 88% which indicated that 88% of the employees were correctly predicted as non-resigned. The recall for class 0 is 82% which indicated that 82% of the actual non-resigned employees were correctly identified. The F1-score of 85% reflects a balanced performance between precision and recall for class 0.

For resigned employees (class 1), the precision is 78% which indicated that 78% predicted resignations were accurate. The recall for class 1 is 85%, showing that 85% of actual resignations were correctly identified. The F1-score of 81% indicates a reasonable balance between precision and recall, though slightly lower than for class 0.

The model's overall accuracy is 83%. Since this is the base model, further tuning is expected to improve its performance particularly in enhancing precision for class 1. This would lead to a more accurate and reliable prediction of employee resignations.

### 5.2.3.2 Learning Curve

```

from sklearn.model_selection import learning_curve, StratifiedKFold
import numpy as np
import matplotlib.pyplot as plt

# Stratified Cross-Validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

train_sizes, train_scores, test_scores = learning_curve(
    estimator=xgb_model,
    X=X_train_resampled,
    y=y_train_resampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=5,
    scoring='f1',
    n_jobs=-1
)

# Mean and Std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

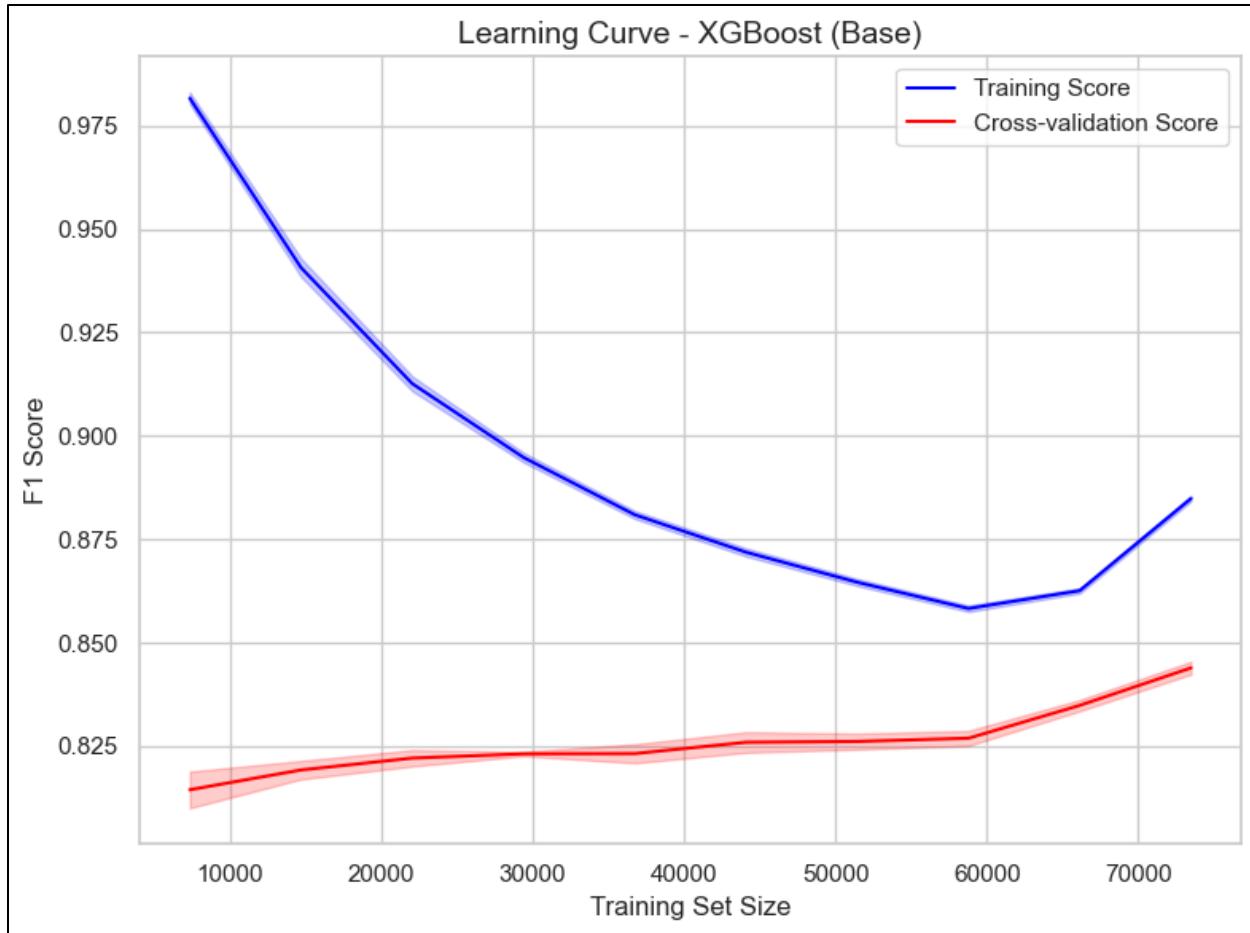
# Learning curve
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label='Training Score', color='blue')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color='blue')
plt.plot(train_sizes, test_mean, label='Cross-validation Score', color='red')
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color='red')
plt.title('Learning Curve - XGBoost (Base)', fontsize=14)
plt.xlabel('Training Set Size')
plt.ylabel('F1 Score')
plt.legend(loc='best')
plt.grid(True)
plt.tight_layout()
plt.show()

```

*Figure 135: Source Code for Learning Curve of XGBoost Base Model\*

The figure above shows the source code for generating a learning curve of the XGBoost Base Model using stratified 5-fold cross-validation. The “learning\_curve()” function from “sklearn.model\_selection” is used to evaluate the model’s performance across varying training set sizes. The F1 score is chosen as the performance metric as it is suitable for imbalanced classification tasks such as employee churn prediction.

The code calculates the mean and standard deviation of both training and cross-validation scores which are plotted using Matplotlib. The blue curve represents the training score while the red curve shows the cross-validation score with shaded areas indicating one standard deviation from the mean.



*Figure 136: Learning Curve of XGBoost Base Model*

The figure above shows the **Learning Curve of the XGBoost Base Model**. It shows a clear gap between the training score (blue line) and the cross-validation score (red line), which indicates there is possible overfitting. The training score starts very high at around 0.98 and gradually declines as the training set size increases, dropping to approximately 0.86 at a training size of 60,000 before slightly rising again. In contrast, the cross-validation score remains relatively flat, starting around 0.81 and increasing modestly to about 0.84 at 60,000 samples.

This persistent gap between the two curves suggests that while the model fits the training data well but its ability to generalize to unseen data is limited. This highlights the need for further tuning or regularization to improve the model's generalization performance.

### 2.3.3 ROC and AUC

```

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

# Get predicted probabilities for the positive class (Resigned = 1)
y_probs = xgb_model.predict_proba(x_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_probs)

# AUC score
auc_score = roc_auc_score(y_test, y_probs)

# ROC curve
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})', color='blue')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - XGBoost (Base)')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()

```

*Figure 137: Source Code for ROC AUC Score of XGBoost Base Model*

The figure above shows the source code of the ROC (Receiver Operating Characteristic) curve for the XGBoost Base Model. It begins by importing a few functions from “sklearn.metrics”, “roc\_curve” to compute the false positive rate (FPR) and true positive rate (TPR) and “roc\_auc\_score” to calculate the AUC (Area Under the Curve) score. The predicted probabilities of the positive class (class 1) are obtained from the model using “predict\_proba”.

Then, The ROC curve is plotted using “matplotlib.pyplot”, where the x-axis represents the false positive rate and the y-axis represents the true positive rate. Additionally, the AUC score is displayed on the graph to provide a numerical summary of the model's performance.

The plot includes a grid for better readability and the dashed line represents the baseline for random classification. The model's performance is assessed by comparing how far the ROC curve deviates from the diagonal line with a greater deviation indicating better performance.

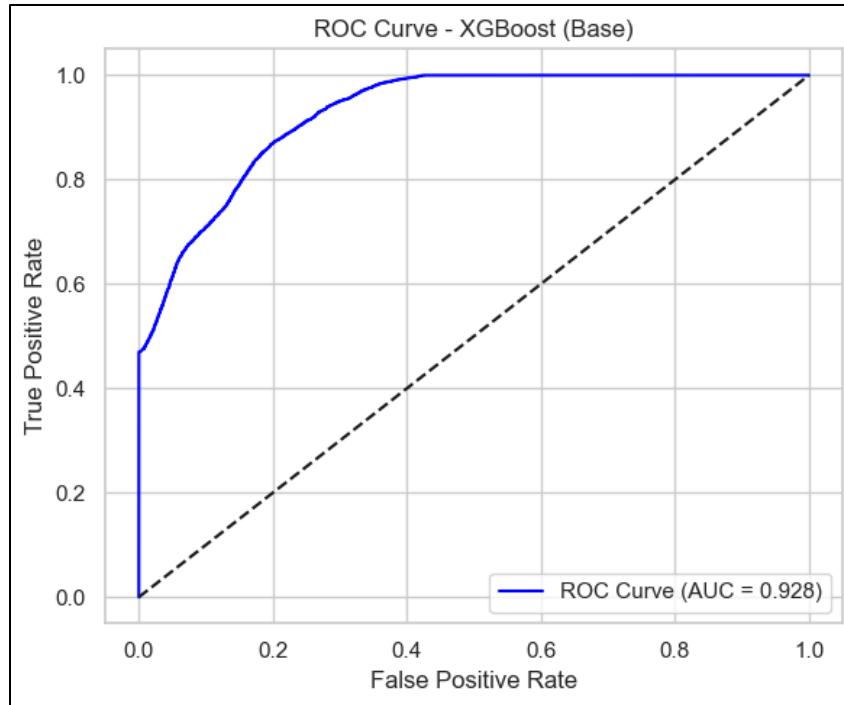


Figure 138: ROC AUC Score of XGBoost Base Model

The **ROC curve** shown in the image represents the performance of the **XGBoost Base Model**. The curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR). The ideal model would have a TPR of 1 (perfect sensitivity) and an FPR of 0 (no false positives) which has leading to a curve that is as close to the top-left corner of the plot as possible.

In this case, the model achieves an AUC (Area Under the Curve) score of **0.928** which is considered a strong performance. An AUC value of 0.5 would suggest no discriminative ability and an AUC of 1.0 indicates perfect classification. With an AUC of 0.928, this XGBoost base model effectively distinguishes between the classes with a high proportion of true positives and a relatively low rate of false positives.

Overall, this result shows that the XGBoost Base Model has a good ability to differentiate between the positive and negative classes in the dataset but its performance could be further improved during the tuning phase.

## 5.2.4 XGBoost Tuned Model

### 5.2.4.1 Confusion Matrix and Classification Report

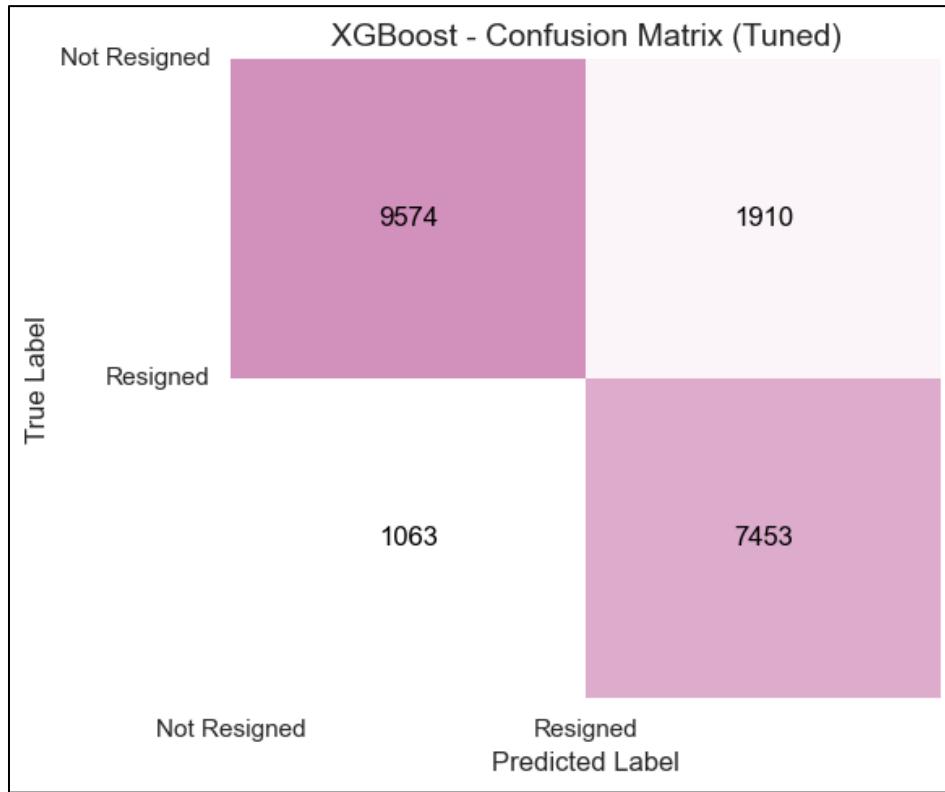


Figure 139: Confusion Matrix for XGBoost Tuned Model

Figure above shows the **confusion matrix of XGBoost Tuned Model** on improved performance in predicting employee resignations. The confusion matrix shows that the model correctly predicted 9,574 non-resigned employees as non-resigned (True Negatives) and 7,453 resigned employees as resigned (True Positives). However, it misclassified 1,910 non-resigned employees as resigned (False Positives) and 1,063 resigned employees as non-resigned (False Negatives).

Confusion Matrix:				
[[9574 1910]				
[1063 7453]]				
Classification Report:				
	precision	recall	f1-score	support
0	0.90	0.83	0.87	11484
1	0.80	0.88	0.83	8516
accuracy			0.85	20000
macro avg	0.85	0.85	0.85	20000
weighted avg	0.86	0.85	0.85	20000
Accuracy Score: 0.85135				

*Figure 140: Classification Report for XGBoost Tuned Model*

The **classification report** above shows the performance of the **XGBoost Tuned Model**. The model achieved a precision of 90% for non-resigned employees (class 0), a recall of 83% and an F1-score of 87%. For resigned employees (class 1), the precision is 80%, the recall is 88% and the F1-score is 83%. These results show an overall improvement in both identifying employees likely to stay and those at risk of resigning.

The model's overall accuracy is 85.13% with the macro average and weighted average for precision, recall and F1-score all at 85%. This indicates consistent performance across both classes and this makes the model more reliable for predicting employee resignations. The balance between precision and recall ensures that both non-resigned and resigned employees are effectively identified to support HR teams in making timely and data-driven decisions.

### 5.2.2.1.2 Comparison Results of Base and Tuned Models

Model Type	Accuracy	Precision (Class 0)	Precision (Class 1)	Recall (Class 0)	Recall (Class 1)	F1-Score (Macro Average)
XGBoost <b>(Base Model)</b>	0.83	0.88	0.78	0.82	0.85	0.83
XGBoost <b>(Tuned Model)</b>	0.85	0.90	0.80	0.83	0.88	0.85

Table 11: Comparison of Base and Tuned Model (XGBoost)

The table above displays the **comparison between the Base and Tuned XGBoost models** with revealing several key improvements. The tuned model shows an increase in overall accuracy from 83% to 85% which has demonstrated a better performance in correctly predicting both non-resigned and resigned employees. For non-resigned employees (class 0), the precision improved from 88% to 90%, meaning the tuned model is more accurate in identifying non-resigned employees and reducing misclassifications. The recall for class 0 also slightly increased from 82% to 83% and this indicated that the tuned model is marginally better at identifying all the actual non-resigned employees.

In terms of resigned employees (class 1), the precision improved from 78% to 80% which reflected the tuned model's enhanced ability to predict resignations accurately. This increase shows that there are fewer false positives, which are the employees who are predicted to resign but is stayed to ensure a more reliable decision-making for HR. Besides, the recall for class 1 saw a significant improvement rising from 85% to 88%. This improvement proved that the tuned model is better at identifying resigned employees, which helps in reducing false negatives and ensuring that fewer resignations are overlooked.

The macro average F1-score improved from 83% to 85%, which reflected a more balanced performance across both classes. Overall, the tuned XGBoost model shows a clear enhancement in its ability to predict employee resignations with notable improvements in both precision and recall for resigned employees. These improvements make the model more reliable and effective for HR decision-making.

The Tuned Model is better than the Base Model because tuning optimizes the model's parameters which is allowing it to more effectively capture patterns in the data. While the Base Model was already performing reasonably well, tuning helps improve its ability to identify key features especially in more complex patterns like predicting resigned employees. For example, tuning leads to a significant increase in recall for resigned employees (from 85% to 88%). This shows that the tuned model is better in identifying employees who are at risk of leaving, which is essential for HR to take timely actions. The precision improvements further ensure that fewer employees are misclassified to reduce both false positives and false negatives to provide more accurate and reliable predictions. By fine-tuning the model, HR teams can now rely on the Tuned Model for more precise decision-making for better workforce planning and retention strategies.

#### 5.2.4.2 Learning Curve

```

from sklearn.model_selection import learning_curve
import numpy as np
import matplotlib.pyplot as plt

# Learning Curve
train_sizes, train_scores, test_scores = learning_curve(
    estimator=best_xgboost,
    X=X_train_resampled,
    y=y_train_resampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

# Mean and std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label='Training Score', color='blue')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color='blue')
plt.plot(train_sizes, test_mean, label='Cross-validation Score', color='red')
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color='red')
plt.title('Learning Curve - XGBoost (Tuned)', fontsize=14)
plt.xlabel('Training Set Size')
plt.ylabel('F1 Score')
plt.legend(loc='best')
plt.grid(True)
plt.tight_layout()
plt.show()

```

*Figure 141: Source Code for Learning Curve of XGBoost Tuned Model*

The figure above shows the source code for generating a learning curve of the XGBoost Tuned Model using stratified 5-fold cross-validation. The “learning\_curve()” function from “sklearn.model\_selection” is used to evaluate the model’s performance across varying training set sizes. The F1 score is chosen as the performance metric as it is suitable for imbalanced classification tasks such as employee churn prediction.

The code calculates the mean and standard deviation of both training and cross-validation scores which are plotted using Matplotlib. The blue curve represents the training score while the red curve shows the cross-validation score with shaded areas indicating one standard deviation from the mean.

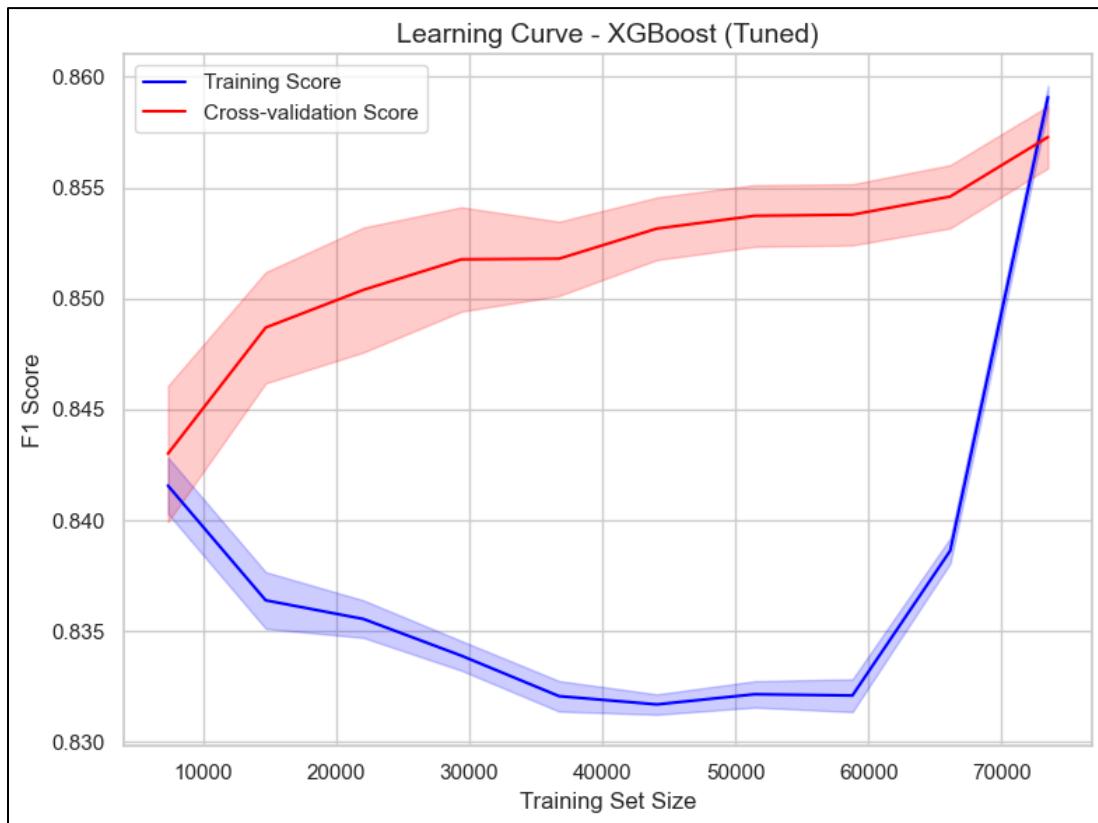


Figure 142: Learning Curve of XGBoost Tuned Model

The figure above displays the **Learning Curve of the XGBoost Tuned Model**. Unlike the base model, this curve shows a narrower and more stable gap between the training score (blue line) and the cross-validation score (red line) which suggests the models have improved generalization. Initially, both scores were close to each other at around 0.84 at 10,000 training samples. Then, the

training score experiences a slight dip and flattens between 0.83 and 0.835 up to approximately 60,000 samples before rising sharply beyond 70,000 samples which is reaching nearly 0.86. Meanwhile, the cross-validation score steadily increases, which is starting around 0.843 and rising consistently to about 0.858.

This gradual upward trend in cross-validation performance coupled with the reduced gap between the two lines indicates that tuning has successfully mitigated overfitting and enhanced the model's generalization capability. Overall, the tuned model shows a more balanced and reliable performance across varying training sizes which has confirms the positive impact of hyperparameter tuning.

#### 5.2.4.3 ROC and AUC

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

# Predict probabilities
y_proba = best_xgboost.predict_proba(x_test)[:, 1]

# Compute ROC curve and AUC
fpr, tpr, _ = roc_curve(y_test, y_proba)
auc_score = roc_auc_score(y_test, y_proba)

# ROC curve
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})', color='blue')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - XGBoost (Tuned)')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()
```

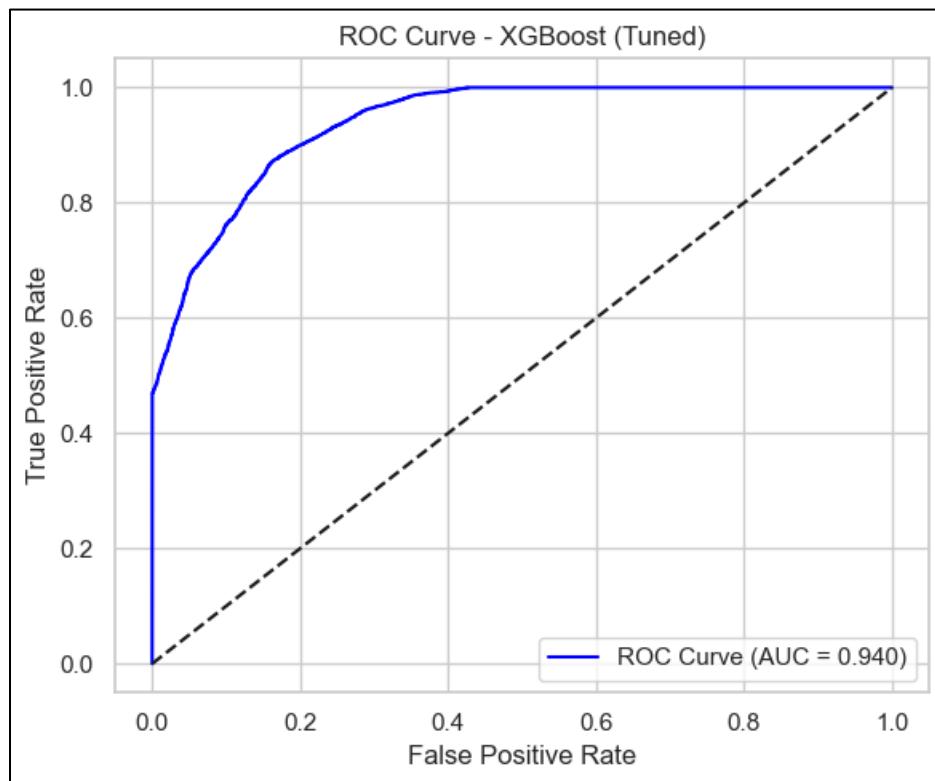
*Figure 143: Source Code for ROC AUC Score of XGBoost Tuned Model*

The figure above shows the source code of the ROC (Receiver Operating Characteristic) curve for the XGBoost Tuned Model. It begins by importing a few functions from “sklearn.metrics”, “roc\_curve” to compute the false positive rate (FPR) and true positive rate (TPR) and

“roc\_auc\_score” to calculate the AUC (Area Under the Curve) score. The predicted probabilities of the positive class (class 1) are obtained from the model using “predict\_proba”.

Then, The ROC curve is plotted using “matplotlib.pyplot”, where the x-axis represents the false positive rate and the y-axis represents the true positive rate. Additionally, the AUC score is displayed on the graph to provide a numerical summary of the model's performance.

The plot includes a grid for better readability and the dashed line represents the baseline for random classification. The model's performance is assessed by comparing how far the ROC curve deviates from the diagonal line with a greater deviation indicating better performance.



*Figure 144: ROC AUC Score of XGBoost Tuned Model*

The figure above displays the **ROC Curve of the XGBoost Tuned Model** with an **AUC** (Area Under the Curve) score of **0.940**. This score signifies a strong predictive performance as the model can distinguish between classes of resigned and non-resigned employees with high accuracy.

The ROC curve shows a steep rise in the True Positive Rate (TPR) as the False Positive Rate (FPR) increases which indicates that the model quickly identifies positive instances (resigned employees)

with minimal false positives. The AUC score of 0.940 is higher than the base model's AUC of 0.928 and this shows a clear improvement in the tuned model's ability to predict resignations correctly. The closer the AUC score is to 1, the better the model's performance. This makes the tuned model more effective in identifying employees at risk of leaving. The curve's proximity to the top-left corner further proves that the model's strong ability to balance both sensitivity and specificity which is making it a more reliable tool for HR decision-making.

## 5.2.5 Gradient Boosting Base Model

### 5.2.5.1 Confusion Matrix and Classification Report

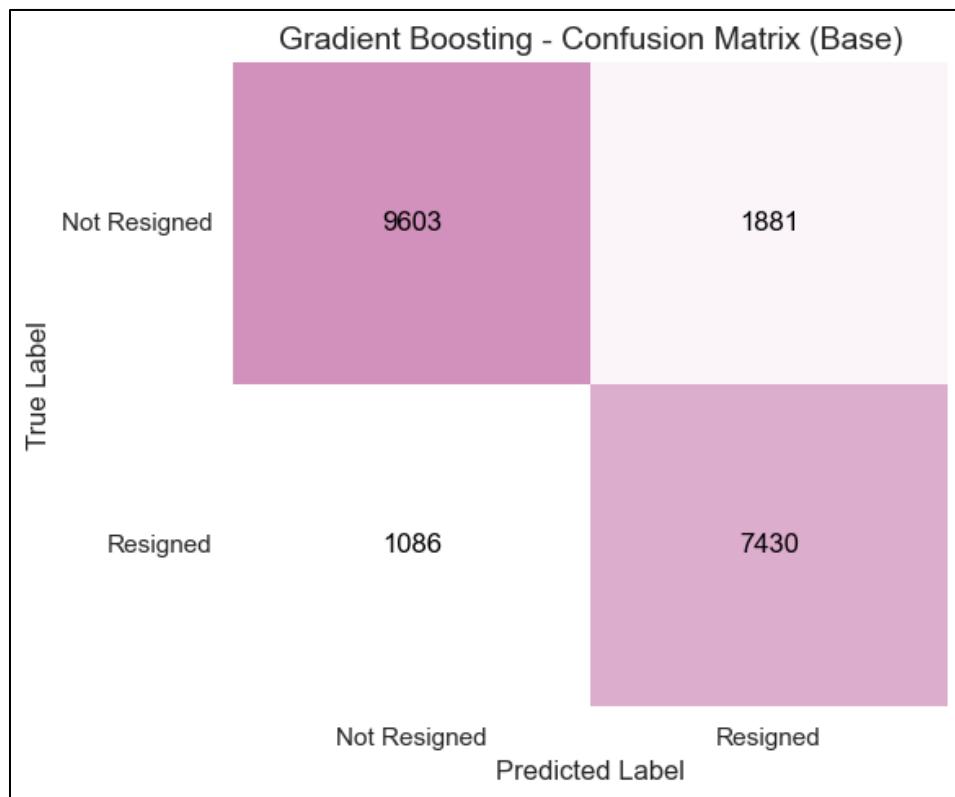
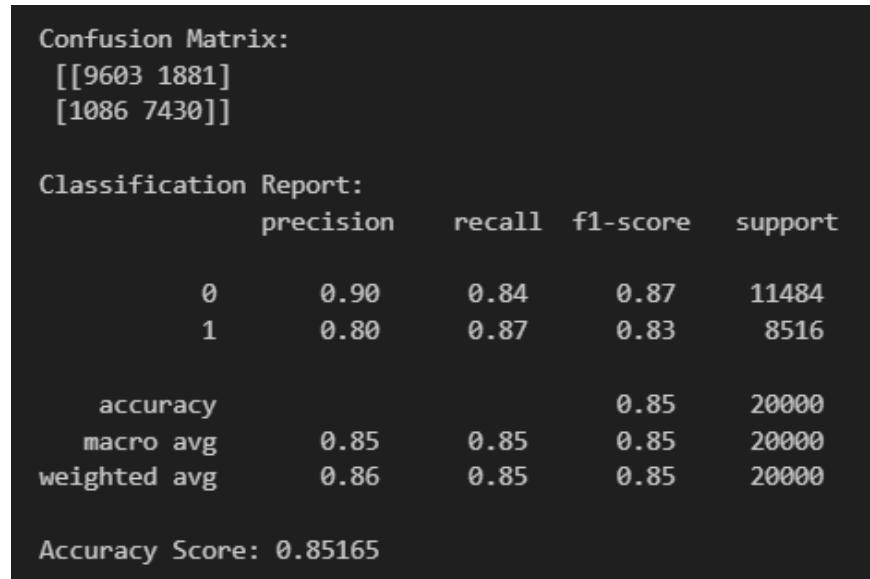


Figure 145: Confusion Matrix for Gradient Boosting Base Model

Figures above show the **confusion matrix for Gradient Boosting Base Model** which is performing well in predicting employee resignations. The confusion matrix shows that the model correctly predicted 9,603 non-resigned employees as non-resigned (True Negatives) and 7,430 resigned employees as resigned (True Positives). However, it misclassified 1,881 non-resigned employees as resigned (False Positives) and 1,086 resigned employees as non-resigned (False Negatives).



*Figure 146: Classification Report for Gradient Boosting Base Model*

The **classification report** for the **Gradient Boosting Base Model** shows that it achieved a precision of 90% for non-resigned employees (class 0), a recall of 84% and an F1-score of 87%. For resigned employees (class 1), the precision is 80%, the recall is 87% and the F1-score is 83%. While these results show that the base model effectively identifies both non-resigned and resigned employees, there may still be room for further improvement, especially in reducing false positives and false negatives

The model's overall accuracy is 85.17% with the macro average and weighted average for precision, recall and F1-score all at 85%. While these results indicate a solid base performance, further tuning is expected to enhance the model's ability to predict employee resignations more accurately. Specifically, tuning could improve recall for class 1 with further minimizing misclassifications and enable better proactive interventions in HR decision-making.

### 5.2.5.2 Learning Curve

```

from sklearn.model_selection import StratifiedKFold, learning_curve
import numpy as np
import matplotlib.pyplot as plt

# Stratified K-Fold cross-validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Learning curve
train_sizes, train_scores, test_scores = learning_curve(
    estimator=gb_model,
    X=X_train_resampled,
    y=y_train_resampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

# Mean and Std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

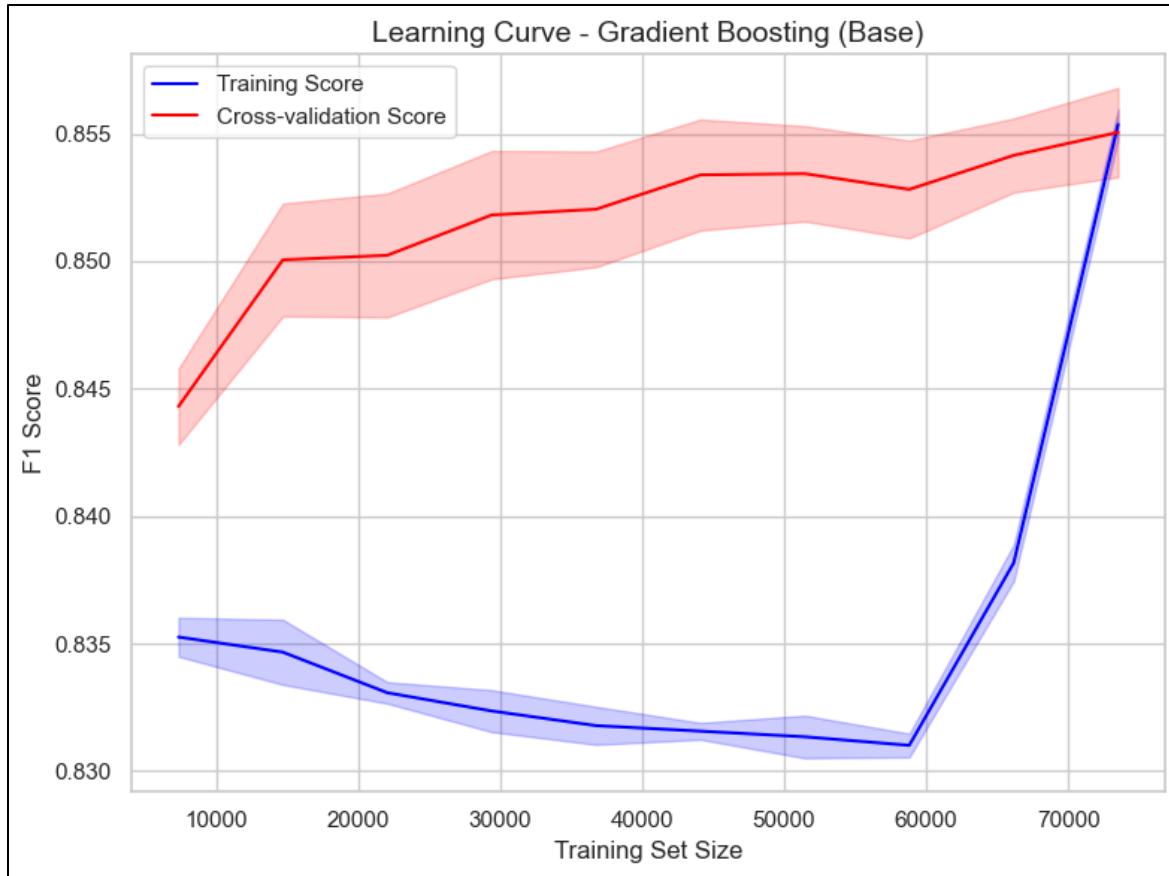
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label='Training Score', color='blue')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color='blue')
plt.plot(train_sizes, test_mean, label='Cross-validation Score', color='red')
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color='red')
plt.title('Learning Curve - Gradient Boosting (Base)', fontsize=14)
plt.xlabel('Training Set Size')
plt.ylabel('F1 Score')
plt.legend(loc='best')
plt.grid(True)
plt.tight_layout()
plt.show()

```

*Figure 147: Source Code for Learning Curve of Gradient Boosting Base Model*

The figure above shows the source code for generating a learning curve of the Gradient Boosting Base Model using stratified 5-fold cross-validation. The “learning\_curve()” function from “sklearn.model\_selection” is used to evaluate the model’s performance across varying training set sizes. The F1 score is chosen as the performance metric as it is suitable for imbalanced classification tasks such as employee churn prediction.

The code calculates the mean and standard deviation of both training and cross-validation scores which are plotted using Matplotlib. The blue curve represents the training score while the red curve shows the cross-validation score with shaded areas indicating one standard deviation from the mean.



*Figure 148: Learning Curve of Gradient Boosting Base Model*

The figure above shows the **learning curve of the Gradient Boosting Base Model**. The cross-validation score (red line) starts at approximately 0.844 and steadily increases as the training set size grows, reaching around 0.855 at 75,000 training set sizes. This consistent rise indicates that the model benefits from additional training data and generalizes well. Besides, the training score (blue line) starts slightly above 0.835 and dips gradually, reaching its lowest point near 0.830 at around 60,000 samples before increasing again sharply towards the end.

The gap between the training and validation scores remains noticeable throughout which suggests the model might still be slightly underfitting. Although the difference is not large, these patterns suggest that additional tuning could bring the training and validation scores closer together and further improve the model's performance.

### 5.2.5.3 ROC and AUC

```

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

y_probs = gb_model.predict_proba(X_test)[:, 1]

# Calculate ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
auc_score = roc_auc_score(y_test, y_probs)

# Plot ROC Curve
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})', color='blue', linewidth=2)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Gradient Boosting (Base)', fontsize=14)
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()

```

*Figure 149: Source Code for ROC AUC Score of Gradient Boosting Base Model*

The figure above shows the source code of the ROC (Receiver Operating Characteristic) curve for the Gradient Boosting Base Model. It begins by importing a few functions from “sklearn.metrics”, “roc\_curve” to compute the false positive rate (FPR) and true positive rate (TPR) and “roc\_auc\_score” to calculate the AUC (Area Under the Curve) score. The predicted probabilities of the positive class (class 1) are obtained from the model using “predict\_proba”.

Then, The ROC curve is plotted using “matplotlib.pyplot”, where the x-axis represents the false positive rate and the y-axis represents the true positive rate. Additionally, the AUC score is displayed on the graph to provide a numerical summary of the model's performance.

The plot includes a grid for better readability and the dashed line represents the baseline for random classification. The model's performance is assessed by comparing how far the ROC curve deviates from the diagonal line with a greater deviation indicating better performance.

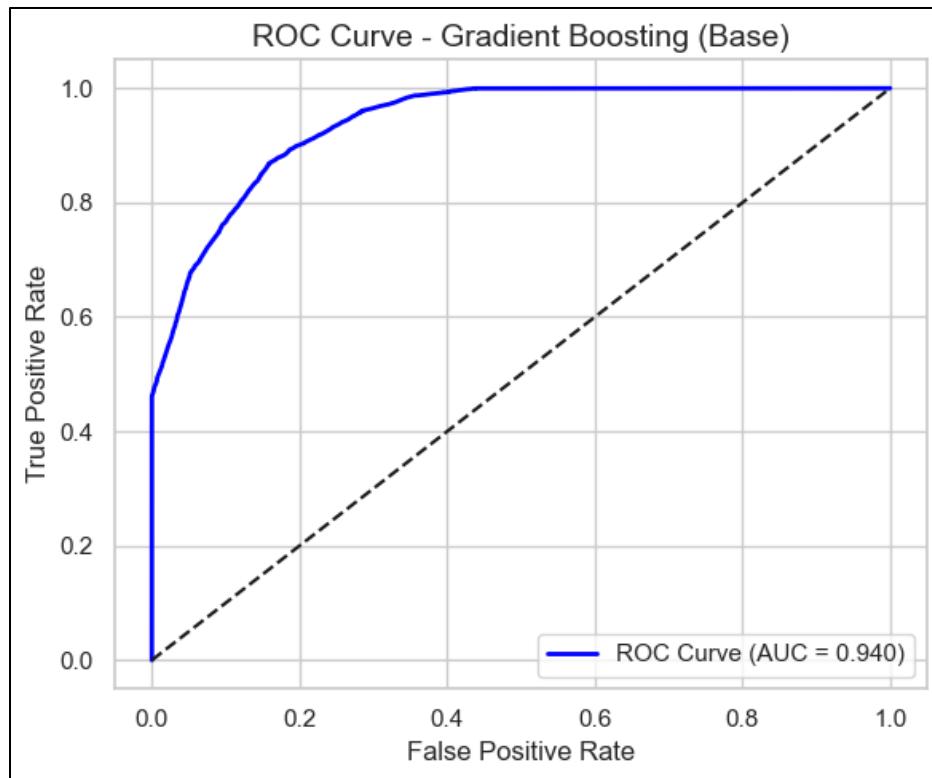


Figure 150: ROC AUC Score of Gradient Boosting Base Model

The **ROC curve for the Gradient Boosting Base Model** shows strong classification performance with an **AUC** score of **0.940**. The curve is steep with approaching the top-left corner which indicates a high true positive rate and a low false positive rate. This suggests that the model is effective at distinguishing between the positive (resigned employees) and negative (non-resigned employees) classes. The AUC value of 0.940 is close to 1.0, which is the ideal score and demonstrates that the base model already performs well in classifying employees accurately. While the model performs well, there may still be room for further improvement with additional tuning or optimization.

## 5.2.6 Gradient Boosting Tuned Model

### 5.2.6.1 Confusion Matrix and Classification Report

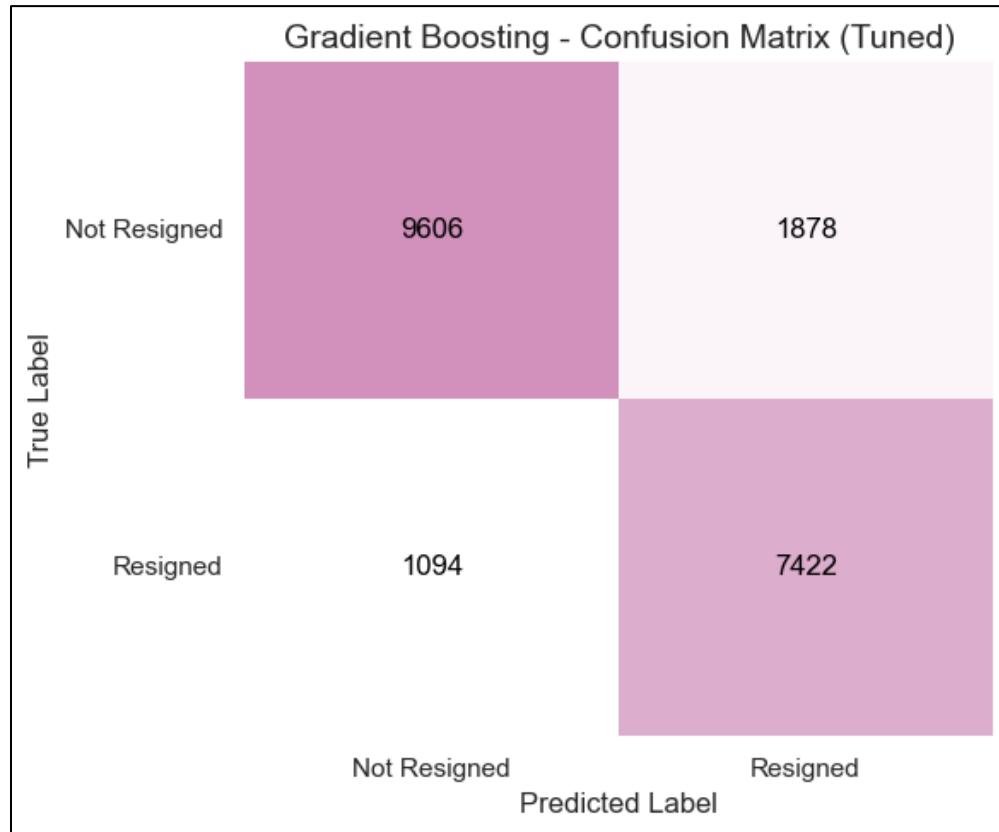


Figure 151: Confusion Matrix for Gradient Boosting Tuned Model

The figure above shows the **confusion matrix for Gradient Boosting Tuned Model** which demonstrating a consistent and balanced performance in predicting employee resignations. The confusion matrix shows that the model correctly predicted 9,606 non-resigned employees as non-resigned (True Negatives) and 7,422 resigned employees as resigned (True Positives). However, it misclassified 1,878 non-resigned employees as resigned (False Positives) and 1,094 resigned employees as non-resigned (False Negatives).

Confusion Matrix:				
[[9606 1878]				
[1094 7422]]				
Classification Report:				
	precision	recall	f1-score	support
0	0.90	0.84	0.87	11484
1	0.80	0.87	0.83	8516
accuracy			0.85	20000
macro avg	0.85	0.85	0.85	20000
weighted avg	0.86	0.85	0.85	20000
Accuracy Score: 0.8514				

*Figure 152: Classification Report for Gradient Boosting Tuned Model*

The **classification report** above shows that the **Gradient Boosting Tuned Model** achieved a precision of 90% for non-resigned employees (class 0), a recall of 84% and an F1-score of 87%. For resigned employees (class 1), the precision is 80%, the recall is 87% and the F1-score is 83%. These results reflect a strong and stable ability to correctly identify both employees likely to stay and those at risk of leaving.

The model's overall accuracy is 85.14% with the macro average and weighted average for precision, recall and F1-score all at 85% which has indicated a reliable and consistent performance across both classes. However, there is no significant performance improvement from the base model to the tuned model. The further explanation and justification for this result will be discussed in the Comparison of Base and Tuned Model (Gradient Boosting) section. Despite this, the tuned model maintains a high level of accuracy and recall especially for class 1 to make it a dependable tool for supporting HR in proactive retention planning.

### 5.2.2.1.3 Comparison Results of Base and Tuned Models

Model Type	Accuracy	Precision (Class 0)	Precision (Class 1)	Recall (Class 0)	Recall (Class 1)	F1-Score (Macro Average)
<b>Gradient Boosting (Base)</b>	0.85	0.90	0.80	0.84	0.87	0.85
<b>Gradient Boosting (Tuned)</b>	0.85	0.90	0.80	0.84	0.87	0.85

Table 12: Comparison of Base and Tuned Model (Gradient Boosting)

The **comparison between the Base and Tuned Gradient Boosting models** reveals that both models exhibit almost identical performance across all metrics. The accuracy for both models is 85%, indicating that tuning did not significantly improve the overall prediction accuracy. This could be because the base model was already well-optimized and effectively captured the patterns in the data. Precision for class 0 (non-resigned employees) is 90% in both models also showing that the model's ability to accurately identify non-resigned employees remains unchanged. Similarly, the precision for class 1 (resigned employees) is 80% for both models, indicating consistent performance in predicting resigned employees.

The recall for class 0 is 84% for both models and shows that both are equally effective at identifying non-resigned employees. For class 1, the recall is 87% for both models, demonstrating the same ability to correctly identify resigned employees. These similar recall rates suggest that both models were already capturing the key features of the dataset effectively and tuning did not significantly change their ability to identify each class. The F1-score is also the same at 85% for both models, which suggests a balanced performance across both classes. Since the base model already had a good balance between precision and recall, tuning did not lead to a substantial improvement in this metric.

In conclusion, the Base and Tuned Gradient Boosting models show no significant improvements after tuning which indicates that the model's performance is stable across these configurations.

The tuning did not have a major impact on the model's ability to predict both non-resigned and resigned employees as the base model was already well-tuned to the dataset and makes both models equally reliable for this task.

### 5.2.6.2 Learning Curve

```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve

train_sizes, train_scores, test_scores = learning_curve(
    estimator=best_gradientboost,
    X=X_train_resampled,
    y=y_train_resampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    n_jobs=-1,
    scoring='f1'
)

# Mean and Std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

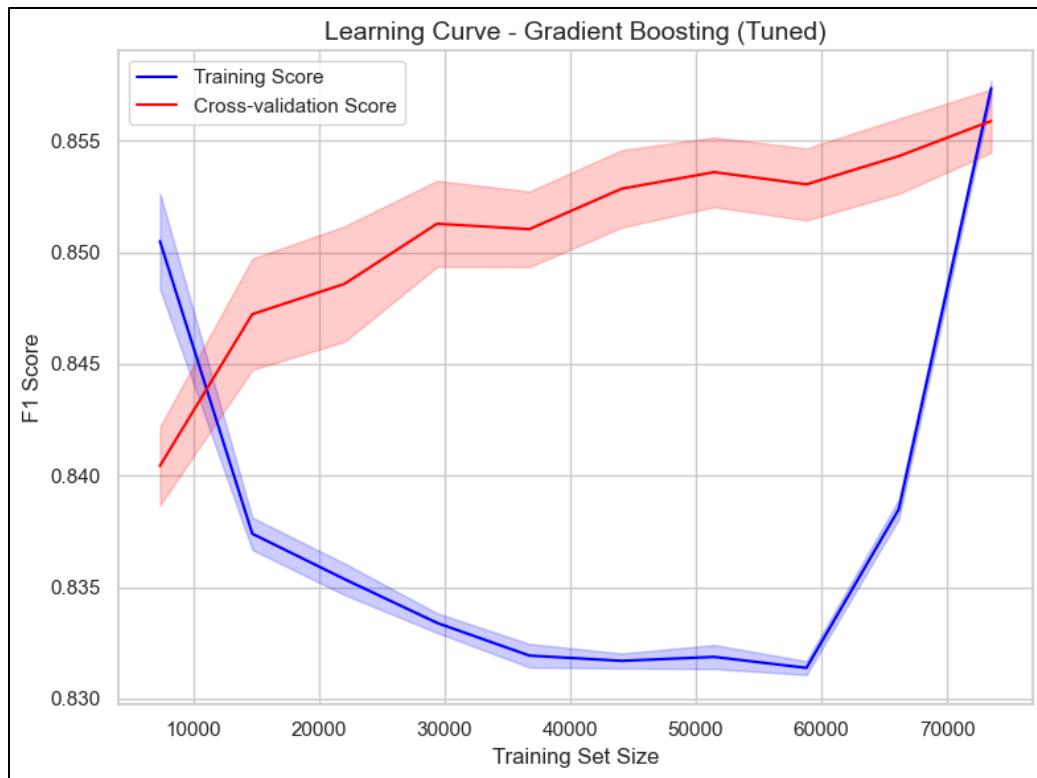
# Learning curve
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label="Training Score", color="blue")
plt.plot(train_sizes, test_mean, label="Cross-validation Score", color="red")
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color="blue")
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color="red")
plt.title("Learning Curve - Gradient Boosting (Tuned)", fontsize=14)
plt.xlabel("Training Set Size")
plt.ylabel("F1 Score")
plt.legend(loc="best")
plt.grid(True)
plt.tight_layout()
plt.show()

```

*Figure 153: Source Code for Learning Curve of Gradient Boosting Tuned Model*

The figure above shows the source code for generating a learning curve of the Gradient Boosting Tuned Model using stratified 5-fold cross-validation. The “learning\_curve()” function from “sklearn.model\_selection” is used to evaluate the model’s performance across varying training set sizes. The F1 score is chosen as the performance metric as it is suitable for imbalanced classification tasks such as employee churn prediction.

The code calculates the mean and standard deviation of both training and cross-validation scores which are plotted using Matplotlib. The blue curve represents the training score while the red curve shows the cross-validation score with shaded areas indicating one standard deviation from the mean.



*Figure 154: Learning Curve of Gradient Boosting Tuned Model*

The figure above shows the **learning curve of the Gradient Boosting Tuned Model**. Both training and cross-validation scores show a more aligned and stable trend compared to the base model.

Initially, the training score (blue line) started above 0.850 but gradually decreased, reaching a low point around 0.830 at approximately 60,000 training samples before rising sharply again. In contrast, the cross-validation score (red line) begins around 0.840 and steadily increases and reaches approximately 0.855 as the training size grows. The gap between the two curves remains narrow and consistent throughout which has indicated improved generalization and reduced variance.

Although the classification report shows that the final performance metrics such as accuracy, precision, recall and F1-score are identical between the Base and Tuned Gradient Boosting models, the learning curve reveals that the tuned model is more stable and less prone to overfitting. This suggests that while the end results are the same, the tuned model achieves them with greater reliability and consistency to make it more dependable when deployed on unseen data.

### 5.2.6.3 ROC and AUC

```

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

y_probs = best_gradientboost.predict_proba(x_test)[:, 1]

# Calculate ROC curve & AUC score
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
auc_score = roc_auc_score(y_test, y_probs)

# 4. Plot ROC curve
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})', color='blue')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Gradient Boosting (Tuned)')
plt.legend(loc='lower right')
plt.grid()
plt.tight_layout()
plt.show()

```

*Figure 155: Source Code for ROC AUC Score of Gradient Boosting Tuned Model*

The figure above shows the source code of the ROC (Receiver Operating Characteristic) curve for the Gradient Boosting Tuned Model. It begins by importing a few functions from “sklearn.metrics”, “roc\_curve” to compute the false positive rate (FPR) and true positive rate (TPR) and “roc\_auc\_score” to calculate the AUC (Area Under the Curve) score. The predicted probabilities of the positive class (class 1) are obtained from the model using “predict\_proba”.

Then, The ROC curve is plotted using “matplotlib.pyplot”, where the x-axis represents the false positive rate and the y-axis represents the true positive rate. Additionally, the AUC score is displayed on the graph to provide a numerical summary of the model's performance.

The plot includes a grid for better readability and the dashed line represents the baseline for random classification. The model's performance is assessed by comparing how far the ROC curve deviates from the diagonal line with a greater deviation indicating better performance.

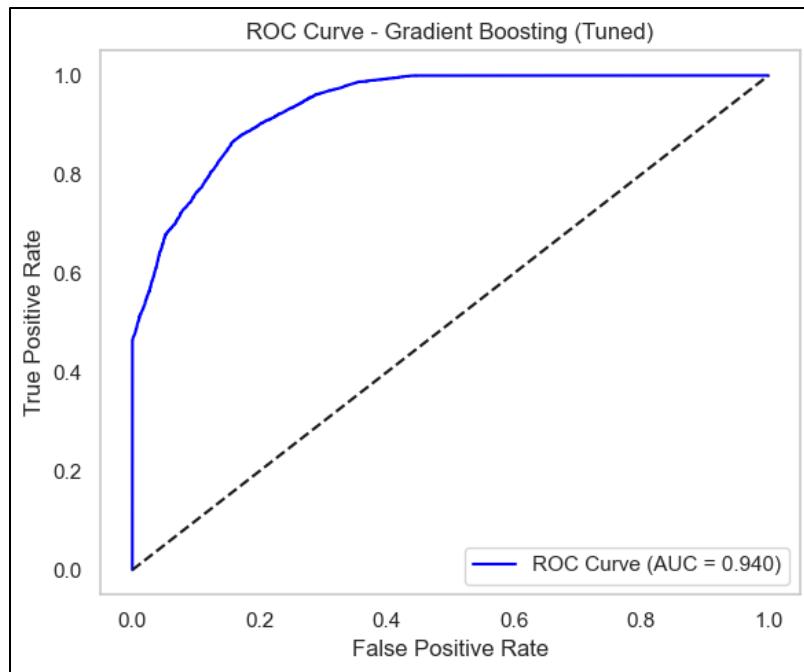


Figure 156: ROC AUC Score of Gradient Boosting Tuned Model

The **ROC curve for the Gradient Boosting Base Model** above also shows strong classification performance with an **AUC score of 0.940**, which is same result as the Base Model. The curve rises steeply toward the top-left corner and is indicating a high true positive rate and a low false positive rate. This suggests that the model is effective at distinguishing between the positive (resigned employees) and negative (non-resigned employees) classes.

Despite the tuning, the ROC curve for the Tuned Gradient Boosting Model shows almost identical performance to the base model with the AUC score remaining at 0.940. While tuning typically aims to enhance model performance, there was no significant change in the AUC score. This suggests that the base model was already highly optimized and capable of effectively distinguishing between the two classes. Therefore, the tuning did not significantly improve this aspect of the model's performance. While the tuning may have contributed to improvements in other areas such as model stability, it did not substantially affect the overall classification performance as reflected by the unchanged AUC score.

## 5.2.7 Support Vector Machine Base Model

### 5.2.7.1 Confusion Matrix and Classification Report

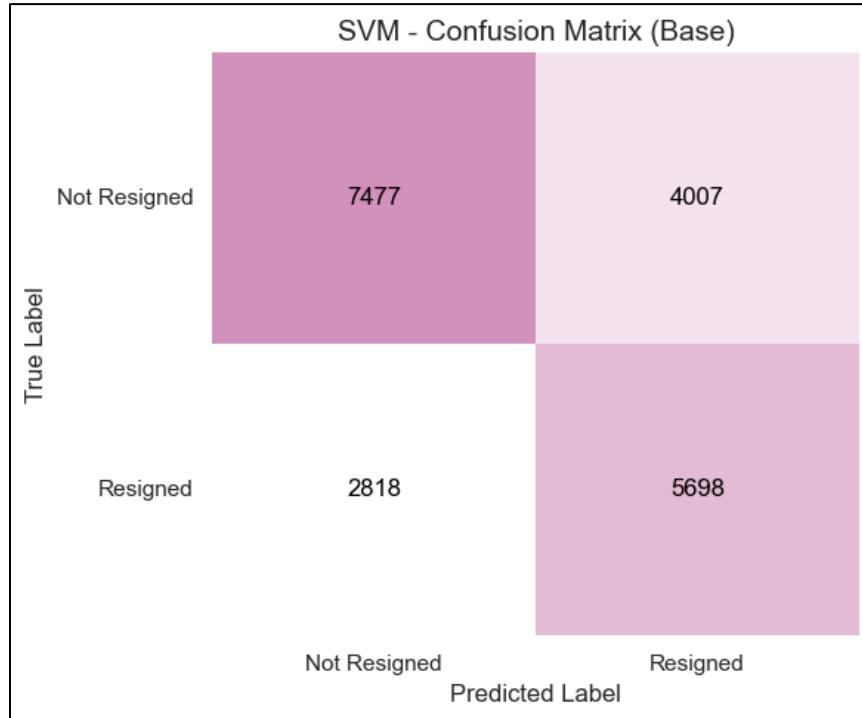


Figure 157: Confusion Matrix for Support Vector Machine Base Model

Figure above shows the **confusion matrix for the Support Vector Machine (SVM) Base Model** with its prediction results through the confusion matrix. The confusion matrix shows that the model correctly predicted 7,477 non-resigned employees as non-resigned (True Negatives) and 5,698 resigned employees as resigned (True Positives). However, it also misclassified 4,007 non-resigned employees as resigned (False Positives) and 2,818 resigned employees as non-resigned (False Negatives).

Confusion Matrix:				
[[7477 4007] [2818 5698]]				
	precision	recall	f1-score	support
0	0.73	0.65	0.69	11484
1	0.59	0.67	0.63	8516
accuracy			0.66	20000
macro avg	0.66	0.66	0.66	20000
weighted avg	0.67	0.66	0.66	20000
Accuracy Score: 0.65875				

*Figure 158: Classification Report for Support Vector Machine Base Model*

The **classification report for Support Vector Machine Base Model** indicates that for class 0 (non-resigned employees), the model achieved a precision of 73%, a recall of 65% and a F1-score of 69%. For class 1 (resigned employees), the precision is 59%, recall is 67% and F1-score is 63%. These results show that the model performs moderately in identifying both classes but with a noticeable imbalance in precision and recall values.

The overall accuracy of the model is 65.88%, with macro and weighted averages for precision, recall and F1-score all at 66%. The results suggest that the SVM Base Model can be further improvement in reducing misclassifications and enhancing its ability to more accurately identify both resigned and non-resigned employees.

### 5.2.7.2 Learning Curve

```

from sklearn.model_selection import StratifiedKFold, learning_curve
import numpy as np
import matplotlib.pyplot as plt

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

train_sizes, train_scores, test_scores = learning_curve(
    estimator=svm_model_subset,
    X=X_train_subset,
    y=y_train_subset,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

# Mean and Std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

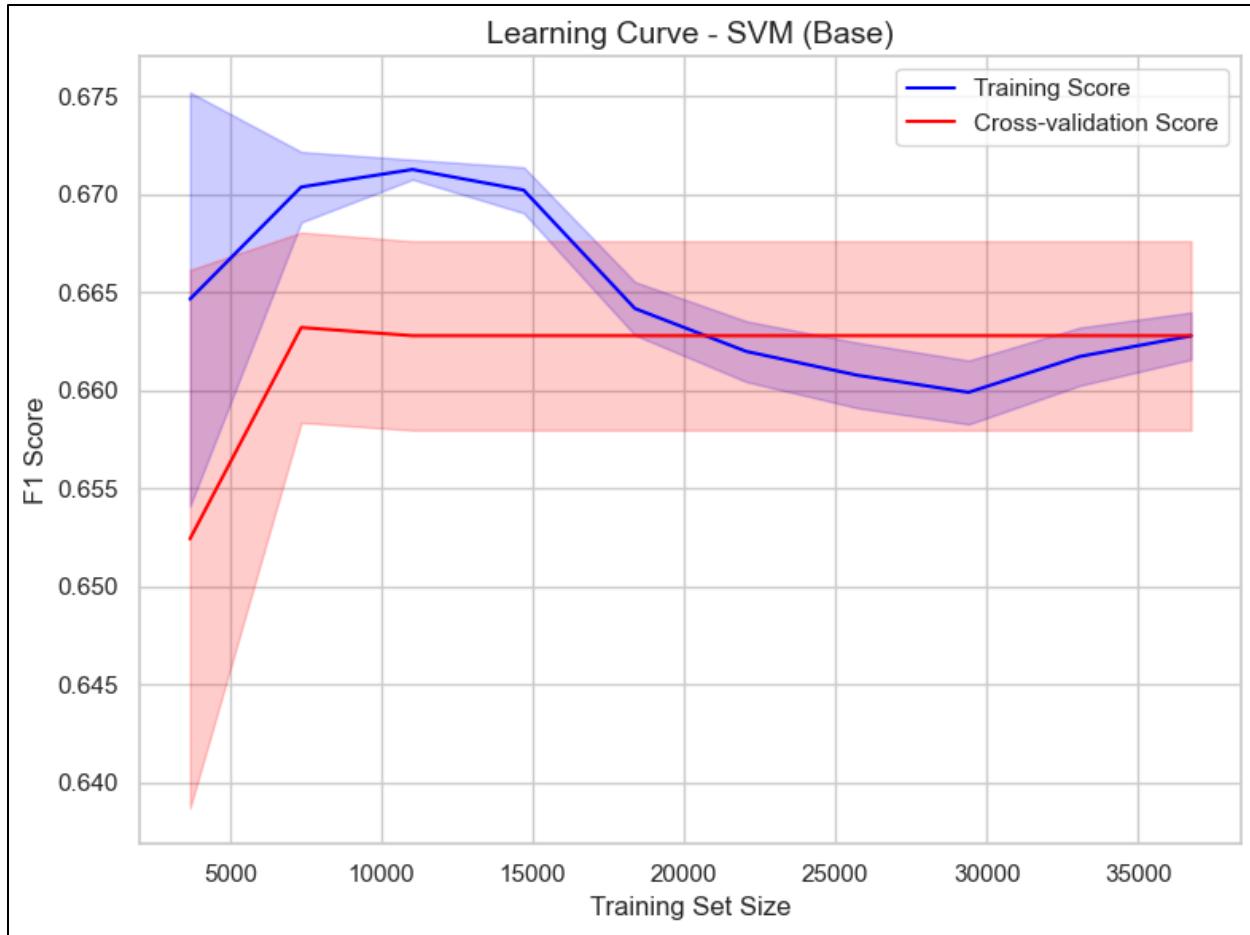
# Plot learning curve
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label="Training Score", color="blue")
plt.plot(train_sizes, test_mean, label="Cross-validation Score", color="red")
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color="blue")
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color="red")
plt.title("Learning Curve - SVM (Base)", fontsize=14)
plt.xlabel("Training Set Size")
plt.ylabel("F1 Score")
plt.legend(loc="best")
plt.grid(True)
plt.tight_layout()
plt.show()

```

*Figure 159: Source Code for Learning Curve of Support Vector Machine Base Model*

The figure above shows the source code for generating a learning curve of the Support Vector Machine Base Model using stratified 5-fold cross-validation. The “learning\_curve()” function from “sklearn.model\_selection” is used to evaluate the model’s performance across varying training set sizes. The F1 score is chosen as the performance metric as it is suitable for imbalanced classification tasks such as employee churn prediction.

The code calculates the mean and standard deviation of both training and cross-validation scores which are plotted using Matplotlib. The blue curve represents the training score while the red curve shows the cross-validation score with shaded areas indicating one standard deviation from the mean.



*Figure 160: Learning Curve of Support Vector Machine Base Model*

The figure above shows the **learning curve of the Support Vector Machine (SVM) Base Model**, which was trained using only 50% of the full training set. The training score (blue line) starts around 0.665 and rises slightly to a peak above 0.670 before gradually declining as the training set size increases. Meanwhile, the cross-validation score (red line) begins lower at approximately 0.652 and has increased quickly and stabilized around 0.662. The narrowing gap between the two curves as the training size grows suggests that early overfitting reduces with more data exposure. However, both scores flatten toward the end which indicates that the model's learning has plateaued under the current settings. This suggests that adding more data alone may not improve performance significantly unless combined with further tuning or algorithmic adjustments.

### 5.2.7.3 ROC and AUC

```

from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

y_probs = svm_model_subset.decision_function(X_test)
fpr, tpr, _ = roc_curve(y_test, y_probs)
auc_score = auc(fpr, tpr)

plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})', color='blue')
plt.plot([0, 1], [0, 1], 'k--', label='Random Guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - SVM (Base)', fontsize=14)
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()

```

*Figure 161: Source Code for ROC AUC Score of Support Vector Machine Base Model*

The figure above shows the source code of the ROC (Receiver Operating Characteristic) curve for the Support Vector Machine Base Model. It begins by importing a few functions from “sklearn.metrics”, “roc\_curve” to compute the false positive rate (FPR) and true positive rate (TPR) and “roc\_auc\_score” to calculate the AUC (Area Under the Curve) score. The predicted probabilities of the positive class (class 1) are obtained from the model using “predict\_proba”.

Then, The ROC curve is plotted using “matplotlib.pyplot”, where the x-axis represents the false positive rate and the y-axis represents the true positive rate. Additionally, the AUC score is displayed on the graph to provide a numerical summary of the model's performance.

The plot includes a grid for better readability and the dashed line represents the baseline for random classification. The model's performance is assessed by comparing how far the ROC curve deviates from the diagonal line with a greater deviation indicating better performance.

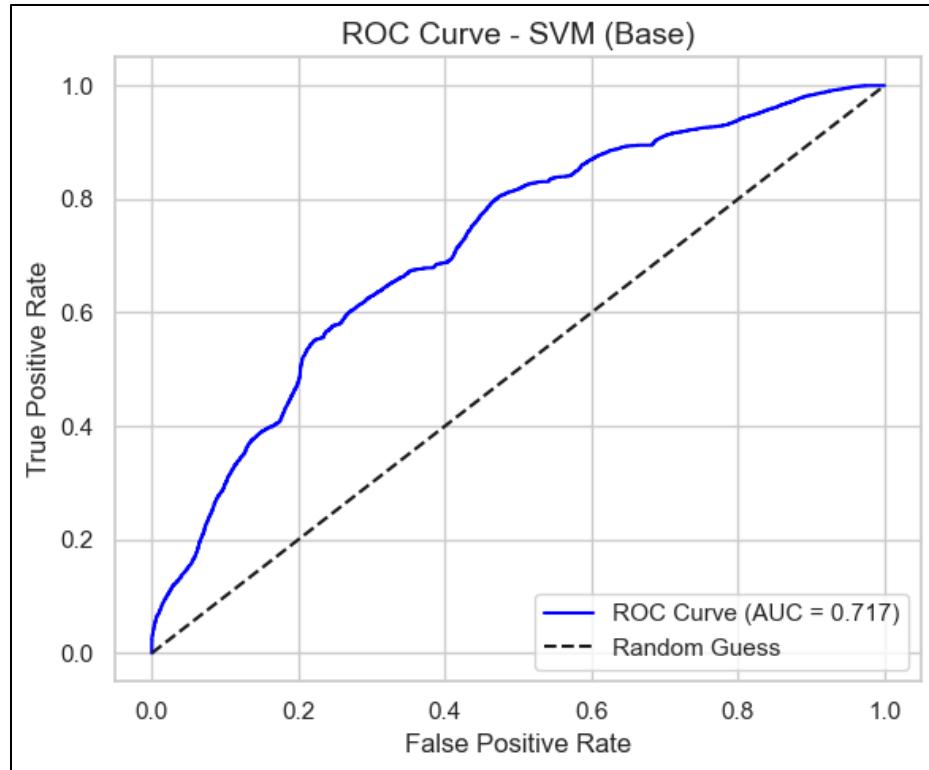
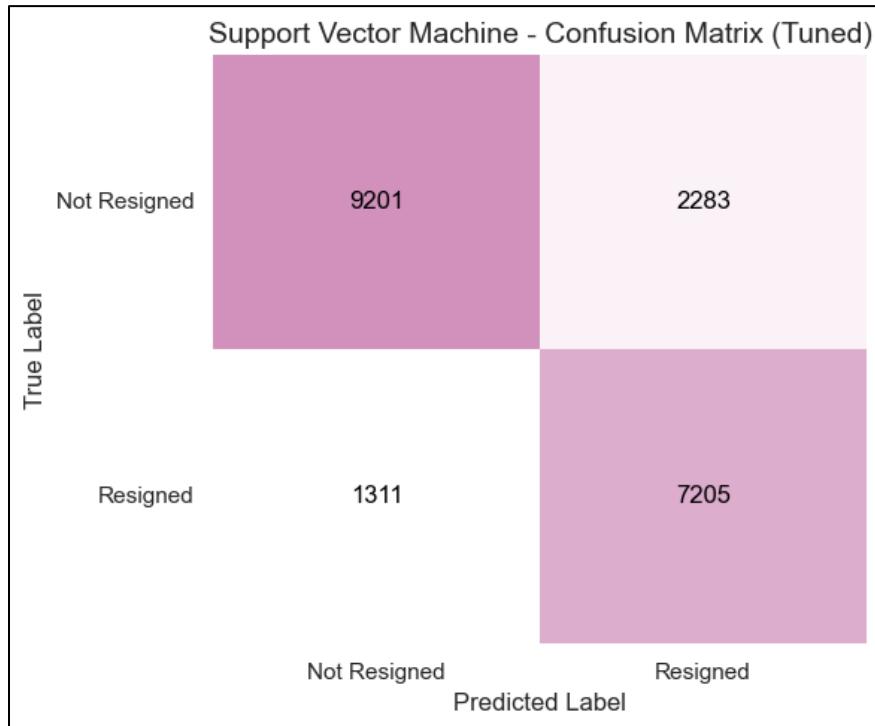


Figure 162: ROC AUC Score of Support Vector Machine Base Model

The ROC curve for the **Support Vector Machine (SVM) Base Model** shows an **AUC score of 0.717**, indicating moderate classification performance. The curve follows a relatively gentle upward trajectory with a noticeable distance from the top-left corner. This suggests that the model has a limited ability to effectively distinguish between resigned (class 1) and non-resigned (class 0) employees. This AUC score indicates that the model performs better than random guessing which is represented by the dashed diagonal line. However, it still faces challenges in correctly identifying true positives while keeping the false positive rate low. Therefore, the base model's performance could be improved particularly in distinguishing between the two classes with higher precision and recall.

## 5.2.8 Support Vector Machine Tuned Model

### 5.2.8.1 Confusion Matrix and Classification Report



*Figure 163: Confusion Matrix for Support Vector Machine Tuned Model*

Figures above show the **confusion matrix for Support Vector Machine (SVM) Tuned Model** which demonstrates markedly stronger predictive performance. According to the confusion matrix, the model correctly classified 9,201 non-resigned employees as non-resigned (True Negatives) and 7,205 resigned employees as resigned (True Positives). The misclassifications are reduced with 2,283 non-resigned employees incorrectly predicted as resigned (False Positives) and 1,311 resigned employees incorrectly predicted as non-resigned (False Negatives).

Confusion Matrix:				
[[9201 2283]				
[1311 7205]]				
Classification Report:				
	precision	recall	f1-score	support
0	0.88	0.80	0.84	11484
1	0.76	0.85	0.80	8516
accuracy			0.82	20000
macro avg	0.82	0.82	0.82	20000
weighted avg	0.83	0.82	0.82	20000
Accuracy Score: 0.8203				

*Figure 164: Classification Report for Support Vector Machine Tuned Model*

The figure above shows the **classification report for Support Vector Machine Tuned Model**. For non-resigned employees (class 0) the model achieves precision 88 %, recall 80 % and F1-score 84 %. For resigned employees (class 1) the metrics rise to precision 76 %, recall 85 %, and F1-score 80 % which indicates model was able to capture actual resignations while keeping false alarms at a reasonable level.

The overall accuracy improves to 82.03 % and for the macro-average precision, recall and F1-score are all 82 %, which is confirming balanced effectiveness across both classes. These results show that tuning has substantially enhanced SVM's capacity to identify employees at risk of leaving while retaining high precision for those predicted to stay. This makes the model a more dependable tool for informed HR retention planning.

### 5.2.2.1.4 Comparison Results of Base and Tuned Models

Model Type	Accuracy	Precision (Class 0)	Precision (Class 1)	Recall (Class 0)	Recall (Class 1)	F1-Score (Macro Average)
SVM (Base Model)	0.66	0.73	0.59	0.65	0.67	0.66
SVM (Tuned Model)	0.82	0.88	0.76	0.80	0.85	0.82

Table 13: Comparison of Base and Tuned Model (Support Vector Machine)

The **comparison between the Base SVM model and the Tuned SVM model** shows significant improvements in the latter although using only 50% of the sample size for tuning. The accuracy of the tuned model increased drastically from 66% to 82% and has indicated that tuning enhanced the model's overall ability to correctly classify both non-resigned and resigned employees. Precision for non-resigned employees (class 0) improved from 73% to 88%, which shows that the tuned model is more effective in identifying non-resigned employees and helps in reducing the misclassification of employees as resigned. Similarly, the precision for resigned employees (class 1) also saw an improvement with rose from 59% to 76% which is demonstrating a better ability to predict employees who have resigned.

In terms of recall, the tuned model outperforms the base model with a significant increase from 65% to 80% for non-resigned employees (class 0) and is better at identifying actual non-resigned employees. The recall for resigned employees (class 1) improved from 67% to 85%, showing that the tuned model is far more capable of identifying employees who have resigned to reduce the false negatives and enabling more timely interventions.

The macro average F1-score also improved from 66% to 82%, which has reflected a more balanced performance across both classes. Overall, the Tuned SVM model shows a clear enhancement in its ability to predict employee resignations, especially in identifying resigned employees. This makes the model a more reliable tool for HR decision-making and retention strategies.

The Tuned SVM model is better than the Base model because tuning allowed the model to capture underlying patterns in the data more effectively and improve its overall classification ability. The

significant improvements in precision and recall for resigned employees (class 1) suggest that the tuning made the model more accurate and reliable in identifying both classes of employees. The increase in precision for non-resigned employees and the higher recall for resigned employees ensure that HR teams can take timely and informed actions such as focusing retention efforts on employees at risk of leaving.

### 5.2.8.2 Learning Curve

```
from sklearn.model_selection import learning_curve

# Generate learning curve data
train_sizes, train_scores, test_scores = learning_curve(
    estimator=best_svm_model,
    X=X_train_sampled,
    y=y_train_sampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

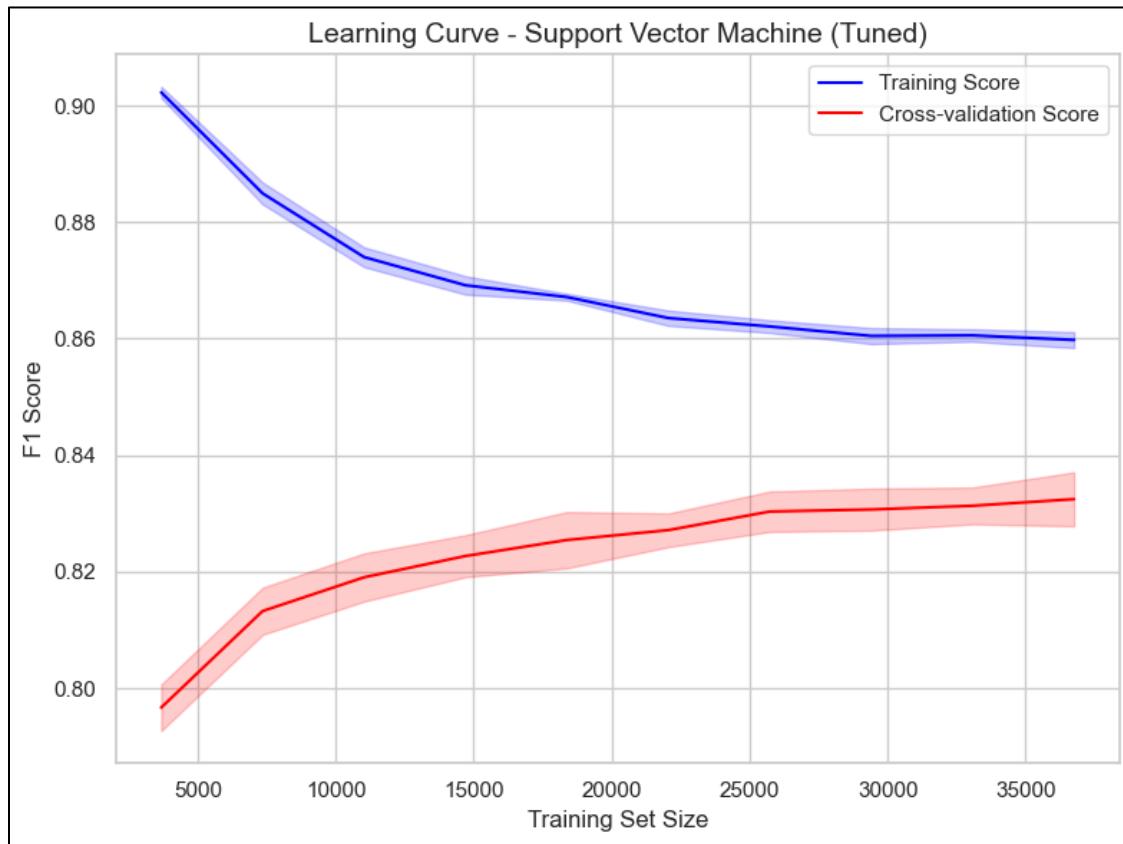
# Calculate mean and standard deviation
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot learning curve
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label="Training Score", color="blue")
plt.plot(train_sizes, test_mean, label="Cross-validation Score", color="red")
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color="blue")
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color="red")
plt.title("Learning Curve - Support Vector Machine (Tuned)", fontsize=14)
plt.xlabel("Training Set Size")
plt.ylabel("F1 Score")
plt.legend(loc="best")
plt.grid(True)
plt.tight_layout()
plt.show()
```

*Figure 165: Source Code for Learning Curve of Support Vector Machine Tuned Model*

The figure above shows the source code for generating a learning curve of the Support Vector Machine Tuned Model using stratified 5-fold cross-validation. The “learning\_curve()” function from “sklearn.model\_selection” is used to evaluate the model’s performance across varying training set sizes. The F1 score is chosen as the performance metric as it is suitable for imbalanced classification tasks such as employee churn prediction.

The code calculates the mean and standard deviation of both training and cross-validation scores which are plotted using Matplotlib. The blue curve represents the training score while the red curve shows the cross-validation score with shaded areas indicating one standard deviation from the mean.



*Figure 166: Learning Curve of Support Vector Machine Tuned Model*

The figure above shows the **learning curve of the Support Vector Machine (SVM) Tuned Model**, which was trained using 50% of the full training dataset. Compared to the base model, the tuned version demonstrates a clear performance improvement. The training score (blue line) starts high at around 0.90 and gradually decreases, stabilizing near 0.86 which is indicating reduced overfitting. In contrast to the base model where the cross-validation score plateaued around 0.66, the tuned model shows a steady and notable rise in the cross-validation score (red line) and is increasing from approximately 0.79 to 0.83 as the training size grows.

This consistent upward trend in validation performance with a narrower gap between training and validation curve has indicates that the tuned model generalizes significantly better to unseen data.

It reflects improved stability and reliability in scenarios with larger datasets. These results confirm that tuning has enhanced the SVM's learning capacity which has allowing the model to maintain higher predictive power while minimizing the risk of overfitting present in the base model.

### 5.2.8.3 ROC and AUC

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc

y_proba = best_svm_model.predict_proba(x_test)[:, 1]

# ROC curve and AUC
fpr, tpr, _ = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure(figsize=(6, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {roc_auc:.3f})')
plt.plot([0, 1], [0, 1], linestyle='--', color='black', label='Random Guess')
plt.xlabel('False Positive Rate', fontsize=12)
plt.ylabel('True Positive Rate', fontsize=12)
plt.title('ROC Curve - Support Vector Machine (Tuned)', fontsize=14)
plt.legend(loc='lower right')
plt.tight_layout()
plt.show()
```

*Figure 167: Source Code for ROC AUC Score of Support Vector Machine Tuned Model*

The figure above shows the source code of the ROC (Receiver Operating Characteristic) curve for the Support Vector Machine Tuned Model. It begins by importing a few functions from “sklearn.metrics”, “roc\_curve” to compute the false positive rate (FPR) and true positive rate (TPR) and “roc\_auc\_score” to calculate the AUC (Area Under the Curve) score. The predicted probabilities of the positive class (class 1) are obtained from the model using “predict\_proba”.

Then, The ROC curve is plotted using “matplotlib.pyplot”, where the x-axis represents the false positive rate and the y-axis represents the true positive rate. Additionally, the AUC score is displayed on the graph to provide a numerical summary of the model's performance.

The plot includes a grid for better readability and the dashed line represents the baseline for random classification. The model's performance is assessed by comparing how far the ROC curve deviates from the diagonal line with a greater deviation indicating better performance.

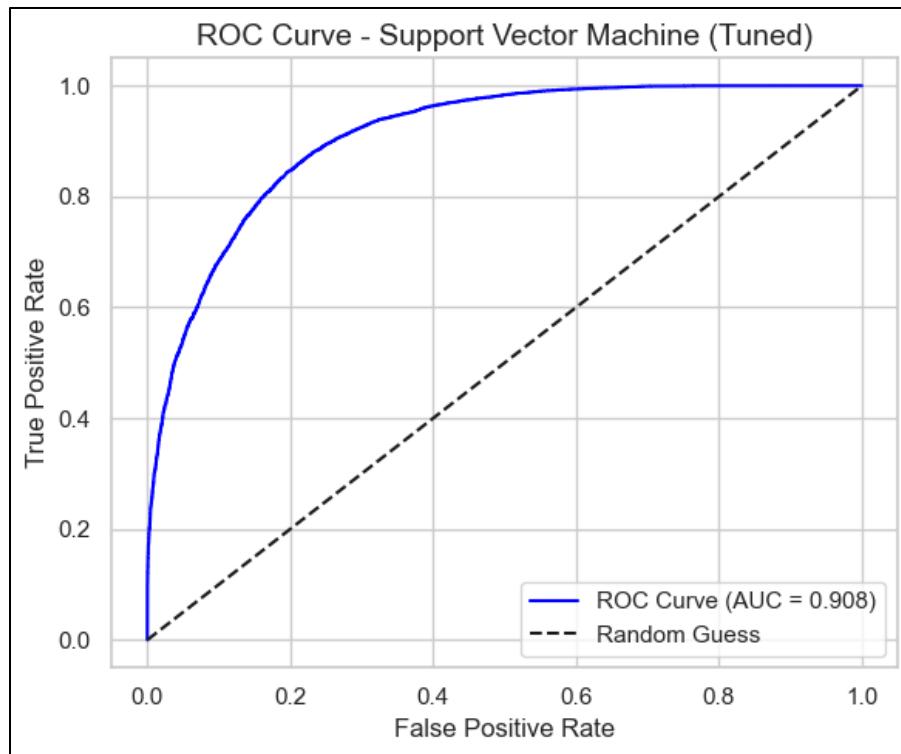


Figure 168: ROC AUC Score of Support Vector Machine Tuned Model

The figure above displays the **ROC curve for the Support Vector Machine (SVM) Tuned Model**, which shows a significant improvement in performance with an AUC score of **0.908**. The ROC curve shows a steep rise toward the top-left corner, and this indicates a higher true positive rate with a substantially lower false positive rate compared to the base model. This improvement in the AUC score reflects the enhanced ability of the tuned model to more effectively discriminate between resigned and non-resigned employees. The tuning process has optimized the model and boosted its ability to correctly identify true positives to make the model reliable for predicting employee churn. The results prove that the tuned model performs significantly better than the base model with the improved classification capabilities.

## 5.2.9 Comparison and Discussion of All Models

### 5.2.9.1 Analysis of Base Models for Employee Churn Prediction

Model Type	Accuracy	Precision (Class 0)	Precision (Class 1)	Recall (Class 0)	Recall (Class 1)	F1-Score (Macro Average)
<b>Random Forest</b>	0.79	0.83	0.75	0.80	0.78	0.79
<b>XGBoost</b>	0.83	0.88	0.78	0.82	0.85	0.83
<b>Gradient Boosting</b>	0.85	0.90	0.80	0.84	0.87	0.85
<b>Support Vector Machine (SVM)</b>	0.66	0.73	0.59	0.65	0.67	0.66

Table 14: Comparison of Machine Learning Algorithms (Base Model)

The table above displays the performance of four machine learning base models which are Random Forest, XGBoost, Gradient Boosting and Support Vector Machine (SVM) for predicting employee churn with the target variable being employee resignation, “Resigned”. All the models were evaluated based on key classification metrics such as accuracy, precision, recall and F1-score.

The Random Forest model achieved an accuracy of 79%. It demonstrated high precision for predicting non-resigned employees (83%) but showed slightly lower precision for predicting resigned employees (75%). This indicates that it accurately identifies employees who stay but has misclassified a notable portion of those who resign. In terms of recall, Random Forest performed well for both non-resigned employees (80%) and resigned employees (78%) although it still missed a significant number of actual resignations. The overall F1-score of 0.79 reflects a balanced trade-off between precision and recall.

XGBoost showed better performance than Random Forest which achieved an accuracy of 83% in both precision and recall for non-resigned employees. Its precision for non-resigned employees was 88% and it also exhibited a slightly higher recall (82%) for correctly identifying non-resigned

employees. For resigned employees, XGBoost's recall was notably higher (85%) than Random Forest which indicated it is better at identifying employees likely to resign. The F1-score of 0.83 further proves that XGBoost's strong and reliable performance for churn prediction.

The Gradient Boosting model outperformed both Random Forest and XGBoost in achieving the highest accuracy (85%) and making it the most accurate model in this comparison. It achieved the highest precision for non-resigned employees (90%) and showed great accuracy in predicting employees who would stay. Additionally, Gradient Boosting excelled in recall for the resigned class (87%), successfully identifying most employees who were likely to leave. The overall F1-score of 0.85 reflects the best balance between precision and recall which proves that it is the most effective model for predicting employee churn among the base model.

In contrast, the Support Vector Machine (SVM) model performed the weakest with an accuracy of only 66%. This could be partly because unlike the other models which used the full training set, SVM was limited to using only 50% of the data due to computational cost and time constraints. As a result, it struggled to classify both non-resigned and resigned employees correctly as shown by its low precision (73% for non-resigned, 59% for resigned) and recall (65% for non-resigned, 67% for resigned). With a low F1-score of 0.66, the SVM model failed to strike an optimal balance between precision and recall, which made it the least suitable choice for this prediction.

Overall, Gradient Boosting Base Model emerged as the top performer which was offering the most balanced and reliable predictions for employee churn. It was closely followed by XGBoost Base Model as both models demonstrate the best accuracy, precision, recall and F1-scores. These results had made them ideal candidates for further optimization in employee churn prediction. Random Forest Base Model showed a solid performance but was outperformed by Base Model of Gradient Boosting and XGBoost. However, the SVM Base Model showed weaker performance compared to other models. Instead, it will be tuned to assess whether performance can be improved.

Based on this analysis, Gradient Boosting Base Model is currently the best model for predicting employee churn and optimizing workforce productivity. However, the final decision will be based on the results of tuning the models. Further evaluation of the tuned models is necessary to assess potential improvements in performance which will be discussed in the next [section 5.2.9.2](#).

### 5.2.9.2 Analysis of Tuned Models for Employee Churn Prediction

Model Type	Accuracy	Precision (Class 0)	Precision (Class 1)	Recall (Class 0)	Recall (Class 1)	F1-Score (Macro Average)
<b>Random Forest</b>	0.85	0.90	0.80	0.83	0.87	0.85
<b>XGBoost</b>	0.85	0.90	0.80	0.83	0.88	0.85
<b>Gradient Boosting</b>	0.85	0.90	0.80	0.84	0.87	0.85
<b>Support Vector Machine (SVM)</b>	0.82	0.88	0.76	0.80	0.85	0.82

Table 15: Comparison of Machine Learning Algorithms (Tuned Model)

The table above displays the performance of the tuned models which are Random Forest, XGBoost, Gradient Boosting and Support Vector Machine (SVM) for predicting employee churn by using the same key metrics as the base models which are accuracy, precision, recall and F1-score.

The Random Forest tuned model achieved an accuracy of 85% and this is showing an improvement over its base model (79%). It demonstrated high precision for non-resigned employees (90%) and precision for resigned employees (80%). The recall for non-resigned employees is 83% while the recall for resigned employees is 87% which is showing balanced identification of both classes. The F1-score (Macro Average) of 0.85 reflects a good balance between precision and recall which are aligning with the accuracy improvements. This indicates that the Random Forest Tuned Model has become a strong contender for predicting employee churn with improved performance over the base model.

The XGBoost Tuned Model also achieved an accuracy of 85%. The precision for non-resigned employees is 90% and for resigned employees is 80%. The recall for non-resigned employees is 83% and for resigned employees which has increased slightly to 88%. This is showing that XGBoost Tuned Model is better at identifying employees who are likely to resign. The F1-score (Macro Average) of 0.85 also has reflecting a well-balanced performance in terms of precision and

recall. All the results shows that XGBoost has reached an optimized state after tuning when compared to Random Forest.

The Gradient Boosting Tuned Model also achieved an accuracy of 85% with matching both Random Forest and XGBoost's accuracy. Precision for non-resigned employees is 90% and precision for resigned employees is 80% which is similar the other models. The recall for non-resigned employees is 84% while recall for resigned employees is 87%. This is indicating that Gradient Boosting slightly outperforms the other two models in terms of recall for non-resigned employees. Its F1-score (Macro Average) is 0.85, consistent with the performance of both Random Forest and XGBoost. As with the other tuned models, Gradient Boosting's performance remains stable after optimization and continues to demonstrate high efficiency in predicting employee churn.

The SVM tuned model achieved an accuracy of 82% which is lower than the 85% achieved by the other three models but it still an improvement over its base model (66%). It showed 88% precision for non-resigned employees and 76% precision for resigned employees which is slightly lower than the precision scores of the other models. The recall for non-resigned employees is 80% and for resigned employees, it improved to 85%. This suggested that the tuning helped SVM better identify employees likely to resign. The F1-score (Macro Average) of 0.82 is the lowest among the tuned models which have shown improvement after tuning. However, it still does not achieve the same balance between precision and recall as the other models which may be due to the use of only 50% of the training set.

Overall, the tuned models show the improved performance compared to their base models. Random Forest, XGBoost and Gradient Boosting all achieved 85% accuracy with similar precision, recall, and F1-scores and has indicating that these models are highly optimized and equally effective for predicting employee churn. SVM showed improvement but still requires additional optimization to match the performance of the other models. Therefore, Gradient Boosting and XGBoost remain the top contenders for predicting employee churn and optimizing workforce productivity based on the overall results.

### 5.2.9.3 Justification for Selecting the Best Model

Based on the analysis of both the XGBoost and Gradient Boosting models, XGBoost was selected as the best model for predicting employee churn and optimizing workforce productivity. Both models have achieved 85% accuracy with similar precision and F1-scores which are demonstrating strong overall performance. However, the key differentiator between the two models lies in their recall for predicting resigned employees. XGBoost demonstrated a slightly higher recall (88%) for identifying employees likely to resign compared to Gradient Boosting's recall of 87%. This small but significant difference indicates that XGBoost is more effective at identifying employees at risk of leaving and this is important for business companies aiming to take early action to prevent turnover.

Besides, the further support for selecting XGBoost comes from the learning curve analysis. XGBoost demonstrated a minimal gap between training and validation scores which has indicating low overfitting and better generalization. Its validation score started slightly lower but quickly converged to match the training score which reached a final value around of 0.855. This pattern suggests that XGBoost generalizes better to unseen data and is more efficient and requiring less data to achieve peak performance compared to Gradient Boosting. While for the Gradient Boosting showed a slightly larger initial gap between training and validation scores with a final validation score of around 0.850. This has indicated that the model needs more data to stabilize and reach optimal performance. The slower convergence and slightly less steady curve in Gradient Boosting make it less efficient for real-time applications. The details of the learning curves for both can be found in [Section 5.2.4.2](#) (XGBoost Tuned Model Learning Curve) and [Section 5.2.6.2](#) (Gradient Boosting Tuned Model Learning Curve).

To contextualize the performance of the XGBoost model, its metrics are compared with those from two previous research studies which had discussed in [Section 2.3](#) in the Literature Review. These studies also identified XGBoost as an optimal solution for employee attrition prediction which is Kovvuri & Dommetti (2023) and Akinode & Bada (2022). This comparison serves to assess whether the current implementation aligns with established benchmarks and to highlight the specific advancements the model offers over prior work.

Metric	Developed Model	Kovvuri & Dommeti (2022)	Akinode & Bada (2022)
<b>Accuracy</b>	85.0%	85.3%	<b>85.4%</b>
<b>Precision (1)</b>	80.0%	<b>88.0%</b>	57.4%
<b>Recall (1)</b>	<b>88.0%</b>	69.0%	45.5%
<b>F1-Score (1)</b>	<b>85.0%</b>	77.0%	50.7%
<b>ROC AUC</b>	<b>94.0%</b>	88.0%	62.5%

*Table 16: Comparison of XGBoost Performance Metrics for Employee Attrition Prediction*

In the comparison table shown above, the developed model that have built achieves 85.0% accuracy which is comparable to the 85.3% from Kovvuri & Dommeti (2022) and 85.4% from Akinode & Bada (2022). However, the developed model stands out in recall (88%) compared to 45.5%–69% in the other studies. This is demonstrating that the developed model has a superior ability to identify employees at risk of leaving. This is very important in churn prediction where early intervention is key to improving retention. It should be noted that the highest results in each metric have been bolded for ease of reference and analysis.

Similarly, the developed model outperforms the others in ROC AUC (94%) while the others range from 62.5% to 88%. A higher ROC AUC reflects the developed model's ability to make more accurate distinctions between employees who will stay and those at risk of leaving. This has showed that the developed model is more reliable predictions across various thresholds.

While the developed model excels in recall and ROC AUC, precision (80%) is lower than Kovvuri & Dommeti (2022) which achieved 88% precision. Precision is essential to avoid false positives which is the incorrectly identifying employees at risk when they are not resigned. The trade-off between precision and recall is common in churn models where improvements in one metric often led to a reduction in the other. In this case, the developed model prioritizes recall to ensure that more at-risk employees are flagged for potential retention interventions. This approach is critical for early intervention for allowing HR professionals to take timely action to retain valuable employees.

These advancements highlight that while the developed model is comparable to previous benchmarks in terms of accuracy, it offers significant improvements in its ability to identify high-risk employees as evidenced by the superior recall and ROC AUC. These enhancements not only

validate the effectiveness of the developed model but also reinforce its practical value in real-world HR applications. This is because actionable insights provided by the model are essential for optimizing employee retention efforts. By choosing XGBoost, it aligns with the project's objective to evaluate, compare and select the best machine learning models based on performance metrics. XGBoost's superior ability to generalize, faster convergence and minimal overfitting support the goal of providing actionable insights for employee retention and productivity optimization. Additionally, its efficiency in handling large datasets, built-in regularization and faster convergence make it a better fit for real-world applications where incremental updates and scalability are crucial. The research support and details of XGBoost can be found in [section 2.2.4.5](#).

The model's ability to offer quicker insights and predictions allows HR professionals to take immediate and data-driven actions to optimize retention strategies. While Gradient Boosting demonstrated strong performance and was similar to XGBoost in many aspects, XGBoost provides the significant advantages in recall for resigned employees, handling imbalanced data and maintaining higher stability. These factors make XGBoost the top choice for this project. Its consistent performance in critical areas and its effective handling of real-world data complexities led to its selection as the best model for predicting employee churn and optimizing workforce productivity.

### 5.3 Feature Importance

```

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

# Create a Series with feature importances
feature_importance = pd.Series(best_xgboost.feature_importances_, index=X.columns)

# Sort top 10 features
top_10_features = feature_importance.sort_values(ascending=False).head(10)

print("\nTop 10 Important Features (XGBoost):")
print(top_10_features)

plt.figure(figsize=(10, 6))
sns.barplot(
    x=top_10_features.values,
    y=top_10_features.index,
    color="#D291BC"
)
plt.title('Top 10 Feature Importances (XGBoost)', fontsize=14)
plt.xlabel('Importance Score')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()

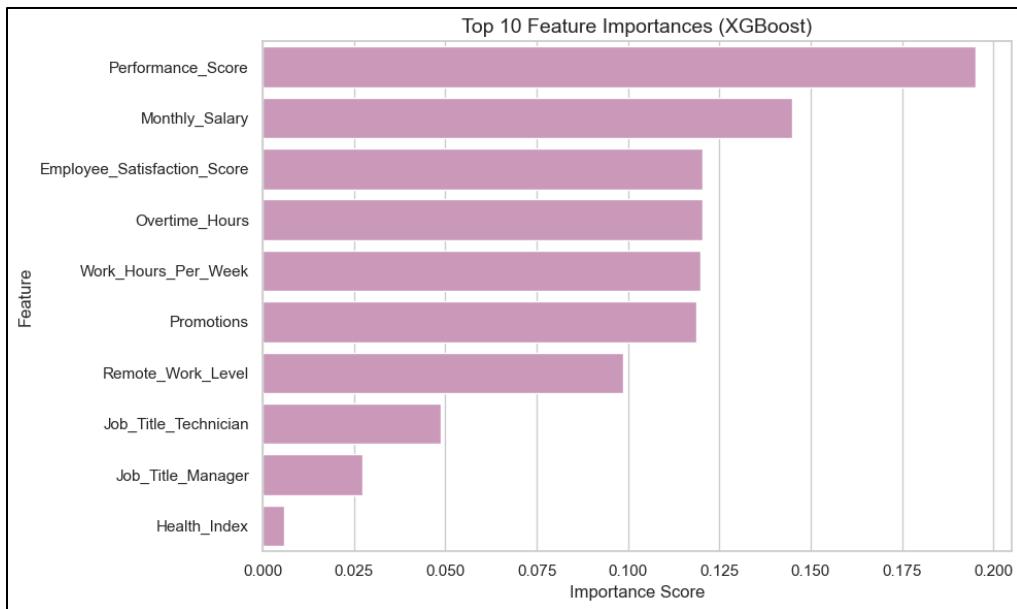
```

*Figure 169: Source Code of Feature Importance for XGBoost*

The source code above shows the calculation and visualizes the top 10 most important features for a machine learning model trained using XGBoost. First, it extracts the feature importance scores from the trained model “best\_xgboost.feature\_importances\_” and associates them with the feature names using a pandas Series. Then, the features are sorted by importance and the top 10 are selected for further analysis. A bar plot is created using Seaborn to visualize these top features to show the importance score on the x-axis and the feature names on the y-axis. This visualization allows for easily interpret which features are most influential in predicting the target variable.

Top 10 Important Features (XGBoost):	
Performance_Score	0.195200
Monthly_Salary	0.144966
Employee_Satisfaction_Score	0.120462
Overtime_Hours	0.120367
Work_Hours_Per_Week	0.119715
Promotions	0.118770
Remote_Work_Level	0.098767
Job_Title_Technician	0.048677
Job_Title_Manager	0.027199
Health_Index	0.005876
dtype: float32	

*Figure 170: Top 10 Importance Features (XGBoost)*



*Figure 171: Horizontal Bar Chart of Top 10 Importance Features (XGBoost)*

Figure above shows the Top 10 Importance Features for the calculation and the bar chart for a machine learning model trained using XGBoost. Upon analyzing the output, the Top 5 Important Features identified by the XGBoost Tuned Model are:

1. Performance\_Score (0.195200)
2. Monthly\_Salary (0.144966)
3. Employee\_Satisfaction\_Score (0.120462)
4. Overtime\_Hours (0.120367)
5. Work\_Hours\_Per\_Week (0.119715)

These features are assigned as the highest importance scores which signal their strong influence on predicting employee outcomes for the resignation.

“Performance\_Score” is the most significant predictor of employee churn. The higher performance employees tend to stay with the company due to better recognition, career advancement and financial incentives (Tirta et al., 2020). However, research shows that high performers are 2.5 times more likely to leave if they feel treated unfairly or see no growth opportunities compared to moderate performers (Gadi et al., 2018). This is especially true in competitive industries like technology where the top talent is often poached by other companies offering better compensation.

A 2023 study revealed that tech companies with poor performance differentiation systems experienced 35% higher annual churn among top-tier employees (Tirta et al., 2020). This highlights the critical need for organizations to align performance management systems with retention strategies for high-potential staff.

“Monthly\_Salary” plays a significant role in employee turnover. Compensation dissatisfaction accounts for nearly 70% of voluntary churn in sectors with transparent pay benchmarks like finance and consulting (Hills & Laura, 2021). Employees whose salaries fall below 80% of the tmarket median are 3.2 times more likely to leave within six months (Duchini et al., 2023). Besides, pay transparency laws have mentioned this issue with organizations that disclose salary ranges publicly experiencing 28% higher churn rates among underpaid employees compared to those with opaque compensation structures (Duchini et al., 2023). This highlights the need to adopt fair and competitive compensation strategies to reduce turnover.

The “Employee\_Satisfaction\_Score” shows a complex relationship with churn. Employees with the lowest 20% satisfaction scores have a 65% chance of leaving each year. However, even employees in the top 20% for satisfaction still face a 12% chance of leaving due to external factors like the job market (Enander & Cardoso, 2020). This suggests that while satisfaction helps reduce churn, it doesn't eliminate it entirely (Thome et al., 2020).

Besides, “Overtime\_Hours” and “Work\_Hours\_Per\_Week” also show a clear relationship with churn. The moderate overtime between 5 to 10 hours per week correlates with an 8% lower churn rate which has indicating higher engagement. However, working more than 15 overtime hours per week leads to a 42% increase in churn probability (Shoman et al., 2021). Additionally, post-Covid-19 pandemic data reveals that remote employees working excessive hours experience burnout to churn progression 30% faster than office-based counterparts (Kelly & Moen, 2020). This highlights the need for revised workload management strategies, especially in hybrid work environments.

While “Promotions” with 0.118770 score is slightly less important than the top 5 features, but it still plays an important role in predicting employee turnover. Historically, “Promotions” have been a strong retention tool, but recent data shows their diminishing effectiveness in today’s labor markets. A study found that the protective effect of promotions lasts only 6 to 9 months compared

to 18 months in the past (Udayar et al., 2024). Similarly, “Remote\_Work\_Level” with 0.098767 impacts churn with fully remote employees experiencing 12% higher churn than hybrid workers, while office-mandated roles suffer a 22% higher churn rate than hybrid positions (Emanuel, et al., 2024). These findings challenge traditional retention strategies and emphasize the need for more tailored approaches.

In conclusion, the top 5 features which is “Performance\_Score”, “Monthly\_Salary”, “Employee\_Satisfaction\_Score”, “Overtime\_Hours” and “Work\_Hours\_Per\_Week” are the most influential factors in predicting employee resignation. While “Promotions” and “Remote\_Work\_Level” are still relevant, their impact on reducing churn is not as strong as other factors in today’s rapidly changing work environment. This analysis has shown the importance of adapting retention strategies focusing on performance differentiation, fair compensation practices and tailored work arrangements to mitigate churn.

## 5.4 Model Deployment

```
import joblib  
  
joblib.dump(best_xgboost, 'employee_churn_model.pkl')  
  
['employee_churn_model.pkl']
```

Figure 172: Source Code of Saving the Trained XGBoost Model

The figure above shows the source code for saving the trained XGBoost model to a file. The joblib library is used to serialize and save the machine learning model, “best\_xgboost” into a file. The model is saved with the filename “employee\_churn\_model.pkl”. This method allows the model to be easily reloaded for the HR system that will be built later and enable efficient predictions without the need to retrain it. Therefore, saving the model in this way is important especially when preparing models for deployment in real-world applications where quick access to pre-trained models is necessary.

### 5.4.1 Generating Data for Dynamic Updates in Model Deployment

To enable periodic updates in the system, it was necessary to extend the original dataset which only contained data up until 2024. Since the system requires up-to-date information which has including data for 2025, the dataset was refreshed with synthetic data calculated based on quarterly trends

To simulate recent patterns and ensure the system remains functional with the latest information, there was over 4,500 rows of synthetic data were generated using Faker. This approach allows the system to reflect changes in the dataset and provides a semi-dynamic experience where updates occur based on user interactions rather than continuous live data streams.

```

import pandas as pd
import numpy as np
from faker import Faker
import time

# Original dataset
df = pd.read_csv('Extended_Employee_Performance_and_Productivity_Data.csv')
df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], dayfirst=True).dt.strftime('%d/%m/%Y')

# Initialize Faker for realistic fake data
fake = Faker()

# Precompute department-job title mappings for faster access
dept_job_mapping = df.groupby('Department')['Job_Title'].unique().to_dict()
job_edu_mapping = df.groupby('Job_Title')['Education_Level'].unique().to_dict()
job_perf_salary_mapping = df.groupby(['Job_Title', 'Performance_Score'])['Monthly_Salary'].apply(list).to_dict()

def generate_new_records_fast(n):
    # Generate employee IDs
    emp_ids = np.arange(1000000, 1000000 + n)

    # Sample departments and genders
    departments = np.random.choice(df['Department'].unique(), size=n)
    genders = np.random.choice(df['Gender'].unique(), size=n)

    # Get job titles based on departments
    job_titles = np.array([np.random.choice(dept_job_mapping[dept]) for dept in departments])

    # Generate ages, years at company within 1 years
    ages = np.random.randint(28, 68, size=n)
    years_at_company = np.random.randint(0, 1, size=n)

    # Generate hire dates between July 1, 2024 to July 1, 2025
    start_date = pd.to_datetime('2024-07-01')
    end_date = pd.to_datetime('2025-07-01')
    date_range = (end_date - start_date).days
    random_days = np.random.randint(0, date_range + 1, size=n)
    hire_dates = pd.Series(start_date + pd.to_timedelta(random_days, unit='D')).dt.strftime('%d/%m/%Y')

    # Get education levels based on job titles
    educations = np.array([np.random.choice(job_edu_mapping.get(job, df['Education_Level'].unique())) for job in job_titles])

    # Generate performance scores
    performances = np.random.choice([1, 2, 3, 4, 5], size=n)

    # Get salaries based on job title and performance
    salaries = np.array([
        np.random.choice(job_perf_salary_mapping.get((job, perf), df[df['Job_Title'] == job]['Monthly_Salary'].values))
        for job, perf in zip(job_titles, performances)
    ])

    # Generate other fields
    work_hours = np.random.randint(30, 60, size=n)
    projects = np.random.randint(0, 50, size=n)
    overtime = np.random.randint(0, 30, size=n)
    sick_days = np.random.randint(0, 15, size=n)
    remote_freq = np.random.randint(30, 80, size=n)
    team_size = np.random.randint(1, 20, size=n)
    training_hours = np.random.randint(0, 20, size=n)
    promotions = np.random.randint(0, 3, size=n)
    satisfaction = np.round(np.random.uniform(1, 5, size=n), 2)

    # Create DataFrame
    new_data = pd.DataFrame({
        'Employee_ID': emp_ids,
        'Department': departments,
        'Gender': genders,
        'Age': ages,
        'Job_Title': job_titles,
        'Hire_Date': hire_dates,
        'Years_At_Company': years_at_company,
        'Education_Level': educations,
        'Performance_Score': performances,
        'Monthly_Salary': salaries,
        'Work_Hours_Per_Week': work_hours,
        'Projects_Handled': projects,
        'Overtime_Hours': overtime,
        'Sick_Days': sick_days,
        'Remote_Work_Frequency': remote_freq,
        'Team_Size': team_size,
        'Training_Hours': training_hours,
        'Promotions': promotions,
        'Employee_Satisfaction_Score': satisfaction,
        'Resigned': False
    })

    return new_data

start_time = time.time()

# Generate 4,500 new records with July 2024-July 2025 hire dates
new_records = generate_new_records_fast(4500)

# Combine with original data
extended_df = pd.concat([df, new_records], ignore_index=True)
extended_df['Hire_Date'] = pd.to_datetime(extended_df['Hire_Date'], dayfirst=True)

```

Figure 173: Source Code for Generating New Data for Dynamic Updates

```

# Generate other fields
work_hours = np.random.randint(30, 60, size=n)
projects = np.random.randint(0, 50, size=n)
overtime = np.random.randint(0, 30, size=n)
sick_days = np.random.randint(0, 15, size=n)
remote_freq = np.random.randint(30, 80, size=n)
team_size = np.random.randint(1, 20, size=n)
training_hours = np.random.randint(0, 20, size=n)
promotions = np.random.randint(0, 3, size=n)
satisfaction = np.round(np.random.uniform(1, 5, size=n), 2)

# Create DataFrame
new_data = pd.DataFrame({
    'Employee_ID': emp_ids,
    'Department': departments,
    'Gender': genders,
    'Age': ages,
    'Job_Title': job_titles,
    'Hire_Date': hire_dates,
    'Years_At_Company': years_at_company,
    'Education_Level': educations,
    'Performance_Score': performances,
    'Monthly_Salary': salaries,
    'Work_Hours_Per_Week': work_hours,
    'Projects_Handled': projects,
    'Overtime_Hours': overtime,
    'Sick_Days': sick_days,
    'Remote_Work_Frequency': remote_freq,
    'Team_Size': team_size,
    'Training_Hours': training_hours,
    'Promotions': promotions,
    'Employee_Satisfaction_Score': satisfaction,
    'Resigned': False
})

return new_data

start_time = time.time()

# Generate 4,500 new records with July 2024-July 2025 hire dates
new_records = generate_new_records_fast(4500)

# Combine with original data
extended_df = pd.concat([df, new_records], ignore_index=True)
extended_df['Hire_Date'] = pd.to_datetime(extended_df['Hire_Date'], dayfirst=True)

```

Figure 174: Source Code for Generating New Data for Dynamic Updates

To ensure the system can accurately calculate churn risk on a quarterly basis and reflect the latest workforce changes, the original dataset was used to continue for this model deployment section. This dataset, named "Extended\_Employee\_Performance\_and\_Productivity\_Data.csv," which contained employee performance and productivity data only up until 2024. Then, it was extended by generating synthetic data for the year 2025. To achieve this, there are over 4,500 new employee records generated using the Faker data generation library which allowed for the creation of realistic and diverse employee attributes such as employee IDs, departments, genders, job titles, performance scores, salaries and other work-related variables.

```

# Create Resignation_Date column (initially empty)
extended_df['Resignation_Date'] = pd.NaT

# Get all resigned employees
resigned_mask = extended_df['Resigned'] == True
resigned = extended_df[resigned_mask]

# Randomly select 885 for Q1 (Jan-Mar 2025) and 478 for Q2 (Apr-Jun 2025)
q1_resignations = resigned.sample(885, random_state=42)
remaining_resigned = resigned.drop(q1_resignations.index)
q2_resignations = remaining_resigned.sample(478, random_state=42)
other_resignations = remaining_resigned.drop(q2_resignations.index)

# Assign Q1 resignation dates (Jan-Mar 2025)
q1_start = pd.to_datetime('01/01/2025', dayfirst=True)
q1_end = pd.to_datetime('31/03/2025', dayfirst=True)
days_in_q1 = (q1_end - q1_start).days
random_days = np.random.randint(0, days_in_q1 + 1, size=len(q1_resignations))
extended_df.loc[q1_resignations.index, 'Resignation_Date'] = q1_start + pd.to_timedelta(random_days, unit='D')

# Assign Q2 resignation dates (Apr-Jun 2025)
q2_start = pd.to_datetime('01/04/2025', dayfirst=True)
q2_end = pd.to_datetime('30/06/2025', dayfirst=True)
days_in_q2 = (q2_end - q2_start).days
random_days = np.random.randint(0, days_in_q2 + 1, size=len(q2_resignations))
extended_df.loc[q2_resignations.index, 'Resignation_Date'] = q2_start + pd.to_timedelta(random_days, unit='D')

# Assign random past dates for other resignations (before 2025)
past_start = pd.to_datetime('01/01/2018', dayfirst=True)
past_end = pd.to_datetime('31/12/2024', dayfirst=True)
days_in_past = (past_end - past_start).days
random_days = np.random.randint(0, days_in_past + 1, size=len(other_resignations))
extended_df.loc[other_resignations.index, 'Resignation_Date'] = past_start + pd.to_timedelta(random_days, unit='D')
extended_df['Hire_Date'] = extended_df['Hire_Date'].dt.strftime('%d/%m/%Y')
extended_df['Resignation_Date'] = extended_df['Resignation_Date'].dt.strftime('%d/%m/%Y')

# Save to new CSV
extended_df.to_csv("Deployment_Used_Extended_Dataset_With_Resignation.csv", index=False)

# Verification
print(f"Execution completed in {time.time() - start_time:.2f} seconds")
print("\nResignation date distribution:")
print(extended_df[resigned_mask]['Resignation_Date'].value_counts().sort_index())
print(f"\nTotal records: {len(extended_df)}")
print(f"Resigned employees: {resigned.sum()}")
print(f"- Q1 2025 resignations (Jan-Mar): {len(q1_resignations)}")
print(f"- Q2 2025 resignations (Apr-Jun): {len(q2_resignations)}")
print(f"- Other resignations: {len(other_resignations)}")
print(f"Current employees (no resignation date): {len(extended_df) - resigned.sum()}")

```

*Figure 175: Source Code for Generating New Data for Dynamic Updates*

Additionally, to calculate churn risk on a quarterly basis, a "Resignation\_Date" column was created and added. This column was important for tracking employee resignations and the data was split into resignation periods for Q1 (Jan-Mar 2025) and Q2 (Apr-Jun 2025) along with employees who had resigned before 2025. By using the Faker library's capabilities, random resignation dates were assigned within these periods to simulate the realistic employee turnover. This allowed the dataset to reflect accurate resignation patterns, which are essential for predicting churn risk in a timely manner.

The extended dataset which named “Deployment\_Use\_Extended\_Dataset\_with\_Resignation.csv” is containing both new employees and resignation data to ensure that the system can calculate up-to-date churn risk to allow for the simulation of real-world employee turnover trends. This method enhances the accuracy of churn predictions and supports the model’s adaptability to future workforce changes.

```
Total records: 104500
Resigned employees: 42578
  - Q1 2025 resignations (Jan-Mar): 805
  - Q2 2025 resignations (Apr-Jun): 478
  - Other resignations: 41295
Current employees (no resignation date): 61922
```

*Figure 176: Output of Generating New Data*

The figure above displays the output for the results from extending the dataset and incorporating resignation data for employee churn risk analysis. The total number of records for the new dataset is 104,500 and 42,578 employees resigned. Out of these, 805 employees resigned in Q1 2025 (Jan-Mar), 478 employees resigned in Q2 2025 (Apr-Jun) and 41,295 employees resigned before 2025. The remaining 61,922 employees have no resignation date which indicates they are current employees. This breakdown allows for an accurate analysis of churn risk based on the latest resignation trends to support the model's ability to simulate real-world employee turnover.

```
import os
import pandas as pd
from tabulate import tabulate

# Define the file path
file_path = "Deployment_Use_Extended_Dataset_with_Resignation.csv"

# Check if the file exists in the current directory
if os.path.isfile(file_path):
    df = pd.read_csv(file_path)
    print("Dataset loaded successfully!")

    # Display the first five rows of the dataset
    print("\nFirst five rows of the dataset:")
    print(tabulate(df.head(), headers='keys', tablefmt='pretty', showindex=False))
else:
    print(f"Error: The file '{file_path}' was not found!")
```

*Figure 177: Source Code for Printing the New Dataset*

The figure above is the source code for printing the first five rows of the dataset in a tabular format using the tabulate library for improved readability. This code is essential for ensuring the dataset

has been loaded correctly and to preview the initial records for verification of the new data structure and columns.

Resignation_Date
nan
nan
23/04/2022
16/10/2024
nan

Figure 178: Output of Resignation\_Date

In the extended dataset, a new column named "Resignation\_Date" was added to track employee resignation details. For employees who have resigned, specific resignation dates were assigned such as "23/04/2022" and "16/10/2024" as shown in the data snippet. For employees who have not resigned, the value in the "Resignation\_Date" column is represented as NaN (Not a Number) which indicates that no resignation date has been assigned. This new column is essential for monitoring employee turnover and calculating churn risk based on resignation trends.

```
print(df.columns)
Index(['Employee_ID', 'Department', 'Gender', 'Age', 'Job_Title', 'Hire_Date',
       'Years_At_Company', 'Education_Level', 'Performance_Score',
       'Monthly_Salary', 'Work_Hours_Per_Week', 'Projects_Handled',
       'Overtime_Hours', 'Sick_Days', 'Remote_Work_Frequency', 'Team_Size',
       'Training_Hours', 'Promotions', 'Employee_Satisfaction_Score',
       'Resigned', 'Resignation_Date'],
      dtype='object')
```

Figure 179: Output for Printing Out of Existing Columns

The dataset now includes a total of 21 columns as shown in the figure output when listing the column names. These columns include key employee attributes such as "Employee\_ID" "Department" "Gender" "Job\_Title" "Hire\_Date" "Years\_At\_Company" and "Employee\_Satisfaction\_Score". One of the newly added columns is "Resignation\_Date," which will help in tracking employees who have resigned and the corresponding dates. This expanded list of columns enables more detailed analysis and facilitates accurate tracking of employee turnover and other relevant metrics.

## 5.4.2 Overview of The Employee Churn Analytics Dashboard System

```

def header():
    if st.session_state.role == "HR":
        st.title("HR Analytics Dashboard")
        st.write(f"Last login: {st.session_state.last_login}")
    elif st.session_state.role == "Talent":
        st.title("Talent Acquisition Dashboard")
        st.write(f"Last login: {st.session_state.last_login}")

    # Refresh button for both HR and Talent Acquisition
    if st.button("Refresh Data"):
        st.session_state.hr_data = load_data()
        st.success("Data refreshed!")
        st.rerun()

    st.markdown("----")

# MAIN FUNCTION
def main():
    init_session_state()

    st.set_page_config(
        page_title="Employee Churn Analytics Dashboard System",
        page_icon="📊",
        layout="wide"
    )

    st.markdown("""
<style>
.main {padding-top: 1rem;}
.stButton button {border-radius: 8px;}
.stAlert {border-radius: 8px;}
</style>
""", unsafe_allow_html=True)

# Check authentication
if not st.session_state.logged_in:
    login_page()
else:
    create_navbar()
    header()

# Page routing based on role
if st.session_state.role == "HR":
    if st.session_state.current_page == "HR Dashboard":
        hr_dashboard()
    elif st.session_state.current_page == "Predict Employee Churn":
        churn_prediction()
    elif st.session_state.current_page == "Overview of Employee Performance":
        employee_performance()
    elif st.session_state.current_page == "Actionable Insights for Retention and Productivity":
        actionable_insights()

elif st.session_state.role == "Talent":
    if st.session_state.current_page == "Talent Insights":
        talent_dashboard()
    elif st.session_state.current_page == "Hiring Recommendations":
        hiring_recommendations()
    elif st.session_state.current_page == "Predict Employee Churn":
        churn_prediction()

if __name__ == "__main__":
    main()

```

Figure 180: Source Code of Employee Churn Analytics Dashboard System 1

The Employee Churn Analytics Dashboard is a web-based application designed to provide key metrics and insights related to HR and talent management. The system accommodates two primary user roles which are HR and Talent. Each role has access to similar core functions but with views tailored specifically to their responsibilities. The application allows users to log in and access their respective dashboards based on their roles. HR users will see a broader overview of organizational metrics while Talent users receive more specialized insights focused on their department or individual performance. The system handles user authentication and ensures that the appropriate dashboard is displayed according to the user's role to provide a personalized experience. Additionally, users can refresh the data displayed on their dashboards using the "Refresh Data" button which reloads and updates the metrics from the excel file. The navigation within the application is dynamic which allows users to switch between various sections.

To ensure security and data integrity, the application features contain a "Login Page" where users must authenticate before accessing their dashboards. This step is important as the system is designed for a company environment where secure access is paramount. For more details about the login process, view [Section 5.4.3.1.3.](#)

For easy access to detailed information about specific functions, the corresponding hyperlinks are provided at the end of each function description. These links guide the relevant section to offer detailed information into the functionality.

Below are the key functions available to **HR Role**:

- **HR Dashboard:** Provides an overview of HR metrics like total employees and churn rate. For detailed information, see [Section 5.4.3.2.1](#) and [Section 5.4.3.2.2](#).
- **Predict Employee Churn:** Offers insights into potential employee resignations to help HR take proactive measures. For detailed information, see [Section 5.4.3.1.5](#).
- **Overview of Employee Performance:** Displays performance data for HR to assess and improve workforce productivity. For detailed information, see [Section 5.4.3.2.3](#).
- **Actionable Insights for Retention and Productivity:** Provides recommendations to enhance employee retention and productivity. For detailed information, see [Section 5.4.3.2.4](#).

Below are the key functions available to **Talent Role**:

- **Talent Dashboard:** Delivers analytics for managing talent, assessing skills and identifying development opportunities. For detailed information, see [Section 5.4.3.3.1](#).
- **Hiring Recommendations:** Suggests hiring strategies based on data-driven insights. For detailed information, see [Section 5.4.3.3.2](#).
- **Predict Employee Churn:** Allows talent to input data and generate churn predictions by helping them identify at-risk employees for targeted retention efforts. For detailed information, see [Section 5.4.3.1.5](#).



Figure 181: Interactive Toolbar for Visualizations

The HR dashboard also offers dynamic and interactive user experience where all the bar charts, graphs and visualizations come equipped with various interactive functions. Users can **zoom in**, **zoom out**, **download** and view the data in **full-screen mode**. These features enhance the analysis

and interpretation of the data with greater flexibility for detailed exploration. This enables HR professionals to effectively monitor and analyze employee metrics to ensure that they have all the necessary tools to make informed decisions.

### 5.4.3 System Functions

```
# Initialize session state variables
def init_session_state():
    if 'logged_in' not in st.session_state:
        st.session_state.logged_in = False
    if 'current_page' not in st.session_state:
        st.session_state.current_page = "Login"
    if 'last_login' not in st.session_state:
        st.session_state.last_login = None
    if 'role' not in st.session_state:
        st.session_state.role = None
    if 'login_attempted' not in st.session_state:
        st.session_state.login_attempted = False
    if 'hr_data' not in st.session_state:
        st.session_state.hr_data = None

import streamlit as st
import pandas as pd
import numpy as np
import joblib
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from io import BytesIO
import plotly.express as px
import plotly.graph_objects as go
from dateutil.relativedelta import relativedelta
import shap
```

Figure 182: Source Code of Employee Churn Analytics Dashboard System 1

```
# Load data
def load_data():
    df_deployment = pd.read_csv('Deployment_Used_Extended_Dataset_with_Resignation.csv')
    df_performance = pd.read_csv('Extended_Employee_Performance_and_Productivity_Data.csv')

    # Store DataFrames separately in session state (recommended)
    st.session_state.df_deployment = df_deployment
    st.session_state.df_performance = df_performance

    return df_deployment, df_performance
```

Figure 183: Source Code of Employee Churn Analytics Dashboard System 2

The source code above includes several functions that are essential for managing session state and loading data in the HR Analytics System. The "`init_session_state()`" function initializes session variables to manage the state of the app.

The "`load_data()`" function is responsible for loading two datasets into Pandas DataFrames. The first dataset, "Deployment\_Used\_Extended\_Dataset\_with\_Resignation.csv" contains simulated employee deployment and resignation data which was generated using the Faker library for testing purposes. The second dataset, "Extended\_Employee\_Performance\_and\_Productivity\_Data.csv" is an original dataset containing real data related to employee performance and productivity. Both datasets are loaded into the session state to be used across various components of the application for analysis and visualization.

These functions work together to ensure that the HR Analytics System functions smoothly by initializing session parameters and loading the necessary data for further processing and analysis.

### 5.4.3.1 Functions Available for Both HR and Talent Acquisition Role

#### 5.4.3.1.1 Employee Churn Analytics System Calculation Functions

```

# Total Employees excluding resigned employees
def get_total_employees(df):
    active_employees = df[df['Resigned'] == False]
    return active_employees.shape[0]

# Churn rate for current quarter and previous quarter
def get_churn_rate(df, start_date, end_date):
    df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], dayfirst=True)
    df['Resignation_Date'] = pd.to_datetime(df['Resignation_Date'], errors='coerce')

    # Employees who resigned in this quarter
    resigned_in_period = df[(df['Resignation_Date'] >= start_date) & (df['Resignation_Date'] <= end_date)]
    active_employees_at_start = df[(df['Hire_Date'] < start_date) & (df['Resigned'] == False)]

    if active_employees_at_start.shape[0] == 0:
        return 0.0

    # Churn rate (Resigned employees / Active employees at the current quarter)
    churn_rate = (resigned_in_period.shape[0] / active_employees_at_start.shape[0]) * 100
    return churn_rate

# Get the start and end dates of the current and previous quarters
def get_quarter_dates(year, quarter):
    """Returns (start_date, end_date) for given year and quarter (1-4)"""
    if quarter == 1:
        return pd.to_datetime(f"{year}-01-01"), pd.to_datetime(f"{year}-03-31")
    elif quarter == 2:
        return pd.to_datetime(f"{year}-04-01"), pd.to_datetime(f"{year}-06-30")
    elif quarter == 3:
        return pd.to_datetime(f"{year}-07-01"), pd.to_datetime(f"{year}-09-30")
    else:
        return pd.to_datetime(f"{year}-10-01"), pd.to_datetime(f"{year}-12-31")

# Get the current year and quarter
def get_current_quarter():
    today = pd.to_datetime('today')
    current_year = today.year
    current_month = today.month
    return current_year, (current_month - 1) // 3 + 1

# Get the comparison quarters based on current date
def get_comparison_quarters():
    """Returns (q1_year, q1_num, q1_start, q1_end, q2_year, q2_num, q2_start, q2_end)"""
    current_year, current_q = get_current_quarter()

    if current_q >= 3: # Compare Q1 vs Q2 of current year
        return (
            current_year, 1, *get_quarter_dates(current_year, 1),
            current_year, 2, *get_quarter_dates(current_year, 2)
        )
    else: # Compare Q3 vs Q4 of previous year
        return (
            current_year-1, 3, *get_quarter_dates(current_year-1, 3),
            current_year-1, 4, *get_quarter_dates(current_year-1, 4)
        )

```

Figure 184: Source Code of Employee Churn Analytics System Calculation Functions 1

```

# Calculate hires in a given period
def get_hired_in_period(df, start_date, end_date):
    df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], format='%m/%d/%Y')
    hired_in_period = df[(df['Hire_Date'] > start_date) & (df['Hire_Date'] <= end_date)]
    return hired_in_period.shape[0]

# Calculate active employees at the end of a period
def get_active_employees_at_end_of_period(df, start_date, end_date):
    df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], dayfirst=True)
    df_period = df[(df['Hire_Date'] <= end_date) & (df['Hire_Date'].dt.year == df['Resigned'].dt.year)]
    active_employees = df_period[df_period['Resigned'] == False]
    return active_employees.shape[0]

# Calculate average satisfaction for the previous quarter
def calculate_average_satisfaction_last_quarter(df):
    last_quarter_cutoff = pd.to_datetime('2025-04-01')
    df_last_quarter = df[df['Hire_Date'] < last_quarter_cutoff]
    avg_satisfaction_last_quarter = df_last_quarter['Employee_Satisfaction_Score'].mean() if not df_last_quarter.empty else 0
    return avg_satisfaction_last_quarter

# Calculate average satisfaction for this quarter (all active employees)
def calculate_average_satisfaction_this_quarter(df):
    avg_satisfaction_this_quarter = df['Employee_Satisfaction_Score'].mean() if not df.empty else 0
    return avg_satisfaction_this_quarter

# Get quarterly metrics
def get_quarterly_metrics(df, year, quarter):
    """Returns all metrics for a given quarter"""
    df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], errors='coerce')

    # Get the start and end date for the quarter
    q_start, q_end = get_quarter_dates(year, quarter)

    # Filter data for the given quarter
    quarter_data = df[
        (df['Hire_Date'] >= q_start) &
        (df['Hire_Date'] <= q_end)
    ]

    active_employees = get_active_employees_at_end_of_period(df, q_start, q_end)
    hires = get_hired_in_period(df, q_start, q_end)
    churn = get_churn_rate(df, q_start, q_end)
    satisfaction = quarter_data['Employee_Satisfaction_Score'].mean()

    return {
        'employees': active_employees,
        'hires': hires,
        'churn': churn,
        'satisfaction': satisfaction if not pd.isna(satisfaction) else 0,
        'year': year,
        'quarter': quarter,
        'start': q_start,
        'end': q_end
    }

```

Figure 185: Source Code of Employee Churn Analytics System Calculation Functions 2

```

# Get previous quarter
def get_previous_quarter(year, quarter):
    if quarter == 1:
        return (year-1, 4)
    return (year, quarter-1)

# Get sorted quarter metrics for trends
def get_sorted_quarters_for_trends(df, current_year, current_quarter):
    """Returns sorted quarter metrics for trends"""

    # Generate last 8 quarters
    quarters = []
    year, quarter = current_year, current_quarter

    for _ in range(8):
        year, quarter = get_previous_quarter(year, quarter)
        if (year < current_year) or (year == current_year and quarter < current_quarter):
            quarters.append((year, quarter))

    # Sort the quarters properly in chronological order
    quarters.sort(key=lambda x: (x[0], x[1]))

    quarter_metrics = []
    for year, quarter in quarters:
        metrics = get_quarterly_metrics(df, year, quarter)
        if metrics:
            quarter_metrics.append(metrics)

    return quarter_metrics

```

Figure 186: Source Code of Employee Churn Analytics System Calculation Functions 3

The figures above show source code of the Employee Churn Analytics System Calculation Functions. These functions calculate various employee metrics such as the total number of active employees, churn rate and employee satisfaction scores. For example, the "`get_total_employees()`" function calculates the number of active employees by excluding resigned individuals while the "`get_churn_rate()`" function computes the churn rate based on employee resignations within a specified period. The "`get_quarter_dates()`" function determines the start and end dates of a given quarter to analyze data over specific time frames.

There are additional functions such as "`get_hired_in_period()`", "`get_active_employees_at_end_of_period()`", and "`calculate_average_satisfaction_last_quarter()`" are helped to track employee turnover, hiring trends and employee satisfaction levels over different quarters. Therefore, together with these functions allow the system to provide important data to analyze employee churn and other HR-related metrics.

### 5.4.3.1.2 Employee Metrics Functions

#### 5.4.3.1.2.1 Common Metrics for HR and Talent

The source code below includes several functions for displaying employee and churn metrics as well as detailed insights into employee data.

```
def display_employee_metrics(q1, q2):
    """Display employee metrics and return button state"""
    employee_change = ((q2['employees'] - q1['employees']) / q1['employees']) * 100 if q1['employees'] > 0 else 0
    st.metric(
        "Total Employees",
        f"{q2['employees']:,}",
        delta=f'{employee_change:.1f}% from Q{q1['quarter']}',
        delta_color="normal" if employee_change >= 0 else "inverse"
    )
    return st.button("View Employee Details", key="employee_details")

def display_churn_metrics(q1, q2):
    """Display churn metrics and return button state"""
    churn_change = q2['churn'] - q1['churn']
    st.metric(
        "Churn Rate",
        f'{(q2['churn']):.2f}%',
        delta=f'{(churn_change):.2f}% from Q{q1['quarter']}',
        delta_color="normal" if churn_change >= 0 else "inverse"
    )
    return st.button("View Churn Details", key="churn_details")
```

Figure 187: Source Code of Display Common Metrics for HR and Talent

The "`display_employee_metrics()`" function shows the total number of employees for a given quarter and calculates the percentage change in employee count compared to the previous quarter. It also provides a button for viewing more detailed employee information. The "`display_churn_metrics()`" function displays the churn rate for the current quarter and the

percentage change compared to the previous quarter. It also offers a button to view more detailed churn data.

```

def display_employee_details(df_deployment, q1, q2):
    """Show detailed employee information"""
    st.subheader(f"Employee Details [ {q2['quarter']} ] {q2['year']} ")
    hire_comparison = ((q2['hires'] - q1['hires']) / q1['hires']) * 100 if q1['hires'] > 0 else 0
    tab1, tab2 = st.tabs(["Hire Comparison", "Department Breakdown"])

    with tab1:
        st.markdown(f"**Employee Hire Comparison (Q{q1['quarter']} vs Q{q2['quarter']}):** {hire_comparison:.1f}%")
        comparison_data = pd.DataFrame({
            'Period': [f"Q{q1['quarter']} {q1['year']}", f"Q{q2['quarter']} {q2['year']}"],
            'Hires': [q1['hires'], q2['hires']]
        })

        fig = px.bar(
            comparison_data,
            x='Period',
            y='Hires',
            text='Hires',
            color='Period',
            color_discrete_sequence=[ '#1f77b4', '#ff7f0e' ]
        )

        fig.update_traces(
            textposition='outside',
            textfont_size=14,
            marker_line_width=0
        )

        fig.update_layout(
            showLegend=False,
            xaxis_title=None,
            yaxis_title='Hires',
            plot_bgcolor='rgba(0,0,0,0)',
            paper_bgcolor='rgba(0,0,0,0)',
            margin=dict(t=30),
            uniformtext_minsize=8,
            uniformtext_mode='hide',
            yaxis=dict(
                gridcolor='rgba(211,211,211,0.3)'
            )
        )
        st.plotly_chart(fig, use_container_width=True)

    with tab2:
        st.markdown("#### Current Employees per Department")
        current_emp = (
            df_deployment[df_deployment['Resigned'] == False]
            .groupby('Department')[['Employee_ID']]
            .count()
            .reset_index(name='Current Employees')
        )
        fig_emp = px.bar(
            current_emp.sort_values('Current Employees', ascending=False),
            x='Department',
            y='Current Employees',
            text='Current Employees'
        )
        fig_emp.update_traces(marker_color= '#1f77b4', textposition='outside')
        fig_emp.update_layout(xaxis_title='', yaxis_title='Employees', showlegend=False)
        st.plotly_chart(fig_emp, use_container_width=True)

```

*Figure 188: Source Code of Display Employee Details for HR and Talent*

The "**display\_employee\_details()**" function presents detailed information about employee hires, comparing the number of hires between two quarters. It includes two tabs, which is one for the Hire Comparison between two quarters to display a bar chart of hires with data labels and another

for the Department Breakdown to show the number of current employees per department using a bar chart.

```

def display_churn_details(df_deployment, current_year, current_quarter):
    """Show detailed churn information"""
    st.subheader("Churn Details")

    tab1, tab2 = st.tabs(["Churn Rate Trend", "Resignations by Department"])

    with tab1:
        completed_quarters = []
        for i in range(6):
            q = current_quarter - 1 - i
            year = current_year
            if q <= 0:
                q += 4
                year -= 1
            completed_quarters.append({'year': year, 'quarter': q})

        quarter_metrics = []
        all_data = get_sorted_quarters_for_trends(df_deployment, current_year, current_quarter)
        for q in completed_quarters:
            match = next((item for item in all_data
                         if item['year'] == q['year'] and item['quarter'] == q['quarter']), None)
            if match:
                quarter_metrics.append(match)

        if quarter_metrics:
            trend_df = pd.DataFrame({
                'Quarter': [f'{q["year"]}{q["quarter"]}' for q in quarter_metrics],
                'Date': [pd.to_datetime(f'{q["year"]}-{q["quarter"]*3}-01') for q in quarter_metrics],
                'Churn Rate': [q['churn'] for q in quarter_metrics]
            }).sort_values('Date')

            fig = px.line(
                trend_df,
                x='Quarter',
                y='Churn Rate',
                range_y=[0, 5],
                markers=True,
                line_shape='linear',
                title='Churn Rate Trend'
            )
            fig.update_traces(
                line=dict(color='red', width=2),
                marker=dict(color='red', size=8, line=dict(color='white', width=1))
            )
            fig.update_layout(
                plot_bgcolor='rgba(0,0,0,0)',
                paper_bgcolor='rgba(0,0,0,0)',
                yaxis=dict(gridcolor='rgba(255,255,255,0.2)'),
                xaxis=dict(tickangle=45),
                font=dict(color='white')
            )
            fig.add_trace(go.Scatter(
                x=trend_df['Quarter'],
                y=trend_df['Churn Rate'],
                fill='tozeroy',
                fillcolor='rgba(30,136,229,0.2)',
                line=dict(width=0),
                showlegend=False
            ))
            st.plotly_chart(fig, use_container_width=True)

```

*Figure 189: Source Code of Display Churn Details for HR and Talent 1*

```

with tab2:
    today = pd.to_datetime("today")
    current_year = today.year
    current_quarter = (today.month - 1) // 3 + 1

    if current_quarter == 1:
        last_qtr = 4
        last_year = current_year - 1
    else:
        last_qtr = current_quarter - 1
        last_year = current_year

    q_start = pd.to_datetime(f"{last_year}-{(last_qtr-1)*3 + 1}-01")
    q_end = pd.to_datetime(f"{last_year}-{(last_qtr*3)}-01") + pd.offsets.MonthEnd[1]

    resigned_qtr = df_deployment[
        (df_deployment['Resigned'] == True) &
        (pd.to_datetime(df_deployment['Resignation_Date'], errors='coerce') >= q_start) &
        (pd.to_datetime(df_deployment['Resignation_Date'], errors='coerce') <= q_end)
    ]

    resigned_df = (resigned_qtr
                    .groupby('Department')['Employee_ID']
                    .count()
                    .reset_index(name='Resigned Count'))

    st.markdown(f"#### Resigned Employees per Department [ Q{last_qtr} {last_year} ]")

    fig_resign = px.bar(
        resigned_df.sort_values('Resigned Count', ascending=False),
        x='Department',
        y='Resigned Count',
        text='Resigned Count',
        title='Resignations by Department'
    )
    fig_resign.update_traces(marker_color="#1f77b4", textposition='outside')
    fig_resign.update_layout(xaxis_title='', yaxis_title='Resigned', showlegend=False)
    st.plotly_chart(fig_resign, use_container_width=True)

```

*Figure 190: Source Code of Display Churn Details for HR and Talent 2*

The "display\_churn\_details()" function provides detailed churn data in two tabs which for the first tab shows the Churn Rate Trend over the last 6 quarters using a line graph and the second tab displays Resignations by Department to show the number of resigned employees per department for the most recent quarter. This function helps track trends and provides deeper insights into employee turnover across departments.

Together with these functions enable the system to present a variety of employee metrics and churn-related insights in an interactive and visually engaging way.

### 5.4.3.1.2.2 HR Role-Specific Metrics

```

def display_satisfaction_trend(
    df_deployment: pd.DataFrame,
    current_year: int,
    current_quarter: int
) -> None:
    """Display satisfaction trend visualization
    Args:
        df_deployment: DataFrame containing employee deployment data
        current_year: The current year as integer
        current_quarter: The current quarter (1-4)
    """
    st.subheader("Satisfaction Trend")

    # Get quarter data for trend analysis
    desired_quarters = []
    latest_complete_quarter = 2
    latest_complete_year = 2025
    for i in range(6): # Last 6 quarters
        quarter_offset = i
        year = latest_complete_year - (quarter_offset // 4)
        quarter = latest_complete_quarter - (quarter_offset % 4)
        if quarter <= 0:
            quarter += 4
            year -= 1
        desired_quarters.append({'year': year, 'quarter': quarter})

    # Get matching quarter metrics
    quarter_metrics = []
    all_quarters = get_sorted_quarters_for_trends(df_deployment, current_year, current_quarter)
    for q in desired_quarters:
        matching_quarter = next(
            (item for item in all_quarters
             if item['year'] == q['year'] and item['quarter'] == q['quarter']),
            None
        )
        if matching_quarter:
            quarter_metrics.append(matching_quarter)

    if not quarter_metrics:
        st.warning("No satisfaction data available for the specified quarters")
    return

```

Figure 191: Source Code of Display Satisfaction Trend Metrics for HR 1

```

# Prepare trend data
quarter_metrics = sorted(quarter_metrics, key=lambda x: (x['year'], x['quarter']))
trend_df = pd.DataFrame({
    'Quarter': [f"(q['year'])'0(q['quarter'])" for q in quarter_metrics],
    'Satisfaction': [q['satisfaction'] for q in quarter_metrics],
    'Date': [q['end'] for q in quarter_metrics]
}).sort_values('Date')

# Calculate average
avg_satisfaction = trend_df['Satisfaction'].mean()

# Create visualization
fig = go.Figure()
fig.add_trace(go.Scatter(
    x=trend_df['Quarter'],
    y=trend_df['Satisfaction'],
    mode='lines+markers',
    name='Satisfaction',
    line=dict(color='skyblue', width=2)
))
fig.add_trace(go.Scatter(
    x=trend_df['Quarter'],
    y=[avg_satisfaction] * len(trend_df),
    mode='lines',
    name='Average',
    line=dict(color='red', width=2, dash='dot')
))
fig.update_layout(
    title='Satisfaction Trend',
    xaxis_title='Quarter',
    yaxis_title='Satisfaction Score',
    height=400
)
st.plotly_chart(fig, use_container_width=True)

# Show latest value
latest = trend_df.iloc[-1]
st.markdown(f"**Latest:** {latest['Satisfaction']:.1f} in {latest['Quarter']}")

```

Figure 192: Source Code of Display Satisfaction Trend Metrics for HR 2

The "`display_satisfaction_trend()`" function is designed specifically for HR to visualize employee satisfaction trends over the last six quarters. It calculates the satisfaction scores for each quarter, compares them to the average score and displays the results in a Plotly line chart. The chart shows the satisfaction trend with a red dashed line representing the average score which will be shown in the [section 5.4.3.2.1.2 “View Satisfaction Trends”](#) later. The most recent quarter's satisfaction score is also displayed. This function helps HR monitor changes in employee satisfaction over time.

#### 5.4.3.1.3 Login Page

```
# Login Page
def login_page():
    st.title("Employee Churn Analytics Dashboard Login")
    st.write("Please enter your credentials to access the HR analytics dashboard")

    with st.form("login_form"):
        username = st.text_input("Username", key="username")
        password = st.text_input("Password", type="password", key="password")
        submitted = st.form_submit_button("Login")

        if submitted:
            st.write(f"Entered username: '{username}', password: '{password}'")

            username = username.strip()
            password = password.strip()

            # Check credentials
            if username == "hr" and password == "ilovetowork":
                st.session_state.logged_in = True
                st.session_state.role = "HR"
                st.session_state.current_page = "HR Dashboard"
                st.session_state.last_login = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                st.session_state.hr_data = load_data()
                st.success("HR login successful! Loading data...")
                st.rerun()

            elif username == "talent" and password == "ilovetoworkalso":
                st.session_state.logged_in = True
                st.session_state.role = "Talent"
                st.session_state.current_page = "Talent Insights"
                st.session_state.last_login = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
                st.session_state.hr_data = load_data()
                st.success("Talent Acquisition login successful! Loading data...")
                st.rerun()

            else:
                st.error("Invalid username or password")
```

*Figure 193: Source Code of Login Page*

The "`login_page()`" function is responsible for handling the user login process in the Employee Churn Analytics Dashboard. When a user visits the login page, they are prompted to enter their username and password. Based on the entered credentials, the system checks if the username and password match the predefined values. If the username and password are correct, the user is directed to their respective role-specific page.

**Employee Churn Analytics Dashboard Login**

Please enter your credentials to access the HR analytics dashboard

Username  
hr

Password  
ilovetowork

**Login**

Figure 194: Interface of Login Page for HR

**Employee Churn Analytics Dashboard Login**

Please enter your credentials to access the HR analytics dashboard

Username  
talent

Password  
ilovetoworkalso

**Login**

Figure 195: Interface of Login Page for Talent

**Employee Churn Analytics Dashboard Login**

Please enter your credentials to access the HR analytics dashboard

Username  
hr

Password  
ilovetowoo

**Login**

Entered username: 'hr', password: 'ilovetowoo'

Invalid username or password

Figure 196: Interface of Login Page for Invalid Username or Password

The figures above show the login page interface. If the username is "hr" and the password is correct, the user is granted access as an HR user and is taken to the HR Dashboard. Similarly, if the username is "talent" and the password matches, the user is granted access as a Talent user and is

directed to the Talent Insights page. If the credentials are incorrect, an error message is displayed. Additionally, the system updates session variables such as login status, role, and the last login time and loads the relevant data for each role.

#### 5.4.3.1.4 Navigation Bar

```
# Navigation bar
def create_navbar():
    st.sidebar.title("Navigation")
    st.sidebar.markdown(f"**Role**: {st.session_state.role}")

    # Common pages for all roles
    pages = {
        "Predict Employee Churn": "Predict Employee Churn"
    }

    # HR-specific pages
    if st.session_state.role == "HR":
        pages.update({
            "HR Dashboard": "HR Dashboard",
            "Employee Performance": "Overview of Employee Performance",
            "Actionable Insights": "Actionable Insights for Retention and Productivity"
        })

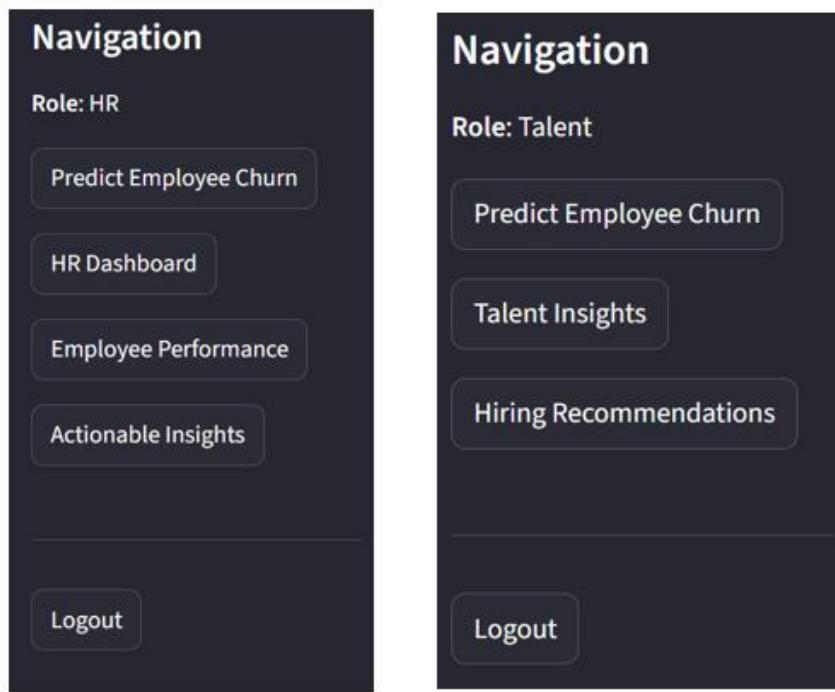
    # Talent-specific pages
    elif st.session_state.role == "Talent":
        pages.update({
            "Talent Insights": "Talent Insights",
            "Hiring Recommendations": "Hiring Recommendations"
        })

    # Navigation buttons
    for page_name, page_key in pages.items():
        if st.sidebar.button(page_name, key=f"nav_{page_key}"):
            st.session_state.current_page = page_key
            st.rerun()

    st.sidebar.markdown("---")
    if st.sidebar.button("Logout", key="logout_btn"):
        st.session_state.logged_in = False
        st.session_state.current_page = "Login"
        st.session_state.role = None
        st.rerun()
```

*Figure 197: Source Code of Navigation Bar*

The "`"create-navbar()`" function is responsible for generating the navigation sidebar in the Employee Churn Analytics Dashboard for both HR and Talent users. The sidebar is updated based on the user's role which provides access to relevant pages for each role.



*Figure 198: Interface of Navigation Bar for HR and Talent*

The figures above are the interface for the navigation bar for the HR and the Talent. First, it displays the user's role at the top of the sidebar. Then, it defines a set of pages based on the user's role. The "Predict Employee Churn" page is accessible to all users regardless of their role. This page provides insights into predicting employee churn which offers valuable information for both HR and Talent users to identify potential risks of employee turnover.

For HR users, additional pages such as HR Dashboard, Employee Performance and Actionable Insights for Retention and Productivity are added to the navigation. For Talent users, the sidebar includes pages like Talent Insights and Hiring Recommendations. Each page is displayed as a button in the sidebar and when a user clicks on a button, the `current_page` session variable is updated and the page is refreshed to display the corresponding content. The sidebar also includes a “logout” button for the user to log out after using.

### 5.4.3.1.5 Employee Churn Prediction Page

```
def churn_prediction():
    st.title("Employee Churn Prediction")
    st.write("""This tool helps predict whether an employee will leave the company.""")

    # Expandable section for variable explanations
    with st.expander("Variable Explanations"):
        st.markdown("""
            *Performance Score*: Employee's performance rating (1-5 scale)
            *Satisfaction Score*: Employee's satisfaction with their job (1-5 scale)
            *Monthly Salary*: Employee's monthly pay
            *Work Hours*: Typical hours worked per week
            *Overtime Hours*: Extra hours worked beyond the normal schedule per year
            *Promotions*: Number of promotions received per year
            *Remote Work Level*: The proportion of time spent working remotely (None/Low/High)
            *Job Title*: Employee's role
            *Health Condition Index*: Employee's health status score (1-10 scale)
        """)
```

Figure 199: Source Code of Employee Churn Prediction 1

```
if st.session_state.hr_data is None:
    st.error("HR data not loaded. Please login again.")
    return

with st.form("churn_prediction_form"):
    col1, col2 = st.columns(2)

    with col1:
        performance_score = st.slider(
            "Performance Score (1-5)", 1, 5, 3,
            help="Employee's performance rating (1-5 scale)"
        )
        employee_satisfaction = st.slider(
            "Satisfaction Score (1-5)", 1, 5, 3,
            help="Employee's satisfaction with their job (1-5 scale)"
        )
        monthly_salary = st.number_input(
            "Monthly Salary ($)", min_value=1000, max_value=20000, value=5000,
            help="Employee's monthly pay in dollars"
        )
        work_hours_per_week = st.slider(
            "Work Hours/Week", 10, 80, 40,
            help="Typical hours worked per week"
        )

    with col2:
        overtime_hours = st.slider(
            "Overtime Hours/Year", 0, 60, 30,
            help="Extra hours worked beyond the normal schedule per year"
        )
        promotions = st.slider(
            "Promotions/Year", 0, 5, 3,
            help="Number of promotions received per year"
        )
        remote_work_level = st.selectbox(
            "Remote Work Level", ["None", "Low", "High"],
            help="Time spent working remotely (None/Low/High) "
        )
        job_title = st.selectbox(
            "Job Title", ["Manager", "Technician", "Other"],
            help="Employee's role in the company"
        )
        health_index = st.slider(
            "Health Condition Index (1-10)", 1, 10, 5,
            help="Employee's health status score (1-10 scale)"
        )

    submitted = st.form_submit_button("Predict Churn Risk")
```

Figure 200: Source Code of Employee Churn Prediction 2

```

if submitted:
    try:
        model = joblib.load('employee_churn_model.pkl')

        input_data = pd.DataFrame([
            if remote_work_level == "Low" else 2 if remote_work_level == "High" else 0,
            promotions,
            1 if job_title == "Technician" else 0,
            1 if job_title == "Manager" else 0,
            monthly_salary,
            work_hours_per_week,
            overtime_hours,
            employee_satisfaction,
            health_index],
            columns=['Performance_Score', 'Remote_Work_Level', 'Promotions',
                     'Job_Title_Technician', 'Job_Title_Manager', 'Monthly_Salary',
                     'Work_Hours_Per_Week', 'Overtime_Hours',
                     'Employee_Satisfaction_Score', 'Health_Index'])

        # Make prediction
        prediction = model.predict(input_data)
        probabilities = model.predict_proba(input_data)
        prob_churn = probabilities[:,1]

        st.subheader("Prediction Results")

        # Recommendations based on churn risk
        if prob_churn > 0.7:
            st.error("High Risk of Churn: (prob_churn:.1%)")
            st.write("This employee has a high probability of leaving. Immediate action recommended to retain this employee.")
            st.subheader("Recommended Actions:")
            st.write("""
                - Schedule 1:1 meeting to discuss concerns.
                - Identify key issues affecting satisfaction.
                - Review compensation and benefits.
                - Consider career development opportunities.
                - Assess and adjust workload balance if needed.
            """)

        elif prob_churn > 0.4:
            st.warning("Moderate Risk of Churn: (prob_churn:.1%)")
            st.write("This employee shows some risk factors. Consider taking proactive measures to overcome potential concerns.")
            st.subheader("Recommended Actions:")
            st.write("""
                - Check-in with the employee to understand their concerns.
                - Provide feedback channels for open discussion.
                - Monitor performance and satisfaction over time.
                - Consider additional training or support.
            """)

        else:
            st.success("Low Risk of Churn: (prob_churn:.1%)")
            st.write("This employee appears stable. Continue regular engagement.")
            st.subheader("Recommended Actions:")
            st.write("""
                - Continue regular check-ins to maintain engagement.
                - Ensure consistent career development opportunities.
                - Watch for any signs of dissatisfaction.
            """)

    except:
        st.error("An error occurred while processing the data.")


# 2. Resignations (Top 5 most recent)
recent_resignations = df[df['Resigned'] == True].sort_values(by='Resignation_Date', ascending=False).head(5)
for _, row in recent_resignations.iterrows():
    satisfaction = row['Employee_Satisfaction_Score']
    performance = row['Performance_Score']
    years = row['Years_At_Company']

    # Refined logic for resignations
    if performance >= 4 or years >= 5:
        priority = "High"
    elif performance == 3:
        priority = "Medium"
    else:
        priority = "Low"

    activity_data.append({
        "Employee ID": str(row['Employee_ID']),
        "Date": pd.to_datetime(row['Resignation_Date']).date(),
        "Job Title": row['Job_Title'],
        "Activity Status": "Resignation",
        "Performance Score": performance,
        "Employee Satisfaction": f'{satisfaction:.2f}',
        "Years At Company": years,
        "Priority": priority
    })

# Display section
st.subheader("Recent Activity")

if activity_data:
    activity_df = pd.DataFrame(activity_data)

    # Priority sorting order
    priority_order = pd.CategoricalDtype(['High', 'Medium', 'Low'], ordered=True)
    activity_df['Priority'] = activity_df['Priority'].astype(priority_order)

    # Sort
    activity_df = activity_df.sort_values(['Date', 'Priority'], ascending=[False, True])
    activity_df.index = np.arange(1, len(activity_df) + 1)

    color_map = {
        'High': 'background-color: rgba(255, 0, 0, 0.15); color: white;',      # very light red
        'Medium': 'background-color: rgba(255, 200, 0, 0.15); color: white;',    # very light yellow
        'Low': 'background-color: rgba(0, 200, 0, 0.15); color: white;'       # very light green
    }

    def highlight_priority(val):
        return color_map.get(val, '')

    styled_df = activity_df.style.applymap(highlight_priority, subset=['Priority'])

    st.dataframe(styled_df, use_container_width=True)

else:
    st.write("No recent activity found.")

```

Figure 201: Source Code of Employee Churn Prediction 3

```

# 2. Resignations (Top 5 most recent)
recent_resignations = df[df['Resigned'] == True].sort_values(by='Resignation_Date', ascending=False).head(5)
for _, row in recent_resignations.iterrows():
    satisfaction = row['Employee_Satisfaction_Score']
    performance = row['Performance_Score']
    years = row['Years_At_Company']

    # Refined logic for resignations
    if performance >= 4 or years >= 5:
        priority = "High"
    elif performance == 3:
        priority = "Medium"
    else:
        priority = "Low"

    activity_data.append({
        "Employee ID": str(row['Employee_ID']),
        "Date": pd.to_datetime(row['Resignation_Date']).date(),
        "Job Title": row['Job_Title'],
        "Activity Status": "Resignation",
        "Performance Score": performance,
        "Employee Satisfaction": f'{satisfaction:.2f}',
        "Years At Company": years,
        "Priority": priority
    })

# Display section
st.subheader("Recent Activity")

if activity_data:
    activity_df = pd.DataFrame(activity_data)

    # Priority sorting order
    priority_order = pd.CategoricalDtype(['High', 'Medium', 'Low'], ordered=True)
    activity_df['Priority'] = activity_df['Priority'].astype(priority_order)

    # Sort
    activity_df = activity_df.sort_values(['Date', 'Priority'], ascending=[False, True])
    activity_df.index = np.arange(1, len(activity_df) + 1)

    color_map = {
        'High': 'background-color: rgba(255, 0, 0, 0.15); color: white;',      # very light red
        'Medium': 'background-color: rgba(255, 200, 0, 0.15); color: white;',    # very light yellow
        'Low': 'background-color: rgba(0, 200, 0, 0.15); color: white;'       # very light green
    }

    def highlight_priority(val):
        return color_map.get(val, '')

    styled_df = activity_df.style.applymap(highlight_priority, subset=['Priority'])

    st.dataframe(styled_df, use_container_width=True)

else:
    st.write("No recent activity found.")

```

Figure 202: Source Code of Employee Churn Prediction 4

```

# SHAP (Feature Importance)
try:
    explainer = shap.TreeExplainer(model)
    shap_values = explainer.shap_values(input_data)

    feature_contributions = pd.DataFrame(shap_values[0], index=input_data.columns, columns=["SHAP Value"])
    feature_contributions['Abs_SHAP Value'] = feature_contributions['SHAP Value'].abs()
    feature_contributions = feature_contributions.sort_values(by='Abs_SHAP Value', ascending=False)

    # Only select positive SHAP values (which indicate positive contribution to churn risk)
    feature_contributions_positive = feature_contributions[feature_contributions['SHAP Value'] > 0]

    fig = px.bar(
        feature_contributions_positive.head(7),
        x="SHAP Value",
        y=feature_contributions_positive.head(7).index,
        title="Top Influencing Features Based on SHAP Values",
        text=feature_contributions_positive.head(7)[['Abs_SHAP Value']].round(1).astype(str) + '%',
        color='SHAP Value',
        color_continuous_scale='Blues',
        orientation='h'
    )

    fig.update_traces(
        hovertemplate='%{y}  
SHAP Value: %{x:.2f}  
Percentage: %{text}%',
    )

    fig.update_layout(
        title={'font': {'size': 25, 'color': 'white'}, 'x': 0.5, 'xanchor': 'center'},
        xaxis_title='SHAP Value',
        yaxis_title='Features',
        font=dict(color='white'),
        plot_bgcolor='rgba(0,0,0,0)',
        paper_bgcolor='rgba(0,0,0,0)'
    )

    st.plotly_chart(fig, use_container_width=True)

except Exception as e:
    st.error(f"Error with SHAP explanation: {str(e)}")
    st.write("Falling back to Feature Importance visualization...")

# Feature importance fallback (if SHAP fails)
st.markdown("----")
st.subheader("Key Factors Influencing This Prediction")
feature_importance = pd.Series(model.feature_importances_, index=input_data.columns)
top_features = feature_importance.sort_values(ascending=False).head(7)

# Calculate the percentage for each feature
total_importance = top_features.sum()
top_features_percentage = (top_features / total_importance) * 100

```

Figure 203: Source Code of Employee Churn Prediction 5

```

# Create the horizontal bar chart
fig = px.bar(
    x=top_features.values,
    y=top_features.index,
    labels={'x': 'Importance Score', 'y': 'Features'},
    title="Top Influencing Factors",
    text=top_features_percentage.round(1).astype(str) + '%',
    color=top_features.values,
    color_continuous_scale='Blues',
    orientation='h'
)

fig.update_traces(
    hovertemplate='%{y}  
Importance: %{x:.2f}  
Percentage: %{text}%',
)

fig.update_layout(
    title={'font': {'size': 25, 'color': 'white'}, 'x': 0.5, 'xanchor': 'center'},
    xaxis_title='Importance Score',
    yaxis_title='Features',
    font=dict(color='white'),
    plot_bgcolor='rgba(0,0,0,0)',
    paper_bgcolor='rgba(0,0,0,0)'
)

# Show the plot
st.plotly_chart(fig, use_container_width=True)

except Exception as e:
    st.error(f"Error loading model: {str(e)}")
    st.write("Please try again or contact support")

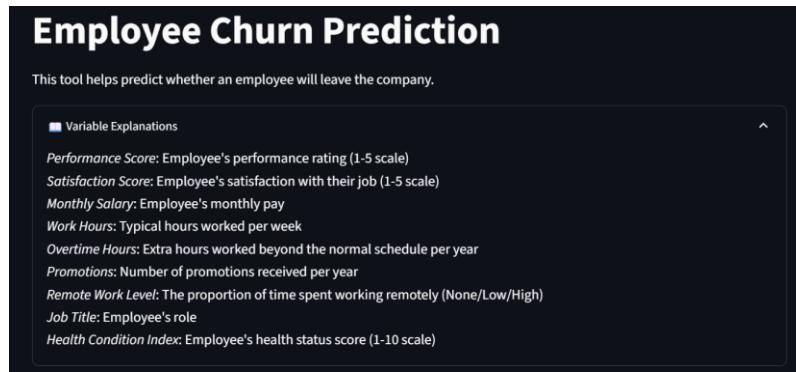
```

Figure 204: Source Code of Employee Churn Prediction 7

The "**churn\_prediction()**" function is designed to predict the employee churn based on several key employee attributes such as performance score, satisfaction score, salary, work hours, overtime, promotions, remote work level, job title and health condition index. After the user provides input for these variables, the function uses a pre-trained machine learning model to predict the probability of the employee leaving the company. Based on this prediction, the system classifies the churn risk as high, moderate or low and offers the corresponding recommendations for HR to take appropriate action.

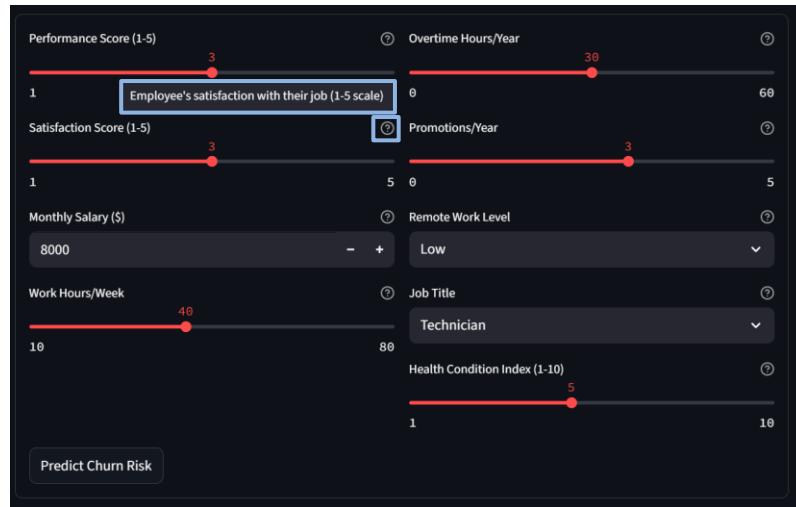
To enhance the interpretability of the model's prediction, the function employs SHAP (SHapley Additive exPlanations) values. SHAP values are derived from cooperative game theory and explain the contribution of each feature to the model's output. They quantify on how much each individual feature impacts the churn prediction by assigning a value to each feature and indicating its contribution to the outcome. A positive SHAP value implies the feature increases the churn, while a negative SHAP value suggests it decreases. These SHAP values are visualized in a bar chart to highlight the most influential features for making it easier for HR to understand which factors are driving the prediction.

In cases where SHAP values cannot be computed, the function defaults to displaying feature importance, which provides an alternative way of understanding the relative impact of each feature on the model's decision. This fallback ensures that even without SHAP explanations, HR can still access the key factors influencing the churn prediction. Overall, the "**churn\_prediction()**" function provides HR with both predictive insights and a clear understanding of the factors influencing employee churn to enable a more informed and proactive retention strategies.



*Figure 205: Interface for Employee Churn Prediction 1*

The system provides a variable explanations section to help users understand the input variables required for the churn prediction model. As shown in Figure above, each variable is clearly defined to ensure that users know what each input represents. For example, the Performance Score represents the employee's rating on a 1-5 scale. These explanations are accessible through an expandable section and are designed to provide clarity and assist users in accurately entering the required data.



*Figure 206: Interface for Employee Churn Prediction 2*

The system also provides tooltips for each input field through the help argument to offer a brief description of each variable to guide users in providing the correct data. These tooltips ensure that users understand what each input represents to improve the overall user experience and help users enter accurate and consistent data.

Below will show **three examples** of the churn risk for **low, moderate and high**.

#### 5.4.3.1.5.1 Low Risk of Churn

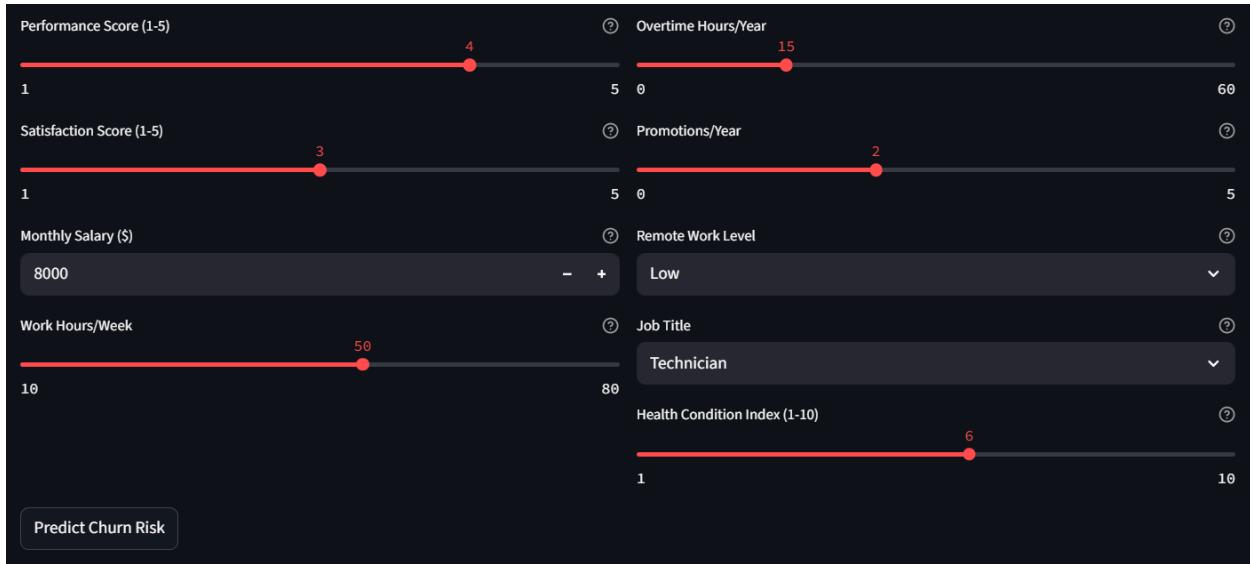
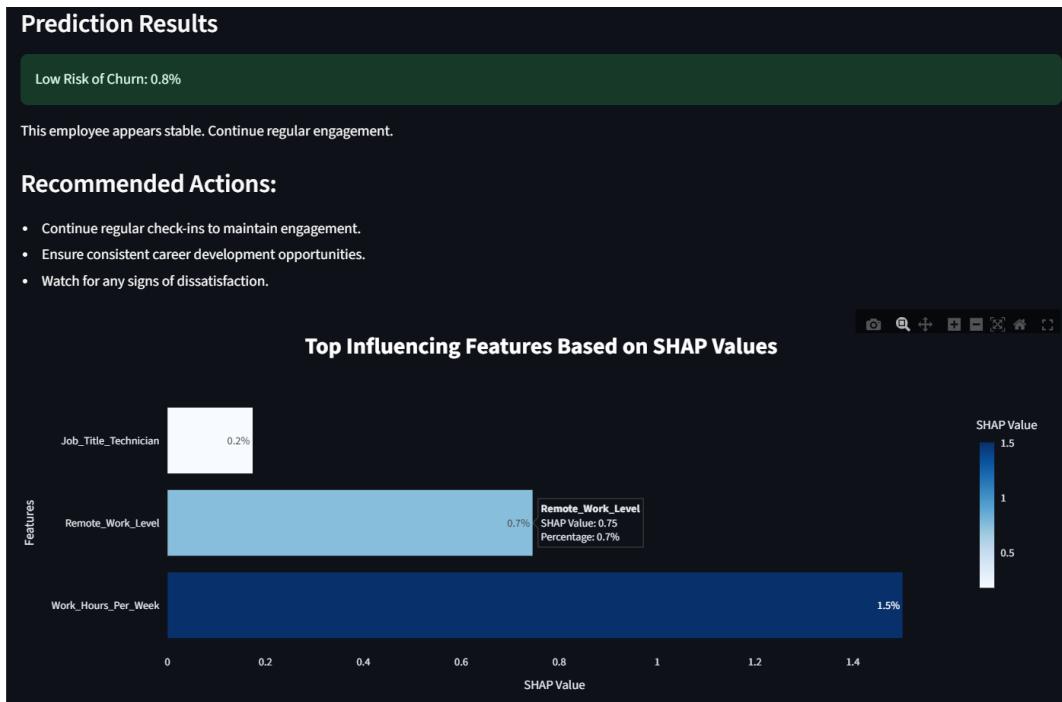


Figure 207: Interface for Predicting Low Risk of Churn

The figure displays the input data interface of the employee churn prediction interface which focuses on the Low Risk of Churn scenario. The interface allows HR or Talent personnel to enter various employee attributes. For example, HR has provided values for Performance Score (4), Satisfaction Score (3), Monthly Salary (8,000 USD), Work Hours per Week (50 hours), Overtime Hours per Year (15 hours), Promotions per Year (2), Remote Work Level (Low), Job Title (Technician) and Health Condition Index (6). These inputs are then processed by the machine learning model to predict the likelihood of the employee leaving the company.



*Figure 208: Interface for Prediction Result of Predicting Low Risk of Churn*

The figure above shows the prediction result along with the recommended actions. The prediction indicates a Low Risk of Churn at 0.8% with green shaded background colour, meaning the employee has a very low probability of leaving the company based on the provided data. The system also suggests actionable recommendations for HR such as continuing regular check-ins, ensuring consistent career development opportunities and monitoring for any signs of dissatisfaction.

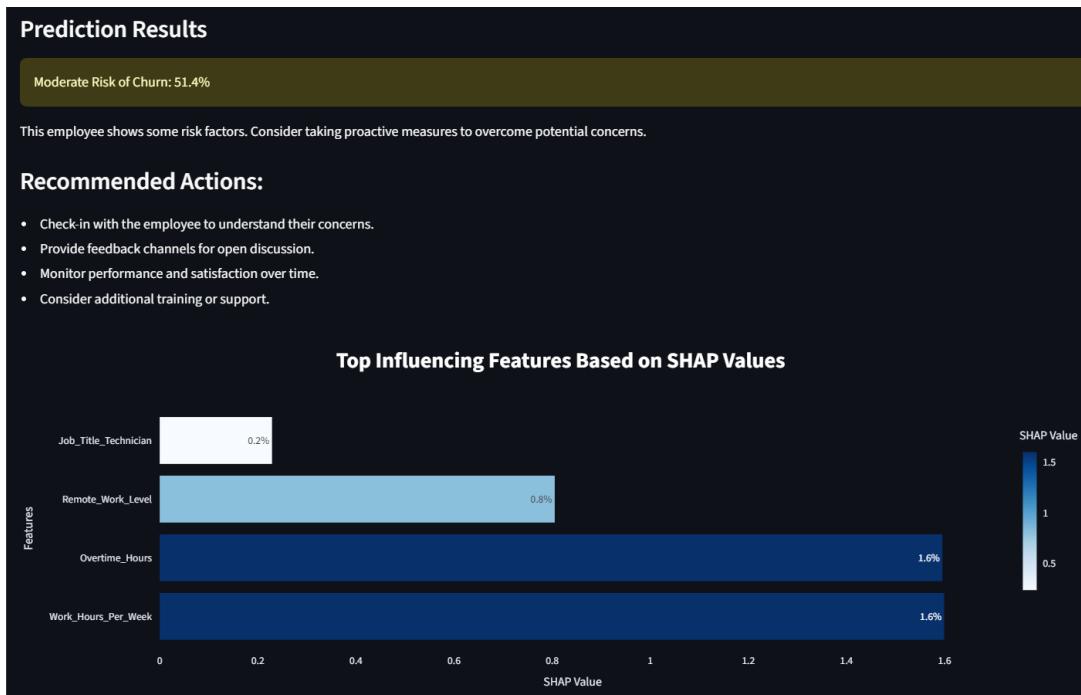
Below the prediction results, the bar chart illustrates the top influencing features based on SHAP values which have highlighted the most significant factors contributing to the churn prediction. “Work \_Hours\_Per\_Week” has the largest impact on the prediction and is contributing 1.5% to the decision. The other notable features include “Remote\_Work\_Level” and “Job\_Title”, each with their corresponding SHAP values, which indicate their role in the model's decision-making process. When the mouse pointer hovers over the bar chart, it displays additional details such as the SHAP value and the percentage contribution of each feature to provide a more detailed insight into the factors influencing the churn prediction. This visualization enables HR to better understand the key factors influencing churn risk and provides valuable insights for developing targeted retention strategies.

### 5.4.3.1.5.2 Moderate Risk of Churn

The figure displays the input data interface for the employee churn prediction system which focuses on the Moderate Risk of Churn scenario. The form allows HR or Talent personnel to enter various employee attributes such as Performance Score (3), Satisfaction Score (3), Monthly Salary (6,000 USD), Work Hours per Week (50 hours), Overtime Hours per Year (20 hours), Promotions per Year (1), Remote Work Level (Low), Job Title (Technician) and Health Condition Index (5). These inputs are processed by the machine learning model to predict the likelihood of the employee leaving the company.

*Figure 209: Interface for Predicting Moderate Risk of Churn*

The figure displays the input data interface for the employee churn prediction system which focuses on the Moderate Risk of Churn scenario. The form allows HR or Talent personnel to enter various employee attributes such as Performance Score (3), Satisfaction Score (3), Monthly Salary (6,000 USD), Work Hours per Week (50 hours), Overtime Hours per Year (20 hours), Promotions per Year (1), Remote Work Level (Low), Job Title (Technician) and Health Condition Index (5). These inputs are processed by the machine learning model to predict the likelihood of the employee leaving the company.



*Figure 210: Interface for Prediction Result of Predicting Moderate Risk of Churn*

The figure above shows the prediction result along with the recommended actions. The prediction result indicates a Moderate Risk of Churn at 51.4% with yellow background shaded colour, suggesting that the employee has a moderate probability of leaving the company based on the provided data. The system also provides actionable recommendations for HR, including checking in with the employee to understand their concerns, providing feedback channels for open discussions, monitoring performance and satisfaction over time and considering additional training or support.

Below the prediction results, the bar chart illustrates the top influencing features based on SHAP values, which show the most significant factors contributing to the churn prediction. "Work\_Hours\_Per\_Week" and "Overtime\_Hours" each contribute 1.6% to the decision and have made them the most influential factors in the prediction. The other notable features include "Remote\_Work\_Level" is contributing 0.8% and "Job\_Title" with a minimal contribution of 0.2%. When the mouse pointer hovers over the bar chart, it displays additional details such as the SHAP value and the percentage contribution of each feature to provide a more detailed insight into the factors influencing the churn prediction. These SHAP values help HR understand the key factors driving the churn prediction and offer insights into where to focus retention strategies.

### 5.4.3.1.5.3 High Risk of Churn

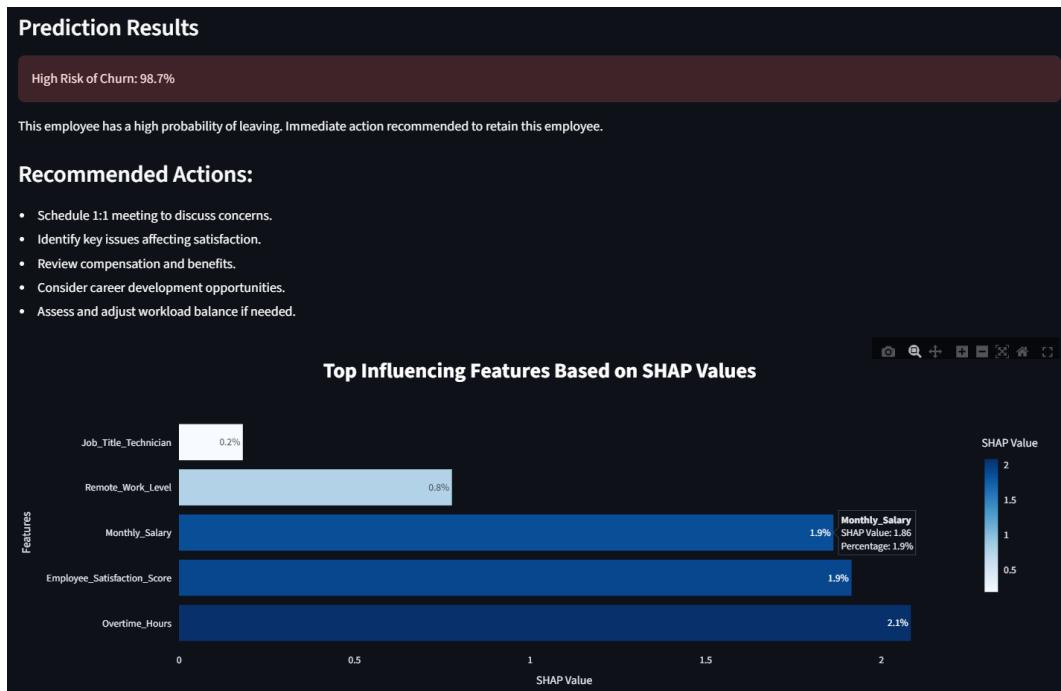
The figure shows a user interface for predicting employee churn risk. It includes the following input fields:

- Performance Score (1-5): Value 2
- Satisfaction Score (1-5): Value 2
- Monthly Salary (\$): Value 5000
- Work Hours/Week: Value 45
- Overtime Hours/Year: Value 20
- Promotions/Year: Value 1
- Remote Work Level: Set to Low
- Job Title: Set to Technician
- Health Condition Index (1-10): Value 3

A "Predict Churn Risk" button is located at the bottom left.

*Figure 211: Interface for Predicting High Risk of Churn*

The figure displays the input data interface for the employee churn prediction system which focuses on the High Risk of Churn scenario. The input form allows HR or Talent personnel to enter various employee attributes such as Performance Score (2), Satisfaction Score (2), Monthly Salary (5,000 USD), Work Hours per Week (45 hours), Overtime Hours per Year (20 hours), Promotions per Year (1), Remote Work Level (Low), Job Title (Technician) and Health Condition Index (3). These inputs are processed by the machine learning model to predict the likelihood of the employee leaving the company.



*Figure 212: Interface for Prediction Result of Predicting High Risk of Churn*

The figure above shows the prediction result along with the recommended actions. The prediction result indicates a High Risk of Churn at 98.7% with red color shaded background, meaning the employee has a very high probability of leaving the company based on the provided data. The system also provides actionable recommendations for HR, including scheduling a 1:1 meeting to discuss concerns, identifying key issues affecting satisfaction, reviewing compensation and benefits, considering career development opportunities and adjusting the workload balance if necessary.

Below the prediction results, the bar chart illustrates the top influencing features based on SHAP values which is highlighting the most significant factors contributing to the churn prediction. “Monthly\_Salary” has the largest impact on the prediction which has contributing 1.9% to the decision. The other notable features include “Employee\_Satisfaction\_Score” (1.9%), “Overtime\_Hours” (2.1%) and “Remote\_Work\_Level” (0.8%) with their corresponding SHAP values indicating their role in the model's decision-making process. When the mouse pointer hovers over the bar chart, it displays additional details such as the SHAP value and the percentage contribution of each feature to provide a more detailed insight into the factors influencing the churn

prediction. This visualization helps HR better understand the key factors influencing churn risk and provides insights for targeted retention strategies.

### 5.4.3.2 HR Specific Functions

#### 5.4.3.2.1 HR Main Dashboard Page

```
def hr_dashboard():
    if st.session_state.role != "HR":
        st.warning("You don't have permission to access this page")
        return

    if st.session_state.hr_data is None:
        st.error("HR data not loaded. Please login again.")
        return

    # Load data
    df_deployment, df_performance = st.session_state.hr_data

    # Get current date info
    today = pd.to_datetime('today')
    current_year = today.year
    current_quarter = (today.month - 1) // 3 + 1

    # Get comparison quarters
    q2_year, q2_quarter = get_previous_quarter(current_year, current_quarter)
    q1_year, q1_quarter = get_previous_quarter(q2_year, q2_quarter)

    # Get metrics
    q1 = get_quarterly_metrics(df_deployment, q1_year, q1_quarter)
    q2 = get_quarterly_metrics(df_deployment, q2_year, q2_quarter)

    # Calculate changes
    satisfaction_change = q2['satisfaction'] - q1['satisfaction']

    # Dashboard Metrics
    col1, col2, col3 = st.columns(3)

    with col1:
        employee_details_button = display_employee_metrics(q1, q2)

    with col2:
        st.metric(
            "Avg. Satisfaction",
            f'{q2["satisfaction"]:.1f}/10',
            delta=f'{satisfaction_change:.1f} from Q{q1["quarter"]}',
            delta_color="normal" if satisfaction_change >= 0 else "inverse"
        )
        satisfaction_details_button = st.button("View Satisfaction Details", key="satisfaction_details")

    with col3:
        churn_details_button = display_churn_metrics(q1, q2)

    st.markdown("---")

    # Conditional displays
    if employee_details_button:
        display_employee_details(df_deployment, q1, q2)

    if satisfaction_details_button:
        display_satisfaction_trend(df_deployment, current_year, current_quarter)

    if churn_details_button:
        display_churn_details(df_deployment, current_year, current_quarter)
```

*Figure 213: Source Code of HR Main Dashboard Page 1*

The figure above shows the source code of HR Main Dashboard Page. The HR Dashboard provides HR professionals with a comprehensive overview of key employee metrics and performance indicators to facilitate informed decision-making. The dashboard includes various sections such as

employee details, satisfaction scores and churn metrics which allows HR to track employee well-being and identify potential retention issues. One of the key features of the HR dashboard is the ability to compare data from previous quarters in real-time based on the current date. This allows HR to observe trends and changes in key metrics such as employee satisfaction, work hours and turnover risk. By using real-time data, HR can track fluctuations and make informed decisions for retention and performance management.

HR can use the dashboard to monitor overall employee satisfaction with visualizations that show satisfaction trends and highlight areas requiring attention. The dashboard also allows HR professionals to drill down into specific details such as churn risk to better understand which factors may be contributing to potential employee turnover. Additionally, it provides a quick snapshot of employee performance and churn metrics to make it easier for HR to take proactive measures and customize strategies for retention. This functionality helps HR teams make data-driven decisions and implement strategies to maintain a healthy, engaged and productive workforce.

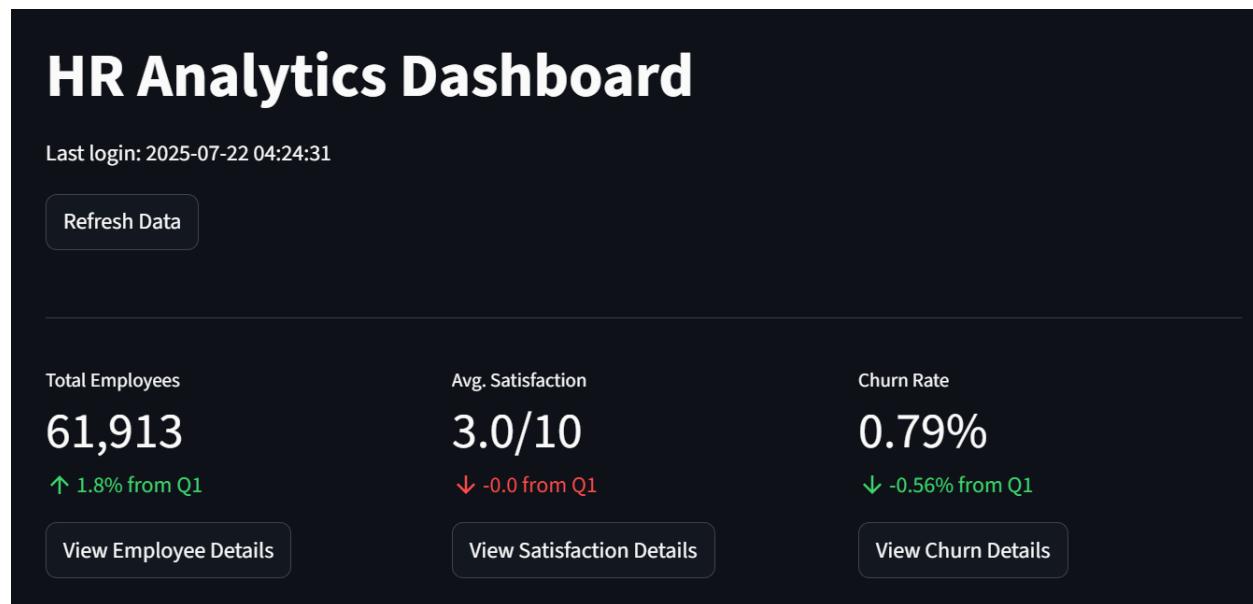


Figure 214: Interface of HR Analytics Dashboard

The HR Analytics Dashboard provides a comprehensive overview of key employee metrics which are displayed on a quarterly basis. The dashboard shows essential data such as Total Employees, Average Satisfaction and Churn Rate with each metric reflecting changes compared to the previous quarter. For example, the Total Employees figure is displayed with the current number of

employees, 61,913 and includes a percentage increase or decrease such as a 1.8% increase from Q1 which is highlighted with color coding for easy visualization. Similarly, the Average Satisfaction score is shown with any increase or decrease from the previous quarter. Since there was no change in the satisfaction score, it is highlighted in red which shows that no improvement has occurred and may indicate a need for attention. On the other hand, the Churn Rate is presented with its changes with a decrease in churn is highlighted in green as this reduction is a positive sign indicating improved employee retention. The dashboard also includes interactive "View Details" buttons for further insights into each metric.

In addition, the **Refresh Data** button ensures that the information displayed is always up to date which offer real-time insights into the workforce's status. It also shows the last login time for HR teams to stay informed of the most recent updates. This functionality enables HR teams to continuously monitor and analyze workforce data to manage and respond to evolving employee needs.

Below shows the visualization diagrams displayed after clicking the 'View Details' for each metric to allow HR for a more detailed analysis.

#### 5.4.3.2.1.1 View Employee Details

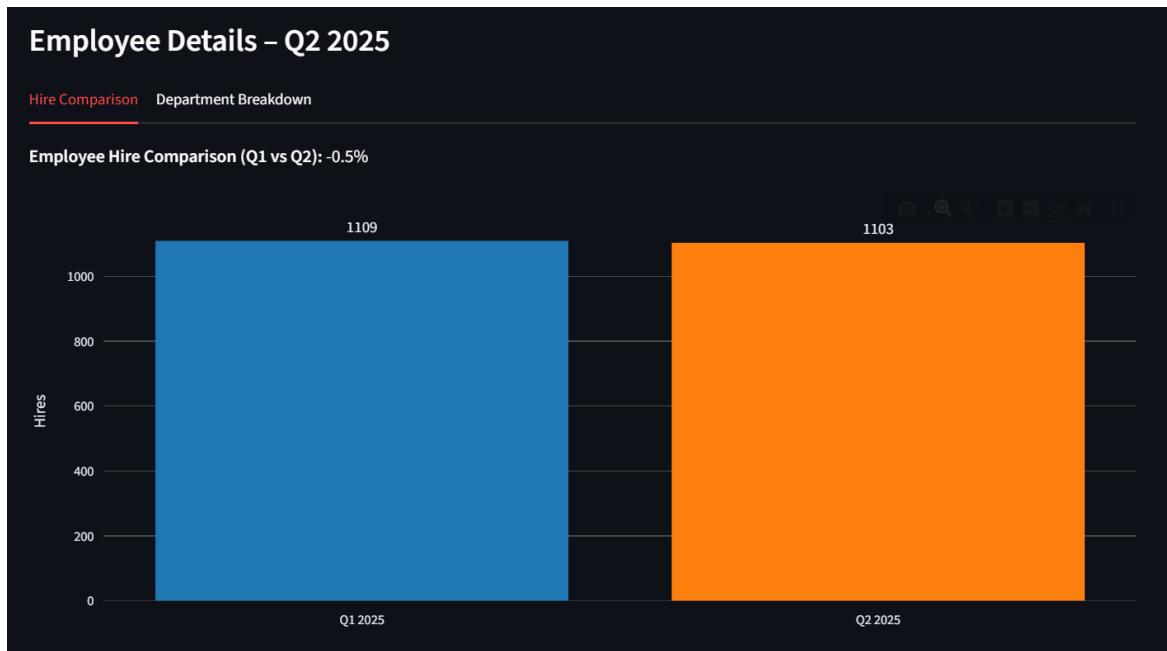


Figure 215: Interface of View Employee Details 1



*Figure 216: Interface of View Employee Details 2*

The figures above show bar charts that provide detailed visualizations for HR personnel. The first chart compares employee hires between different quarters while the second chart displays the distribution of employees across departments.

The "View Employee Details" function allows HR personnel to access deeper insights into the data. By clicking on it, HR can view visualizations like the employee hire comparisons and department breakdowns. These charts highlight key trends such as hiring patterns over time and workforce distribution by departments which are helping HR to make informed decisions based on the data.

#### 5.4.3.2.1.2 View Satisfaction Trends

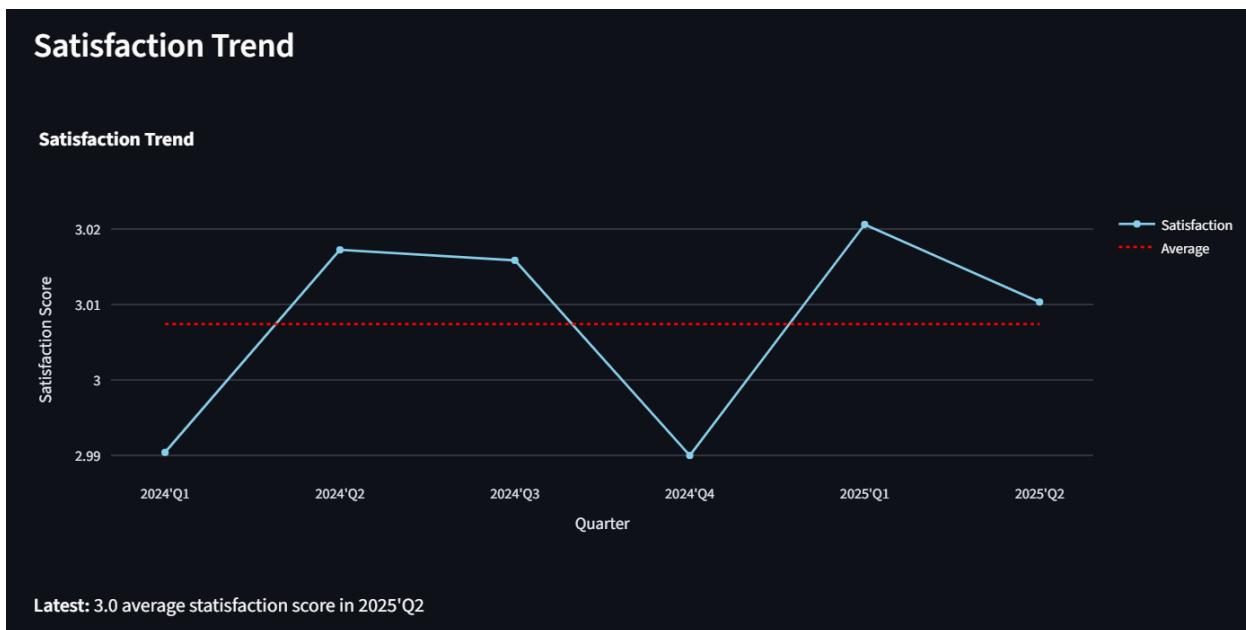


Figure 217: Interface of View Satisfaction Trend

The figure above shows the "Satisfaction Trend" chart which visualizes the employee satisfaction scores over multiple quarters. The chart includes two key components, which is the blue line representing the actual satisfaction score for each quarter and the red dashed line indicating the average satisfaction score.

This visualization allows HR personnel to track fluctuations in employee satisfaction over time and compare it against the average score. It helps identify periods where satisfaction was particularly high or low and assesses whether there are any noticeable trends. For example, in the chart, the satisfaction score peaks in 2024'Q2 and then dips again in 2024'Q3 which indicates that satisfaction has fluctuated during the observed period. This trend analysis helps HR teams to monitor and solve issues related to employee satisfaction and develop strategies for improvement.

Besides, the "Satisfaction Trend" feature also includes a label showing the latest satisfaction score for 2025'Q2 to provide HR with the most current data available.

### 5.4.3.2.1.3 View Employee Churn Details

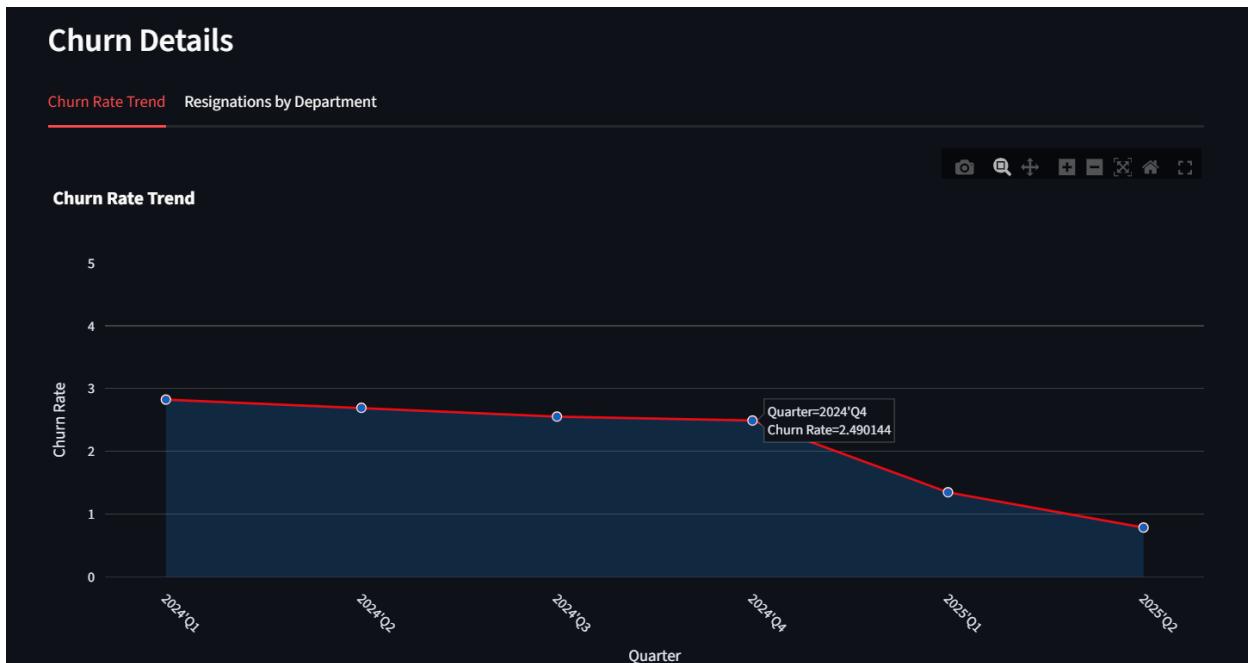


Figure 218: Interface of View Employee Churn Details 1



Figure 219: Interface of View Employee Churn Details 2

The figures above show two different visualizations related to employee churn which is the "Churn Rate Trend" and "Resigned Employees per Department" charts.

The first chart, "Churn Rate Trend," tracks the changes in churn rate over several quarters. It shows a steady decrease in churn overtime, from 2024'Q1 to 2025'Q2. The visualization allows HR to monitor how employee turnover is evolving and whether any interventions have been effective in reducing churn. This trend provides valuable insight into the company's retention efforts and helps HR teams assess the effectiveness of their strategies.

The second chart, "Resigned Employees per Department," provides a detailed breakdown of employee resignations across different departments for 2025'Q2. The chart shows that the Sales department has the highest number of resignations followed closely by Customer Support and Legal. This information allows HR to identify departments that may need more attention or targeted retention efforts. By understanding the resignation trends by department is important for pinpointing specific areas where employee dissatisfaction or turnover may be more prevalent.

Overall, these visualizations enable HR teams to gain deeper insights into churn trends and departmental resignation rates to facilitate more informed decision-making and effective retention strategies.

#### 5.4.3.2.1.4 Quick Actions

```
# Quick actions
st.subheader("Quick Actions")
action_col1, action_col2, action_col3 = st.columns(3)
with action_col1:
    if st.button("Run Churn Prediction"):
        st.session_state.current_page = "Predict Employee Churn"
        st.rerun()
with action_col2:
    if st.button("View Performance"):
        st.session_state.current_page = "Overview of Employee Performance"
        st.rerun()
with action_col3:
    if st.button("Get Insights"):
        st.session_state.current_page = "Actionable Insights for Retention and Productivity"
        st.rerun()

st.markdown("---")
recent_activity()
```

Figure 220: Source Code of HR Main Dashboard Page 2

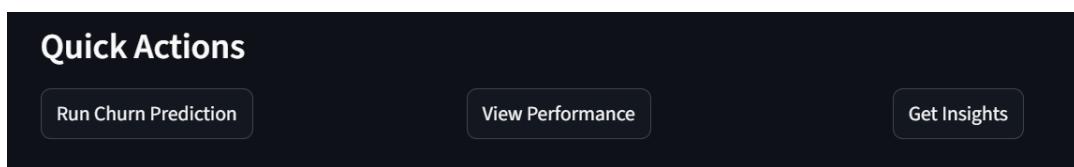


Figure 221: Interface for Quick Actions

The figure above shows the "Quick Actions" section in the HR Analytics Dashboard which allows HR personnel to quickly navigate between key functions. There are three buttons which is "Run Churn Prediction," "View Performance," and "Get Insights" for providing easy access to various predictive and analytical tools. These quick action buttons are designed to enhance navigation and streamline the decision-making process for HR to offer an immediate access to the most critical features of the HR Analytics Dashboard.

#### 5.4.3.2.2 Recent Activity

```

def recent_activity():
    if 'df_deployment' not in st.session_state:
        st.warning("HR data not loaded. Please log in again.")
        return

    df = st.session_state.df_deployment

    required_columns = ['Employee_ID', 'Hire_Date', 'Job_Title',
                        'Employee_Satisfaction_Score', 'Performance_Score',
                        'Years_At_Company', 'Resigned', 'Resignation_Date']
    missing_columns = [col for col in required_columns if col not in df.columns]

    if missing_columns:
        st.warning(f"Missing columns: {', '.join(missing_columns)}")
        return

    activity_data = []

    # 1. New Hires (Top 5 most recent)
    recent_hires = df.sort_values(by='Hire_Date', ascending=False).head(5)
    for _, row in recent_hires.iterrows():
        satisfaction = row['Employee_Satisfaction_Score']
        performance = row['Performance_Score']
        years = row['Years_At_Company']

        # Priority based on satisfaction
        if satisfaction >= 4:
            priority = "Low"
        elif satisfaction == 3:
            priority = "Medium"
        else:
            priority = "High"

        activity_data.append({
            "Employee ID": str(row['Employee_ID']),
            "Date": pd.to_datetime(row['Hire_Date']).date(),
            "Job Title": row['Job_Title'],
            "Activity Status": "New Hire",
            "Performance Score": performance,
            "Employee Satisfaction": f"{satisfaction:.2f}",
            "Years At Company": years,
            "Priority": priority
        })

```

Figure 222: Source Code of Recent Activity 1

```

# 2. Resignations (Top 5 most recent)
recent_resignations = df[df['Resigned'] == True].sort_values(by='Resignation_Date', ascending=False).head(5)
for _, row in recent_resignations.iterrows():
    satisfaction = row['Employee_Satisfaction_Score']
    performance = row['Performance_Score']
    years = row['Years_At_Company']

    # Refined logic for resignations
    if performance >= 4 or years >= 5:
        priority = "High"
    elif performance == 3:
        priority = "Medium"
    else:
        priority = "Low"

    activity_data.append({
        "Employee ID": str(row['Employee_ID']),
        "Date": pd.to_datetime(row['Resignation_Date']).date(),
        "Job Title": row['Job_Title'],
        "Activity Status": "Resignation",
        "Performance Score": performance,
        "Employee Satisfaction": f"{satisfaction:.2f}",
        "Years At Company": years,
        "Priority": priority
    })

# Display section
st.subheader("Recent Activity")

if activity_data:
    activity_df = pd.DataFrame(activity_data)

    # Priority sorting order
    priority_order = pd.CategoricalDtype(['High', 'Medium', 'Low'], ordered=True)
    activity_df['Priority'] = activity_df['Priority'].astype(priority_order)

    # Sort
    activity_df = activity_df.sort_values(['Date', 'Priority'], ascending=[False, True])
    activity_df.index = np.arange(1, len(activity_df)) + 1

    color_map = {
        'High': 'background-color: rgba(255, 0, 0, 0.15); color: white;',      # very light red
        'Medium': 'background-color: rgba(255, 200, 0, 0.15); color: white;',      # very light yellow
        'Low': 'background-color: rgba(0, 200, 0, 0.15); color: white;'           # very light green
    }

    def highlight_priority(val):
        return color_map.get(val, '')

    styled_df = activity_df.style.applymap(highlight_priority, subset=['Priority'])

    st.dataframe(styled_df, use_container_width=True)

else:
    st.write("No recent activity found.")

```

*Figure 223: Source Code of Recent Activity 2*

The recent\_activity() function is designed to provide HR personnel with a table that displays recent employee activities such as new hires and resignations. First, the function checks whether the required data including employee details and relevant performance metrics is available in the session state. If the data is missing, the user is prompted to log in again.

Once the necessary data is loaded, the function will fetch the top 5 most recent hires and resignations. For each employee, it calculates the performance and satisfaction scores and assigns a priority level based on these metrics.

Then, it generates a table that displays these recent activities. The priority is color-coded for better clarity which for the "High" priority is marked in red, "Medium" in yellow and "Low" in green. This recent activity interface allows HR personnel to quickly assess recent workforce activities and take appropriate actions. If no recent activity is found, the function displays a message for "No recent activity found."

Recent Activity								
	Employee ID	Date	Job Title	Activity Status	Performance Score	Employee Satisfaction	Years At Company	Priority
1	1000573	2025-07-01	Consultant	New Hire	3	1.30	0	High
2	1002635	2025-07-01	Technician	New Hire	1	2.94	0	High
3	1003354	2025-07-01	Technician	New Hire	5	2.49	0	High
4	1002146	2025-07-01	Engineer	New Hire	3	3.72	0	High
5	1000697	2025-07-01	Technician	New Hire	1	4.48	0	Low
6	43600	2025-06-30	Developer	Resignation	1	3.37	6	High
7	30572	2025-06-30	Analyst	Resignation	1	2.61	9	High
8	70106	2025-06-30	Technician	Resignation	1	1.50	0	Low
9	64636	2025-06-30	Technician	Resignation	1	4.16	2	Low
10	46851	2025-06-30	Technician	Resignation	2	3.20	2	Low

Figure 224: Interface for Overview of Recent Activity

The figure above displays the interface for recent activity which is a useful tool for HR personnel to assess the latest developments in the workforce. This table shows the most recent employee hires and resignations with essential data such as Employee ID, Job Title, Activity Status (New Hire or Resignation), Performance Score, Employee Satisfaction Score, Years at Company and Priority. The Priority column is color-coded for clarity with "High" marked in red, "Medium" in yellow and "Low" in green for allowing HR to quickly identify critical cases.

The data is divided into two categories which is New Hires and Resignations. For the new hires are prioritized based on satisfaction scores. For resignations, priority is determined by performance scores and years of service. Employees with higher performance and more years at the company are assigned a "High" priority which signalling the need for immediate attention for consultation.

	Employee ID	Date	Job Title	Activity Status	Performance Score	Employee Satisfaction	Years At Company	Priority
1	1000573	2025-07-01	Consultant	New Hire	3	1.30	0	High
2	1002635	2025-07-01	Technician	New Hire	1	2.94	0	High
3	1003354	2025-07-01	Technician	New Hire	5	2.49	0	High
4	1002146	2025-07-01	Engineer	New Hire	3	3.72	0	High
5	1000697	2025-07-01	Technician	New Hire	1	4.48	0	Low
6	43600	2025-06-30	Developer	Resignation	1	3.37	6	High
7	30572	2025-06-30	Analyst	Resignation	1	2.61	9	High
8	70106	2025-06-30	Technician	Resignation	1	1.50	0	Low
9	64636	2025-06-30	Technician	Resignation	1	4.16	2	Low
10	46851	2025-06-30	Technician	Resignation	2	3.20	2	Low

Figure 225: Interface for Recent Activity Advanced Functions 1

The figure above shows the table is sorted in descending order by date to ensure that HR teams can quickly access the most recent activities. Additionally, the table offers advanced functions such as sorting by any column which allows HR to easily identify trends or outliers. Users can also pin important columns for quick reference, hide columns for a cleaner view and autosize columns for better readability. These features enhance the user experience which has making it easier for HR to track trends, make informed decisions and manage workforce changes effectively.

This layout provides flexibility to ensure that HR teams can adjust the table view according to their specific needs and preferences. With these interactive functions, HR personnel can gain insights in real-time and efficiently navigate through large amounts of employee data.

Recent Activity								
	Employee ID	Date	Job Title	Activity Status	Performance Score	↑ Employee Satisfaction	Years At Company	Priority
1	1000573	2025-07-01	Consultant	New Hire	3	1.30	0	High
8	70106	2025-06-30	Technician	Resignation	1	1.50	0	Low
3	1003354	2025-07-01	Technician	New Hire	5	2.49	0	High
7	30572	2025-06-30	Analyst	Resignation	1	2.61	9	High
2	1002635	2025-07-01	Technician	New Hire	1	2.94	0	High
10	46851	2025-06-30	Technician	Resignation	2	3.20	2	Low
6	43600	2025-06-30	Developer	Resignation	1	3.37	6	High
4	1002146	2025-07-01	Engineer	New Hire	3	3.72	0	High
9	64636	2025-06-30	Technician	Resignation	1	4.16	2	Low
5	1000697	2025-07-01	Technician	New Hire	1	4.48	0	Low

Figure 226: Interface for Recent Activity Advanced Functions 2

The figure above shows an example of the "Recent Activity" table with the "Employee Satisfaction" column sorted in ascending order. This sorting allows HR to easily identify employees with the lowest satisfaction scores at the top of the list to help them prioritize attention and intervention for those at risk of leaving the company. The ability to sort data provides HR teams with a more effective way to monitor workforce trends and address potential concerns promptly.

### 5.4.3.2.3 Employee Performance Page

```
def create_dept_chart(data, x_col, y_col, title, company_avg, line_color="red"):

    data['Pct_Diff'] = ((data[y_col] - company_avg) / company_avg * 100)

    fig = px.bar(
        data,
        x=x_col,
        y=y_col,
        title=title,
        text=f"{{val:.1f}} ({pct:+.1%}%)" for val, pct in zip(data[y_col], data['Pct_Diff']),
        color_discrete_sequence=[ "#4B7881"]
    )

    fig.update_traces(
        textposition="outside",
        textfont_size=10,
        marker_color="#17222A",
        textangle=0,
        opacity=0.9
    )

    fig.add_hline(
        y=company_avg,
        line_dash="dash",
        line_color=line_color,
        line_width=2,
        annotation_text="Company Avg: {company_avg:.1f}",
        annotation_position="top right",
        annotation_font=dict(size=12, color="black"),
        annotation_bgcolor="rgba(255,255,255,0.7)",
        annotation_y=company_avg + 1.25
    )

    y_max = max(data[y_col].max() * 1.5, company_avg * 1.5)
    fig.update_layout(
        yaxis_range=[0, y_max],
        margin=dict(t=100, b=40),
        plot_bgcolor="rgba(0,0,0,0)"
    )
    return fig
```

Figure 227: Source Code of Employee Performance Page 1

```
def employee_performance():
    st.title("Employee Performance Overview")

    if 'df_deployment' not in st.session_state:
        st.warning("HR data not loaded. Please log in again.")
        return

    df = st.session_state.df_deployment

    # Calculate company-wide averages
    company_avg_perf = df['Performance_Score'].mean()
    company_avg_satisfaction = df['Employee_Satisfaction_Score'].mean()
    company_avg_overtime = df['Overtime_Hours'].mean()

    # 1. Department Performance
    st.subheader("Department Performance")

    dept_performance = df.groupby('Department').agg(
        Avg_Performance=('Performance_Score', 'mean'),
        Avg_Satisfaction=('Employee_Satisfaction_Score', 'mean'),
        Avg_Overtime=('Overtime_Hours', 'mean'),
        Employee_Count=('Employee_ID', 'count')
    ).reset_index()

    # Display metrics
    col1, col2, col3 = st.columns(3)
    with col1:
        st.metric("Total Departments", len(dept_performance))
    with col2:
        st.metric("Highest Performing Dept",
                  dept_performance.loc[dept_performance['Avg_Performance'].idxmax()]['Department'])
    with col3:
        st.metric("Most Satisfied Dept",
                  dept_performance.loc[dept_performance['Avg_Satisfaction'].idxmax()]['Department'])

    # Department comparison charts
    tab1, tab2, tab3 = st.tabs(["Performance", "Satisfaction", "Overtime"])

    with tab1:
        fig = create_dept_chart(
            dept_performance,
            'Department',
            'Avg_Performance',
            'Average Performance Score by Department',
            company_avg_perf
        )
        st.plotly_chart(fig, use_container_width=True)
```

Figure 228: Source Code of Employee Performance Page 2

```

with tab2:
    fig = create_dept_chart(
        dept_performance,
        'Department',
        'Avg_Satisfaction',
        'Average Satisfaction by Department',
        company_avg_satisfaction
    )
    st.plotly_chart(fig, use_container_width=True)

with tab3:
    fig = create_dept_chart(
        dept_performance,
        'Department',
        'Avg_Overtime',
        'Average Overtime Hours by Department',
        company_avg_overtime
    )
    st.plotly_chart(fig, use_container_width=True)

st.markdown("---")

```

*Figure 229: Source Code of Employee Performance Page 3*

The `create_dept_chart()` function is used to create bar charts that visualize department performance data, such as average performance score, employee satisfaction, and overtime hours. It compares these metrics to the company average and shows the percentage difference for each department. The chart includes color-coded bars and a reference line indicating the company average, helping HR teams easily compare departments and identify areas that require attention.

In the `employee_performance()` function, this charting tool is used to visualize and compare key metrics across departments, including performance, satisfaction, and overtime. The function also displays department-specific statistics, such as the total number of departments, the highest-performing department, and the most satisfied department, helping HR make informed decisions about employee engagement and department-level improvements.

#### 5.4.3.2.3.1 Employee Performance Overview



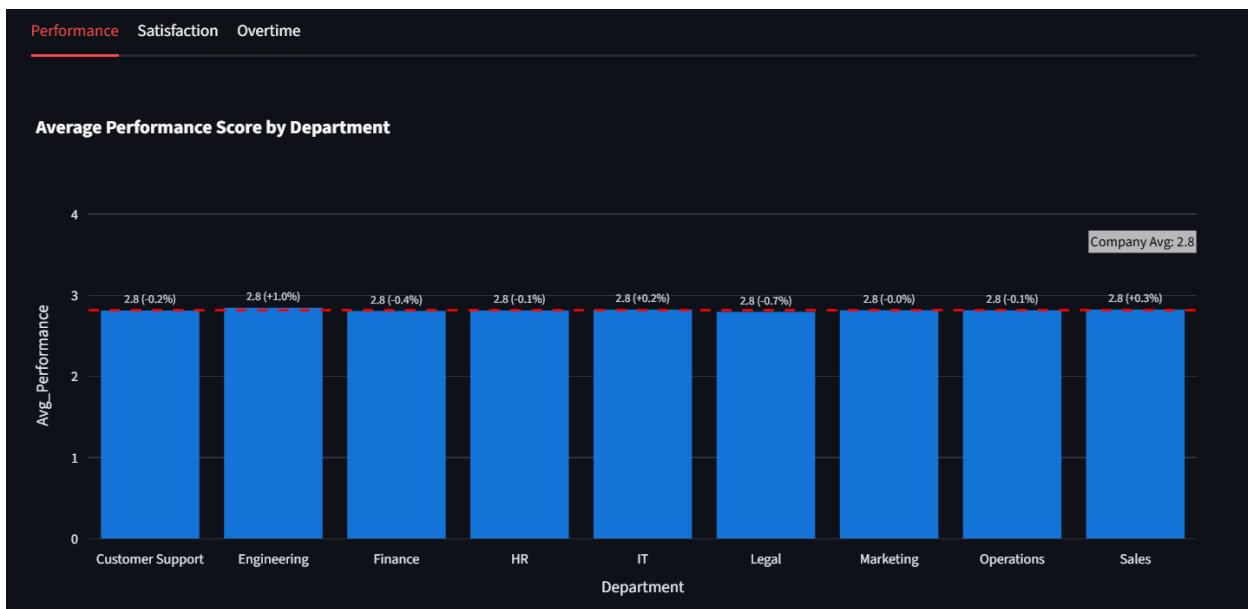
*Figure 230: Interface of Employee Performance Overview*

The figure above shows the **Employee Performance Overview** interface. This is a component for HR personnel to gain valuable insights into the performance and satisfaction levels across various

departments within the organization. This interface provides key metrics which include “Total Departments”, “Highest Performing Department” and “Most Satisfied Department”.

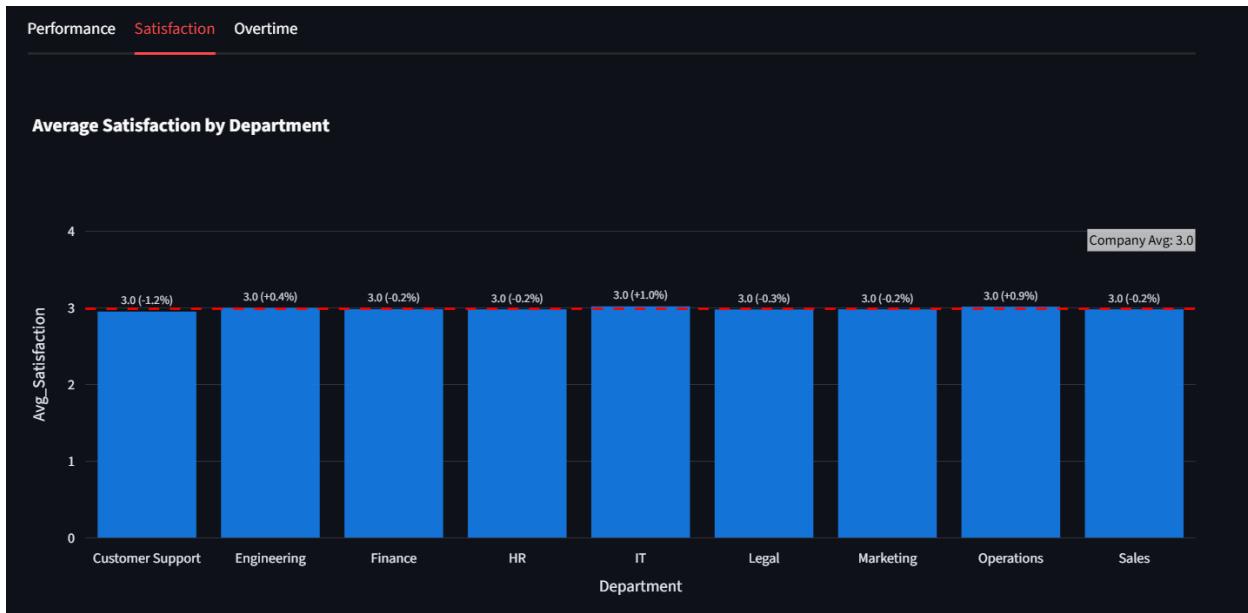
The “Total Departments” metric displays the total number of departments within the company which is currently 9. This metric enables HR to identify areas of excellence that could potentially be applied to improve performance in other departments. Besides, the “Highest Performing Department” highlights the department with the highest average performance score, with Engineering being the top performer in the present period. Similarly, the "Most Satisfied Department" identifies the department with the highest employee satisfaction score, with IT leading in this area. This data helps HR recognize departments where employee engagement is strong as it offers insights into effective practices that can be replicated across the organization.

All data displayed in this dashboard is sourced directly from the dataset to ensure the information is both real and at the latest. This feature empowers HR teams by providing them with up-to-date data that aids in making informed, strategic decisions regarding workforce management, performance improvements and initiatives aimed at enhancing employee satisfaction. By applying this data, HR can make data-driven decisions to further strengthen organizational performance and engagement.



*Figure 231: Interface of Average Performance Score by Department*

The figure above displays the "Average Performance Score by Department." It shows the performance score for each department compared to the company's average performance score of 2.8 with marked by a red dashed line. The percentage change from the previous quarter (Q1) is shown in brackets next to each department's score such as (-0.2%) or (+0.4%). This allows HR to quickly identify departments that are performing above or below the company average and assess whether the performance is improving or declining from the previous quarter. The visual comparison between departments along with the performance change percentage helps HR focus on departments that may need further improvement or adjustments.



*Figure 232: Interface of Average Satisfaction by Department*

The figure above shows the "Average Satisfaction by Department" reflecting the satisfaction level of employees across different departments. Similarly to the performance chart, the company's average satisfaction score of 3.0 is indicated with a red dashed line. Each department's satisfaction score is accompanied by the percentage change from the previous quarter such as (-1.2%) or (+0.3%). This provides HR with a clear view of any significant shifts in employee satisfaction and helps identify areas where employee engagement or satisfaction has improved or worsened. The percentage change enables HR to take targeted actions in addressing employee concerns in specific departments.



*Figure 233: Average Overtime Hours by Department*

The figure above presents the "Average Overtime Hours by Department" displaying the average overtime hours worked in each department with the company average of 14.6 hours marked by the red dashed line. Each department's overtime hours are followed by the percentage change from the last quarter such as (+0.7%) or (-0.1%). This helps HR assess whether departments are overworking employees or managing their workloads efficiently. By tracking the percentage change in overtime hours from the last quarter, HR gains valuable insight into workload trends. This enables them to take proactive steps to prevent employee burnout and maintain a healthy work-life balance across departments.

Overall, these visualizations with percentage changes from the previous quarter offer HR the valuable insights into departmental trend. This helps them make data-driven decisions to improve employee engagement, satisfaction, performance and workload management.

### 5.4.3.2.3.2 Quarterly Trends

```
# 2. Quarterly Trends
st.subheader("Quarterly Trends")

df['Quarter'] = df['Hire_Date'].dt.to_period('Q').astype(str)
df['Quarter'] = df['Quarter'].apply(lambda x: f'{x[:4]}-{x[5:]}')
current_quarter = pd.to_datetime('today').to_period('Q')
df['Quarter_Period'] = df['Quarter'].apply(lambda x: pd.to_datetime(x[:4] + '-' + x[5:]).to_period('Q'))
df = df[df['Quarter_Period'] <= current_quarter]

# Select the latest 6 quarters
latest_quarters = df['Quarter'].sort_values(ascending=False).unique()[:6]
df_filtered = df[df['Quarter'].isin(latest_quarters)]

quarterly_trends = df_filtered.groupby('Quarter').agg(
    Avg_Performance=('Performance_Score', 'mean'),
    Avg_Satisfaction=('Employee_Satisfaction_Score', 'mean'),
    Hire_Count=('Employee_ID', 'count')
).reset_index().sort_values('Quarter')

fig = px.line(quarterly_trends,
              x='Quarter',
              y='Avg_Performance',
              title='Performance Score Trend (Last 6 Quarters)',
              markers=True,
              text=quarterly_trends['Avg_Performance'].round(2))

fig.update_traces(textposition="top center",
                   line=dict(width=2),
                   marker=dict(size=10))

fig.add_hline(y=company_avg_perf, line_dash="dash", line_color="red",
              annotation_text=f"Company Avg: {company_avg_perf:.1f}")

st.plotly_chart(fig, use_container_width=True)

st.markdown("----")
```

*Figure 234: Source Code of Quarterly Trends*

The source code above defines a function to display **Quarterly Trends** where it calculates average performance and satisfaction scores for the latest 6 quarters. A line graph is created to compare these scores with the company average to provide HR with insights into performance changes. The Overtime Analysis categorizes employees by overtime hours worked by using bar and pie charts to visualize how performance and satisfaction correlate with overtime hours worked. This helps HR assess workload trends and make data-driven decisions.



*Figure 235: Interface of Quarterly Trends*

The figure above shows the Performance Score Trend over the last 6 quarters. It presents the average performance score for each quarter and highlights how these scores compare to the company average which marked by the red dashed line. The chart visually demonstrates fluctuations in employee performance across the periods with HR able to identify any significant increases or declines in performance.

By viewing the line chart, HR can benefit from this visualization by identifying patterns in employee performance over time. For example, if performance dips in certain quarters, HR can correlate this with external factors or departmental issues to take necessary actions. The comparison with the company average helps HR evaluate whether performance is above or below the company standard and prompting targeted interventions for underperforming teams or departments.

### 5.4.3.2.3.3 Overtime Analysis

```

# 3. Overtime Analysis
st.subheader("Overtime Analysis")

df['Overtime_Category'] = pd.cut(df['Overtime_Hours'],
                                 bins=[-1, 5, 10, 15, 20, 100],
                                 labels=['0-5 hrs', '6-10 hrs', '11-15 hrs', '16-20 hrs', '20+ hrs'])

overtime_summary = df.groupby('Overtime_Category').agg(
    Avg_Performance=('Performance_Score', 'mean'),
    Employee_Count=('Employee_ID', 'count'),
    Avg_Satisfaction=('Employee_Satisfaction_Score', 'mean')
).reset_index()

coll, col2 = st.columns(2)
with coll:
    fig = px.bar(overtime_summary,
                  x='Overtime_Category',
                  y='Avg_Performance',
                  color='Avg_Performance',
                  title='Average Performance by Overtime Range',
                  text='Avg_Performance')
    fig.update_traces(texttemplate='%{text:.2f}', textposition='outside')
    st.plotly_chart(fig, use_container_width=True)

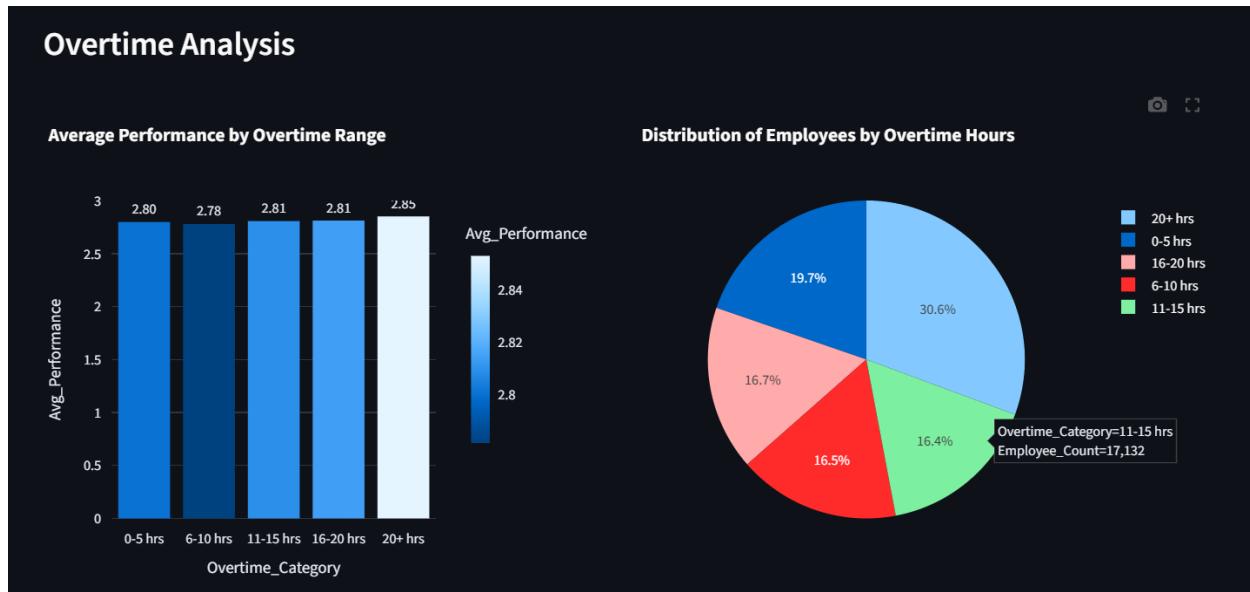
with col2:
    fig = px.pie(overtime_summary,
                  names='Overtime_Category',
                  values='Employee_Count',
                  title='Distribution of Employees by Overtime Hours')
    st.plotly_chart(fig, use_container_width=True)

st.markdown("---")

```

*Figure 236: Source Code of Overtime Analysis*

The Overtime Analysis section of the dashboard categorizes employees based on overtime hours worked by using ranges like 0-5 hours, 6-10 hours and so on. The data is grouped by these categories by calculating the average performance score and employee count for each group. Then, there are two visualizations that are created which are a bar chart showing average performance across overtime categories and a pie chart depicting the distribution of employees in each category. This analysis helps HR assess the impact of overtime on employee performance and gain insights into the distribution of employees across different overtime categories.



*Figure 237: Interface of Overtime Analysis*

The Overtime Analysis section of the dashboard provides HR with valuable insights into how overtime hours affect employee performance. By categorizing employees into different overtime ranges (0-5 hours, 6-10 hours, 11-15 hours, 16-20 hours and 20+ hours), HR can assess the average performance score for each group. So, the bar chart allows the HR to understand whether employees who work more overtime tend to perform better or worse. If performance decreases as overtime increases, HR can use this information to implement strategies that limit excessive overtime to prevent burnout and ensure that employees maintain a good work-life balance.

In addition, the pie chart shows the distribution of employees across these overtime categories to offer HR a clear view of how overtime is spread across the workforce. If many employees are working excessive overtime like more than 20 hours, HR can take action to distribute workloads more evenly. This can help prevent overburdening certain employees and ensure a healthier and more sustainable working environment. Overall, these visualizations provide HR with data-driven insights that enable them to make informed decisions regarding overtime policies, employee well-being and performance management.

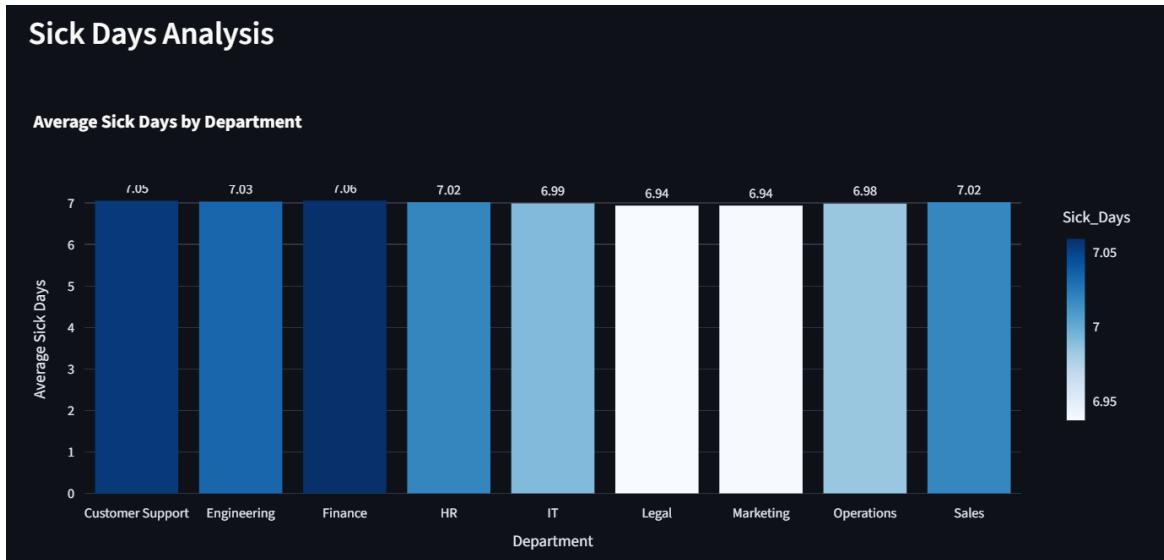
#### 5.4.3.2.3.4 Sick Day Analysis

```
# 4. Sick Days Analysis
st.subheader("Sick Days Analysis")

sick_by_dept = df.groupby('Department')['Sick_Days'].mean().reset_index()
fig = px.bar(
    sick_by_dept,
    x='Department',
    y='Sick_Days',
    title='Average Sick Days by Department',
    color='Sick_Days',
    color_continuous_scale='Blues',
    text='Sick_Days'
)
fig.update_traces(texttemplate=' %{text:.2f}', textposition='outside')
fig.update_layout(yaxis_title="Average Sick Days")
st.plotly_chart(fig, use_container_width=True)
```

*Figure 238: Source Code of Sick Day Analysis*

The figure above shows the average sick days taken by employees in each department. The bar chart visualizes the average number of sick days per department, with the bars colored in shades of blue based on the number of sick days. The exact average sick days are displayed on top of each bar, making it easy for HR to compare the sick day trends across departments.



*Figure 239: Interface of Sick Day Analysis*

The figure above shows the average sick days taken by employees in each department. The bar chart allows HR to easily compare the number of sick days across different departments. The bars are color-coded in shades of blue with the darker bars indicating higher average sick days and the lighter bars indicating lower averages. So, HR can use this chart to quickly identify which

departments are experiencing higher absenteeism. For example, departments like Customer Support and Engineering have higher average sick days which around 7 while departments like Legal and Marketing have lower averages which around 6.94.

This analysis helps HR identify trends in absenteeism and determine if certain departments may require closer attention such as improving employee well-being or reviewing workload distribution. By analysing the bar chart, HR can use these insights to develop targeted strategies for reducing sick days such as offering wellness programs or investigating underlying causes for higher absenteeism in specific departments.

#### 5.4.3.2.3.5 Download Summary Report

```

# Executive Report Section
st.markdown("----")
st.subheader("Overall Performance Report Summary Download")

# Create summary data for report
overtime_summary = df.groupby('Overtime_Hours').agg(
    Avg_Performance=('Performance_Score', 'mean'),
    Employee_Count=('Employee_ID', 'count')
).reset_index()

# Calculate sick days metrics
max_sick_dept = df.groupby('Department')['Sick_Days'].mean().idxmax()
min_sick_dept = df.groupby('Department')['Sick_Days'].mean().idxmin()

report_data = {
    "Department Performance": dept_performance,
    "Quarterly Trends": quarterly_trends,
    "Overtime Analysis": overtime_summary,
    "Performance Correlations": df[['Employee_Satisfaction_Score', 'Performance_Score', 'Department', 'Job_Title']],
    "Sick Days Analysis": df.groupby('Department').agg(
        {'Sick_Days': 'mean',
         'Performance_Score': 'mean'
    ).reset_index()
}

# Convert to Excel
def create_excel_report(data_dict):
    output = BytesIO()
    with pd.ExcelWriter(output, engine='xlsxwriter') as writer:
        for sheet_name, df in data_dict.items():
            df.to_excel(writer, sheet_name=sheet_name[:31], index=False)
    return output.getvalue()

excel_report = create_excel_report(report_data)

st.download_button(
    label="⬇️ Download Full Performance Report (Excel)",
    data=excel_report,
    file_name="employee_performance_report.xlsx",
    mime="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
    help="Download the full performance report in Excel format."
)

# Text summary version
text_report = f"""
Employee Performance Report
Generated on: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}
"""

```

Figure 240: Source Code of Download Summary Report 1

```

# Text summary version
text_report = """
Employee Performance Report
Generated on: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}

==== Key Metrics ===
- Total Departments: {len(dept_performance)}
- Highest Performing Department: {dept_performance.loc[dept_performance['Avg_Performance'].idxmax()]['Department']}
- Most Satisfied Department: {dept_performance.loc[dept_performance['Avg_Satisfaction'].idxmax()]['Department']}

==== Overtime Insights ===
- Highest performing overtime range: {overtime_summary.loc[overtime_summary['Avg_Performance'].idxmax()]['Overtime_Hours']}
- Most common overtime range: {overtime_summary.loc[overtime_summary['Employee_Count'].idxmax()]['Overtime_Hours']}
- Percentage in most common range: {overtime_summary['Employee_Count'].max()/overtime_summary['Employee_Count'].sum()*100:.1f}%

==== Sick Days Insights ===
- Department with most sick days: {max_sick_dept}
- Department with least sick days: {min_sick_dept}
- Average sick days across company: {df['Sick_Days'].mean():.1f} days

==== Recommendations ===
1. Investigate why {dept_performance.loc[dept_performance['Avg_Performance'].idxmax()]['Department']} performs best
2. Address satisfaction in {dept_performance.loc[dept_performance['Avg_Satisfaction'].idxmin()]['Department']}
3. Monitor employees with {overtime_summary.loc[overtime_summary['Employee_Count'].idxmax()]['Overtime_Hours']} overtime for burnout
4. Investigate high sick days in {max_sick_dept} department
5. Review wellness programs in departments with above-average sick days
6. Analyze correlation between performance and sick days
"""

st.download_button(
    label="Download Quick Summary (Text)",
    data=text_report,
    file_name="performance_summary.txt",
    mime="text/plain",
    help="Download a quick summary of the performance report in text format."
)

```

*Figure 241: Source Code of Download Summary Report 2*

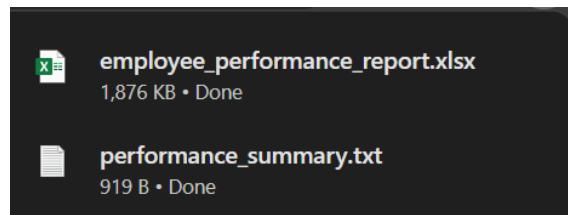
Figure above shows the Executive Report Section where HR can generate and download a comprehensive employee performance report for executives to view. The report includes various insights such as the average performance scores and employee counts for different overtime ranges as well as the department with the most and least sick days. So, HR can download the report in Excel format and share it with executives, to provide them with a high-level overview of performance, overtime and sick day trends.

Additionally, a text-based summary is generated with key metrics like the highest performing department, the most common overtime range and recommendations for HR. This summary can also be downloaded in text format to have a quick overview of the key insights and actionable recommendations for HR to address areas like employee performance, overtime management and sick days trends.



*Figure 242: Interface of Download Summary Report*

Figure above shows the interface with two download options for the Overall Performance Report Summary which are one in Excel format and the other in text file format.



*Figure 243: Downloaded Summary Report*

Once the user clicks the download button, the file will start downloading. So once the download is complete, it will be saved in the downloads section of the computer as shown in the figure above.

```
Employee Performance Report
Generated on: 2025-07-18 01:20:37

==== Key Metrics ====
- Total Departments: 9
- Highest Performing Department: Engineering
- Most Satisfied Department: IT

==== Overtime Insights ====
- Highest performing overtime range: 26.0
- Most common overtime range: 24.0
- Percentage in most common range: 3.5%

==== Sick Days Insights ====
- Department with most sick days: Finance
- Department with least sick days: Legal
- Average sick days across company: 7.0 days

==== Recommendations ====
1. Investigate why Engineering performs best
2. Address satisfaction in Customer Support
3. Monitor employees with 24.0 overtime for burnout
4. Investigate high sick days in Finance department
5. Review wellness programs in departments with above-average sick days
6. Analyze correlation between performance and sick days
```

*Figure 244: Employee Performance Report in Text File*

Figure above shows the performance summary text file which includes a detailed overview of key metrics, overtime insights, sick days insights and recommendations based on the current data at the time of download. The date and time of the download are displayed at the top for HR and executives to know that the data is up to date.

The Key Metrics section summarizes important department-related information like the total number of departments, the highest performing department and the most satisfied department. Besides, Overtime Insights provide details on the highest-performing overtime range, the most common overtime range and the percentage of employees in the most common overtime range. Additionally, the Sick Days Insights section highlights the departments with the most and least sick days with the average sick days across the entire company. Finally, the Recommendations section offers actionable insights such as investigating high performance in certain departments and monitoring overtime to prevent burnout.

In conclusion, this summary is designed to help HR and executives quickly review the most important trends and make data-driven decisions to improve performance, employee well-being and productivity within the company.

	A	B	C	D	E	F
1	Department	Avg_Performance	Avg_Satisfaction	Avg_Overtime	Employee_Count	Pct_Diff
2	Customer Support	2.811925423	2.950249162	14.47632958	11639	-0.945612134
3	Engineering	2.845885938	2.998587723	14.71562908	11485	0.691796391
4	Finance	2.806868791	2.982710689	14.45769099	11676	-1.073146829
5	HR	2.812798887	2.980036519	14.71324233	11501	0.675465005
6	IT	2.823539504	3.018564811	14.78356352	11657	1.156638297
7	Legal	2.796354924	2.978832891	14.62916061	11687	0.100135233
8	Marketing	2.815604659	2.980758822	14.57100034	11676	-0.29782678
9	Operations	2.815378007	3.014691581	14.58393471	11640	-0.209323294
10	Sales	2.824161539	2.980922957	14.60291186	11539	-0.079471968

*Figure 245: Employee Performance Report in Excel File*

The figure above shows the downloaded file, "employee\_performance\_report" which contains a table with key performance data for each department in the company. The table includes several important metrics such as Average Performance, Average Satisfaction, Average Overtime, Employee Count and Percentage Difference.

The Average Performance column reflects the performance scores of employees in each department while the Average Satisfaction column shows the overall employee satisfaction within

those departments. Furthermore, the Average Overtime column indicates how many overtime hours on average for employees in each department are working and for the Employee Count represents the total number of employees in each department. Lastly, the Percentage Difference (Pct\_Diff) indicates how much a department's performance or satisfaction is above or below the company average where helping HR and executives team to determine whether the department is meeting expectations.

This table offers HR and executives a comprehensive view of department-wise performance to help them assess trends in employee satisfaction, overtime work and overall performance across departments. It serves as a tool for identifying which departments may need attention like improving satisfaction, reducing overtime or increasing performance. With this data, HR and executives can make informed decisions to optimize resource allocation, target improvements and ensure better work-life balance for employees.

#### 5.4.3.2.4 Actionable Insights for Retention and Productivity Page

```
def generate_dynamic_insights(df, insight_type, churn_rate):
    high_performers = df[df['Performance_Score'] > 4]
    low_satisfaction = df[df['Employee_Satisfaction_Score'] < 2.5]
    high_overtime = df[df['Overtime_Hours'] > 25]
    optimal_performance = df[df['Performance_Score'] == 5]

    # Retention Insights
    if insight_type == "Retention":
        insights = []

        if churn_rate > 20:
            insights.append(f"High churn risk (churn rate: {churn_rate}%). Consider retention strategies.")
            insights.append("Suggested Actions: Consider improving engagement and career development programs, review compensation, and increase job satisfaction.")

        if len(high_performers) > 0:
            insights.append(f"High-performing employees: {len(high_performers)} found. Reward and retain them.")
            insights.append("Suggested Actions: Reward high performers with promotions, incentives, or leadership opportunities.")

        if len(low_satisfaction) > 0:
            insights.append(f"Employees with low satisfaction: {len(low_satisfaction)} need immediate attention.")
            insights.append("Suggested Actions: Schedule 1:1 meetings, understand their concerns, and offer support.")

        return insights

    # Productivity Insights
    elif insight_type == "Productivity":
        insights = []

        if len(high_overtime) > 0:
            insights.append(f"Employees working >25 overtime hours: {len(high_overtime)}. Monitor for burnout.")
            insights.append("Suggested Actions: Consider workload adjustments or introducing wellness programs.")

        if len(optimal_performance) > 0:
            insights.append(f"Optimal performers: {len(optimal_performance)} found. Foster leadership roles.")
            insights.append("Suggested Actions: Nurture these employees for future leadership roles or mentorship programs.")

        return insights
```

Figure 246: Source Code of Actionable Insights for Retention and Productivity Page 1

```

# Engagement Insights
elif insight_type == "Engagement":
    insights = []

    high_engaged = df[(df['Performance_Score'] > 4) & (df['Employee_Satisfaction_Score'] > 4)]
    if len(high_engaged) > 0:
        insights.append("Highly engaged employees: {len(high_engaged)} found. Leverage them for mentorship roles.")
        insights.append("Suggested Actions: Leverage high-engagement employees for mentoring new hires or driving culture initiatives.")

    low_engaged = df[(df['Performance_Score'] <= 3) & (df['Employee_Satisfaction_Score'] <= 3)]
    if len(low_engaged) > 0:
        insights.append("Low engagement risk: {len(low_engaged)} employees with both low satisfaction and performance.")
        insights.append("Suggested Actions: Address underlying issues, including performance coaching, or reassignment to more suitable roles.")

    return insights

# Compensation Insights
elif insight_type == "Compensation":
    insights = []

    underpaid_performers = df[(df['Performance_Score'] == 5) & (df['Monthly_Salary'] < df['Monthly_Salary'].median())]
    if len(underpaid_performers) > 0:
        insights.append(f"{len(underpaid_performers)} high performers are underpaid. Consider salary adjustments.")
        insights.append("Suggested Actions: Ensure competitive compensation packages for high performers to avoid attrition.")

    low_salary_high_satisfaction = df[(df['Monthly_Salary'] < df['Monthly_Salary'].median()) & (df['Employee_Satisfaction_Score'] > 4)]
    if len(low_salary_high_satisfaction) > 0:
        insights.append(f"{len(low_salary_high_satisfaction)} employees with low salary and high satisfaction found.")
        insights.append("Suggested Actions: Review salary adjustments to ensure satisfaction remains high.")

    return insights

return []

```

Figure 247: Source Code of Actionable Insights for Retention and Productivity Page 2

```

def actionable_insights():
    st.set_page_config(layout="centered")
    st.title("Actionable Insights for Retention and Productivity")

    # Insight Selector
    st.subheader("🔍 Insight Selector", anchor=False)
    insight_type = st.selectbox(
        "What type of insights would you like to view and analyse?", 
        ["Compensation", "Productivity", "Retention", "Engagement"], 
        index=1,
        help="Select the HR metric you want to analyze"
    )

    st.markdown("<br>", unsafe_allow_html=True)
    generate_clicked = st.button("➡ Generate Insights")

    if "df_deployment" not in st.session_state:
        st.error("HR data not loaded. Please log in again.")
        return

    df = st.session_state.df_deployment
    dummy_churn = 8.6

    if generate_clicked:
        today = pd.to_datetime("today")
        current_year = today.year
        current_quarter = (today.month - 1) // 3 + 1
        start_date, end_date = get_quarter_dates(current_year, current_quarter)
        churn_rate = get_churn_rate(df, start_date, end_date)

        insights = generate_dynamic_insights(df, insight_type, churn_rate)
        st.subheader(f"({insight_type}) Insights")

        if insights:
            insight_text = ""
            numbered = 1
            for i in range(0, len(insights), 2):
                main = insights[i]
                action = insights[i+1] if i+1 < len(insights) else None

                st.markdown(f"***{numbered}. {main}***")
                if action:
                    for prefix in ["Suggested Actions:", "Suggested Action:"]:
                        if action.startswith(prefix):
                            action = action.replace(prefix, "").strip()
                    st.markdown(f"  &nbsp;&nbsp;&nbsp;➡ Suggested Action: {action}", unsafe_allow_html=True)

                insight_text += f"(numbered). {main}\n"
                insight_text += f"  ➡ Suggested Action: {action}\n"
            else:
                insight_text += f"(numbered). {main}\n"

            numbered += 1

    # Action Plan and Report
    st.markdown("---")
    st.subheader("Report Summary and Action Plan")

```

Figure 248: Source Code of Actionable Insights for Retention and Productivity Page 3

```

    action_plans = {
        "Retention": """Sample Retention Action Plan:
1. Identify at-risk employees (churn risk >40%)
2. Schedule stay interviews within 2 weeks
3. Create personalized development plans
4. Review compensation benchmarks
5. Establish mentorship pairings""",
        "Productivity": """Sample Productivity Action Plan:
1. Identify teams with overtime >50 hrs
2. Introduce flexible work schedules or task rotation
3. Launch wellness or fatigue-monitoring initiatives
4. Monitor impact of training programs on performance
5. Adjust workloads for balance and sustainability""",
        "Engagement": """Sample Engagement Action Plan:
1. Conduct anonymous engagement surveys
2. Launch mentorship or buddy programs
3. Recognize and reward contributions regularly
4. Organize quarterly engagement townhalls
5. Support career growth and learning pathways""",
        "Compensation": """Sample Compensation Action Plan:
1. Benchmark current salary against market rates
2. Identify underpaid high performers
3. Develop a transparent performance-based bonus system
4. Review compensation equity across roles and departments
5. Communicate compensation philosophy to employees"""
    }

    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    report = f"""Employee {insight_type} Insights Report
Generated on: {timestamp}

--- INSIGHTS ---
{insight_text}

--- ACTION PLAN ---
{action_plans.get(insight_type, "No action plan available.")}
"""

    st.download_button(
        label="Download Full Insight & Action Plan Report",
        data=report,
        file_name=f"{insight_type.lower()}_insight_report.txt",
        mime="text/plain",
        help="Download the full insight and action plan report in text format."
    )

```

*Figure 249: Source Code of Actionable Insights for Retention and Productivity Page 4*

The figure above code defines a section of the dashboard where HR users can generate actionable insights for improving employee retention, productivity, engagement and compensation. The “generate\_dynamic\_insights” function analyzes employee data and provides insights based on selected categories such as identifying high performers, employees at risk of churn or those with low satisfaction. HR can choose the type of insights that they want to analyze and the system generates customized recommendations for the actions that HR should take such as improving engagement, addressing burnout or adjusting compensation. After generating insights, HR can download a text report that includes both the insights and suggested action plans to help them make data-driven decisions to enhance employee satisfaction and retention.



Figure 250: Interface of Insight Selector

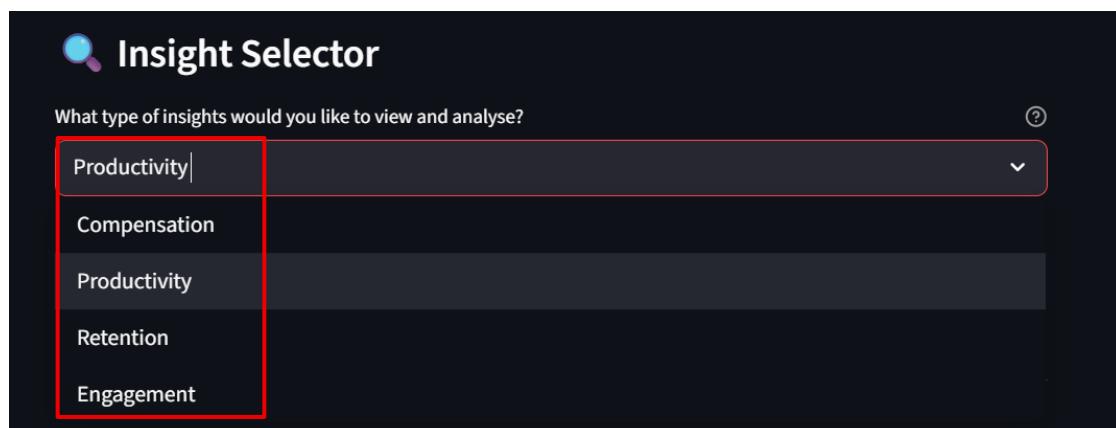


Figure 251: Option of Insight Selector

The figure above shows the Insight Selector interface where HR can choose from four options which has included Compensation, Productivity, Retention and Engagement. These categories allow HR to focus on specific aspects of employee performance and well-being for providing customized insights and recommendations. By viewing these insights, it could help the company to improve retention, enhance productivity, adjust compensation and boost overall employee engagement.

Each section above provides a focused action plan based on the insights generated respectively for offering the strategic guidance to improve compensation, productivity, retention and engagement within the organization.

### 5.4.3.2.4.1 Compensation Insights

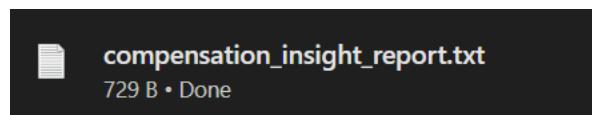
The screenshot shows a dark-themed user interface titled "Insight Selector". At the top, there is a search bar with the placeholder text "What type of insights would you like to view and analyse?". Below the search bar is a dropdown menu set to "Compensation". A button labeled "Generate Insights" with a star icon is located below the dropdown. The main content area is titled "Compensation Insights" and contains two numbered items:

1. 5212 high performers are underpaid. Consider salary adjustments.  
👉 Suggested Action: Ensure competitive compensation packages for high performers to avoid attrition.
2. 12885 employees with low salary and high satisfaction found.  
👉 Suggested Action: Review salary adjustments to ensure satisfaction remains high.

At the bottom, there is a section titled "Report Summary and Action Plan" with a "Download Full Insight & Action Plan Report" button.

*Figure 252: Interface of Compensation Insights*

The figure above displays the Compensation Insights generated from the current employee data. This analysis focuses on compensation related factors such as identifying high-performing employees who are underpaid and employees with low salaries but high satisfaction. Based on this, the system recommends adjusting compensation packages for high performers to prevent potential attrition. Additionally, for employees with low pay but high satisfaction, it suggests reviewing their salary adjustments to ensure their continued satisfaction. The report summary and action plan below can be shared with executives such as CEO for further review and informed decision-making.



*Figure 253: Downloaded Compensation Insight Report*

```

Employee Compensation Insights Report
Generated on: 2025-07-18 14:37:28

--- INSIGHTS ---
1. 5212 high performers are underpaid. Consider salary adjustments.
   ➡ Suggested Action: Ensure competitive compensation packages for high performers to avoid attrition.
2. 12885 employees with low salary and high satisfaction found.
   ➡ Suggested Action: Review salary adjustments to ensure satisfaction remains high.

--- ACTION PLAN ---
Sample Compensation Action Plan:
1. Benchmark current salary against market rates
2. Identify underpaid high performers
3. Develop a transparent performance-based bonus system
4. Review compensation equity across roles and departments
5. Communicate compensation philosophy to employees

```

*Figure 254: Compensation Insights Report in text file*

After downloading the file, the Compensation Insights Report will display detailed analysis as shown in figure above that are also including the current date and time to reflect the latest available information. This report highlights the key findings and provides suggested actions to help HR and executives make informed decisions about compensation adjustments.

#### 5.4.3.2.4.2 Productivity Insights

The screenshot shows a dark-themed web application interface. At the top, there's a header titled "Insight Selector" with a magnifying glass icon. Below it is a dropdown menu labeled "Productivity". A button labeled "Generate Insights" is visible. The main content area is titled "Productivity Insights" and contains two numbered items:

- Employees working >25 overtime hours: 14222. Monitor for burnout.  
➡ Suggested Action: Consider workload adjustments or introducing wellness programs.
- Optimal performers: 18519 found. Foster leadership roles.  
➡ Suggested Action: Nurture these employees for future leadership roles or mentorship programs.

At the bottom, there's a section titled "Report Summary and Action Plan" with a "Download Full Insight & Action Plan Report" button.

*Figure 255: Interface of Productivity Insights*

The figure above shows the Productivity Insights derived from the current employee data. This analysis identifies employees who are working more than 25 overtime hours and was highlighting potential risks of burnout. It also highlights optimal performers who have the potential to take on leadership roles. The suggested actions based on these insights include adjusting workloads for employees working excessive overtime to mitigate burnout. The report summary and action plan below can be shared with executives for strategic decision-making regarding employee productivity.

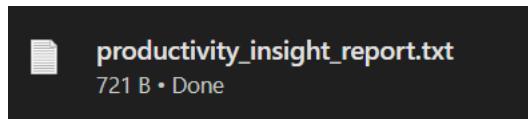


Figure 256 :Downloaded Productivity Insight Report

```
Employee Productivity Insights Report
Generated on: 2025-07-18 14:38:53

--- INSIGHTS ---
1. Employees working >25 overtime hours: 14222. Monitor for burnout.
   ➡ Suggested Action: Consider workload adjustments or introducing wellness programs.
2. Optimal performers: 18519 found. Foster leadership roles.
   ➡ Suggested Action: Nurture these employees for future leadership roles or mentorship programs.

--- ACTION PLAN ---
Sample Productivity Action Plan:
1. Identify teams with overtime >50 hrs
2. Introduce flexible work schedules or task rotation
3. Launch wellness or fatigue-monitoring initiatives
4. Monitor impact of training programs on performance
5. Adjust workloads for balance and sustainability
```

Figure 257: Productivity Insights Report in text file

After downloading the file, the Productivity Insights Report will display detailed analysis as shown in figure above that are also including the current date and time to reflect the latest available information. This report highlights the key findings and provides suggested actions to help HR and executives make informed decisions about compensation adjustments.

#### 5.4.3.2.4.3 Retention Insights

The screenshot shows a dark-themed user interface titled "Insight Selector". At the top, a search bar asks "What type of insights would you like to view and analyse?" with a help icon ( ⓘ ) and a dropdown menu set to "Retention". Below the search bar is a button labeled "Generate Insights". The main section is titled "Retention Insights" and contains two numbered items:

- 1. High-performing employees: 18519 found. Reward and retain them.**
  - Suggested Action:** Reward high performers with promotions, incentives, or leadership opportunities.
- 2. Employees with low satisfaction: 39614 need immediate attention.**
  - Suggested Action:** Schedule 1:1 meetings, understand their concerns, and offer support.

At the bottom, there is a "Report Summary and Action Plan" section with a "Download Full Insight & Action Plan Report" button.

Figure 258: Interface of Retention Insights

The figure above presents the **Retention Insights** derived from the current employee data. This analysis identifies high-performing employees who need to be rewarded and retained as well as employees with low satisfaction levels who need immediate attention. The system recommends rewarding high performers with promotions, incentives or leadership opportunities to ensure their retention. For employees with low satisfaction, the suggested action is to schedule 1:1 meeting to understand their concerns and offer support improve their satisfaction and reduce turnover. The report summary and action plan below provide a comprehensive overview for executives such as CEO to guide retention strategies and actions.

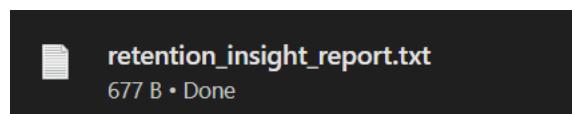


Figure 259: Downloaded Retention Insight Report

```

Employee Retention Insights Report
Generated on: 2025-07-18 14:39:38

--- INSIGHTS ---
1. High-performing employees: 18519 found. Reward and retain them.
   ➡ Suggested Action: Reward high performers with promotions, incentives, or leadership opportunities.
2. Employees with low satisfaction: 39614 need immediate attention.
   ➡ Suggested Action: Schedule 1:1 meetings, understand their concerns, and offer support.

--- ACTION PLAN ---
Sample Retention Action Plan:
1. Identify at-risk employees (churn risk >40%)
2. Schedule stay interviews within 2 weeks
3. Create personalized development plans
4. Review compensation benchmarks
5. Establish mentorship pairings

```

*Figure 260: Retention Insights Report in text file*

After downloading the file, the Retention Insights Report will display detailed analysis as shown in figure above that are also including the current date and time to reflect the latest available information. This report highlights the key findings and provides suggested actions to help HR and executives make informed decisions about compensation adjustments.

#### 5.4.3.2.4.4 Engagement Insights

The screenshot shows a dark-themed user interface for engagement insights. At the top, there's a header "Insight Selector" with a magnifying glass icon. Below it is a dropdown menu set to "Engagement". A prominent red button labeled "Generate Insights" is centered. The main content area is titled "Engagement Insights" and contains two numbered sections: 1. Highly engaged employees: 4436 found. Leverage them for mentorship roles. (Suggested Action: Leverage high-engagement employees for mentoring new hires or driving culture initiatives.) and 2. Low engagement risk: 34046 employees with both low satisfaction and performance. (Suggested Action: Address underlying issues, including performance coaching, or reassignment to more suitable roles.). At the bottom, there's a section titled "Report Summary and Action Plan" with a "Download Full Insight & Action Plan Report" button.

*Figure 261: Interface of Engagement Insights*

The figure above displays the **Engagement Insights** generated from the current employee data. This analysis focuses on employee engagement levels by identifying highly engaged employees who can be leveraged for mentorship roles as well as those with low engagement risks, exhibiting both low satisfaction and poor performance. The system suggests using engaged employees for leadership roles or mentorship programs to foster a positive work culture. The summary report and action plan below offer a detailed view for executives to make informed decisions on enhancing employee engagement and organizational culture.

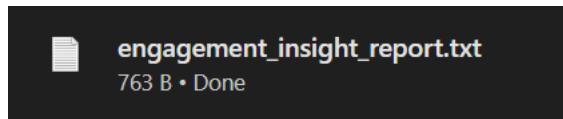


Figure 262: Downloaded Engagement Insight Report

```
Employee Engagement Insights Report
Generated on: 2025-07-18 14:40:23

--- INSIGHTS ---
1. Highly engaged employees: 4436 found. Leverage them for mentorship roles.
   ➔ Suggested Action: Leverage high-engagement employees for mentoring new hires or driving culture initiatives.
2. Low engagement risk: 34046 employees with both low satisfaction and performance.
   ➔ Suggested Action: Address underlying issues, including performance coaching, or reassignment to more suitable roles.

--- ACTION PLAN ---
Sample Engagement Action Plan:
1. Conduct anonymous engagement surveys
2. Launch mentorship or buddy programs
3. Recognize and reward contributions regularly
4. Organize quarterly engagement townhalls
5. Support career growth and learning pathways
```

Figure 263: Engagement Insights Report in text file

After downloading the file, the Engagement Insights Report will display detailed analysis as shown in figure above that are also including the current date and time to reflect the latest available information. This report highlights the key findings and provides suggested actions to help HR and executives make informed decisions about compensation adjustments.

### 5.4.3.3 Talent Acquisition-Specific Functions

#### 5.4.3.3.1 Talent Dashboard Page

```

def talent_dashboard():
    if st.session_state.role != "Talent":
        st.warning("You don't have permission to access this page")
        return

    if 'df_deployment' not in st.session_state:
        st.warning("HR data not loaded. Please log in again.")
        return

    df_deployment = st.session_state.df_deployment

    # Convert date columns to datetime
    df_deployment['Hire_Date'] = pd.to_datetime(df_deployment['Hire_Date'], errors='coerce')
    df_deployment['Resignation_Date'] = pd.to_datetime(df_deployment['Resignation_Date'], errors='coerce')

    # Get quarter info
    today = pd.to_datetime('today')
    current_year = today.year
    current_quarter = (today.month - 1) // 3 + 1

    # Get previous quarters
    def get_previous_quarters(year, quarter, count):
        quarters = []
        for i in range(count):
            if quarter == 1:
                quarter = 4
                year -= 1
            else:
                quarter -= 1
            quarters.append((year, quarter))
        return quarters

    # Get last 2 quarters for comparison
    previous_quarters = get_previous_quarters(current_year, current_quarter, 2)
    q2_year, q2_quarter = previous_quarters[0]
    q1_year, q1_quarter = previous_quarters[1]

    # Get metrics
    q1 = get_quarterly_metrics(df_deployment, q1_year, q1_quarter)
    q2 = get_quarterly_metrics(df_deployment, q2_year, q2_quarter)

    # Calculate Average Tenure
    def calculate_avg_tenure(df, quarter_end_date):
        active_employees = df[(df['Hire_Date'] <= quarter_end_date) &
                               ((df['Resignation_Date'] > quarter_end_date) | (df['Resignation_Date'].isna()))]
        return active_employees['Years_At_Company'].mean() if not active_employees.empty else 0

    q1_avg_tenure = calculate_avg_tenure(df_deployment, q1['end'])
    q2_avg_tenure = calculate_avg_tenure(df_deployment, q2['end'])
    avg_tenure_change = q2_avg_tenure - q1_avg_tenure

```

Figure 264: Source Code of Talent Dashboard Page 1

```

# Three main metrics at the top
col1, col2, col3 = st.columns(3)

with col1:
    employee_button = display_employee_metrics(q1, q2)

with col2:
    st.metric(
        "Average Tenure",
        f"{q2_avg_tenure:.1f} years",
        delta=f"{avg_tenure_change:.1f} years from Q{q1['quarter']}",
        delta_color="normal" if avg_tenure_change >= 0 else "inverse"
    )
    tenure_button = st.button("View Tenure Details", key="tenure_details")

with col3:
    churn_button = display_churn_metrics(q1, q2)

# Display detailed sections based on button clicks
if employee_button:
    st.markdown("----")
    display_employee_details(df_deployment, q1, q2)

if churn_button:
    st.markdown("----")
    display_churn_details(df_deployment, current_year, current_quarter)

if tenure_button:
    st.markdown("----")
    st.subheader("Average Tenure Trend")

    # Get last 6 quarters
    last_six_quarters = get_previous_quarters(current_year, current_quarter, 6)
    quarters = []

    for year, quarter in last_six_quarters:
        quarter_data = get_quarterly_metrics(df_deployment, year, quarter)
        quarter_data['avg_tenure'] = calculate_avg_tenure(df_deployment, quarter_data['end'])
        quarters.append(quarter_data)

    # Sort chronologically
    quarters = sorted(quarters, key=lambda x: (x['year'], x['quarter']))

    trend_df = pd.DataFrame({
        'Quarter': [f"Q{q['quarter']} {q['year']}" for q in quarters],
        'Average Tenure': [q['avg_tenure'] for q in quarters],
        'Date': [q['end'] for q in quarters]
    })

```

Figure 265: Source Code of Talent Dashboard Page 2

```

fig = go.Figure()
fig.add_trace(go.Scatter(
    x=trend_df['Quarter'],
    y=trend_df['Average Tenure'],
    mode='lines+markers',
    name='Average Tenure',
    line=dict(color='skyblue', width=2)
))
fig.add_trace(go.Scatter(
    x=trend_df['Quarter'],
    y=[trend_df['Average Tenure'].mean() * len(trend_df)],
    mode='lines',
    name='Average',
    line=dict(color='red', width=2, dash='dot'),
    hoverlabel=dict(font=dict(color='black')),
    hoverinfo='y+name'
))
fig.update_layout(
    title='Average Tenure Trend (Last 6 Quarters)',
    xaxis_title='Quarter',
    yaxis_title='Tenure (years)',
    height=400,
    autosize=True,
    margin=dict(l=20, r=20, t=60, b=20),
    plot_bgcolor='rgba(0,0,0,0)',
    paper_bgcolor='rgba(0,0,0,0)',
    hovermode='x unified'
)
st.plotly_chart(fig, use_container_width=True)

# Quick navigation to hiring recommendations
st.markdown("---")
if st.button("Get Hiring Recommendations", key="go_to_recommendations"):
    st.session_state.current_page = "Hiring Recommendations"
    st.rerun()

```

*Figure 266: Source Code of Talent Dashboard Page 3*

The figure above shows the source codes which are defining for a Talent Acquisition Dashboard page specifically for the Talent Acquisition team in the HR management system. It provides key insights into employee metrics such as churn rate, average tenure and overall performance. The page displays the current and previous quarters' data which is including metrics like employee tenure and churn for the Talent team to track changes over time.

It also includes interactive elements such as buttons to view detailed employee and churn metrics and graph to show the trend of average tenure over the last six quarters. Additionally, the page has a quick navigation option for users to easily access sections like hiring recommendations. Overall, the dashboard is designed to help HR professionals make data-driven decisions regarding employee retention and hiring strategies.



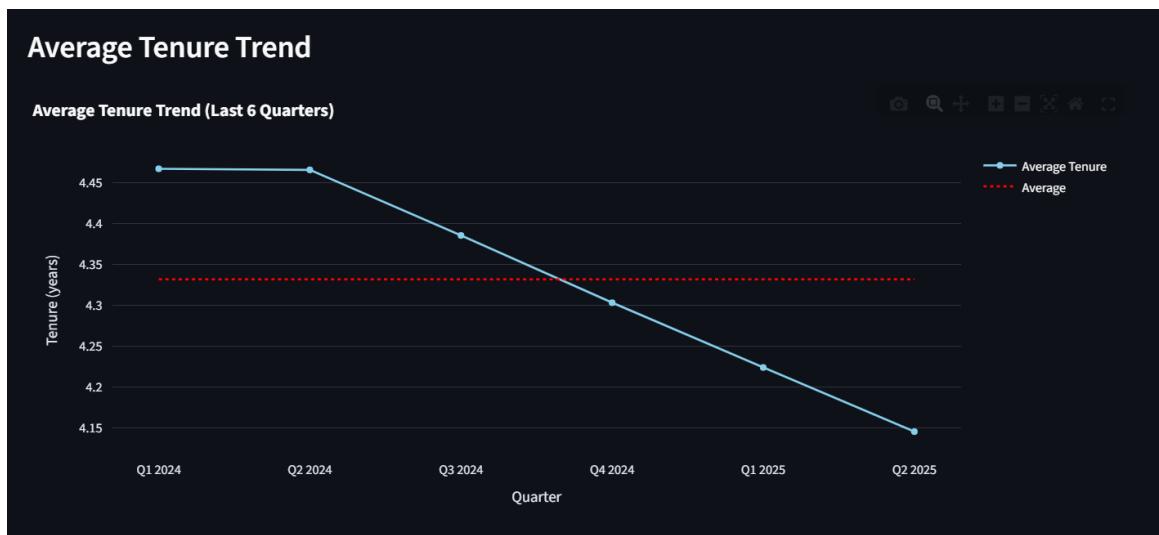
*Figure 267: Interface of Source Code Talent Dashboard*

The figure above shows the Talent Acquisition Dashboard which is displaying the last login date and time for easy reference. It also includes a data refresh feature that updates the dataset after uploading the latest CSV file to ensure timely data updates.

The Total Employees and Churn Rate have already been discussed in the HR section because the features are the same metrics in both areas. The "View Employee Details" at [Section 5.4.3.2.1.1](#) and "View Churn Details" at [Section 5.4.3.2.1.3](#). Both features provide valuable insights into employee turnover patterns which are helping Talent Acquisition teams identify at risk employees and develop effective retention strategies.

Additionally, the Average Tenure section on the Talent Acquisition Dashboard indicates the average number of years employees have stayed with the company. It is currently showing 4.1 years which is representing a 0.1-year decrease from the previous quarter. This data is based on the employee dataset and the trend is displayed with a green arrow for a slight increase and a red arrow for a decrease which help the talent acquisition to visualize employee retention patterns easily.

The "View Tenure Details" button provides a deeper dive into tenure information for allowing Talent Acquisition professionals to track the average length of employment. This helps them make data-driven decisions aimed at improving employee retention. These metrics help Talent Acquisition teams assess how well the organization is retaining talent and adjust hiring and retention strategies accordingly. The line chart visualization will be discussed in the next section.



*Figure 268: Interface of Average Tenure Trend 1*

After clicking the "View Tenure Details", the system will display the Average Tenure Trend chart. This chart shows the average employee tenure over the last six quarters. The line represents the average tenure for each quarter while the red dotted line shows the overall average across these quarters for a benchmark to evaluate the trend. As shown in the figure above, the average tenure has been gradually decreasing from 4.45 years in Q1 2024 to 4.15 years in Q2 2025.

For Talent Acquisition teams, this chart provides valuable insights into employee retention patterns. By analysing the downward trend in average tenure, they can identify potential issues with employee engagement or satisfaction. A decreasing tenure could indicate higher turnover and this may prompt the need for retention strategies such as improving employee engagement, career development opportunities or compensation adjustments.

By comparing the average tenure trend against the overall benchmark (red dotted line), Talent Acquisition can quickly identify if the organization is retaining employees at or above the expected average. If the trend continues downward, HR professionals can take proactive steps to adjust hiring, onboarding and retention strategies. For example, it may be necessary to focus on employee development programs or refine the company culture to better align with employee expectations.

In summary, this chart helps Talent Acquisition teams gain insights into how well the organization is retaining talent over time and they can use this data to make informed decisions to improve employee retention and overall workforce stability.

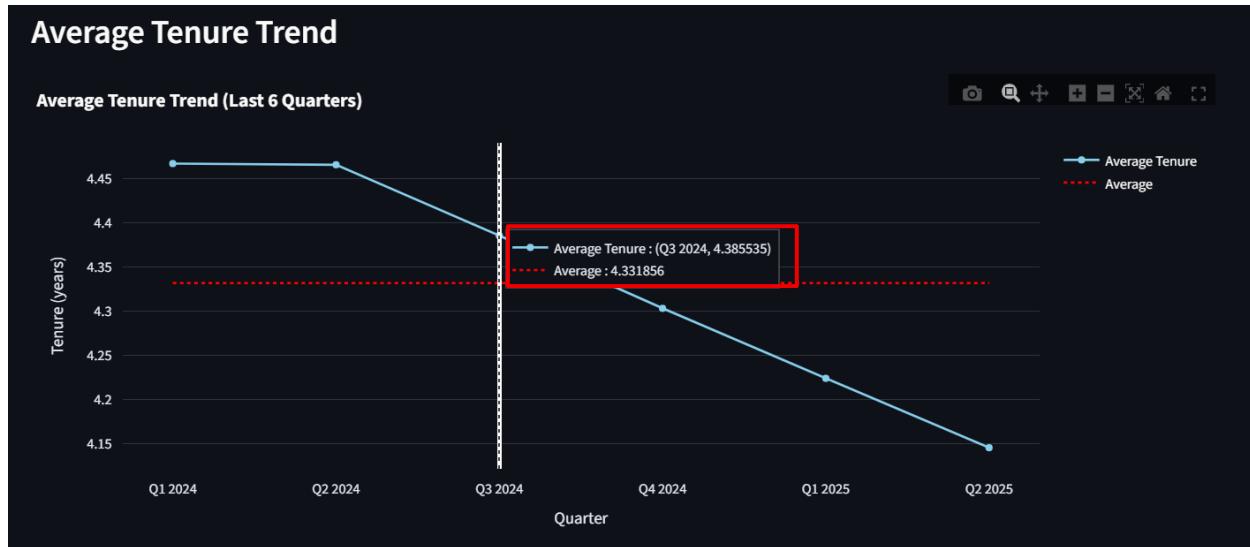


Figure 269: Interface of Average Tenure Trend 2

Additionally, when the mouse pointer hovers over the dotted blue line, it will display detailed data such as the average tenure for each quarter and the average years. This interactive feature provides a quick view of the specific data points for Talent Acquisition teams to easily track the average tenure for each quarter and compare it to the overall average.

### 5.4.3.3.2 Hiring Recommendations Page

```

def hiring_recommendations():
    if st.session_state.role != "Talent":
        st.warning("You don't have permission to access this page")
        return

    st.title("Hiring Recommendations")

    if 'df_deployment' not in st.session_state:
        st.warning("HR data not loaded. Please log in again.")
        return

    df = st.session_state.df_deployment
    st.subheader("Education and Experience Distribution of Top Performers")

    successful_employees = df[
        (df['Performance_Score'] >= 4) &
        (df['Employee_Satisfaction_Score'] >= 4) &
        (df['Resigned'] == False)
    ]

    if not successful_employees.empty:
        col1, col2 = st.columns(2)

        with col1:
            education_dist = successful_employees['Education_Level'].value_counts()
            fig1 = px.pie(
                education_dist,
                names=education_dist.index,
                values=education_dist.values,
                title='Education Distribution'
            )
            st.plotly_chart(fig1, use_container_width=True, key=f"education_chart_{str(np.random.randint(100000))}")

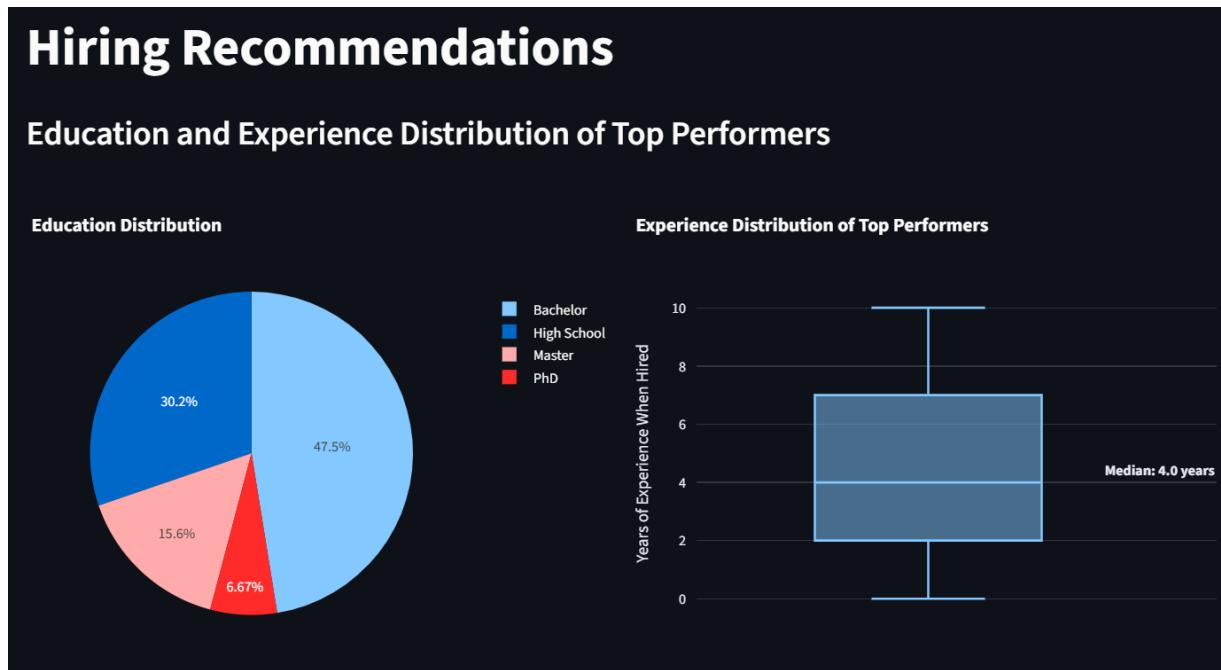
        with col2:
            fig2 = px.box(
                successful_employees,
                y='Years_At_Company',
                title='Experience Distribution of Top Performers',
                labels={'Years_At_Company': 'Years of Experience When Hired'}
            )
            fig2.update_traces(
                hoverinfo='y',
                hovertemplate='<b>%{y:.1f} years</b>',
                boxpoints=False
            )
            q1, med, q3 = np.percentile(successful_employees['Years_At_Company'], [25, 50, 75])
            fig2.add_annotation(
                x=0.5, y=med,
                text=f"<b>Median: {med:.1f} years</b>",
                showarrow=False,
                yshift=10
            )
            st.plotly_chart(fig2, use_container_width=True, key=f"experience_chart_{str(np.random.randint(100000))}")

```

*Figure 270: Source Code of Hiring Recommendation Page*

The Hiring Recommendations page in the Talent Acquisition dashboard helps HR professionals identify key candidate profiles by analyzing the characteristics of top-performing employees. It filters employees based on high performance and satisfaction scores while excluding those who have resigned. The page visualizes the education distribution and experience distribution of successful employees using pie charts and box plots. These visualizations provide insights into the common attributes of top performers for Talent Acquisition teams to make informed and data-driven decisions when shaping future hiring and recruitment strategies.

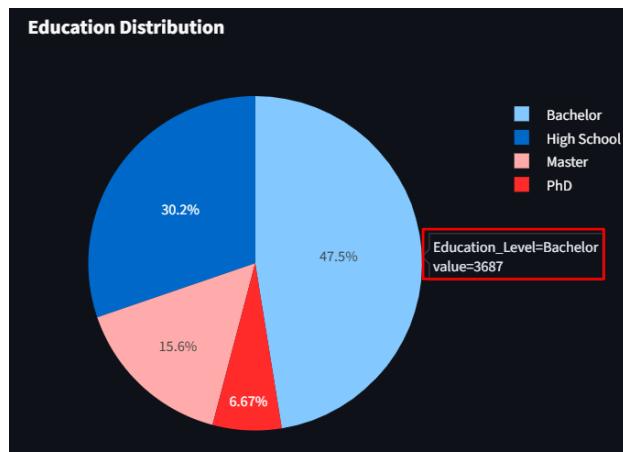
#### 5.4.3.3.2.1 Education and Experience Distribution of Top Performers



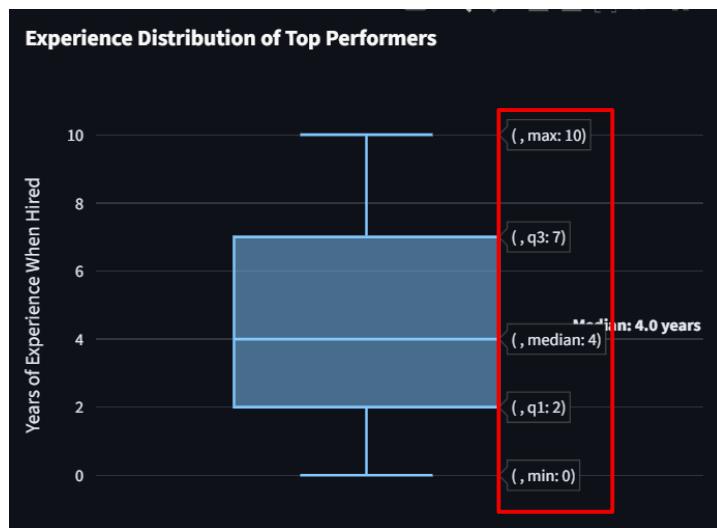
*Figure 271: Interface for Hiring Recommendations*

The figure above shows the Education and Experience Distribution of Top Performers on the Talent Acquisition dashboard. The pie chart displays the education distribution of successful employees with categories for Bachelor's, High School, Master's and PhD. It shows the percentage of each education level to help Talent Acquisition teams easily view which educational background is most common among top performers. For example, 47.5% of successful employees have a Bachelor's degree while 6.67% have a PhD. This information helps HR teams identify the most common educational qualifications for successful hires and adjust their recruitment strategies accordingly.

The box plot visualizes the years of experience of successful employees when they were hired which is showing the distribution of experience across employees. The median experience is 4.0 years and the plot also provides the maximum (10 years) and minimum (0 years) experience along with quartile values. This gives Talent Acquisition teams an understanding of the typical experience level of successful employees for helping them target candidates with similar profiles in future hiring.



*Figure 272: Pie Chart on Education Distribution for Hiring Recommendations*



*Figure 273: Boxplot on Experience Distribution for Hiring Recommendations*

When the mouse pointer hovers over any part of the chart, it will display detailed information such as the exact percentage or value for the data point. This feature allows Talent Acquisition professionals to quickly access more specific insights. Additionally, both the pie chart and the box plot are interactive where they can be zoomed in and viewed in full-screen mode for a closer look at the details.

For the box plot, hovering over the chart will show key statistical details like the maximum, median, quartiles and minimum values to provide a clearer understanding of the data distribution. This interactivity makes the analysis process more intuitive and accessible for Talent Acquisition teams.

### 5.4.3.3.2 Priority Hiring Recommendations

```

# Hiring priority recommendations
st.markdown("----")
st.subheader("Priority Hiring Recommendations")

# Get departments with highest churn
resigned = df[df['Resigned'] == True]
churn_by_dept = resigned.groupby('Department').size().reset_index(name='Resignations')
total_by_dept = df.groupby('Department').size().reset_index(name='Total')
churn_data = pd.merge(churn_by_dept, total_by_dept, on='Department')
churn_data['Churn Rate'] = (churn_data['Resignations'] / churn_data['Total']) * 100

high_churn_depts = churn_data.sort_values('Churn Rate', ascending=False).head(3)

for _, row in high_churn_depts.iterrows():
    with st.expander(f'{row["Department"]} Department (Churn Rate: {row["Churn Rate"]:.1f}%)'):
        st.write(f"**Recommended Actions:**")
        st.write("- Prioritize hiring for {row['Department']} roles")
        st.write("- Target candidates with 2+ years experience in similar roles")
        st.write("- Look for evidence of stability in previous positions")
        st.write("- Consider offering competitive benefits for these roles")

# Downloadable report
st.markdown("----")
st.subheader("Generate Hiring Strategy Report")

report_text = f"""
Hiring Strategy Recommendations
Generated on: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}

--- Priority Departments ---
{high_churn_depts[['Department', 'Churn Rate']].to_string(index=False)}

--- Ideal Candidate Profile ---
Education: {education_dist.index[0]} ({(education_dist.values[0]/len(successful_employees)*100):.1f}%)  

Average Experience When Hired: {successful_employees['Years_At_Company'].mean():.1f} years

--- Recommended Actions ---
1. Prioritize hiring for {high_churn_depts.iloc[0]['Department']}
2. Target candidates with 2+ years experience
3. Implement skills testing for technical roles
"""

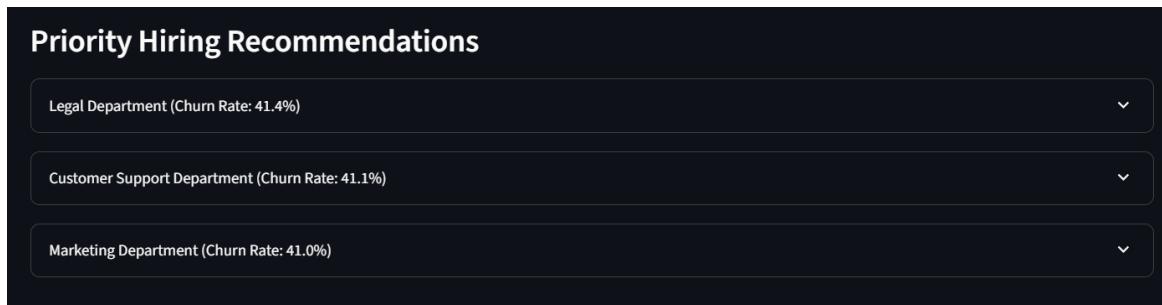
st.download_button(
    label="Download Hiring Strategy Report",
    data=report_text,
    file_name="hiring_recommendations.txt",
    mime="text/plain",
    key=f"download_{str(np.random.randint(100000))}"
)

```

*Figure 274: Source Code of Hiring Priority Recommendations*

The Hiring Priority Recommendations section on the Talent Acquisition dashboard helps HR teams identify departments with the highest employee turnover and provides actionable insights for improving recruitment efforts. It will filter employees who have resigned and calculate the churn rate for each department by comparing the number of resignations to the total employees. Then, the top three departments with the highest churn rates are highlighted and provide recommendations. These recommendations are displayed in an expander format for Talent Acquisition teams to access detailed information.

Additionally, a Hiring Strategy Report can be generated and downloaded. The report includes a summary of the priority departments, the education distribution and average experience of top performers and key hiring recommendations. This feature allows HR professionals to have a summary report document that consolidates all insights and strategies for helping them make informed decisions on recruitment and retention efforts.



*Figure 275: Interface of Priority Hiring Recommendations 1*

The figure above displays the Priority Hiring Recommendations for the top three departments with the highest churn rates with their respective percentages. The Legal Department, Customer Support Department and Marketing Department each have a churn rate above 40% with their respective percentages. This provides Talent Acquisition teams with an overview of the departments experiencing the highest turnover which are allowing them to prioritize hiring efforts in these areas. The detailed recommendations for each department can be accessed to help HR professionals act on critical hiring needs to improve retention and stability within these departments.

**Priority Hiring Recommendations**

**Legal Department (Churn Rate: 41.4%)**

**Recommended Actions:**

- Prioritize hiring for Legal roles
- Target candidates with 2+ years experience in similar roles
- Look for evidence of stability in previous positions
- Consider offering competitive benefits for these roles

**Customer Support Department (Churn Rate: 41.1%)**

**Recommended Actions:**

- Prioritize hiring for Customer Support roles
- Target candidates with 2+ years experience in similar roles
- Look for evidence of stability in previous positions
- Consider offering competitive benefits for these roles

**Marketing Department (Churn Rate: 41.0%)**

**Recommended Actions:**

- Prioritize hiring for Marketing roles
- Target candidates with 2+ years experience in similar roles
- Look for evidence of stability in previous positions
- Consider offering competitive benefits for these roles

*Figure 276: Interface of Priority Hiring Recommendations 2*

Once clicked, the expander reveals each department's recommended actions. For example, the Legal Department with a 41.4% churn rate is recommended on prioritizing hiring for legal roles with targeting candidates with 2+ years of experience in similar roles, looking for evidence of stability in previous positions and considering offering competitive benefits. Based on the recommendations, the Talent Acquisition team can gain valuable insights into which departments need immediate hiring attention and what candidate qualities to focus on. This helps the team tailor their recruitment strategies, target the right candidates and solve turnover issues more effectively by ensuring that the department's needs are met with the best-fit talent.

**Generate Hiring Strategy Report**

[Download Hiring Strategy Report](#)

*Figure 277: Interface of Generate Hiring Strategy Report*

After reviewing the recommended actions and churn rates, the summary report can be downloaded for a quick overview of all the insights. This report has included the key findings such as departments with the highest churn rates, ideal candidate profiles and suggested actions for recruitment and retention. By downloading the report, Talent Acquisition teams can easily access all the information needed to make informed and strategic decisions for improving hiring and addressing turnover.

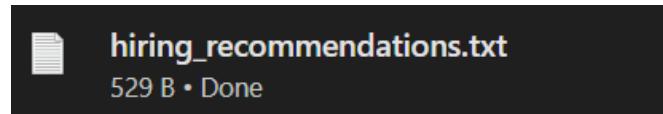


Figure 278: Downloaded Hiring Recommendations Report Summary

```
Hiring Strategy Recommendations
Generated on: 2025-07-19 05:05:57

--- Priority Departments ---
  Department    Churn Rate
    Legal        41.430649
Customer Support 41.094596
  Marketing     40.981501

--- Education and Experience Distribution of Top Performers ---
Education: Bachelor (47.5%)
Average Experience When Hired: 4.2 years

--- Recommended Actions ---
1. Prioritize hiring for Legal
2. Target candidates with 2+ years experience
3. Implement skills testing for technical roles
```

Figure 279: Hiring Recommendations Report Summary in Text File

After downloading the file, the Hiring Strategy Report will display a detailed analysis and the current date and time to reflect the most up-to-date information. The report provides insights into the key findings and recommended actions to help Talent Acquisition teams prioritize recruitment efforts. It highlights the departments with the highest churn rates, identifies the ideal candidate profiles and suggests strategic actions for improving hiring and retention. By offering data-driven recommendations, this report enables Talent Acquisition professionals to make informed decisions to enhance workforce stability and optimize recruitment strategies.

#### 5.4.4 Conclusion of Model Deployment

In conclusion, the deployment of the Employee Churn Analytics Dashboard System marks a significant step in applying machine learning to predict employee churn and optimize workforce productivity. By using the XGBoost model which delivers the best overall results in identifying employees at risk of resignation, the system equips HR professionals with the ability to make data-driven predictions. HR users can analyze the churn risk across various employee demographics, performance metrics and satisfaction levels to take timely actions to retain valuable talent.

The system helps HR professionals to predict employee churn by providing them with predictive insights into employee churn, categorized into low, moderate and high-risk groups. This predictive capability allows HR to proactively intervene before an employee decides to leave by identifying underlying factors such as job satisfaction, performance issues or work-life balance concerns. Through the interactive dashboards, HR can easily visualize employee trends, churn rates and other key metrics that are important for optimizing workforce productivity.

The inclusion of SHAP values ensures that HR can interpret the model's predictions with transparency of showing how each factor influences the churn risk. This enables HR professionals to understand the reason behind the predictions for allowing them to focus on the most critical factors that drive turnover. Moreover, the system's real-time data refresh and dynamic functionalities help HR stay on top of employee trends, track performance and adjust retention strategies based on the latest insights.

For Talent Acquisition, the system provides additional functionality such as identifying high-turnover departments and recommending targeted hiring strategies. These insights contribute to optimizing the hiring process to ensure that new recruits are better aligned with the company's needs to reduce future churn and improve overall workforce productivity.

In conclusion, the Employee Churn Analytics Dashboard not only helps HR predict and manage employee churn but also optimizes workforce productivity by providing actionable insights. By combining machine learning with a user-friendly interface, the system enhances HR's ability to make informed decisions that support employee retention, satisfaction and overall organizational success.

## 5.5 Summary

In conclusion, Chapter 5 presents the results of the evaluation and implementation of the employee churn prediction models and the deployment of the Employee Churn Analytics Dashboard System. The chapter is structured into multiple sections which starts with an introduction and followed by the evaluation of various machine learning models such as Random Forest, XGBoost, Gradient Boosting and Support Vector Machine.

The evaluation of the models focused on key performance metrics such as accuracy, precision, recall, F1-score and ROC AUC score. Among the models tested, XGBoost was selected as the best model for predicting employee churn due to its higher recall rate for identifying employees at risk of resignation with 88% result compared to Gradient Boosting's 87%. Additionally, XGBoost demonstrated superior generalization, minimal overfitting and faster convergence to make it the most effective model for real-time applications. Additionally, the learning curve analysis supports XGBoost's superior efficiency which has demonstrated that it requires less data to achieve peak performance.

In terms of feature importance, XGBoost effectively handled large datasets and identified the key variables influencing employee churn. This was followed by the deployment of the predictive model into the Employee Churn Analytics Dashboard System which is customized for HR professionals. The dashboard allows HR and talent acquisition teams to gain actionable insights into employee churn such as identifying employees at low, moderate or high risk of leaving the company. The system's functions include employee metrics, churn predictions, performance tracking and retention insights for supporting HR in making informed and data-driven decisions.

The results confirm that XGBoost is the most suitable model for this project by offering superior performance in predicting employee churn and optimizing workforce productivity. The model's deployment into a user-friendly dashboard enhances HR's ability to make timely interventions to improve retention and overall employee satisfaction.

## Chapter 6: Conclusion

### 6.1 Critical Evaluation

#### 6.1.1 The achievement of the overall of the project

This project successfully achieved its primary goal which aim to design and implement a machine learning-based system capable of predicting employee churn and optimizing workforce productivity. The motivation behind the project stemmed from the critical challenge of high employee attrition on disrupting organizational workflows, increases recruitment and training costs which is leading to the loss of valuable institutional knowledge. The developed analytics system aims to support Human Resource (HR) departments with a proactive approach which is enabling them to detect early signs of dissatisfaction or disengagement and take preventive measures before the resignations occur.

The project followed by the CRISP-DM methodology to ensure a structured and systematic process throughout. It began with a deep understanding of the business problem on the employee churn and its far-reaching impact on organizations. This helps in understanding formed the foundation for data collection, preprocessing and model development.

The data analysis phase involved exploring key features such as employee satisfaction, performance scores, overtime hours and other relevant factors. The dataset was meticulously cleaned with missing values handled, categorical variables encoded, numerical features scaled and feature selection applied. Additionally, class imbalance was solved through SMOTE resampling techniques to ensure that the model received balanced data for better prediction accuracy. This comprehensive preprocessing ensured that the machine learning models were trained on reliable, high-quality data to result in a more accurate prediction.

There are various machine learning algorithms were tested which is including Random Forest, XGBoost , Gradient Boosting and Support Vector Machine (SVM). After evaluating performance using metrics such as precision, recall, F1-score and ROC AUC, XGBoost proved as the best-performing model. It showed superior predictive accuracy in handling class imbalance which is a common issue in churn prediction in nowadays where the number of employees leaving the organization is much smaller than those remaining. The model's ability to generalize well to unseen data confirmed its effectiveness for real-world HR applications.

To make the model accessible to HR professionals without technical expertise, an interactive dashboard was developed as the user interface for the system. This dashboard offers a dynamic tool for visualizing and tracking various employee metrics with including churn risk levels, employee satisfaction trends, absenteeism, overtime hours and other productivity indicators. The dashboard also allows HR to explore relationships between key factors such as how satisfaction or overtime work influences employee churn with providing a deeper understanding of the data. HR also can predict employee churn in the analytics system by inputting the required factors. Based on these predictions, employees are categorized into low, moderate or high-risk groups. Then, the system provides tailored recommendations and insights for each risk level for HR to prioritize retention efforts more effectively. Additionally, dynamic filtering features enable users to drill down into employee data and gaining valuable insights into specific departments or individual.

The dashboard includes an actionable insights section and gives recommendations based on the current data such as implementing flexible work arrangements or retention programs. Additionally, HR professionals can generate summary reports for executive reviews to support data-driven decisions regarding workforce management. A key feature of the system is the interactive data update system which allows HR to manually import new employee data through CSV file and refresh the dashboard to provide up-to-date insights. This ensures that the system displays the latest information for HR to act promptly on emerging trends.

In conclusion, the project successfully delivered a complete machine learning solution that integrates predictive analytics with intuitive visualization tools to solve the critical issue of employee churn. The system combines technical expertise with practical application for offering HR departments a powerful tool to reduce turnover, enhance workforce productivity and improve employee engagement. Its deployment into real-world HR environments demonstrates its potential for interactive data updates and its readiness for widespread application in organizations. Furthermore, the project aligns with Sustainable Development Goal (SDG) 8: Decent Work and Economic Growth, promoting fair and stable employment practices. By applying predictive analytics, the system helps organizations enhance workforce productivity, reduce turnover and create a more sustainable working environment. This leads to better inclusion, engagement and job security for achieving SDG 8.

### **6.1.2 Contribution of the project towards community/ industries**

This project provides a practical contribution to both the community and industry by addressing a critical real-world challenge in nowadays which is employee attrition and declining workforce productivity. High employee turnover disrupts organizational workflows, increases recruitment and training costs, lowers employee morale and often results in the loss of valuable institutional knowledge. It is because predicting employee churn is no longer an optional practice but a necessity for modern businesses that aim to remain competitive, agile and stable.

The analytics system developed in this project offers a scalable and data-driven solution to this employee churn issue by applying machine learning techniques with the XGBoost model that was selected as the best-performing algorithm based on its high predictive accuracy and generalization capabilities. By analysing employee records dataset from 2013 to 2024, the model identifies key patterns and risk factors such as job satisfaction, performance scores, overtime hours and team size to predict the employee resigning. These predictions empower HR teams to take timely and targeted actions before resignations have occurred such as implementing retention programmes, offering flexible work arrangements or adjusting workloads to improve employee satisfaction and engagement.

In a real-world business environment, this analytics system would offer immediate and measurable benefits. Human Resource departments could shift from reactive to proactive workforce management by reducing the time and effort spent addressing turnover after it has occurred. The integrated interactive dashboard enables HR professionals to access real-time visualisations of employee churn risk levels, productivity metrics, satisfaction trends, absenteeism and overtime analysis. With this tool, HR managers can quickly identify at-risk employees, monitor departmental health index and prioritise interventions based on churn risk categories such as low, moderate and high. Furthermore, talent acquisition teams can use these insights to improve recruitment strategies by targeting candidates who align better with long-term retention patterns. For business leaders, this system supports informed decision-making related to workforce planning, resource allocation and HR budgeting.

Beyond its immediate operational impact, the system also contributes to long-term organisational success. By reducing voluntary turnover, companies can save significantly on recruitment and training costs while maintaining performance continuity and preserving team dynamics.

Additionally, improved productivity monitoring facilitates a better workload distribution and enables the early detection of burnout. This also supports a more accurate performance evaluations, which directly contribute to enhancing employee well-being. These improvements not only boost employee engagement but also strengthen the internal culture within the organization. As a result, the company is better positioned to retain top talent and this is essential for fostering long-term and sustainable business growth.

This system also supports the United Nations Sustainable Development Goal 8 (Decent Work and Economic Growth) by promoting fair, inclusive and stable employment practices. From an academic and research perspective, this project serves as a concrete example of how machine learning can be ethically and effectively applied in the field of HR analytics. It highlights the potential of data science to solve complex organisational problems while promoting responsible AI usage and supporting future research into predictive workforce management.

### **6.1.3 Strength of the Project**

This project stands out for its combination of a strong data foundation, insightful analysis and practical applicability in HR decision-making. One of the project's key strengths is its use of a large and diverse dataset with 1,000,000 rows. This expansive dataset allows the system to analyse for a wide range of employee characteristics and behaviours which helps in improving the accuracy and generalizability of churn predictions. By incorporating more data, the system enhances its ability to predict employee turnover across various sectors and roles and is making it more adaptable for real-world business environments.

The project also excels in analysing the relationships between various factors such as education level, monthly salary, overtime and health index which are influence on employee resignation. By understanding these relationships enables HR departments to identify key areas that contribute to churn risk. This analysis goes beyond simple prediction as it provides HR with actionable insights for targeted retention strategies. For example, if an employee's overtime hours or low satisfaction levels are linked to higher churn risk, HR can implement specific measures such as adjusting workloads or offering more flexible working hours to prevent employee turnover.

In terms of machine learning model selection, the project was compared several models with using Random Forest, XGBoost, Gradient Boosting and Support Vector Machine (SVM) to identify the

best suited for predicting employee churn. XGBoost emerged as the best model due to its ability to handle class imbalance, high predictive accuracy and efficiency in processing large datasets. The comparison with other models like Random Forest and Gradient Boosting highlights how XGBoost outperforms them in terms of recall and ROC AUC to ensure that the system is more effective at identifying at-risk employees and offering accurate predictive capabilities.

Another key strength is the ability to provide actionable insights for HR in critical areas such as productivity, compensation, retention and employee engagement. By using employee churn analytics system, the system can automatically identify at-risk employees and track satisfaction trends based on the latest data file that have captured. The system also allows HR to take proactive action by offering recommendations tailored to specific employee risk levels. These actionable insights such as identifying employees who may require more engagement or adjustments in their compensation package can help HR teams to make data-driven decisions.

Additionally, the system's ability to refresh the data in real-time by importing new dataset files and updating predictions. This ensures that HR teams are always working with the most up-to-date information. It is important for making timely decisions as employee turnover patterns may shift quickly. With real-time updates and interactive visualizations, HR professionals can instantly track trends, monitor at-risk employees and take immediate action to reduce the employee loss.

Besides, the system's ability to generate summary reports based on analysis results provides HR departments with easy-to-read and concise overviews that are essential for executive decision-making. These reports summarize critical insights such as "Which departments have higher turnover rates?" and "Which factors are most correlated with employee dissatisfaction?". These reports allow HR leaders to prioritize efforts and resources based on data-backed insights.

Overall, the project offers a scalable and data-driven solution for HR departments to manage employee churn effectively. By combining machine learning with dynamic updates and interactive dashboards, HR professionals are equipped with a tool that predicts churn and provides actionable insights for retention. These features are valuable in the real world where HR departments are under pressure to make quick decisions that reduce turnover, enhance employee satisfaction and improve organizational performance. All the features ensure HR teams can take timely and

proactive steps to optimize their workforce. This is making the system serves as an essential tool for any organization aiming to reduce churn and maintain a productive and engaged workforce.

## 6.2 Limitations of the Project

Although the project provides actionable insights in predicting employee churn and offers valuable support to HR departments, there are several limitations that should be acknowledged. One of the key limitations is related to hardware constraints which is the memory and processing speed. Due to these limitations, only 50% of the dataset could be used when tuning the Support Vector Machine (SVM) model. This inability to process large datasets efficiently could potentially hinder the system's performance when applied to real-world business environments where HR departments deal with large amounts of data. In such cases, the system might struggle to scale and may be impacting its overall effectiveness. To overcome this, a more powerful hardware setup or cloud-based solutions would be required to support larger datasets without compromising performance.

Another limitation arises from the use of a static dataset sourced from Kaggle which is only contains historical data up to 2024. Although this data is useful for training the model, but it doesn't reflect the most current HR activities or employee behaviour which is making it less useful for making decisions based on today's situation. In a real-world scenario, businesses need systems that automatically integrate new data to continuously update predictions. The current setup requires HR professionals to manually upload updated CSV files to refresh the system and cause the model is unable to provide real-time insights. This reliance on manual data updates limits the system's adaptability in fast-changing environments where employee trends and behaviours evolve quickly.

The system also lacks the ability to generate personalized intervention plans for every individual employee. While the model categorizes employees into low, moderate and high-risk groups, it does not offer specific recommendations for each individual based on their unique circumstances. In real-world HR settings, personalized insights are important for effectively managing employee churn. Without this level of detail, HR professionals may miss out on opportunities to customize retention strategies to individual needs and reduce the system's overall impact on employee retention.

Additionally, ethical concerns surrounding employee data privacy present another limitation. Although anonymized public datasets were used in this project, real-world implementation would require compliance with strict privacy regulations such as the General Data Protection Regulation (GDPR). This regulation ensures that organizations handle personal data responsibly, requiring explicit consent, secure data storage and clear communication about data usage. In real-world applications, the system would need to adhere to these standards to protect employee privacy and avoid legal or reputational risks.

Despite these limitations, the project demonstrates the potential of machine learning in predicting employee churn and offering actionable insights for HR departments. By solving these limitations through hardware improvements, real-time data integration and personalized intervention plans could enhance the system's applicability and overall effectiveness in real-world business environments.

### **6.3 Suggestion for Future Recommendations**

To enhance the overall functionality and effectiveness of the system, there are several key improvements are recommended.

First, real-time data integration should be implemented to move beyond manual CSV uploads. This can be achieved by incorporating API-based data pipelines that automatically update the model with the latest employee data. By connecting the system to existing HR platforms such as payroll systems and employee engagement tools, HR professionals will have access to continuous and have a real-time insight that better reflect current trends and behaviours. This will ensure more accurate predictions and allow for timely interventions.

Secondly, the system should create personalized intervention plans for each employee. While the model currently provides generalized recommendations based on broad categories of churn risk, a more individualized approach would significantly enhance its impact. By incorporating factors like individual performance feedback, career aspirations and team dynamics, the system could provide HR with customized action plans. These plans may include recommendations for career development opportunities or offering flexible work arrangements for at-risk employees. This level of personalization would not only help reduce turnover more effectively but also improve employee satisfaction and retention.

Besides, the current system is limited by the hardware specifications especially for the processing speed and memory capacity of the local machine. During model tuning, this constraint prevented the use of the full dataset for SVM due to computationally intensive. To overcome this limitation, applying cloud-based platforms such as AWS and Microsoft Azure for data storage and processing would enable the system to handle larger datasets. These platforms offer scalable computing resources that would help to improve processing speed and allow for more efficient model tuning. By using distributed computing, the system could also better accommodate growing data needs and make it more suitable for large-scale applications in real-world HR environments.

Lastly, to ensure that the system remains relevant and accurate over time, automated model retraining should be introduced. The system should periodically retrain its models with the latest available data to account for changing employee behaviours and trends. Therefore, the system will maintain its predictive accuracy and adapt to new patterns to ensure it remains a reliable tool for HR teams. This feature would help to streamline the system's functionality by reducing the need for manual updates and is making it more efficient and user-friendly.

In conclusion, these recommendations aim to address the current limitations of the system while enhancing its applicability and effectiveness in real-world HR environments. By implementing these improvements, the system will provide more accurate and actionable insights for HR departments to proactively manage employee churn and optimize workforce productivity. Furthermore, these enhancements align with United Nations Sustainable Development Goal 8 (Decent Work and Economic Growth), which promotes inclusive and sustainable economic growth and decent work for all. By improving employee retention, satisfaction and engagement, the system contributes to a more stable and productive workforce which fosters a positive impact on both business performance and employee well-being.

## References

- Admin. (2023, September 24). The impact of talent management strategies on employee retention. International Journal of Science and Business (IJSB). [https://www.ijsab.com/volume-28-issue-1/6237?utm\\_source](https://www.ijsab.com/volume-28-issue-1/6237?utm_source)
- Akinode, L., & Bada, O. Employee attrition prediction using machine learning algorithms. In *Proceedings of the 3rd International Conference, The Federal Polytechnic, Ilaro* (Vol. 16). [https://www.researchgate.net/profile/Akinode-John-Lekan/publication/364322002\\_EMPLOYEE\\_ATTRITION\\_PREDICTION\\_USING\\_MACHINE\\_LEARNING\\_ALGORITHMS/links/6347f5a39cb4fe44f324df14/EMPLOYEE-ATTRITION-PREDICTION-USING-MACHINE-LEARNING-ALGORITHMS.pdf](https://www.researchgate.net/profile/Akinode-John-Lekan/publication/364322002_EMPLOYEE_ATTRITION_PREDICTION_USING_MACHINE_LEARNING_ALGORITHMS/links/6347f5a39cb4fe44f324df14/EMPLOYEE-ATTRITION-PREDICTION-USING-MACHINE-LEARNING-ALGORITHMS.pdf)
- Alamsyah, A., & Salma, N. (2022). A Comparative Study of Employee Churn Prediction Model. Journal of Applied Science and Technology Trends, 3(2), 45-50. <https://jazindia.com/index.php/jaz/article/view/3526>
- Alcala, G. E. S., & Murcia, J. V. B. (2024b, March 4). Machine Learning Techniques in Employee Churn Prediction. TWIST Journal. <https://twistjournal.net/twist/article/view/161>
- Almeida, F., & Duarte, C. (2024). Enhancing Work Productivity through Generative Artificial Intelligence: A Comprehensive Review. Sustainability, MPDI. <https://www.mdpi.com/2071-1050/16/3/1166>
- AlShourbaji, I., Helian, N., Sun, Y., Hussien, A. G., Abualigah, L., & Elnaim, B. (2023). An efficient churn prediction model using gradient boosting machine and metaheuristic optimization. Scientific Reports, 13(1). <https://doi.org/10.1038/s41598-023-41093-6>
- Al-Suraihi, W. A., Samikon, S. A., Alsuraihi, A., & Ibrahim, I. (2021). Employee Turnover: Causes, Importance and Retention Strategies. European Journal of Business and Management Research, 6(3), 1-10. <https://zendy.io/title/10.24018/ejbmr.2021.6.3.893>
- Anand, N., Sharma, S., & Yadav, S. (Eds.). (2023). SUSTAINABLE ECONOMIC AND MANAGEMENT PRACTICES: CHALLENGES AND FUTURE PROSPECTS (Vol. 1). <https://www.empyrealphublishinghouse.com/pdf/sustainable-economic-and-management-practices-challenges-and-future-prospects-vol-1.pdf#page=89>

- Bandyopadhyay, N., & Jadhav, A. (2021). Churn prediction of employees using machine learning techniques. *Tehnički Glasnik*, 15(1), 51–59. <https://doi.org/10.31803/tg-20210204181812>
- Bangura, S., & Lourens, M. E. (2024). Unveiling the Hidden Dynamics: Exploring Causative Factors and Impact of Employee Turnover on Organisational Performance. *British Journal of Multidisciplinary and Advanced Studies*, 5(2), 1-12. <https://doi.org/10.37745/bjmas.2022.0439>
- Castro, O., Bruneau, P., Sottet, J., & Torregrossa, D. (2023b, February 7). Landscape of high-performance Python to develop data science and machine learning applications. arXiv.org. [https://arxiv.org/abs/2302.03307?utm\\_source](https://arxiv.org/abs/2302.03307?utm_source)
- Chung, D., Yun, J., Lee, J., & Jeon, Y. (2023). Predictive model of employee attrition based on stacking ensemble learning. *Expert Systems with Applications*, 215, 119364. <https://doi.org/10.1016/j.eswa.2022.119364>
- Dimas, A., & Mukti, S. S. (2021). Performance comparison of grid search and random search methods for hyperparameter tuning in extreme gradient boosting algorithm to predict chronic kidney failure. *International Journal of Intelligent Engineering and Systems*, 14(6), 198–207. <https://doi.org/10.22266/ijies2021.1231.19>
- Duchini, E., Simion, S., & Arthur Turrell. (2023). A review of the effects of pay transparency. [https://emmaduchini.com/wp-content/uploads/2023/09/duchini\\_et\\_al-2023-a-review-on-the-effects-of-pay-transparency.pdf](https://emmaduchini.com/wp-content/uploads/2023/09/duchini_et_al-2023-a-review-on-the-effects-of-pay-transparency.pdf)
- Emanuel, N., & Harrington, E. (2024). Working remotely? Selection, treatment, and the market for Remote work. *American Economic Journal Applied Economics*, 16(4), 528–559. <https://doi.org/10.1257/app.20230376>
- Employee Performance and Productivity Data. (2024, September 4). Kaggle. <https://www.kaggle.com/datasets/mexwell/employee-performance-and-productivity-data>
- Enander, A., & Cardoso, J. (2020). How is employee turnover related to employee retention? A systematic review on two sets of meta-analyses. LUP Student Papers. <https://lup.lub.lu.se/student-papers/search/publication/9022943>

- Fallucchi, F., Coladangelo, M., Giuliano, R., & De Luca, E. W. (2020). Predicting Employee Attrition Using Machine Learning Techniques. *Computers*, 9(4), 86. <https://doi.org/10.3390/computers9040086>
- Fangohr, H., Beg, M., Bergemann, M., Bondar, V., Brockhauser, S., Campbell, A., Carinan, C., Costa, R., Dall'Antonia, F., Danilevski, C., E, J., Ehsan, W., Esenov, S., Fabbri, R., Fangohr, S., Fernandez-Del-Castillo, E., Flucke, G., Fortmann-Grote, C., Marsa, D. F., . . . Zhu, J. (2019). Data Exploration and Analysis with Jupyter Notebooks. HAL (Le Centre Pour La Communication Scientifique Directe). <https://doi.org/10.18429/jacow-icalepcs2019-tucpr02>
- Finnegan, R. (2025, March 19). *Gallagher Report: Why Turnover is Still #1 Concern in 2025*. C-Suite Analytics. <https://c-suiteanalytics.com/gallagher-turnover-is-1-concern-2025/>
- Gadi, P. D., & Kee, D. H. M. (2018). Human resource management practices and turnover intention: The mediating role of perceived organizational support in tertiary institutions in Nigeria. *International Journal of Engineering & Technology*, 7(3.25), 715-722. [https://www.researchgate.net/profile/Paul\\_Gadi/publication/328527915\\_Human\\_Resource\\_Management\\_Practices\\_and\\_Turnover\\_Intention\\_the\\_Mediating\\_Role\\_of\\_Perceived\\_Organizational\\_Support\\_in\\_Tertiary\\_Institutions\\_in\\_Nigeria/links/5bd26f1f4585150b2b876552/Human-Resource-Management-Practices-and-Turnover-Intention-the-Mediating-Role-of-Perceived-Organizational-Support-in-Tertiary-Institutions-in-Nigeria.pdf](https://www.researchgate.net/profile/Paul_Gadi/publication/328527915_Human_Resource_Management_Practices_and_Turnover_Intention_the_Mediating_Role_of_Perceived_Organizational_Support_in_Tertiary_Institutions_in_Nigeria/links/5bd26f1f4585150b2b876552/Human-Resource-Management-Practices-and-Turnover-Intention-the-Mediating-Role-of-Perceived-Organizational-Support-in-Tertiary-Institutions-in-Nigeria.pdf)
- Gazi, N. M. S., Nasiruddin, N. M., Dutta, N. S., Sikder, N. R., Huda, N. C. B., & Islam, N. M. Z. (2024). Employee Attrition Prediction in the USA: A Machine Learning Approach for HR Analytics and Talent Retention Strategies. *Journal of Business and Management Studies*, 6(3), 47–59. <https://doi.org/10.32996/jbms.2024.6.3.6>
- Gazi, N. M. S., Nasiruddin, N. M., Dutta, N. S., Sikder, N. R., Huda, N. C. B., & Islam, N. M. Z. (2024b). Employee Attrition Prediction in the USA: A machine learning approach for HR analytics and talent retention strategies. *Journal of Business and Management Studies*, 6(3), 47–59. <https://doi.org/10.32996/jbms.2024.6.3.6>
- Gill, M. S., Westermann, T., Schieseck, M., & Fay, A. (2023). Integration of Domain Expert-Centric Ontology Design into the CRISP-DM for Cyber-Physical Production Systems.

2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), 1–8. <https://doi.org/10.1109/etfa54631.2023.10275612>

Gill, M. S., Westermann, T., Steindl, G., Gehlhoff, F., & Fay, A. (2024, July 9). Integrating Ontology Design with the CRISP-DM in the context of Cyber-Physical Systems Maintenance. arXiv.org. [https://arxiv.org/abs/2407.06930?utm\\_source](https://arxiv.org/abs/2407.06930?utm_source)

Guido, R., Groccia, M.C. & Conforti, D (2023). A hyper-parameter tuning approach for cost-sensitive support vector machine classifiers. *Soft Computer* **27**. <https://doi.org/10.1007/s00500-022-06768-8>

Hills, Laura, DA., (2021). Managing Overworked Employees - ProQuest. <https://www.proquest.com/docview/2504559003?pq-origsite=gscholar&fromopenview=true>

Hong, W., Pai, P., Huang, Y., & Yang, S. (2005). Application of support vector machines in predicting employee turnover based on job performance. In Lecture notes in computer science (pp. 668–674). [https://doi.org/10.1007/11539087\\_85](https://doi.org/10.1007/11539087_85)

Hosen, M. E. (2022). FACTORS AFFECTING EMPLOYEE TURNOVER IN MULTINATIONAL COMPANIES IN MALAYSIA. *Malaysian Management Journal*, 26. <https://doi.org/10.32890/mmj2022.26.2>

Hosen, M. E. (2022). Factors Affecting Employee Turnover in Multinational Companies in Malaysia. *Malaysian Management Journal*, 26, 31-54. <https://doi.org/10.32890/mmj2022.26.2>

Huber, S., Wiemer, H., Schneider, D., & Ihlenfeldt, S. (2019). DMME: Data mining methodology for engineering applications – a holistic extension to the CRISP-DM model. *Procedia CIRP*, 79, 403–408. <https://doi.org/10.1016/j.procir.2019.02.106>

Imani, M., & Arabnia, H. R. (2023). Hyperparameter optimization and combined data sampling techniques in machine learning for customer churn prediction: A Comparative analysis. *Technologies*, 11(6), 167. <https://doi.org/10.3390/technologies11060167>

- Jaggia, S., Kelly, A., Lertwachara, K., & Chen, L. (2020). Applying the CRISP-DM framework for teaching business analytics. *Decision Sciences Journal of Innovative Education*, 18(4), 612–634. <https://doi.org/10.1111/dsji.12222>
- Jain, N., Tomar, A., & Jana, P. K. (2020). A novel scheme for employee churn problem using multi-attribute decision making approach and machine learning. *Journal of Intelligent Information Systems*, 56(2), 279–302. <https://doi.org/10.1007/s10844-020-00614-9>
- Jana, D. (2020). A novel scheme for employee churn problem using multi-attribute decision-making and machine learning. *Journal of Ambient Intelligence and Humanized Computing*, 12, 1-12. <https://link.springer.com/article/10.1007/s10844-020-00614-9>
- Kakad, S., Kadam, R., Deshpande, P., Karde, S., & Lalwani, R. (2020). Employee Attrition Prediction System. *IJISSET - International Journal of Innovative Science, Engineering & Technology*, 7(9), 57–58. [https://ijiset.com/vol7/v7s9/IJISSET\\_V7\\_I9\\_07.pdf](https://ijiset.com/vol7/v7s9/IJISSET_V7_I9_07.pdf)
- Kanchana, L., & Jayathilaka, R. (2023). Factors impacting employee turnover intentions among professionals in Sri Lankan startups. *PLoS ONE*, 18(2), e0281729. <https://doi.org/10.1371/journal.pone.0281729>
- Khattak, A., Mehak, Z., Ahmad, H., Asghar, M. U., Asghar, M. Z., & Khan, A. (2023). Customer churn prediction using composite deep learning technique. *Scientific Reports*, 13(1). <https://doi.org/10.1038/s41598-023-44396-w>
- Klop, G. (2021). Predicting employee turnover and reducing turnover costs using machine learning techniques. [https://pure.tue.nl/ws/portalfiles/portal/183311084/Master\\_Thesis\\_Guido\\_Klop.pdf](https://pure.tue.nl/ws/portalfiles/portal/183311084/Master_Thesis_Guido_Klop.pdf)
- Kovvuri, S. R., & Dommeti, L. S. D. (2022). Employee Turnover Prediction - A comparative study of supervised machine learning models. DIVA. <https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1679511&dswid=1896>
- Lewis, A., & Sequeira, A. H. (2016). Effectiveness of employee retention strategies in industry. *Employee social responsibility and HR Practices e-journal CMBO*. <https://dx.doi.org/10.2139/ssrn.2733517>

- Lin, C., Huang, C., & Zhang, H. (2018). Enhancing Employee Job satisfaction via E-learning: The Mediating Role of an Organizational Learning Culture. International Journal of Human-Computer Interaction, 35(7), 584–595. <https://doi.org/10.1080/10447318.2018.1480694>
- Lin, C.-Y., & Huang, C.-K. (2021). Employee Turnover Intentions and Job Performance from a Planned Change: The Effects of an Organizational Learning Culture and Job Satisfaction. International Journal of Manpower, 42(3), 409-423. <https://doi.org/10.1108/IJM-08-2018-0281>
- Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. ACM computing surveys (CSUR), 54(6), 1-35. <https://doi.org/10.1145/3457607>
- Murcia, J. (2023). Machine Learning Techniques in Employee Churn Prediction. TWIST Journal. <https://twistjournal.net/twist/article/view/161>
- Najafi-Zangeneh, S., Shams-Gharneh, N., Arjomandi-Nezhad, A., & Zolfani, S. H. (2021). An Improved Machine Learning-Based Employees Attrition Prediction Framework with Emphasis on Feature Selection. Mathematics, 9(11), 1226. <https://doi.org/10.3390/math9111226>
- Nan, L., & Zhang, H. (2023). A model for analyzing employee turnover in enterprises based on improved XGBOOST algorithm. International Journal of Advanced Computer Science and Applications, 14(11). <https://doi.org/10.14569/ijacsa.2023.01411104>
- Nandal, M., Grover, V., Sahu, D., & Dogra, M. (2024). Employee Attrition: Analysis of Data Driven Models. EAI Endorsed Transactions on Internet of Things, 10. <https://doi.org/10.4108/eetiot.4762>
- Naz, K., Siddiqui, I. F., Koo, J., Khan, M. A., & Qureshi, N. M. F. (2022). Predictive modeling of employee churn analysis for IoT-Enabled software industry. Applied Sciences, 12(20), 10495. <https://doi.org/10.3390/app122010495>
- Nguyen, T. D., Nguyen, T. T., & Nguyen, P. C. (2023). Job embeddedness and turnover intention in the public sector: the role of life satisfaction and ethical leadership. International Journal

of Public Sector Management, 36(4/5), 463–479. <https://doi.org/10.1108/ijpsm-03-2023-0070>

Nguyen, T. D., Nguyen, T. T., & Nguyen, P. C. (2023). Job embeddedness and turnover intention in the public sector: the role of life satisfaction and ethical leadership. International Journal of Public Sector Management, 36(4/5), 463–479. <https://doi.org/10.1108/ijpsm-03-2023-0070>

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2012c, January 2). SciKit-Learn: Machine Learning in Python. arXiv.org. [https://arxiv.org/abs/1201.0490?utm\\_source](https://arxiv.org/abs/1201.0490?utm_source)

Pereira, L. A. S., 152022027. (2024). Using Machine Learning to predict Employee Turnover: a case study of the Willis Towers Watson Lisbon Hub. In P. A. Fernandes, Universidade Católica Portuguesa, & Willis Towers Watson, MSc in Business Analytics [Thesis]. Universidade Católica Portuguesa.

<https://repositorio.ucp.pt/bitstream/10400.14/44817/1/203590686.pdf>

Pimentel, J. F., Murta, L., Braganholo, V., & Freire, J. (2019, May). A large-scale study about quality and reproducibility of jupyter notebooks. In 2019 IEEE/ACM 16th international conference on mining software repositories (MSR) (pp. 507-517). IEEE. <https://leomurta.github.io/papers/pimentel2019a.pdf>

Predicting Employee Attrition using XGBoost Machine Learning Approach. (2018, November 1). IEEE Conference Publication | IEEE Xplore. <https://ieeexplore.ieee.org/document/8746940>

Predicting employee churn in Python – machine learning geek. (2023, March 24). <https://machinelearninggeek.com/predicting-employee-churn-in-python/>

Punnoose, R., & Ajit, P. (2016). Prediction of Employee Turnover in Organizations using Machine Learning Algorithms. INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN ARTIFICIAL INTELLIGENCE, 5(9). <https://doi.org/10.14569/ijarai.2016.050904>

RBF        SVM        parameters.        (n.d.).        Scikit-learn.        [https://scikit-learn.org/stable/auto\\_examples/svm/plot\\_rbf\\_parameters.html](https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html)

Reyes, A. G. (2024, December 31). An analysis of Herzberg's Two-Factor theory of motivation and employee turnover in the BPO industry. [https://ijmehd.com/index.php/ijmehd/article/view/304?utm\\_source](https://ijmehd.com/index.php/ijmehd/article/view/304?utm_source)

Schröer, C., Kruse, F., & Gómez, J. M. (2021). A Systematic Literature Review on Applying CRISP-DM Process model. Procedia Computer Science, 181, 526–534. <https://doi.org/10.1016/j.procs.2021.01.199>

Shoman et al., (2021). Predictors of Occupational Burnout: A Systematic review. International Journal of Environmental Research and Public Health, 18(17), 9188. <https://doi.org/10.3390/ijerph18179188>

Study and Prediction Analysis of the Employee Turnover using Machine Learning Approaches. (2021, September 24). IEEE Conference Publication | IEEE Xplore. <https://ieeexplore.ieee.org/abstract/document/9573759>

T. Vijaya Saradhi (2025). A Study on Hyperparameter Tuning in Support Vector Machines and its Impact on Model Accuracy. A Study on Hyperparameter Tuning in Support Vector Machines and Its Impact on Model Accuracy, 5(1). <https://doi.org/10.33425/3066-1226.1063>

Thome, M. J., & Greenwald, J. M. (2020). Job and community embeddedness on voluntary turnover. Journal of Business and Industrial Marketing, 35(10), 1573–1580. <https://doi.org/10.1108/jbim-01-2019-0021>

Tirta, A. H., Enrika, A., & Binus Business School, Jakarta, Indonesia. (2020). Understanding the impact of reward and recognition, work life balance, on employee retention with job satisfaction as mediating variable on millennials in Indonesia. In Centre for Business & Economic Research (CBER), Journal of Business and Retail Management Research (JBRMR) (Vol. 14, Issue 3). [https://jbrmr.com/cdn/article\\_file/2020-07-23-01-05-10-AM.pdf](https://jbrmr.com/cdn/article_file/2020-07-23-01-05-10-AM.pdf)

Udayar, S., Urbanaviciute, I., Maggiori, C., & Rossier, J. (2024). Does promotion foster career sustainability? A comparative three-wave study on the role of promotion in work stress, job satisfaction, and career-related performance. International Journal for Educational and Vocational Guidance. <https://doi.org/10.1007/s10775-024-09694-3>

Van Tuan, N., & Quang, T. M. (2020). Ethical Implications of Artificial Intelligence: A Systematic Review of Bias, Fairness, and Accountability. Artificial Intelligence and Machine Learning Review, 1(1), 1-7. <https://doi.org/10.69987/>

Vengai Musanga Zimbabwe National Defense University vengaimusanga@gmail.com Colin Chibaya\* Sol Plaatje University, Kimberley, SA colin.chibaya@spu.ac.za. (n.d.). A MACHINE LEARNING MODEL TO FORECAST EMPLOYEE CHURN FOR HR ANALYTICS. Nemisa AI - AI Courses South Africa. [https://ai.nemisa.co.za/articles/a-machine-learning-model-to-forecast-employee-churn-for-hr-analytics/?utm\\_source](https://ai.nemisa.co.za/articles/a-machine-learning-model-to-forecast-employee-churn-for-hr-analytics/?utm_source)

View of an analysis of Herzberg's Two-Factor theory of motivation and employee turnover in the BPO industry. (n.d.). <https://ijmehd.com/index.php/ijmehd/article/view/304/295>

View of Employee Turnover: Causes, importance and retention Strategies. (n.d.). <https://www.ejbmrr.org/index.php/ejbmrr/article/view/893/488>

View of FACTORS AFFECTING EMPLOYEE TURNOVER IN MULTINATIONAL COMPANIES IN MALAYSIA. (n.d.). <https://ejournal.uum.edu.my/index.php/mmj/article/view/13208/3899>

View of Factors Affecting Turnover Intention: A literature review. (n.d.). <https://ijbtob.org/index.php/ijbtob/article/view/277/131>

Villalobos-Arias, L., Quesada-López, C., Guevara-Coto, J., Martínez, A., & Jenkins, M. (2020). Evaluating hyper-parameter tuning using random search in support vector machines for software effort estimation. Evaluating Hyper-parameter Tuning Using Random Search in Support Vector Machines for Software Effort Estimation. <https://doi.org/10.1145/3416508.3417121>

Wainer, J., Fonseca, P. How to tune the RBF SVM hyperparameters? An empirical evaluation of 18 search algorithms. *Artif Intell Rev* **54**, 4771–4797 (2021).  
<https://doi.org/10.1007/s10462-021-10011-5>

Yadav, G. (2025). AI and analytics conundrum: unpacking the barriers in modern HR with ISM and MICMAC analysis. *International Journal of Organizational Analysis*.  
<https://doi.org/10.1108/ijoa-08-2024-4782>

Yao, Y. (2024). Employee Turnover Prediction based on Particle Swarm Optimization - Support Vector Machine. *Applied and Computational Engineering*, *112*(1), 1–7.  
<https://doi.org/10.54254/2755-2721/2025.17878>

## Appendices

### Appendix A – Project Plan Form

The screenshot shows a web-based project proposal form. At the top right, there is a user profile icon and the name "YEONG MAN WEI". The main content area has a blue header bar with the text "Project Title Proposal". Below this, the page title is "Project Title Status (Students)". The form fields include:

- Project Title:** Predicting Employee Churn and Optimizing Workforce Productivity Using Machine Learning
- Status:** APPROVED
- Proposed By:** YEONG MAN WEI
- Description:** A detailed description of the project's purpose, methodology, and objectives. It highlights the use of Machine Learning to predict employee churn and optimize productivity by analyzing historical data.
- Aim:** To develop a machine learning-based framework that enhances employee retention and productivity.
- Objectives:**
  - Develop a Machine Learning Model for Predicting Employee Churn.
  - Analyze the Relationship Between Productivity Metrics and Employee Performance.
  - Design a Predictive Dashboard for HR Professionals.
  - Evaluate the Effectiveness of the Dashboard Using Historical Employee Data.
- Targeted Users:** Human Resources (HR) professionals and organizational decision-makers.
- SDG:** SDG8
- Keywords:** HR Analytics, Machine Learning, Analytics and Data, Employee Engagement
- Preferred Supervisor(s):** ASSOC. PROF. DR. RAJA RAJESWARI A/P PONNUSAMY, FARHANA ILLIANI BINTI HASSAN, DR. PREETHI SUBRAMANIAN, JUSTIN GILBERT A/L ALEXIUS SILVESTER, TS. MOHAMMAD NAMAZEE BIN MOHD NIZAM
- Assigned Supervisor:** farhana.illiani
- Assigned Second Marker:** nuramira
- Remarks:** A table showing a single entry: Dr. Devi Octaviani, Poor, Jan 28, 2025, 4:36:38 AM.
- Print:** A button to print the document.

At the bottom of the page, a footer note states: Asia Pacific University of Technology and Innovation (APU). All rights reserved.

Figure 280: Project Plan Form

## Appendix B – Ethics Forms (Fast Track)

The screenshot displays the 'Ethics Form' interface for a 'Fast-track' application. At the top, there's a navigation bar with 'Home > View Ethics Form'. Below it, the 'Ethics Form Type' is set to 'Fast-track'. A section titled 'Supervisor Remarks' contains two entries:

Name	Remarks	Date
FARHANA ILLIANI BINTI HASSAN	Please provide explanation in 'Description' column. Briefly explain about your dataset & its source.	Feb 21, 2025, 3:11:54 AM
FARHANA ILLIANI BINTI HASSAN	Good.	Feb 21, 2025, 3:26:07 AM

The main content area is titled 'Ethics Form (Fast-Track)' and includes a navigation bar with four steps: 1. Participant Confidentiality, 2. Nature of Research, 3. Target Participants, and 4. Support Information. The current step is 'Participant Confidentiality'. The form consists of several questions with radio button options:

- Will you describe the main procedures to participants in advance, so that they are informed about what to expect? Options: Yes (radio), No (radio), N/A (radio).
- Will you tell participants that their participation is voluntary? Options: Yes (radio), No (radio), N/A (radio).
- Will you obtain written consent for participation? Options: Yes (radio), No (radio), N/A (radio).
- If the research is observational, will you ask participants for their consent to being observed? Options: Yes (radio), No (radio), N/A (radio).
- Will you tell participants that they may withdraw from the research at any time and for any reason? Options: Yes (radio), No (radio), N/A (radio).
- With questionnaires and interviews will you give participants the option of omitting questions they do not want to answer? Options: Yes (radio), No (radio), N/A (radio).
- Will you tell participants that their data will be treated with full confidentiality and that, if published, it will not be identifiable as theirs? Options: Yes (radio), No (radio), N/A (radio).
- Will you give participants the opportunity to be debriefed i.e. to find out more about the study and its results? Options: Yes (radio), No (radio), N/A (radio).

A note at the bottom of the questionnaire states: 'Note: If you have ticked No to any of questions above, you should complete the Full Ethics Approval Form. Full Ethics Form'.

At the bottom right, there are 'Prev' and 'Next' buttons, and a 'Print' link.

Figure 281: Ethnics Form

Ethics Form

Home > View Ethics Form

Ethics Form Type: Fast-track

### Supervisor Remarks

Name	Remarks	Date
FARHANA ILLIANI BINTI HASSAN	Please provide explanation in 'Description' column. Briefly explain about your dataset & its source.	Feb 21, 2025, 3:11:54 AM
FARHANA ILLIANI BINTI HASSAN	Good.	Feb 21, 2025, 3:26:07 AM

### Ethics Form (Fast-Track)

1 Participant Confidentiality      2 Nature of Research      3 Target Participants      4 Support Information

#### Nature of Research

Will your project/assignment deliberately mislead participants in any way?  Yes  No  N/A

Is there any realistic risk of any participants experiencing either physical or psychological distress or discomfort? This should include details of what you will tell participants to do if they should experience any problems (e.g., who they can contact for help). You may also need to consider risk assessment issues.  Yes  No  N/A

Is the nature of the research such that contentious or sensitive issues might be involved? This includes research which could induce psychological stress, anxiety or humiliation, or cause more than minimal pain.  Yes  No  N/A

Does your research involve the use of sensitive materials? E.g., records of personal or sensitive confidential information.  Yes  No  N/A

Does your research require external agency approval?  Yes  No  N/A

Does your research use hazardous or controlled substance?  Yes  No  N/A

Does your research require you to visit participants in their home or non-public space?  Yes  No  N/A

Does your research investigate illegal activities or behaviours?  Yes  No  N/A

Does your research involve discussion or collection of information on potentially sensitive, embarrassing or distressing topics, administrative or secure data? This includes research involving respondents through internet where visual images are used, and where sensitive issues are discussed  Yes  No  N/A

Will your participants be receiving financial compensation for participating in your research?  Yes  No  N/A

Will your research data be used in the future after the conclusion of your project?  Yes  No  N/A

Will your research involve in processing sensitive data belonging to an organisation/persons?  Yes  No  N/A

Will your research be collecting photographs, videos, and audio recordings of the participants?  Yes  No  N/A

Will the participants' personal particulars be known to any third party?  Yes  No  N/A

Will the participants' data confidentiality be made known to the public?  Yes  No  N/A

Will the research be conducted where the safety of the researchers maybe in question?  Yes  No  N/A

Will the research be conducted outside of Malaysia and/or UK?  Yes  No  N/A

**Note:** If you have ticked YES to any of questions above, you should complete the Full Ethics Approval Form.  
Full Ethics Form

Prev Next

Print

Asia Pacific University of Technology and Innovation (APU); All rights reserved.

Figure 282:Ethnics Form

Home > View Ethics Form

Ethics Form Type: Fast-track

### Supervisor Remarks

Name	Remarks	Date
YEONG MAN WEI		

### Ethics Form (Fast-Track)

1 Participant Confidentiality    2 Nature of Research    3 Target Participants    4 Support Information

#### Target Participants

Do participants fall into any of the following special groups?

- Yes
- No
- N/A

- Children (under 18 years of age)
- People with communication or learning difficulties
- Patients
- People in custody
- People who could be regarded as vulnerable
- People engaged in illegal activities (e.g., drug taking)
- Groups of people whose relationship among each other allow one to have influence over the other such as: Carers and patients with chronic conditions; teachers and their students; prison authorities and prisoners; employers and employees
- Groups where permission of a gatekeeper is normally required for initial access to members.

**Note:** You may also need to obtain satisfactory clearance from the relevant authorities.

Does the project/assignment involve external funding or external collaboration where the funding body or external collaborative partner requires the University to provide evidence that the project/assignment had been subject to ethical scrutiny?

**Note:** If you have ticked Yes to any of questions above, you should complete the Full Ethics Approval Form. There is an obligation on student and supervisor to bring to the attention of the APU School Research Ethics Committee any issues with ethical implications not clearly covered by the above checklist.  
Full Ethics Form

Prev    Next

Print

Asia Pacific University of Technology and Innovation (APU). All rights reserved.

Figure 283:Ethnics Form

Ethics Form

Home > View Ethics Form

Ethics Form Type: Fast-track

### Supervisor Remarks

Name	Remarks	Date
FARHANA ILLIANI BINTI HASSAN	Please provide explanation in 'Description' column. Briefly explain about your dataset & its source.	Feb 21, 2025, 3:11:54 AM
FARHANA ILLIANI BINTI HASSAN	Good.	Feb 21, 2025, 3:26:07 AM

### Ethics Form (Fast-Track)

1 Participant Confidentiality    2 Nature of Research    3 Target Participants    4 Support Information

#### Support Information

I consider that this project/assignment has no significant ethical implications requiring a full ethics submission to the APU Research Ethics Committee.

I am aware of APU liability policy and will make the necessary arrangement for insurance coverage of all researchers and participants of the project/assignment.

Description: This project aims to predict employee churn and optimize workforce productivity using machine learning by applying the CRISP-DM methodology. The project will apply an open dataset sourced from Kaggle.com with contains 100,000 records with 20 attributes related to employee demographics, job roles, and performance metrics. The methodology involves data preprocessing, including handling missing values, data transformation and normalization to ensure data quality. Machine learning models, such as Decision Trees will be applied to train and evaluate predictive models. The

Consent Form (Maximum 3 files)

Information Sheet

Additional Files

I also confirm that:

**Note:** Please check either 1 of the checkboxes.

i) All key documents e.g., consent form, information sheet, questionnaire/interview, and all material such as emails and posters for the purpose of recruitment of participants are appended to this application.

ii) Any key documents e.g. consent form, information sheet, questionnaire/interview schedules which need to be finalised following initial investigations will be submitted for approval by the project/assignment supervisor/module lecturer before they are used in primary data collection.

Prev Next

Print

Asia Pacific University of Technology and Innovation (APU). All rights reserved.

Figure 284:Ethnics Form

## Appendix C – Log Sheets

The screenshot shows a web-based application for managing meeting logs. At the top, there's a navigation bar with 'Meeting Scheduling' on the left and a user icon with 'YEONG MAN WEI' on the right. Below the navigation, a breadcrumb trail shows 'Home > Meeting Scheduling > My Meeting Logs'. A green button labeled 'APPROVED' is visible. The main content area is titled 'Meeting Log' and contains the following details:

<b>Title</b>	Predicting Employee Churn and Optimizing Workforce Productivity Using Machine Learning
<b>Supervisor Name</b>	farhana illiani
<b>Created At</b>	10-03-2025 04:32 PM
<b>Meeting Name</b>	Meeting 1
<b>Meeting Schedule Date</b>	13-02-2025 02:15 pm
<b>Location</b>	OTHERS
<b>Others</b>	SoC Office

Below this is a section titled 'Content' with two subsections: 'Items For Discussion' and 'Content of Discussion'. Under 'Items For Discussion', there are three numbered points: 1. Review the Project Proposal Form., 2. Ensure that the aim and objectives align with the project title., 3. Identify whether the dataset is suitable for predicting employee churn.. Under 'Content of Discussion', there are two numbered points: 1. Minor revisions are needed for the objectives to ensure they are achievable., 2. Focus on conducting a literature review based on past research.. Under 'Action List', there are three numbered points: 1. Revise the project objectives accordingly., 2. Start doing literature review to analyse relevant past research., 3. Do the initial data preprocessing and understanding of data..

A 'Download' button is located at the bottom of this section. The next section is titled 'Meeting Log Remarks' and contains a table with one row:

Name	Remarks	Date
FARHANA ILLIANI BINTI HASSAN	ok	Mar 11, 2025, 1:43:11 AM

At the bottom of the page, there are 'Back' and 'Print' buttons, and a footer note: 'Asia Pacific University of Technology and Innovation (APU). All rights reserved.'

Figure 285: Log Sheets (Meeting 1)

The screenshot shows a web-based application interface for managing meeting logs. At the top, there's a header with a logo and the name "YEONG MAN WEI". Below the header, the main content area is divided into several sections:

- Meeting Logs**: A green button labeled "APPROVED".
- Meeting Log**: Displays various details about the meeting:
 

Title	Predicting Employee Churn and Optimizing Workforce Productivity Using Machine Learning
Supervisor Name	farhana.illiani
Created At	10-03-2025 04:33 PM
Meeting Name	Meeting 2
Meeting Schedule Date	26-02-2025 03:45 pm
Location	OTHERS
Others	SoC Office
- Content**: Contains sections for **Items For Discussion**, **Content of Discussion**, and **Action List**. Each section lists numbered tasks or steps.
  - Items For Discussion**: 1. Review the IR chapter 1. 2. Review the problem statement and technical research for the literature review. 3. Clarify about the data understanding and data preprocessing step and ask for suggestions.
  - Content of Discussion**: 1. It's better to break down each problem in the problem statement into subtopics. 2. Discuss the technical tools to be used later and choose the appropriate ones.
  - Action List**: 1. Revise the problem statement for better alignment and a tidier structure. 2. Complete Chapter 1 and Chapter 2. 3. Continue working on data understanding and data preprocessing.
- Meeting Log Remarks**: A table showing a single entry:
 

Name	Remarks	Date
FARHANA ILLIANI BINTI HASSAN	ok	Mar 11, 2025, 1:43:25 AM

At the bottom right, there are "Print" and "Asia Pacific University of Technology and Innovation (APU). All rights reserved." links.

*Figure 286: Log Sheets (Meeting 2)*

The screenshot displays a web-based application interface for managing meeting logs. It features a header with navigation links and user information, followed by three main content sections: Meeting Log, Content, and Meeting Log Remarks.

### Meeting Log

**APPROVED**

**Meeting Log**

<b>Title</b>	Predicting Employee Churn and Optimizing Workforce Productivity Using Machine Learning
<b>Supervisor Name</b>	farhana.illiani
<b>Created At</b>	12-03-2025 03:17 AM
<b>Meeting Name</b>	Meeting 3
<b>Meeting Schedule Date</b>	11-03-2025 03:30 pm
<b>Location</b>	OTHERS
<b>Others</b>	SoC Office

### Content

**Items For Discussion**

1. Asking about target variables that have class imbalance issues and requesting suggestion
2. Inquiring about the discussion of SDGs for each chapter.
3. Asking about data preprocessing tasks, such as standardization and feature engineering.

**Content of Discussion**

1. Explore other techniques to identifies the correlations
2. A strong justification is required, and each chapter must be better related to the SDGs.

**Action List**

1. Test the correlations using other techniques
2. Recheck back each chapter and relate it to the SDGs with providing strong justification.
3. Ensure document formatting aligns with the requirements.

**Download**

### Meeting Log Remarks

Name	Remarks	Date
FARHANA ILLIANI BINTI HASSAN	ok	Mar 12, 2025, 2:39:48 AM

**Print**

Asia Pacific University of Technology and Innovation (APU). All rights reserved.

*Figure 287: Log Sheets (Meeting 3)*

The screenshot displays a web-based application for managing meeting logs. At the top, there's a navigation bar with 'Meeting Scheduling' and a user icon labeled 'YEONG MAN WEI'. Below the header, a breadcrumb trail shows 'Home > Meeting Scheduling > My Meeting Logs'. A blue header bar indicates the current section is 'Meeting Logs'. A green button labeled 'APPROVED' is visible. The main content area is titled 'Meeting Log' and contains the following details:

<b>Title</b>	Predicting Employee Churn and Optimizing Workforce Productivity Using Machine Learning
<b>Supervisor Name</b>	farhana.illiani
<b>Created At</b>	26-06-2025 04:49 PM
<b>Meeting Name</b>	Meeting 4
<b>Meeting Schedule Date</b>	03-06-2025 04:00 pm
<b>Location</b>	ONLINE MEETING

A second blue header bar is labeled 'Content'. Under this section, there are two lists:

- Items For Discussion**
  - Feedback on Investigation Report (IR) marks and suggestions for improvement.
  - Model performance after applying class imbalance techniques especially for the confusion matrix results for the model building section.
  - Clarification on whether hyperparameter tuning needs to be supported by literature review.
- Content of Discussion**
  - IR can be improved in the summary of Chapter 2 (Literature Review) to better highlight and justify the best-performing models.
  - During the presentation, only the final results need to be presented and keep the explanation focused.
  - Hyperparameter tuning is recommended that the relevant methods and justifications should be included in Chapter 2 (Literature Review) rather than in the model building chapter.

Under 'Action List', there are three items:

- Revise Chapter 2 summary in the Investigation Report to emphasize the rationale for model selection and include supporting studies.
- Focus presentation content on final model performance and minimize explanation of model building's results.
- Update Chapter 2 (Literature Review) to include background and justification for hyperparameter tuning methods used.

A 'Download' button is located at the bottom of this section. The next section is titled 'Meeting Log Remarks' with a blue header bar. It contains a table:

Name	Remarks	Date
FARHANA ILLIANI BINTI HASSAN	ok	Jul 16, 2025, 5:05:17 AM

Buttons for 'Back' and 'Print' are located at the bottom of this section. The footer of the page states: 'Asia Pacific University of Technology and Innovation (APU). All rights reserved.'

*Figure 288: Log Sheets (Meeting 4)*

The screenshot shows a web-based application interface for managing meeting logs. At the top, there's a header bar with the user's name 'YEONG MAN WEI' and a profile icon. Below the header, the main content area has a title 'Meeting Logs' and a green button labeled 'APPROVED'. The main section is titled 'Meeting Log' and contains the following data:

<b>Title</b>	Predicting Employee Churn and Optimizing Workforce Productivity Using Machine Learning
<b>Supervisor Name</b>	farhana illiani
<b>Created At</b>	18-07-2025 09:23 AM
<b>Meeting Name</b>	Meeting 5
<b>Meeting Schedule Date</b>	10-07-2025 02:45 pm
<b>Location</b>	OTHERS
<b>Others</b>	SoC Office

Below this is a 'Content' section with three subsections:

- Items For Discussion**: A list of 5 numbered points related to model performance and tuning.
- Content of Discussion**: A list of 3 numbered points regarding HR login requirements and dashboard data.
- Action List**: A list of 3 numbered points for setting up HR logins and updating the dashboard.

A 'Download' button is located at the bottom of this section. The final section is titled 'Meeting Log Remarks' and contains a table:

Name	Remarks	Date
FARHANA ILLIANI BINTI HASSAN	ok	Jul 19, 2025, 2:41:19 AM

At the bottom right of the page, there are 'Print' and 'Asia Pacific University of Technology and Innovation (APU). All rights reserved.' links.

Figure 289: Log Sheets (Meeting 5)

**Meeting Scheduling**

 YEONG MAN WEI ▾

[Home](#) > Meeting Scheduling > Meeting Log Details

**Meeting Logs**

**APPROVED**

### Meeting Log

<b>Title</b>	Predicting Employee Churn and Optimizing Workforce Productivity Using Machine Learning
<b>Supervisor Name</b>	farhana.illiani
<b>Created At</b>	20-07-2025 04:38 PM
<b>Meeting Name</b>	Meeting 6
<b>Meeting Schedule Date</b>	18-07-2025 03:30 pm
<b>Location</b>	OTHERS
<b>Others</b>	SoC office

**Content**

**Items For Discussion**

1. Discussed the use of the Faker library to generate fake data for model deployment purposes.
2. Clarified the preferred structure for presenting source code and outputs in the document model deployment section.
3. Reviewed the poster information and confirmed if it's acceptable.

**Content of Discussion**

1. Using the Faker library to generate fake data for deployment is acceptable and can work for the project.
2. It's preferred to present one source code with its corresponding output together rather than all source code followed by the outputs.
3. The poster information is satisfactory and doesn't require changes.

**Action List**

1. Proceed with using the Faker library to generate fake data for deployment.
2. Restructure the model deployment section to show one source code and its corresponding output together.
3. No changes needed for the poster as it's already fine.

**Download**

**Meeting Log Remarks**

Name	Remarks	Date
FARHANA ILLIANI BINTI HASSAN	ok	Jul 21, 2025, 2:07:25 AM

**Print**

Asia Pacific University of Technology and Innovation (APU). All rights reserved.

*Figure 290: Log Sheets (Meeting 6)*

## Appendix D – Poster

# Predicting Employee Churn and Optimizing Workforce Productivity Using Machine Learning

Yeong Man Wei | TP065870 | APU3F2411CS(DA)

BSC (Hons) In Computer Science With Specialism In Data Analytics  
Supervisor : Ms. Farhana Illiani Binti Hassan  
Second Marker: Ms. Nur Amira Binti Abdul Majid

## INTRODUCTION

In the fast-paced world of business, **employee turnover** and **workforce productivity** are crucial for organizational success. High **turnover rates** result in costly disruptions highlighting the **importance of retaining talented employees**. According to Gallagher's 2024 Workforce Trends Report, it mentioned that over 4,000 organizations, **66% of HR executives still say retention is their biggest workforce challenge** (Finnegan et al., 2025).

This project focuses on **applying machine learning to predict employee turnover** and **identify factors influencing productivity** to enhance workforce performance. It includes an **interactive analytics system** where HR can explore real-time data visualizations with predictive models. The system **identifies trends and provides actionable insights** to improve retention and **performance** which helps in **optimizing business outcomes** and fostering a stable and efficient work environment.

## OBJECTIVES

- To develop machine learning models for predicting employee churn based on key factors.
- To analyze the relationship between **productivity metrics** and **employee performance**.
- To **design dashboard** that provides HR professionals with model predictions on employee churn and insights into factors affecting productivity.
- To **evaluate, compare and select the best machine learning models** based on performance metrics to **provide actionable insights** for employee retention and productivity optimization.

## PROBLEM STATEMENT

- Employee churn increases recruitment and training costs with **existing methods failing to predict turnover accurately**.
- Low productivity** hampers business growth and many organizations **lack real time tools** to overcome it.
- Lack of integrated frameworks** that address **both churn prediction and productivity analysis** which are limiting workforce management.
- Data quality issues and ethical concerns** such as privacy and bias are hinder the accuracy and fairness of predictive models in HR analytics.

## IMPLEMENTATION

### METHODOLOGY

CRISP-DM (Cross-Industry Standard Process for Data Mining)

### MODEL

- Random Forest
- XGBoost
- Gradient Boosting
- Support Vector Machine (SVM)

### RESULT

XGBoost model achieves the highest performance to predict employee churn.

- Accuracy 85%
- Precision 80%
- Recall 88%
- F1 score 85%

### Example of Analysis Charts

## DEPLOYMENT

### Login Page

### HR Dashboard

### Employee Performance Overview

### Employee Churn Prediction

## CONCLUSION

This project successfully developed a machine learning-based system to predict employee churn and provide productivity insights. By using XGBoost and an interactive HR dashboard, it enables real-time analysis of churn risk and performance trends. The system supports proactive decision-making to help organisations reduce turnover, improve workforce stability and optimise HR strategies through data-driven insights.

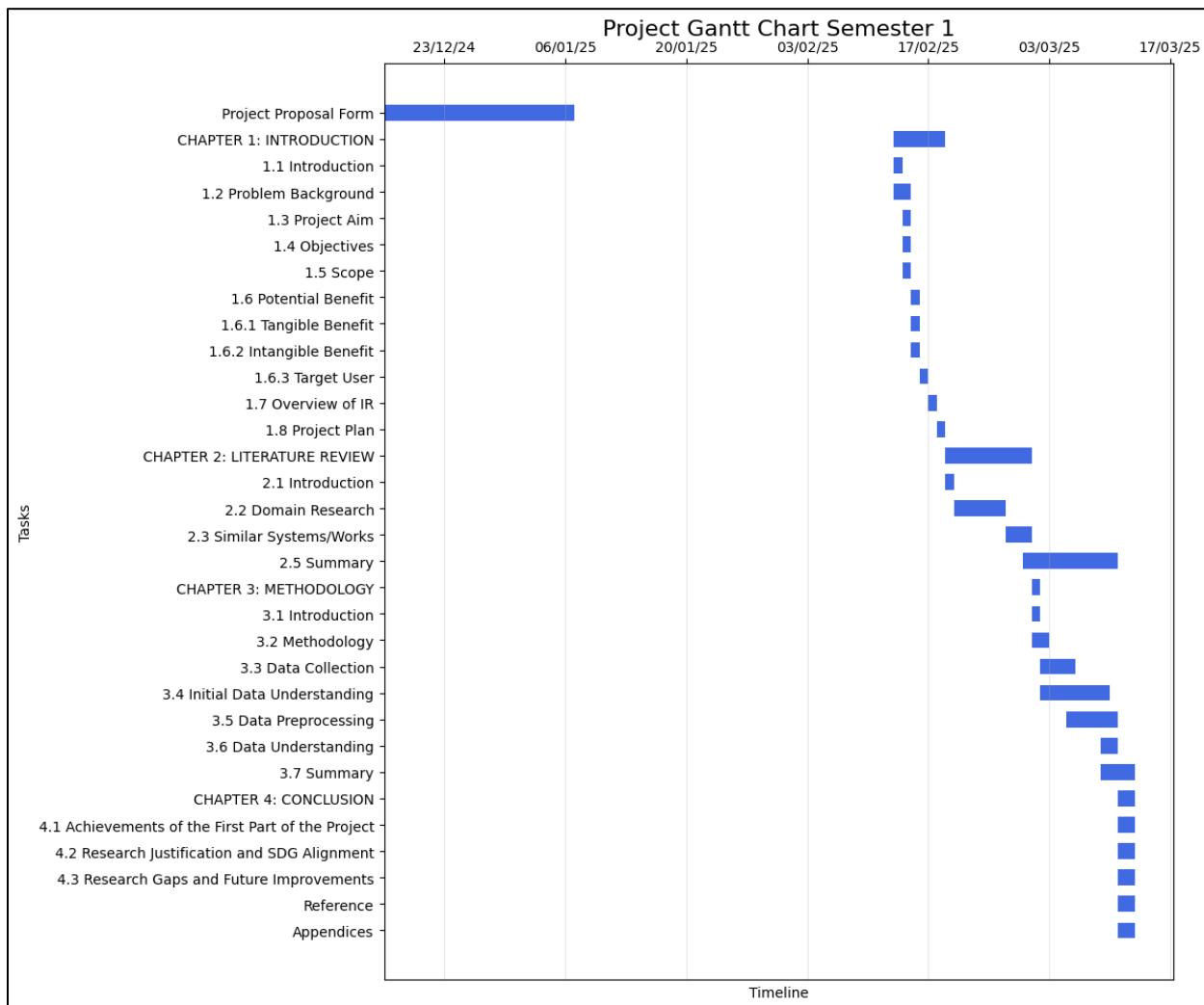
Figure 291: Poster

333

ASIA PACIFIC UNIVERSITY OF TECHNOLOGY & INNOVATION

## Appendix E – Gantt Chart

### Gantt Chart for FYP IR Semester 1



*Figure 292: Gantt Chart of Semester 1*

## Gantt Chart for FYP Semester 2

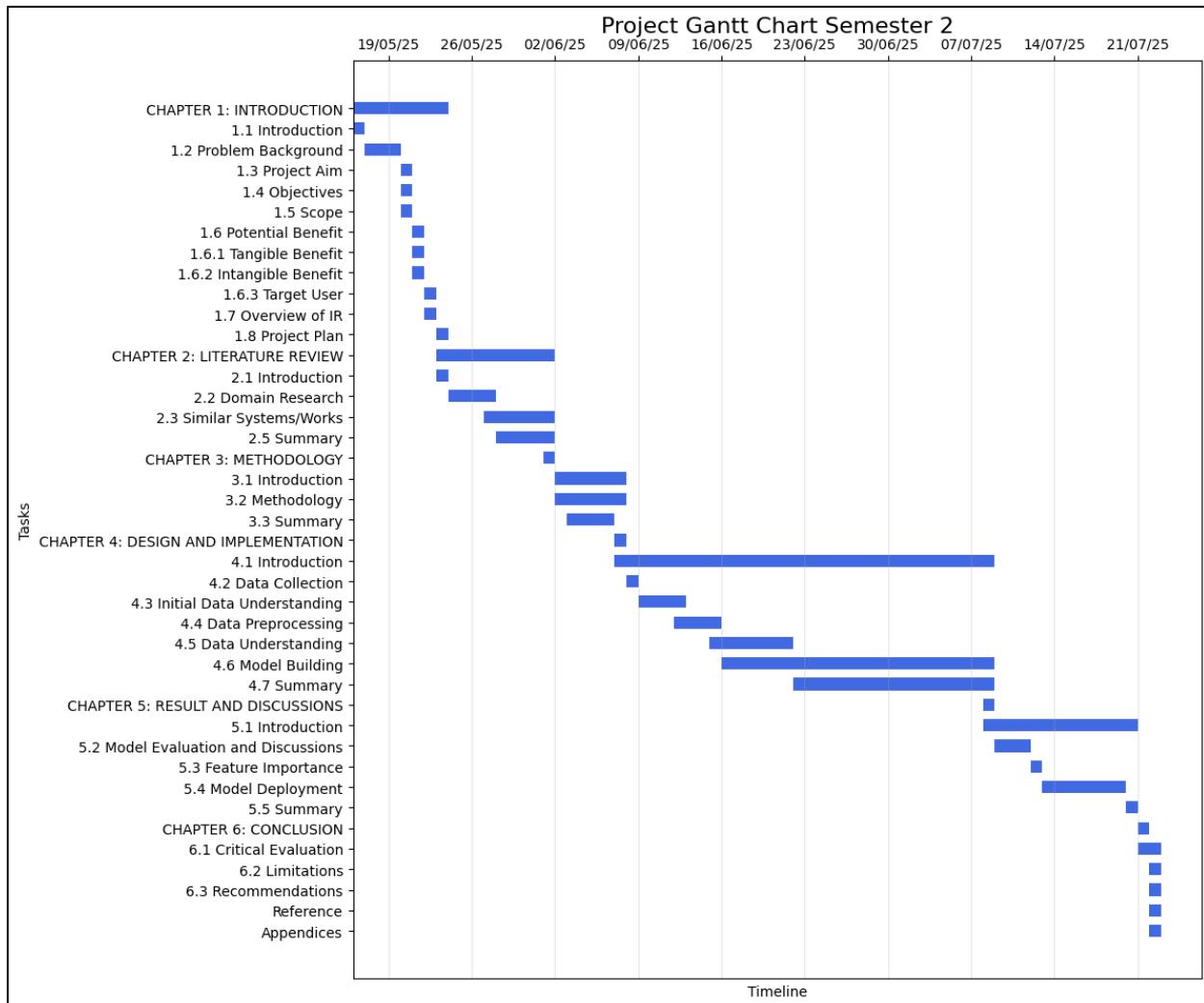


Figure 293: Gantt Chart of Semester 2

## Appendix F – Source Codes

### Preprocessing To Model Building

#### 1.0 Initial Data Understanding

##### 1.1 Load Datasets

```
import os
import pandas as pd
from tabulate import tabulate

# Define the file path
file_path = "Extended_Employee_Performance_and_Productivity_Data.csv"

# Check if the file exists in the current directory
if os.path.isfile(file_path):
    df = pd.read_csv(file_path)
    print("Dataset loaded successfully!")

    # Display the first five rows of the dataset
    print("\nFirst five rows of the dataset:")
    print(tabulate(df.head(), headers='keys', tablefmt='pretty', showindex=False))
else:
    print(f"Error: The file '{file_path}' was not found!")

[]
```

##### 1.2 Observations

```
# (Rows, Columns)
print("(Rows, Columns)")
print(df.shape , "\n")

# View the data types and missing values
print(df.info(), "\n")

# Count of duplicate rows
print("[Count of duplicate rows]")
print(df.duplicated().sum(), "\n")

# check columns
print("[Columns]")
print(df.columns, "\n")
```

##### 1.3 Mean, Std

```
df.describe()
```

```

import pandas as pd
import matplotlib.pyplot as plt

# Summary Statistics
summary = df.describe().transpose()
fig, ax = plt.subplots(figsize=(14, 6))
ax.axis("tight")
ax.axis("off")

# Create Table
table = ax.table(cellText=summary.round(2).values,
                  colLabels=summary.columns,
                  rowLabels=summary.index,
                  cellLoc="center",
                  loc="center")

# Adjust table properties
table.auto_set_font_size(False)
table.set_fontsize(12)

# Column width
for i in range(len(summary.columns)):
    table.auto_set_column_width([i])

for key, cell in table.get_celld().items():
    cell.set_height(0.05)
    cell.set_width(0.1)

plt.show()

```

## 1.4 View Categorical Variables

```

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

categorical_cols = ['Department', 'Gender', 'Job_Title', 'Education_Level', 'Performance_Score', 'Promotions']

sns.set_style("white")

for col in categorical_cols:
    plt.figure(figsize=(12, 6))
    ax = sns.countplot(data=df, x=col, edgecolor="black", alpha=0.7, color="#D291BC")

    # Add count labels on top of bars
    for p in ax.patches:
        ax.annotate(f'{int(p.get_height())}', 
                    (p.get_x() + p.get_width() / 2., p.get_height(), 
                     ha='center', va='bottom', fontsize=10, color='black')

    plt.title(f"Bar Chart of {col}")
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.grid(False)
    plt.show()

```

## 1.5 View Numerical Variables

```

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

numerical_cols = [
    'Age', 'Years_At_Company', 'Monthly_Salary', 'Work_Hours_Per_Week',
    'Projects_Handled', 'Overtime_Hours', 'Sick_Days', 'Remote_Work_Frequency',
    'Team_Size', 'Training_Hours', 'Employee_Satisfaction_Score'
]

sns.set_style("whitegrid")

for col in numerical_cols:
    plt.figure(figsize=(12, 7))

    counts, bins, patches = plt.hist(df[col], bins=10, edgecolor="black", alpha=0.7, color="#D291BC")
    bin_centers = (bins[:-1] + bins[1:]) / 2
    plt.plot(bin_centers, counts, marker="o", linestyle="--", color="#C21E56", label=f"{col} Distribution", linewidth=2)

    # Counts & percentages
    total = counts.sum()
    for count, patch in zip(counts, patches):
        height = patch.get_height()
        if height > 0:
            plt.text(patch.get_x() + patch.get_width()/2, height + 100,
                     f"{int(count)}\n({count/total:.1%})",
                     ha="center", va="bottom", fontsize=10, color="black")

    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.title(f"Histogram of {col}")
    plt.legend()
    plt.ylim(0, max(counts) * 1.2)
    plt.grid(False)
    plt.show()

```

## 1.4 Check Total Number of Resigned and Non-Resigned Employees

```

> 
import pandas as pd
import matplotlib.pyplot as plt

# Count the number of employees who resigned and did not resign
resigned_counts = df["Resigned"].value_counts()

# Counts and Percentages
labels = [f"{category}\n{count} ({count/sum(resigned_counts)*100:.1f}%)"
          for category, count in resigned_counts.items()]
plt.figure(figsize=(6, 7))
plt.pie(resigned_counts, labels=labels, autopct='%1.1f%%',
        colors=[ "#D291BC", "#EE82EE"], startangle=90, shadow=True)
plt.title("Employee Resignation Distribution")
plt.axis('equal')

print(resigned_counts.to_string(index=True))
print(f"Total Employees: {resigned_counts.sum()}")

plt.show()

```

## 1.5 Correlation Heatmap

```

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

# Convert categorical to numerical
df["Resigned"] = df["Resigned"].astype(int)

numeric_df = df.select_dtypes(include=[np.number])
correlation_matrix = numeric_df.corr()
plt.figure(figsize=(12, 8)) # Set figure size
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title("Correlation Heatmap of Employee Attributes")
plt.show()

```

## 2.0 Data Preprocessing - Missing Values and Outliers

### 2.1 Missing Values

```

missing_values = df.isnull().sum()
print(missing_values)
[8]

```

## 2. Check Outliers

### 2.1 Outliers for Categorical Variables

```

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

categorical_cols = ['Department', 'Gender', 'Job_Title', 'Education_Level', 'Performance_Score', 'Promotions']

sns.set_style("white")

# Define outliers
threshold = 5

# Identify outliers
for col in categorical_cols:
    category_counts = df[col].value_counts()
    rare_categories = category_counts[category_counts < threshold].index.tolist()

    if rare_categories:
        print(f"Outliers in {col}: {rare_categories}")
    else:
        print(f"Outliers in {col}: None")

for col in categorical_cols:
    plt.figure(figsize=(12, 6))
    ax = sns.countplot(data=df, x=col, edgecolor="black", alpha=0.7, color="#D291BC")

    for p in ax.patches:
        ax.annotate(f'{int(p.get_height())}', 
                    (p.get_x() + p.get_width() / 2., p.get_height()), 
                    ha='center', va='bottom', fontsize=10, color='black')

    plt.title(f"Bar Chart for {col}")
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.grid(False)
    plt.show()

```

## 2.2 Outliers for Numerical Variables

```

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

df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], errors='coerce')

numerical_columns = ['Age', 'Years_At_Company', 'Monthly_Salary',
                     'Work_Hours_Per_Week', 'Projects_Handled',
                     'Overtime_Hours', 'Sick_Days', 'Remote_Work_Frequency',
                     'Team_Size', 'Training_Hours', 'Employee_Satisfaction_Score']

# IQR
def detect_outliers(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return data[(data < lower_bound) | (data > upper_bound)]

outlier_results = []

for col in numerical_columns:
    outliers = detect_outliers(df[col])
    if not outliers.empty:
        outlier_results.append(f"Outliers in {col}: {outliers.shape[0]}")
    else:
        outlier_results.append(f"Outliers in {col}: None")

    # Boxplot
    plt.figure(figsize=(8, 6))
    sns.boxplot(y=df[col], color="#D291BC", orient='h')
    plt.title(f'Boxplot for {col}')
    plt.show()

for result in outlier_results:
    print(result)

```

## 3.0 Second part Data Preprocessing

### 3.1 Data Transformation - Resigned

```

import pandas as pd
from tabulate import tabulate

df["Resigned"] = df["Resigned"].replace({'True': 1, 'False': 0}).astype(int)

random_rows = df[["Employee_ID", "Resigned"]].sample(n=10)

print("\n [Resigned] - 10 random rows printed\n")
print(tabulate(random_rows, headers="keys", tablefmt="pretty", showindex=False))

```

### 3.2 Feature Engineering

#### Combine Hired Date and Remove Time into (Tenure\_Years)

```

import pandas as pd
from tabulate import tabulate
from dateutil.relativedelta import relativedelta

df['Hire_Date'] = pd.to_datetime(df['Hire_Date']).dt.date
df['Hire_Year'] = pd.to_datetime(df['Hire_Date']).dt.year
df['Hire_Month'] = pd.to_datetime(df['Hire_Date']).dt.month

# Calculate Tenure in years from today's date
df['Tenure_Years'] = df['Hire_Date'].apply(lambda d: round(
    relativedelta(pd.to_datetime('today').date(), d).years +
    relativedelta(pd.to_datetime('today').date(), d).months / 12, 2
))

random_rows = df[['Employee_ID', 'Hire_Date', 'Hire_Year', 'Hire_Month', 'Tenure_Years']].sample(n=10)
random_rows_reset = random_rows.reset_index(drop=True)

print(tabulate(random_rows_reset, headers="keys", tablefmt="pretty"))

# After showing drop hire date and years at company
df = df.drop(columns=['Hire_Date', 'Years_At_Company'], errors='ignore')

```

#### Drop and Show Current Columns

```

from dateutil.relativedelta import relativedelta
import pandas as pd
from tabulate import tabulate

# Health Index Calculation
max_lost_hours = (df['Sick_Days'] * (df['Work_Hours_Per_Week'] / 5)).max()
df['Health_Index'] = round((df['Sick_Days'] * (df['Work_Hours_Per_Week'] / 5)) / max_lost_hours, 2)

# Remote Work Level Categorization
df['Remote_Work_Level'] = pd.cut(df['Remote_Work_Frequency'],
                                   bins=[29, 47, 64, 80],
                                   labels=['Low', 'Medium', 'High'],
                                   include_lowest=True)

# Drop unnecessary columns
columns_to_drop = ['Remote_Work_Frequency', 'Gender']
df = df.drop(columns=[col for col in columns_to_drop if col in df.columns], errors='ignore')

# --- Data Formatting ---
df = df.round(2)

# --- Output Verification ---
print(tabulate(df[['Tenure_Years', 'Health_Index']].head(), headers="keys", tablefmt="pretty"))
print("\nColumns in the DataFrame after feature engineering:\n")
print(df.columns)

```

#### Check New Variables

```
df[['Health_Index', 'Tenure_Years']].describe()
```

### Distribution for health index and tenure years

```

import matplotlib.pyplot as plt
import seaborn as sns

fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# Health_Index
sns.histplot(df['Health_Index'], kde=True, bins=30, ax=axes[0], alpha=0.7, color="#D291BC")
axes[0].set_title('Distribution of Health Index')

# Tenure_Years
sns.histplot(df['Tenure_Years'], kde=True, bins=30, ax=axes[1], alpha=0.7, color="#D291BC")
axes[1].set_title('Distribution of Tenure Years')

plt.show()
]

```

### Encoding

```

import pandas as pd
from tabulate import tabulate
from sklearn.preprocessing import LabelEncoder

# One-Hot Encoding for Department and Job_Title
df_encoded = pd.get_dummies(df, columns=['Department', 'Job_Title'], drop_first=True)

# Ordinal Encoding for Education_Level
education_order = ['High School', 'Bachelor', 'Master', 'PhD']
df_encoded['Education_Level'] = pd.Categorical(df_encoded['Education_Level'], categories=education_order, ordered=True)
df_encoded['Education_Level'] = df_encoded['Education_Level'].cat.codes

# Ordinal Encoding for Remote_Work_Level
df_encoded['Remote_Work_Level'] = pd.Categorical(df_encoded['Remote_Work_Level'], categories=['Low', 'Medium', 'High'], ordered=True)
df_encoded['Remote_Work_Level'] = df_encoded['Remote_Work_Level'].cat.codes

# Treat 'Promotions' and 'Performance_Score' as categorical
df_encoded['Promotions'] = df_encoded['Promotions'].astype('category')
df_encoded['Performance_Score'] = df_encoded['Performance_Score'].astype('category')

# Select columns to display
selected_columns = ['Employee_ID', 'Education_Level', 'Remote_Work_Level']

print("Employee_ID, Education_Level, Remote_Work_Level (First 5 rows):")
print(tabulate(df_encoded[selected_columns].head(), headers="keys", tablefmt="pretty"))

# One-Hot Encoded Department and Job_Title
print("\nOne-Hot Encoded Department and Job_Title (First 5 rows):")
print(tabulate(df_encoded[df_encoded.columns[df_encoded.str.startswith('Department') | df_encoded.columns.str.startswith('Job_Title')]].head(), headers="keys", tablefmt="pretty"))

```

### Check data types after conversion

```

# Check the data types after conversion
print(df_encoded.dtypes)

```

## 4.0 Data Understanding after Data Preprocessing

### 4.1 Visualiation

```

import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")
bar_color = "#D291BC"

# Remote Work Levels
plt.figure(figsize=(9, 7))
ax = sns.countplot(x='Remote_Work_Level', data=df_encoded, color=bar_color, edgecolor="black")
remote_counts = df_encoded['Remote_Work_Level'].value_counts()
total_remote = remote_counts.sum()

for p in ax.patches:
    x = p.get_x() + p.get_width() / 2
    y = p.get_height()
    percentage = (y / total_remote) * 100
    ax.annotate(f'{int(y)}\n({percentage:.2f}%)', (x, y + 50), ha='center', va='bottom', fontsize=10, color='black')

plt.title('Bar Chart of Remote Work Levels')
plt.ylabel('Count')
plt.grid(False)
plt.show()

# Education Levels
plt.figure(figsize=(9, 7))
ax = sns.countplot(x='Education_Level', data=df_encoded, color=bar_color, edgecolor="black")
education_counts = df_encoded['Education_Level'].value_counts()
total_education = education_counts.sum()

for p in ax.patches:
    x = p.get_x() + p.get_width() / 2
    y = p.get_height()
    percentage = (y / total_education) * 100
    ax.annotate(f'{int(y)}\n({percentage:.2f}%)', (x, y + 50), ha='center', va='bottom', fontsize=10, color='black')

plt.title('Bar Chart of Education Levels')
plt.ylabel('Count')
plt.grid(False)
plt.show()

# Department
plt.figure(figsize=(10, 8))
department_columns = [col for col in df_encoded.columns if col.startswith('Department')]
department_counts = df_encoded[department_columns].sum()
total_department = department_counts.sum()
ax = department_counts.plot(kind='bar', color=bar_color, edgecolor="black")

for i, count in enumerate(department_counts):
    percentage = (count / total_department) * 100
    plt.text(i, count + 0.05, f'{count}\n({percentage:.2f}%)', ha='center', va='bottom', fontsize=10, color='black')

```

```
plt.title('Bar Chart of Education Levels')
plt.ylabel('Count')
plt.grid(False)
plt.show()

# Department
plt.figure(figsize=(10, 8))
department_columns = [col for col in df_encoded.columns if col.startswith('Department')]
department_counts = df_encoded[department_columns].sum()
total_department = department_counts.sum()
ax = department_counts.plot(kind='bar', color=bar_color, edgecolor="black")

for i, count in enumerate(department_counts):
    percentage = (count / total_department) * 100
    plt.text(i, count + 0.05, f'{count}\n({percentage:.2f}%)', ha='center', va='bottom', fontsize=10, color='black')

plt.title('Bar Chart of Department Categories')
plt.ylabel('Count')
plt.xticks(rotation=45, ha="right")
plt.grid(False)
plt.show()

# Job Title
plt.figure(figsize=(10, 8))
job_title_columns = [col for col in df_encoded.columns if col.startswith('Job_Title')]
job_title_counts = df_encoded[job_title_columns].sum()
total_job_title = job_title_counts.sum()
ax = job_title_counts.plot(kind='bar', color=bar_color, edgecolor="black")

for i, count in enumerate(job_title_counts):
    percentage = (count / total_job_title) * 100
    plt.text(i, count + 0.05, f'{count}\n({percentage:.2f}%)', ha='center', va='bottom', fontsize=10, color='black')

plt.title('Bar Chart of Job Title Categories')
plt.ylabel('Count')
plt.xticks(rotation=45, ha="right")
plt.grid(False)
plt.show()
```

## 5.0 Feature Selection

### Numerical

```

from sklearn.feature_selection import mutual_info_classif
import matplotlib.pyplot as plt

# Select the numerical columns (excluding certain columns from the list)
numerical_columns = [col for col in df.select_dtypes(include=['int64', 'float64']).columns
                     if col not in ['Promotions', 'Performance_Score', 'Remote_Work_Level', 'Employee_ID', 'Resigned']]

# Define X as features and y as target variable
X_mi = df[numerical_columns]
y = df['Resigned']

# Calculate mutual information
mi_numerical = mutual_info_classif(X_mi, y, random_state=42)

# Create DataFrame to store MI scores
print("\n Numerical Features - Mutual Information: \n")
mi_df = pd.DataFrame({'Variable': X_mi.columns, 'MI Score': mi_numerical}).sort_values(by='MI Score', ascending=False)

# Reset index and start from 1
mi_df.reset_index(drop=True, inplace=True)
mi_df.index = mi_df.index + 1

# Print MI DataFrame without the old index
print(mi_df)

# Different thresholds to see the impact on selected features
thresholds = [0.001, 0.005]
for threshold in thresholds:
    selected_variable = mi_df[mi_df['MI Score'] > threshold]['Variable'].tolist()
    print(f"Top 5 Variables with threshold {threshold}: {selected_variable}")

# Plotting the mutual information scores
plt.figure(figsize=(12, 6))
plt.barh(mi_df['Variable'], mi_df['MI Score'], color="#D291BC")
plt.xlabel('Mutual Information Score')
plt.title('Mutual Information Scores for Numerical Variables')
plt.grid(False)
plt.gca().set_facecolor('white')
plt.tight_layout()
plt.show()

```

## Categorical

```

from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn.feature_selection import mutual_info_classif
import matplotlib.pyplot as plt

# List of categorical columns
categorical_columns = [col for col in df.select_dtypes(include=['object']).columns if col not in ['Employee_ID', 'Resigned']]

# Variables that have already been encoded with LabelEncoder
label_encode_columns = ['Promotions', 'Performance_Score', 'Education_Level', 'Remote_Work_Level']

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Apply LabelEncoder to specific columns
for col in label_encode_columns:
    df[col] = label_encoder.fit_transform(df[col])

# One-hot encode only the remaining categorical columns
remaining_categorical_columns = [col for col in categorical_columns if col not in label_encode_columns]
encoder = OneHotEncoder(drop='first')
X_encoded_sparse = encoder.fit_transform(df[remaining_categorical_columns])
X_encoded_dense = X_encoded_sparse.toarray()

# Combine label-encoded and one-hot encoded data
X_label_encoded = df[label_encode_columns]
X_combined = pd.concat([X_label_encoded,
                        pd.DataFrame(X_encoded_dense, columns=encoder.get_feature_names_out(remaining_categorical_columns))], axis=1)
y = df['Resigned']

# Calculate mutual information for categorical variables
mutual_info_cat = mutual_info_classif(X_combined, y, random_state=42)

# DataFrame for Categorical Mutual Information
mutual_info_cat_df = pd.DataFrame({
    'Feature': X_combined.columns,
    'MI Score': mutual_info_cat
}).sort_values(by='MI Score', ascending=False)

# Reset the index to start from 1
mutual_info_cat_df.reset_index(drop=True, inplace=True)
mutual_info_cat_df.index = mutual_info_cat_df.index + 1

# Print Mutual Information Scores
print("\nCategorical Features - Mutual Information: \n")
print(mutual_info_cat_df)
top_n_cat_features = mutual_info_cat_df.head(5)['Feature'].tolist()
print("\nTop 5 Categorical Features:")
print(top_n_cat_features)
plt.figure(figsize=(12, 6))
plt.barh(mutual_info_cat_df['Feature'], mutual_info_cat_df['MI Score'], color="#D291BC")
plt.xlabel('Mutual Information Score')
plt.title('Mutual Information Scores for Categorical Features')
plt.grid(False)
plt.show()

```

### Merge and Rank for Top 10

```

# Top 5 numerical features by MI score
top_n_num_features = mi_df.sort_values(by='MI Score', ascending=False).head(5)['Variable'].tolist()

# Top 5 categorical features by MI score
top_n_cat_features = mutual_info_cat_df.sort_values(by='MI Score', ascending=False).head(5)['Feature'].tolist()

# Combine them into Top 10 Features
top_10_features = top_n_num_features + top_n_cat_features

print("Top 5 Numerical Features (by MI):")
for i, feat in enumerate(top_n_num_features, start=1):
    print(f"{i}. {feat}")

print("\nTop 5 Categorical Features (by MI):")
for i, feat in enumerate(top_n_cat_features, start=1):
    print(f"{i}. {feat}")

print("\nCombination of Top 10 Features:")
for i, feat in enumerate(top_10_features, start=1):
    print(f"{i}. {feat}")

```

### 6.0 Preparation before building model - Train and Test , SMOTE

#### Check Target Variable Count]

```

# Check Target Variable
print(df['Resigned'].value_counts())

```

### Split TRAIN and TEST

```

from sklearn.model_selection import train_test_split

X_cat_top = X_combined[top_n_cat_features].reset_index(drop=True)
X_num_top = df[top_n_num_features].reset_index(drop=True)
X = pd.concat([X_cat_top, X_num_top], axis=1)

y = df['Resigned']

# Split into train (80%) and test (20%)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# Train set
print("[Train] class distribution:")
train_counts = y_train.value_counts()
print(train_counts)
print("Total:", train_counts.sum(), "\n")

# Test set
print("[Test] class distribution:")
test_counts = y_test.value_counts()
print(test_counts)
print("Total:", test_counts.sum())

```

## SMOTE Techniques for class imbalance

```
from imblearn.over_sampling import SMOTE

# Apply SMOTE only to training set
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

print("Class distribution before SMOTE:\n", y_train.value_counts())
print("\nClass distribution after SMOTE:\n", pd.Series(y_train_resampled).value_counts())
```

## Visualition after doing class imbalance

```
import matplotlib.pyplot as plt

# Before SMOTE
ax = y_train.value_counts().plot(kind='bar', title='Class Distribution Before SMOTE', color="#D291BC")
plt.xticks([0, 1], ['Not Resigned', 'Resigned'], rotation=0)
plt.ylabel('Count')

for p in ax.patches:
    ax.annotate(f'{p.get_height()}', 
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', fontsize=10, color='black', xytext=(2, 10), textcoords='offset points')
plt.ylim(0, max(y_train.value_counts()) + 10000)
plt.grid(False)
plt.tick_params(axis="x", bottom=False)
plt.show()

# After SMOTE
ax = y_train_resampled.value_counts().plot(kind='bar', title='Class Distribution After SMOTE', color="#D291BC")
plt.xticks([0, 1], ['Not Resigned', 'Resigned'], rotation=0)
plt.ylabel('Count')

for p in ax.patches:
    ax.annotate(f'{p.get_height()}', 
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', fontsize=10, color='black', xytext=(0, 10), textcoords='offset points')
plt.ylim(0, max(y_train.value_counts()) + 10000)
plt.grid(False)
plt.tick_params(axis="x", bottom=False)
plt.show()
```

```
print("[Test] class distribution:")
test_counts = y_test.value_counts()
print(test_counts)
print("Total:", test_counts.sum())
```

## 7.0 Model Building

### 7.1 Random Forest

```
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.ensemble import RandomForestClassifier

# Train on resampled data
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train_resampled, y_train_resampled)

# Predict on test set
y_pred = rf_model.predict(X_test)

# Evaluation
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D291BC"])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('Random Forest - Confusion Matrix (Base)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'])
plt.yticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()
```

### Random Forest - Learning Curve (Base)

```

from sklearn.model_selection import learning_curve, StratifiedKFold
import numpy as np

# Stratified Cross-Validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Learning curve
train_sizes, train_scores, test_scores = learning_curve(
    estimator=rf_model,
    X=X_train_resampled,
    y=y_train_resampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

# Mean and Std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Learning curve
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label='Training Score', color='blue')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color='blue')
plt.plot(train_sizes, test_mean, label='Cross-validation Score', color='red')
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color='red')
plt.title('Learning Curve - Random Forest (Base)', fontsize=14)
plt.xlabel('Training Set Size')
plt.ylabel('F1 Score')
plt.legend(loc='best')
plt.grid(True)
plt.tight_layout()
plt.show()

```

### Random Forest - ROC and AUC (Base)

```

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

y_probs = rf_model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_probs)
auc_score = roc_auc_score(y_test, y_probs)

plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})')
plt.plot([0,1], [0,1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Random Forest (Base)', fontsize=14)
plt.legend()
plt.grid()
plt.show()

```

### Random Forest (Tuned)

```

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from matplotlib.colors import LinearSegmentedColormap
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

# Parameter space
param_dist = {
    'n_estimators': [100, 200, 300, 400, 500],
    'max_depth': [10, 20, 30, 40],
    'min_samples_split': [5, 10, 15],
    'min_samples_leaf': [2, 4, 6],
    'max_features': ['sqrt', 'log2'],
    'class_weight': ['balanced']
}

# Stratified CV
stratified_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

rf = RandomForestClassifier(random_state=42)
random_search = RandomizedSearchCV(
    estimator=rf,
    param_distributions=param_dist,
    n_iter=360,
    cv=stratified_cv,
    scoring='f1',
    verbose=1,
    random_state=42,
    n_jobs=-1
)
random_search.fit(X_train_resampled, y_train_resampled)

# Evaluate best model
best_randomforest = random_search.best_estimator_
y_pred_best = best_randomforest.predict(X_test)

# Best parameters
best_params = random_search.best_params_
print("\n---- Best Hyperparameters for Random Forest ---")
print(f"Number of Trees (n_estimators): {best_params['n_estimators']}")
print(f"Maximum Depth (max_depth): {best_params['max_depth']}")
print(f"Minimum Samples to Split (min_samples_split): {best_params['min_samples_split']}")
print(f"Minimum Samples at Leaf (min_samples_leaf): {best_params['min_samples_leaf']}")
print(f"Max Features per Split (max_features): {best_params['max_features']}")
print(f"Class Weight: {best_params['class_weight']}")

# Confusion matrix
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_best))
print("\nClassification Report:\n", classification_report(y_test, y_pred_best))
print("Accuracy Score:", accuracy_score(y_test, y_pred_best))

# Confusion matrix plot
cm = confusion_matrix(y_test, y_pred_best)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D291BC"])

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('Random Forest - Confusion Matrix (Tuned)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'])
plt.yticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

### Random Forest - Learning Curve (Tuned)

```

from sklearn.model_selection import learning_curve
import numpy as np
import matplotlib.pyplot as plt

# Learning curve
train_sizes, train_scores, test_scores = learning_curve(
    estimator=best_randomforest,
    X=X_train_resampled,
    y=y_train_resampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

# Means and Std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Learning curve
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label='Training Score', color='blue')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color='blue')
plt.plot(train_sizes, test_mean, label='Cross-validation Score', color='red')
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color='red')
plt.title('Learning Curve - Random Forest (Tuned)', fontsize=14)
plt.xlabel('Training Set Size')
plt.ylabel('F1 Score')
plt.legend(loc='best')
plt.grid(True)
plt.tight_layout()
plt.show()

```

### Random Forest - ROC and AUC (Tuned)

```

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

y_probs = best_randomforest.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_probs)
auc_score = roc_auc_score(y_test, y_probs)

plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})')
plt.plot([0,1], [0,1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Random Forest (Tuned)', fontsize=14)
plt.legend()
plt.grid()
plt.show()

```

### Random Forest - Feature Importance

```

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

# Get feature importances from the best Random Forest model
rf_importances = pd.Series(best_randomforest.feature_importances_, index=X.columns)
top_10_rf = rf_importances.sort_values(ascending=False).head(10)

# Print top 10
print("Top 10 Important Features (Random Forest):")
print(top_10_rf)

# Plot
plt.figure(figsize=(8, 6))
sns.barplot(
    x=top_10_rf.values,
    y=top_10_rf.index,
    color="#D91BC"
)
plt.title('Top 10 Important Features - Random Forest (Tuned)', fontsize=14)
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.tight_layout()
plt.show()

```

### 7.2 XGBoost

```

import xgboost as xgb
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap

# Initialize and fit the model
xgb_model = xgb.XGBClassifier(random_state=42)
xgb_model.fit(X_train_resampled, y_train_resampled)

# Predict on the test set
y_pred = xgb_model.predict(X_test)

# Evaluation
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Plot confusion matrix
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D91BC" ])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('XGBoost - Confusion Matrix (Base)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'])
plt.yticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

### XG Boost - Learning Curve (Base)

```

from sklearn.model_selection import learning_curve, StratifiedKFold
import numpy as np
import matplotlib.pyplot as plt

# Stratified Cross-Validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Learning curve data
train_sizes, train_scores, test_scores = learning_curve(
    estimator=xgb_model,
    X=X_train_resampled,
    y=y_train_resampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

# Mean and std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Learning curve
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label='Training Score', color='blue')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color='blue')
plt.plot(train_sizes, test_mean, label='Cross-validation Score', color='red')
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color='red')
plt.title('Learning Curve - XGBoost (Base)', fontsize=14)
plt.xlabel('Training Set Size')
plt.ylabel('F1 Score')
plt.legend(loc='best')
plt.grid(True)
plt.tight_layout()
plt.show()

```

### XG Boost - ROC and AUC (Base)

```

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

# Get predicted probabilities for class 1 (Resigned)
y_probs = xgb_model.predict_proba(X_test)[:, 1]

# Compute ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
auc_score = roc_auc_score(y_test, y_probs)

# Plot ROC curve
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})', color='blue')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - XGBoost (Base)')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()

```

XG Boost (Random) - Tuned

```

import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from scipy.stats import randint, uniform
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap
import numpy as np

# StratifiedKFold
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# XGBoost model
xgb_model = xgb.XGBClassifier(
    objective='binary:logistic',
    eval_metric='logloss',
    random_state=42
)

# Hyperparameter space
param_dist = {
    'n_estimators': randint(100, 500),
    'max_depth': randint(3, 15),
    'learning_rate': uniform(0.01, 0.1),
    'subsample': uniform(0.6, 0.3),
    'colsample_bytree': uniform(0.6, 0.3),
    'gamma': uniform(0, 5),
    'reg_lambda': uniform(0.1, 1),
    'reg_alpha': uniform(0, 1)
}

# Randomized Search
random_search = RandomizedSearchCV(
    estimator=xgb_model,
    param_distributions=param_dist,
    n_iter=500,
    scoring='f1',
    cv=cv,
    verbose=2,
    random_state=42,
    n_jobs=-1
)
random_search.fit(X_train_resampled, y_train_resampled)

# Best estimator
best_xgboost = random_search.best_estimator_
y_pred = best_xgboost.predict(X_test)

# Best hyperparameter
best_params = random_search.best_params_
print("\n--- Best Hyperparameters for XGBoost ---")
print(f"Number of Trees (n_estimators): {best_params['n_estimators']}")
print(f"Maximum Tree Depth (max_depth): {best_params['max_depth']}")
print(f"Learning Rate: {best_params['learning_rate']:.4f}")
print(f"Subsample Ratio: {best_params['subsample']:.4f}")
print(f"Column Subsample (colsample_bytree): {best_params['colsample_bytree']:.4f}")
print(f"Gamma (Minimum Loss Reduction): {best_params['gamma']:.4f}")
print(f"L2 Regularization (reg_lambda): {best_params['reg_lambda']:.4f}")
print(f"L1 Regularization (reg_alpha): {best_params['reg_alpha']:.4f}")

# Evaluation
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Confusion Matrix Plot
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#fffffe", "#0291BC" ])

plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('XGBoost - Confusion Matrix (Tuned)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks([0, 1], ['Not Resigned', 'Resigned'])
plt.yticks([0, 1], ['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

### XG Boost - Learning Curve (Tuned)

```

from sklearn.model_selection import learning_curve
import numpy as np
import matplotlib.pyplot as plt

# Learning Curve
train_sizes, train_scores, test_scores = learning_curve(
    estimator=best_xgboost,
    X=X_train_resampled,
    y=y_train_resampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

# Mean and std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label='Training Score', color='blue')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color='blue')
plt.plot(train_sizes, test_mean, label='Cross-validation Score', color='red')
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color='red')
plt.title('Learning Curve - XGBoost (Tuned)', fontsize=14)
plt.xlabel('Training Set Size')
plt.ylabel('F1 Score')
plt.legend(loc='best')
plt.grid(True)
plt.tight_layout()
plt.show()

```

### XG Boost - ROC and AUC (Tuned)

```

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

# Predict probabilities
y_proba = best_xgboost.predict_proba(X_test)[:, 1]

# Compute ROC curve and AUC
fpr, tpr, _ = roc_curve(y_test, y_proba)
auc_score = roc_auc_score(y_test, y_proba)

# ROC curve
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})', color='blue')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - XGBoost (Tuned)')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()

```

### XG Boost - Feature Importance

```

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

# Create a Series with feature importances
feature_importance = pd.Series(best_xgboost.feature_importances_, index=X.columns)

# Sort top 10 features
top_10_features = feature_importance.sort_values(ascending=False).head(10)

print("\nTop 10 Important Features (XGBoost):")
print(top_10_features)

plt.figure(figsize=(10, 6))
sns.barplot(
    x=top_10_features.values,
    y=top_10_features.index,
    color="#D291BC"
)
plt.title('Top 10 Feature Importances (XGBoost)', fontsize=14)
plt.xlabel('Importance Score')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()

```

### 7.3 Gradient Boost

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

gb_model = GradientBoostingClassifier()
gb_model.fit(X_train_resampled, y_train_resampled)

# Predict on test set
y_pred = gb_model.predict(X_test)

# Evaluate
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Plot confusion matrix
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D291BC" ])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('Gradient Boosting - Confusion Matrix (Base)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks([0.5, 1.5], ['Not Resigned', 'Resigned'])
plt.yticks([0.5, 1.5], ['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

### Gradient Boost - Learning Curve (Base)

```

from sklearn.model_selection import StratifiedKFold, learning_curve
import numpy as np
import matplotlib.pyplot as plt

# Stratified K-Fold cross-validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Learning curve
train_sizes, train_scores, test_scores = learning_curve(
    estimator=gb_model,
    X=X_train_resampled,
    y=y_train_resampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

# Mean and Std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label='Training Score', color='blue')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color='blue')
plt.plot(train_sizes, test_mean, label='Cross-validation Score', color='red')
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color='red')
plt.title('Learning Curve - Gradient Boosting (Base)', fontsize=14)
plt.xlabel('Training Set Size')
plt.ylabel('F1 Score')
plt.legend(loc='best')
plt.grid(True)
plt.tight_layout()
plt.show()

```

### Gradient Boost - ROC and AUC (Base)

```

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

y_probs = gb_model.predict_proba(X_test)[:, 1]

# Calculate ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
auc_score = roc_auc_score(y_test, y_probs)

# Plot ROC Curve
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})', color='blue', linewidth=2)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Gradient Boosting (Base)', fontsize=14)
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()

```

```

Gradient Boost - Hyperparameters (Random)

from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from scipy.stats import randint, uniform
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap
import numpy as np

param_dist = {
    'n_estimators': randint(50, 150),
    'learning_rate': uniform(0.01, 0.05),
    'max_depth': randint(3, 5),
    'min_samples_split': randint(10, 30),
    'min_samples_leaf': randint(5, 15),
    'subsample': uniform(0.7, 0.3)
}

# Cross-validation strategy
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Initialize model
gb = GradientBoostingClassifier(random_state=42)

# Randomized Search CV
random_search = RandomizedSearchCV(
    estimator=gb,
    param_distributions=param_dist,
    n_iter=500,
    scoring='f1',
    cv=cv,
    verbose=2,
    n_jobs=-1,
    random_state=42
)
random_search.fit(X_train_resampled, y_train_resampled)

# Best model
best_gradientboost = random_search.best_estimator_
y_pred = best_gradientboost.predict(X_test)

# Best hyperparameter
best_params = random_search.best_params_
print("\n--- Best Hyperparameters for Gradient Boosting ---")
print(f"Number of Trees (n_estimators): {best_params['n_estimators']}")
print(f"Learning Rate: {best_params['learning_rate']:.4f}")
print(f"Maximum Tree Depth (max_depth): {best_params['max_depth']}")
print(f"Min Samples to Split (min_samples_split): {best_params['min_samples_split']}")
print(f"Min Samples at Leaf (min_samples_leaf): {best_params['min_samples_leaf']}")
print(f"Subsample Ratio: {best_params['subsample']:.4f}")

# Evaluation
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D291BC" ])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('Gradient Boosting - Confusion Matrix (Tuned)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'])
plt.yticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

### Gradient Boost - Learning Curve

```

import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve

train_sizes, train_scores, test_scores = learning_curve(
    estimator=best_gradientboost,
    X=X_train_resampled,
    y=y_train_resampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    n_jobs=-1,
    scoring='f1'
)

# Mean and Std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Learning curve
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label="Training Score", color="blue")
plt.plot(train_sizes, test_mean, label="Cross-validation Score", color="red")
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color="blue")
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color="red")
plt.title("Learning Curve - Gradient Boosting (Tuned)", fontsize=14)
plt.xlabel("Training Set Size")
plt.ylabel("F1 Score")
plt.legend(loc="best")
plt.grid(True)
plt.tight_layout()
plt.show()

```

### Gradient Boost - ROC and AUC (Tuned)

```

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

y_probs = best_gradientboost.predict_proba(X_test)[:, 1]

# Calculate ROC curve & AUC score
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
auc_score = roc_auc_score(y_test, y_probs)

# 4. Plot ROC curve
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})', color='blue')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Gradient Boosting (Tuned)')
plt.legend(loc='lower right')
plt.grid()
plt.tight_layout()
plt.show()

```

### Gradient Boost - Feature Importance

```

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

# Get feature importances
feature_importance = pd.Series(best_gradientboost.feature_importances_, index=X.columns)
top_10_features = feature_importance.sort_values(ascending=False).head(10)

# Print top features
print("Top 10 Important Features (Gradient Boosting):")
print(top_10_features)

# Plot with custom color
plt.figure(figsize=(8, 6))
sns.barplot(
    x=top_10_features.values,
    y=top_10_features.index,
    color="#D91BC"
)
plt.title('Top 10 Important Features - Gradient Boosting', fontsize=14)
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.tight_layout()
plt.show()

```

### 7.4 Support Vector Machine

#### SVM (Base)- Random Run 50%

```

from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap
import numpy as np

# Take 50% sample subset from the resampled training data
subset_indices = np.random.choice(len(X_train_resampled), size=45938, replace=False)
X_train_subset = X_train_resampled.iloc[subset_indices]
y_train_subset = y_train_resampled.iloc[subset_indices]

svm_model_subset = SVC()
svm_model_subset.fit(X_train_subset, y_train_subset)

y_pred = svm_model_subset.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D91BC" ])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('SVM - Confusion Matrix (Base)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks([0.5, 1.5], ['Not Resigned', 'Resigned'])
plt.yticks([0.5, 1.5], ['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

### SVM - Learning Curve (Base)

```

from sklearn.model_selection import StratifiedKFold, learning_curve
import numpy as np
import matplotlib.pyplot as plt

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

train_sizes, train_scores, test_scores = learning_curve(
    estimator=svm_model_subset,
    X=X_train_subset,
    y=y_train_subset,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

# Mean and Std
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot learning curve
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label="Training Score", color="blue")
plt.plot(train_sizes, test_mean, label="Cross-validation Score", color="red")
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color="blue")
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color="red")
plt.title("Learning Curve - SVM (Base)", fontsize=14)
plt.xlabel("Training Set Size")
plt.ylabel("F1 Score")
plt.legend(loc="best")
plt.grid(True)
plt.tight_layout()
plt.show()

```

### SVM - ROC and AUC (Base)

```

from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

y_probs = svm_model_subset.decision_function(X_test)
fpr, tpr, _ = roc_curve(y_test, y_probs)
auc_score = auc(fpr, tpr)

plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})', color='blue')
plt.plot([0, 1], [0, 1], 'k--', label='Random Guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - SVM (Base)', fontsize=14)
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()

```

Support Vector Machine - Random (50%)

```

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from scipy.stats import uniform

param_dist = {
    'svm_C': uniform(0.1, 10),
    'svm_gamma': uniform(0.001, 1),
    'svm_kernel': ['rbf']
}

svm_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('svm', SVC(probability=True, random_state=42))
])

# Take 50% sample subset from resampled training data
subset_indices = np.random.choice(X_train_resampled.shape[0], size=45938, replace=False)
X_train_sampled = X_train_resampled.iloc[subset_indices]
y_train_sampled = y_train_resampled.iloc[subset_indices]

# Stratified 5-Fold Cross-Validation
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Perform RandomizedSearchCV
random_search = RandomizedSearchCV(
    estimator=svm_pipeline,
    param_distributions=param_dist,
    n_iter=20,
    scoring='f1',
    cv=cv,
    verbose=2,
    n_jobs=-1,
    random_state=42
)
random_search.fit(X_train_sampled, y_train_sampled)

best_svm_model = random_search.best_estimator_
y_pred = best_svm_model.predict(X_test)

# Print best parameters
best_params = random_search.best_params_
print("\n--- Best Hyperparameters for Support Vector Machine ---")
print(f"Regularization Parameter (C): {best_params['svm_C']:.4f}")
print(f"Kernel Coefficient (gamma): {best_params['svm_gamma']:.4f}")
print(f"Kernel Type: {best_params['svm_kernel']}")

# Evaluation metrics
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy Score:", accuracy_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
custom_cmap = LinearSegmentedColormap.from_list("custom", [ "#ffffff", "#D9EAD3" ])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap=custom_cmap, cbar=False, annot_kws={"color": "black"})
plt.title('Support Vector Machine - Confusion Matrix (Tuned)', fontsize=14)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'])
plt.yticks(ticks=[0.5, 1.5], labels=['Not Resigned', 'Resigned'], rotation=0)
plt.tight_layout()
plt.show()

```

### Support Vector Machine(Tuned) - Learning Curve

```

from sklearn.model_selection import learning_curve

# Generate learning curve data
train_sizes, train_scores, test_scores = learning_curve(
    estimator=best_svm_model,
    X=X_train_sampled,
    y=y_train_sampled,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=cv,
    scoring='f1',
    n_jobs=-1
)

# Calculate mean and standard deviation
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

# Plot learning curve
plt.figure(figsize=(8, 6))
plt.plot(train_sizes, train_mean, label="Training Score", color="blue")
plt.plot(train_sizes, test_mean, label="Cross-validation Score", color="red")
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, alpha=0.2, color="blue")
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, alpha=0.2, color="red")
plt.title("Learning Curve - Support Vector Machine (Tuned)", fontsize=14)
plt.xlabel("Training Set Size")
plt.ylabel("F1 Score")
plt.legend(loc="best")
plt.grid(True)
plt.tight_layout()
plt.show()

```

### Support Vector Machine (Tuned) - ROC and AUC Score

```

from sklearn.metrics import roc_curve, auc

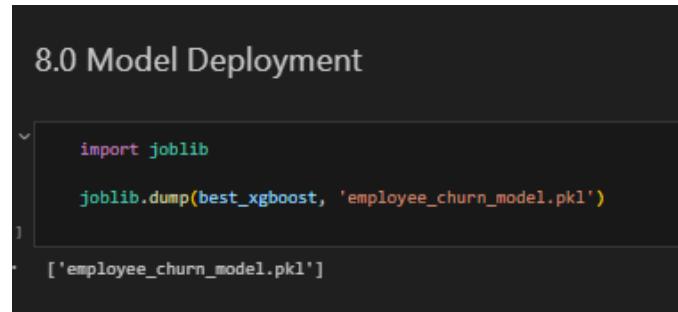
# Get predicted probabilities
y_probs = best_svm_model.predict_proba(X_test)[:, 1]

# Calculate ROC curve and AUC
fpr, tpr, _ = roc_curve(y_test, y_probs)
auc_score = auc(fpr, tpr)

# Plot ROC curve
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})', color='blue')
plt.plot([0, 1], [0, 1], 'k--', label='Random Guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Support Vector Machine (Tuned)', fontsize=14)
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()

```

## Model Deployment Save File



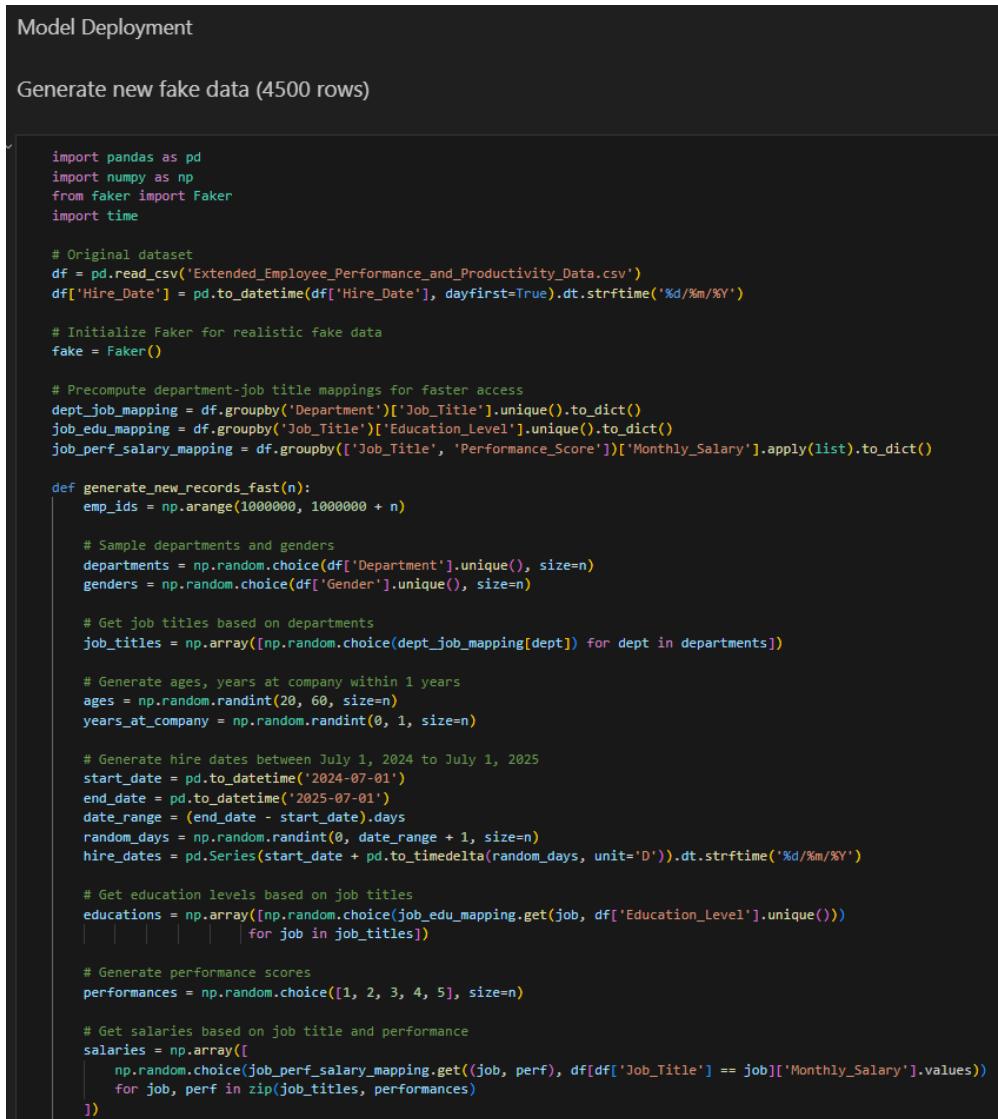
```

8.0 Model Deployment

import joblib
joblib.dump(best_xgboost, 'employee_churn_model.pkl')
['employee_churn_model.pkl']

```

## Generate Fake Data



Model Deployment

Generate new fake data (4500 rows)

```

import pandas as pd
import numpy as np
from faker import Faker
import time

# Original dataset
df = pd.read_csv('Extended_Employee_Performance_and_Productivity_Data.csv')
df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], dayfirst=True).dt.strftime('%d/%m/%Y')

# Initialize Faker for realistic fake data
fake = Faker()

# Precompute department-job title mappings for faster access
dept_job_mapping = df.groupby('Department')['Job_Title'].unique().to_dict()
job_edu_mapping = df.groupby('Job_Title')['Education_Level'].unique().to_dict()
job_perf_salary_mapping = df.groupby(['Job_Title', 'Performance_Score'])['Monthly_Salary'].apply(list).to_dict()

def generate_new_records(n):
    emp_ids = np.arange(1000000, 1000000 + n)

    # Sample departments and genders
    departments = np.random.choice(df['Department'].unique(), size=n)
    genders = np.random.choice(df['Gender'].unique(), size=n)

    # Get job titles based on departments
    job_titles = np.array([np.random.choice(dept_job_mapping[dept]) for dept in departments])

    # Generate ages, years at company within 1 years
    ages = np.random.randint(20, 60, size=n)
    years_at_company = np.random.randint(0, 1, size=n)

    # Generate hire dates between July 1, 2024 to July 1, 2025
    start_date = pd.to_datetime('2024-07-01')
    end_date = pd.to_datetime('2025-07-01')
    date_range = (end_date - start_date).days
    random_days = np.random.randint(0, date_range + 1, size=n)
    hire_dates = pd.Series(start_date + pd.to_timedelta(random_days, unit='D')).dt.strftime('%d/%m/%Y')

    # Get education levels based on job titles
    educations = np.array([np.random.choice(job_edu_mapping.get(job, df['Education_Level'].unique()))
                           for job in job_titles])

    # Generate performance scores
    performances = np.random.choice([1, 2, 3, 4, 5], size=n)

    # Get salaries based on job title and performance
    salaries = np.array([
        np.random.choice(job_perf_salary_mapping.get((job, perf), df[df['Job_Title'] == job]['Monthly_Salary'].values))
        for job, perf in zip(job_titles, performances)
    ])

```

```

# Generate other fields
work_hours = np.random.randint(30, 60, size=n)
projects = np.random.randint(0, 50, size=n)
overtime = np.random.randint(0, 30, size=n)
sick_days = np.random.randint(0, 15, size=n)
remote_freq = np.random.randint(30, 80, size=n)
team_size = np.random.randint(1, 20, size=n)
training_hours = np.random.randint(0, 20, size=n)
promotions = np.random.randint(0, 3, size=n)
satisfaction = np.round(np.random.uniform(1, 5, size=n), 2)

new_data = pd.DataFrame({
    'Employee_ID': emp_ids,
    'Department': departments,
    'Gender': genders,
    'Age': ages,
    'Job_Title': job_titles,
    'Hire_Date': hire_dates,
    'Years_At_Company': years_at_company,
    'Education_Level': educations,
    'Performance_Score': performances,
    'Monthly_Salary': salaries,
    'Work_Hours_Per_Week': work_hours,
    'Projects_Handled': projects,
    'Overtime_Hours': overtime,
    'Sick_Days': sick_days,
    'Remote_Work_Frequency': remote_freq,
    'Team_Size': team_size,
    'Training_Hours': training_hours,
    'Promotions': promotions,
    'Employee_Satisfaction_Score': satisfaction,
    'Resigned': False
})

return new_data

start_time = time.time()

# Generate 4,500 new records with July 2024-July 2025 hire dates
new_records = generate_new_records_fast(4500)

# Combine with original data
extended_df = pd.concat([df, new_records], ignore_index=True)
extended_df['Hire_Date'] = pd.to_datetime(extended_df['Hire_Date'], dayfirst=True)

# Create Resignation_Date column (initially empty)
extended_df['Resignation_Date'] = pd.NaT

# Get all resigned employees
resigned_mask = extended_df['Resigned'] == True
resigned = extended_df[resigned_mask]

# Randomly select 805 for Q1 (Jan-Mar 2025) and 478 for Q2 (Apr-Jun 2025)
q1_resignations = resigned.sample(805, random_state=42)
remaining_resigned = resigned.drop(q1_resignations.index)
q2_resignations = remaining_resigned.sample(478, random_state=42)
other_resignations = remaining_resigned.drop(q2_resignations.index)

```

```

# Assign Q1 resignation dates (Jan-Mar 2025)
q1_start = pd.to_datetime('01/01/2025', dayfirst=True)
q1_end = pd.to_datetime('31/03/2025', dayfirst=True)
days_in_q1 = (q1_end - q1_start).days
random_days = np.random.randint(0, days_in_q1 + 1, size=len(q1_resignations))
extended_df.loc[q1_resignations.index, 'Resignation_Date'] = q1_start + pd.to_timedelta(random_days, unit='D')

# Assign Q2 resignation dates (Apr-Jun 2025)
q2_start = pd.to_datetime('01/04/2025', dayfirst=True)
q2_end = pd.to_datetime('30/06/2025', dayfirst=True)
days_in_q2 = (q2_end - q2_start).days
random_days = np.random.randint(0, days_in_q2 + 1, size=len(q2_resignations))
extended_df.loc[q2_resignations.index, 'Resignation_Date'] = q2_start + pd.to_timedelta(random_days, unit='D')

# Assign random past dates for other resignations (before 2025)
past_start = pd.to_datetime('01/01/2018', dayfirst=True)
past_end = pd.to_datetime('31/12/2024', dayfirst=True)
days_in_past = (past_end - past_start).days
random_days = np.random.randint(0, days_in_past + 1, size=len(other_resignations))
extended_df.loc[other_resignations.index, 'Resignation_Date'] = past_start + pd.to_timedelta(random_days, unit='D')
extended_df['Hire_Date'] = extended_df['Hire_Date'].dt.strftime('%d/%m/%Y')
extended_df['Resignation_Date'] = extended_df['Resignation_Date'].dt.strftime('%d/%m/%Y')

extended_df.to_csv('Deployment_Used_Extended_Dataset_with_Resignation.csv', index=False)

print(f"Execution completed in {time.time() - start_time:.2f} seconds")
print("\nResignation date distribution:")
print(extended_df[resigned_mask]['Resignation_Date'].value_counts().sort_index())
print(f"\nTotal records: {len(extended_df)}")
print(f"Resigned employees: {resigned_mask.sum()}")
print(f" - Q1 2025 resignations (Jan-Mar): {len(q1_resignations)}")
print(f" - Q2 2025 resignations (Apr-Jun): {len(q2_resignations)}")
print(f" - Other resignations: {len(other_resignations)}")
print(f"Current employees (no resignation date): {len(extended_df) - resigned_mask.sum()}")

```

## Analytics System Dashboard

```

import streamlit as st
import pandas as pd
import numpy as np
import joblib
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from io import BytesIO
import plotly.express as px
import plotly.graph_objects as go
from dateutil.relativedelta import relativedelta
import shap

# Initialize session state variables
def init_session_state():
    if 'logged_in' not in st.session_state:
        st.session_state.logged_in = False
    if 'current_page' not in st.session_state:
        st.session_state.current_page = "Login"
    if 'last_login' not in st.session_state:
        st.session_state.last_login = None
    if 'role' not in st.session_state:
        st.session_state.role = None
    if 'login_attempted' not in st.session_state:
        st.session_state.login_attempted = False
    if 'hr_data' not in st.session_state:
        st.session_state.hr_data = None

# Load data
def load_data():
    df_deployment = pd.read_csv('Deployment_Used_Extended_Dataset_with_Resignation.csv')
    df_performance = pd.read_csv('Extended_Employee_Performance_and_Productivity_Data.csv')

    # Store DataFrames separately in session state (recommended)
    st.session_state.df_deployment = df_deployment
    st.session_state.df_performance = df_performance

    return df_deployment, df_performance

# =====#
# Functions for Both HR and Talents #

# Login Page
def login_page():
    st.title("Employee Churn Analytics Dashboard Login")
    st.write("Please enter your credentials to access the HR analytics dashboard")

    with st.form("login_form"):
        username = st.text_input("Username", key="username")
        password = st.text_input("Password", type="password", key="password")
        submitted = st.form_submit_button("Login")

        if submitted:
            st.write(f"Entered username: '{username}', password: '{password}'")

            username = username.strip()
            password = password.strip()

# End of code block

```

```

if username == "hr" and password == "ilovetowork":
    st.session_state.logged_in = True
    st.session_state.role = "HR"
    st.session_state.current_page = "HR Dashboard"
    st.session_state.last_login = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    st.session_state.hr_data = load_data()
    st.success("HR login successful! Loading data...")
    st.rerun()

elif username == "talent" and password == "ilovetoworkalso":
    st.session_state.logged_in = True
    st.session_state.role = "Talent"
    st.session_state.current_page = "Talent Insights"
    st.session_state.last_login = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    st.session_state.hr_data = load_data()
    st.success("Talent Acquisition login successful! Loading data...")
    st.rerun()

else:
    st.error("Invalid username or password")

# -----
# Navigation bar
def create_navbar():
    st.sidebar.title("Navigation")
    st.sidebar.markdown(f"**Role**: {st.session_state.role}")

# Common pages for all roles
pages = {
    "Predict Employee Churn": "Predict Employee Churn"
}

# HR-specific pages
if st.session_state.role == "HR":
    pages.update({
        "HR Dashboard": "HR Dashboard",
        "Employee Performance": "Overview of Employee Performance",
        "Actionable Insights": "Actionable Insights for Retention and Productivity"
    })

# Talent-specific pages
elif st.session_state.role == "Talent":
    pages.update({
        "Talent Insights": "Talent Insights",
        "Hiring Recommendations": "Hiring Recommendations"
    })

# Navigation buttons
for page_name, page_key in pages.items():
    if st.sidebar.button(page_name, key=f"nav_{page_key}"):
        st.session_state.current_page = page_key
        st.rerun()

st.sidebar.markdown("---")
if st.sidebar.button("Logout", key="logout_btn"):
    st.session_state.logged_in = False
    st.session_state.current_page = "Login"
    st.session_state.role = None
    st.rerun()

```

```

def get_total_employees(df):
    active_employees = df[df['Resigned'] == False]
    return active_employees.shape[0]

# Churn rate for current quarter and previous quarter
def get_churn_rate(df, start_date, end_date):
    df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], dayfirst=True)
    df['Resignation_Date'] = pd.to_datetime(df['Resignation_Date'], errors='coerce')

    # Employees who resigned in this quarter
    resigned_in_period = df[(df['Resignation_Date'] >= start_date) & (df['Resignation_Date'] <= end_date)]
    active_employees_at_start = df[(df['Hire_Date'] < start_date) & (df['Resigned'] == False)]

    if active_employees_at_start.shape[0] == 0:
        return 0.0

    # Churn rate (Resigned employees / Active employees at the current quarter)
    churn_rate = (resigned_in_period.shape[0] / active_employees_at_start.shape[0]) * 100
    return churn_rate

# Get the start and end dates of the current and previous quarters
def get_quarter_dates(year, quarter):
    """Returns (start_date, end_date) for given year and quarter (1-4)"""
    if quarter == 1:
        return pd.to_datetime(f"{year}-01-01"), pd.to_datetime(f"{year}-03-31")
    elif quarter == 2:
        return pd.to_datetime(f"{year}-04-01"), pd.to_datetime(f"{year}-06-30")
    elif quarter == 3:
        return pd.to_datetime(f"{year}-07-01"), pd.to_datetime(f"{year}-09-30")
    else:
        return pd.to_datetime(f"{year}-10-01"), pd.to_datetime(f"{year}-12-31")

# Get the current year and quarter
def get_current_quarter():
    today = pd.to_datetime('today')
    current_year = today.year
    current_month = today.month
    return current_year, (current_month - 1) // 3 + 1

# Get the comparison quarters based on current date
def get_comparison_quarters():
    """Returns (q1_year, q1_num, q1_start, q1_end, q2_year, q2_num, q2_start, q2_end)"""
    current_year, current_q = get_current_quarter()

    if current_q >= 3: # Compare Q1 vs Q2 of current year
        return (
            current_year, 1, *get_quarter_dates(current_year, 1),
            current_year, 2, *get_quarter_dates(current_year, 2)
        )
    else: # Compare Q3 vs Q4 of previous year
        return (
            current_year-1, 3, *get_quarter_dates(current_year-1, 3),
            current_year-1, 4, *get_quarter_dates(current_year-1, 4)
        )

# Calculate hires in a given period
def get_hired_in_period(df, start_date, end_date):
    df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], format='%m/%d/%Y')
    hired_in_period = df[(df['Hire_Date'] >= start_date) & (df['Hire_Date'] <= end_date)]
    return hired_in_period.shape[0]

```

```

# Calculate active employees at the end of a period
def get_active_employees_at_end_of_period(df, start_date, end_date):
    df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], dayfirst=True)
    df_period = df[(df['Hire_Date'] <= end_date)]
    active_employees = df_period[df_period['Resigned'] == False]
    return active_employees.shape[0]

# Calculate average satisfaction for the previous quarter
def calculate_average_satisfaction_last_quarter(df):
    last_quarter_cutoff = pd.to_datetime("2025-04-01")
    df_last_quarter = df[df['Hire_Date'] < last_quarter_cutoff]
    avg_satisfaction_last_quarter = df_last_quarter['Employee_Satisfaction_Score'].mean() if not df_last_quarter.empty else 0
    return avg_satisfaction_last_quarter

# Calculate average satisfaction for this quarter (all active employees)
def calculate_average_satisfaction_this_quarter(df):
    avg_satisfaction_this_quarter = df['Employee_Satisfaction_Score'].mean() if not df.empty else 0
    return avg_satisfaction_this_quarter

# Get quarterly metrics
def get_quarterly_metrics(df, year, quarter):
    """Returns all metrics for a given quarter"""
    df['Hire_Date'] = pd.to_datetime(df['Hire_Date'], errors='coerce')

    # Get the start and end date for the quarter
    q_start, q_end = get_quarter_dates(year, quarter)

    # Filter data for the given quarter
    quarter_data = df[
        (df['Hire_Date'] >= q_start) &
        (df['Hire_Date'] <= q_end)
    ]

    active_employees = get_active_employees_at_end_of_period(df, q_start, q_end)
    hires = get_hired_in_period(df, q_start, q_end)
    churn = get_churn_rate(df, q_start, q_end)
    satisfaction = quarter_data['Employee_Satisfaction_Score'].mean()

    return {
        'employees': active_employees,
        'hires': hires,
        'churn': churn,
        'satisfaction': satisfaction if not pd.isna(satisfaction) else 0,
        'year': year,
        'quarter': quarter,
        'start': q_start,
        'end': q_end
    }

# Get previous quarter
def get_previous_quarter(year, quarter):
    if quarter == 1:
        return (year-1, 4)
    return (year, quarter-1)

```

```

# Get sorted quarter metrics for trends
def get_sorted_quarters_for_trends(df, current_year, current_quarter):
    """Returns sorted quarter metrics for trends"""

    # Generate last 8 quarters
    quarters = []
    year, quarter = current_year, current_quarter

    for _ in range(8):
        year, quarter = get_previous_quarter(year, quarter)
        if (year < current_year) or (year == current_year and quarter < current_quarter):
            quarters.append((year, quarter))

    # Sort the quarters properly in chronological order
    quarters.sort(key=lambda x: (x[0], x[1]))

    quarter_metrics = []
    for year, quarter in quarters:
        metrics = get_quarterly_metrics(df, year, quarter)
        if metrics:
            quarter_metrics.append(metrics)

    return quarter_metrics

```

```

# ----- #
# Employee Metrics Page #

def display_employee_metrics(q1, q2):
    """Display employee metrics and return button state"""
    employee_change = ((q2['employees'] - q1['employees']) / q1['employees']) * 100 if q1['employees'] > 0 else 0
    st.metric(
        "Total Employees",
        f'{q2["employees"]:,}',
        delta=f'{employee_change:.1f}% from Q{q1["quarter"]}',
        delta_color="normal" if employee_change >= 0 else "inverse"
    )
    return st.button("View Employee Details", key="employee_details")

def display_churn_metrics(q1, q2):
    """Display churn metrics and return button state"""
    churn_change = q2['churn'] - q1['churn']
    st.metric(
        "Churn Rate",
        f'{q2["churn"]:.2f}%',
        delta=f'{churn_change:.2f}% from Q{q1["quarter"]}',
        delta_color="inverse" if churn_change >= 0 else "normal"
    )
    return st.button("View Churn Details", key="churn_details")

def display_employee_details(df_deployment, q1, q2):
    """Show detailed employee information"""
    st.subheader(f"Employee Details [ Q{q2['quarter']} ] {q2['year']} ")
    hire_comparison = ((q2['hires'] - q1['hires']) / q1['hires']) * 100 if q1['hires'] > 0 else 0

    tab1, tab2 = st.tabs(["Hire Comparison", "Department Breakdown"])

    with tab1:
        st.markdown(f"**Employee Hire Comparison (Q{q1['quarter']} vs Q{q2['quarter']}):** {hire_comparison:.1f}%")
        comparison_data = pd.DataFrame({
            'Period': [f"Q{q1['quarter']} {q1['year']}", f"Q{q2['quarter']} {q2['year']}"],
            'Hires': [q1['hires'], q2['hires']]
        })

        fig = px.bar(
            comparison_data,
            x='Period',
            y='Hires',
            text='Hires',
            color='Period',
            color_discrete_sequence=[ '#1f77b4', '#ff7f0e' ]
        )

        fig.update_traces(
            textposition='outside',
            textfont_size=14,
            marker_line_width=0
        )

        fig.update_layout(
            showlegend=False,
            xaxis_title=None,
            yaxis_title='Hires',
            plot_bgcolor='rgba(0,0,0,0)',
            paper_bgcolor='rgba(0,0,0,0)',
            margin=dict(t=30),
            uniformtext_minsize=8,
            uniformtext_mode='hide',
            yaxis=dict(
                gridcolor='rgba(211,211,211,0.3)'
            )
        )
        st.plotly_chart(fig, use_container_width=True)

```

```

with tab2:
    today = pd.to_datetime("today")
    current_year = today.year
    current_quarter = (today.month - 1) // 3 + 1

    if current_quarter == 1:
        last_qtr = 4
        last_year = current_year - 1
    else:
        last_qtr = current_quarter - 1
        last_year = current_year

    q_start = pd.to_datetime(f"{last_year}-{(last_qtr-1)*3 + 1}-01")
    q_end = pd.to_datetime(f"{last_year}-{last_qtr*3}-01") + pd.offsets.MonthEnd(1)

    resigned_qtr = df_deployment[
        (df_deployment['Resigned'] == True) &
        (pd.to_datetime(df_deployment['Resignation_Date'], errors='coerce') >= q_start) &
        (pd.to_datetime(df_deployment['Resignation_Date'], errors='coerce') <= q_end)
    ]

    resigned_df = (resigned_qtr
                    .groupby('Department')['Employee_ID']
                    .count()
                    .reset_index(name='Resigned Count'))

    st.markdown(f"#### Resigned Employees per Department [ Q{last_qtr} {last_year}]")

    fig_resign = px.bar(
        resigned_df.sort_values('Resigned Count', ascending=False),
        x='Department',
        y='Resigned Count',
        text='Resigned Count',
        title='Resignations by Department'
    )
    fig_resign.update_traces(marker_color='#1f77b4', textposition='outside')
    fig_resign.update_layout(xaxis_title='', yaxis_title='Resigned', showlegend=False)
    st.plotly_chart(fig_resign, use_container_width=True)

# Only HR #
def display_satisfaction_trend(
    df_deployment: pd.DataFrame,
    current_year: int,
    current_quarter: int
) -> None:
    """Display satisfaction trend visualization
    Args:
        df_deployment: DataFrame containing employee deployment data
        current_year: The current year as integer
        current_quarter: The current quarter (1-4)
    """
    st.subheader("Satisfaction Trend")

    # Get quarter data for trend analysis
    desired_quarters = []
    latest_complete_quarter = 2
    latest_complete_year = 2025
    for i in range(6):
        quarter_offset = i
        year = latest_complete_year - (quarter_offset // 4)
        quarter = latest_complete_quarter - (quarter_offset % 4)
        if quarter <= 0:
            quarter += 4
            year -= 1
        desired_quarters.append({'year': year, 'quarter': quarter})

```

```

# Get matching quarter metrics
quarter_metrics = []
all_quarters = get_sorted_quarters_for_trends(df_deployment, current_year, current_quarter)
for q in desired_quarters:
    matching_quarter = next(
        (item for item in all_quarters
         | if item['year'] == q['year'] and item['quarter'] == q['quarter']),
        None
    )
    if matching_quarter:
        quarter_metrics.append(matching_quarter)

if not quarter_metrics:
    st.warning("No satisfaction data available for the specified quarters")
    return

# Prepare trend data
quarter_metrics = sorted(quarter_metrics, key=lambda x: (x['year'], x['quarter']))
trend_df = pd.DataFrame({
    'Quarter': [f"{q['year']}Q{q['quarter']}" for q in quarter_metrics],
    'Satisfaction': [q['satisfaction'] for q in quarter_metrics],
    'Date': [q['end'] for q in quarter_metrics]
}).sort_values('Date')

# Calculate average
avg_satisfaction = trend_df['Satisfaction'].mean()

# Create visualization
fig = go.Figure()
fig.add_trace(go.Scatter([
    x=trend_df['Quarter'],
    y=trend_df['Satisfaction'],
    mode='lines+markers',
    name='Satisfaction',
    line=dict(color='skyblue', width=2)
]))
fig.add_trace(go.Scatter(
    x=trend_df['Quarter'],
    y=[avg_satisfaction] * len(trend_df),
    mode='lines',
    name='Average',
    line=dict(color='red', width=2, dash='dot')
))
fig.update_layout(
    title='Satisfaction Trend',
    xaxis_title='Quarter',
    yaxis_title='Satisfaction Score',
    height=400
)
st.plotly_chart(fig, use_container_width=True)

# Show latest value
latest = trend_df.iloc[-1]
st.markdown(f"**Latest:** {latest['Satisfaction']:.1f} average satisfaction score in {latest['Quarter']}")

# -----
# Churn Prediction Page
def churn_prediction():
    st.title("Employee Churn Prediction")
    st.write("""This tool helps predict whether an employee will leave the company.""")

    with st.expander("■ Variable Explanations"):
        st.markdown("""
*Performance Score*: Employee's performance rating (1-5 scale)
*Satisfaction Score*: Employee's satisfaction with their job (1-5 scale)
*Monthly Salary*: Employee's monthly pay
*Work Hours*: Typical hours worked per week
*Overtime Hours*: Extra hours worked beyond the normal schedule per year
*Promotions*: Number of promotions received per year
*Remote Work Level*: The proportion of time spent working remotely (None/Low/High)
*Job Title*: Employee's role
*Health Condition Index*: Employee's health status score (1-10 scale)
""")

```

```

if st.session_state.hr_data is None:
    st.error("HR data not loaded. Please login again.")
    return

with st.form("churn_prediction_form"):
    col1, col2 = st.columns(2)

    with col1:
        performance_score = st.slider(
            "Performance Score (1-5)", 1, 5, 3,
            help="Employee's performance rating (1-5 scale)"
        )
        employee_satisfaction = st.slider(
            "Satisfaction Score (1-5)", 1, 5, 3,
            help="Employee's satisfaction with their job (1-5 scale)"
        )
        monthly_salary = st.number_input(
            "Monthly Salary ($)", min_value=1000, max_value=20000, value=5000,
            help="Employee's monthly pay in dollars"
        )
        work_hours_per_week = st.slider(
            "Work Hours/Week", 10, 80, 40,
            help="Typical hours worked per week"
        )

    with col2:
        overtime_hours = st.slider(
            "Overtime Hours/Year", 0, 60, 30,
            help="Extra hours worked beyond the normal schedule per year"
        )
        promotions = st.slider(
            "Promotions/Year", 0, 5, 3,
            help="Number of promotions received per year"
        )
        remote_work_level = st.selectbox(
            "Remote Work Level", ["None", "Low", "High"],
            help="Time spent working remotely (None/Low/High) "
        )
        job_title = st.selectbox(
            "Job Title", ["Manager", "Technician", "Other"],
            help="Employee's role in the company"
        )
        health_index = st.slider(
            "Health Condition Index (1-10)", 1, 10, 5,
            help="Employee's health status score (1-10 scale)"
        )

    submitted = st.form_submit_button("Predict Churn Risk")

if submitted:
    try:
        model = joblib.load('employee_churn_model.pkl')

        input_data = pd.DataFrame([[performance_score,
                                    1 if remote_work_level == "Low" else 2 if remote_work_level == "High" else 0,
                                    promotions,
                                    1 if job_title == "Technician" else 0,
                                    1 if job_title == "Manager" else 0,
                                    monthly_salary,
                                    work_hours_per_week,
                                    overtime_hours,
                                    employee_satisfaction,
                                    health_index]],
                               columns=['Performance_Score', 'Remote_Work_Level', 'Promotions',
                                         'Job_Title_Technician', 'Job_Title_Manager', 'Monthly_Salary',
                                         'Work_Hours_Per_Week', 'Overtime_Hours',
                                         'Employee_Satisfaction_Score', 'Health_Index'])

        # Make prediction
        prediction = model.predict(input_data)
        probabilities = model.predict_proba(input_data)
        prob_churn = probabilities[0][1]
    
```

```

st.subheader("Prediction Results")

# Recommendations based on churn risk
if prob_churn > 0.7:
    st.error(f"High Risk of Churn: {prob_churn:.1%}")
    st.write("This employee has a high probability of leaving. Immediate action recommended to retain this employee.")
    st.subheader("Recommended Actions:")
    st.write("""
        - Schedule 1:1 meeting to discuss concerns.
        - Identify key issues affecting satisfaction.
        - Review compensation and benefits.
        - Consider career development opportunities.
        - Assess and adjust workload balance if needed.
    """)
elif prob_churn > 0.4:
    st.warning(f"Moderate Risk of Churn: {prob_churn:.1%}")
    st.write("This employee shows some risk factors. Consider taking proactive measures to overcome potential concerns.")
    st.subheader("Recommended Actions:")
    st.write("""
        - Check-in with the employee to understand their concerns.
        - Provide feedback channels for open discussion.
        - Monitor performance and satisfaction over time.
        - Consider additional training or support.
    """)
else:
    st.success(f"Low Risk of Churn: {prob_churn:.1%}")
    st.write("This employee appears stable. Continue regular engagement.")
    st.subheader("Recommended Actions:")
    st.write("""
        - Continue regular check-ins to maintain engagement.
        - Ensure consistent career development opportunities.
        - Watch for any signs of dissatisfaction.
    """)

# SHAP (Feature Importance)
try:
    explainer = shap.TreeExplainer(model)
    shap_values = explainer.shap_values(input_data)

    feature_contributions = pd.DataFrame(shap_values[0], index=input_data.columns, columns=["SHAP Value"])
    feature_contributions['Abs_SHAP Value'] = feature_contributions['SHAP Value'].abs()
    feature_contributions = feature_contributions.sort_values(by='Abs_SHAP Value', ascending=False)

    # Only select positive SHAP values (which indicate positive contribution to churn risk)
    feature_contributions_positive = feature_contributions[feature_contributions['SHAP Value'] > 0]

    fig = px.bar(
        feature_contributions_positive.head(7),
        x='SHAP Value',
        y=feature_contributions_positive.head(7).index,
        title='Top Influencing Features Based on SHAP Values',
        text=feature_contributions_positive.head(7)[['Abs_SHAP Value']].round(1).astype(str) + '%',
        color='SHAP Value',
        color_continuous_scale='Blues',
        orientation='h'
    )

    fig.update_traces(
        hovertemplate='%{y}  
SHAP Value: %{x:.2f}  
Percentage: %{text}',
    )

    fig.update_layout(
        title={'font': {'size': 25, 'color': 'white'}, 'x': 0.5, 'xanchor': 'center'},
        xaxis_title='SHAP Value',
        yaxis_title='Features',
        font=dict(color='white'),
        plot_bgcolor='rgba(0,0,0,0)',
        paper_bgcolor='rgba(0,0,0,0)'
    )

```

```

        st.plotly_chart(fig, use_container_width=True)

    except Exception as e:
        st.error(f"Error with SHAP explanation: {str(e)}")
        st.write("Falling back to Feature Importance visualization...")

    # Feature importance fallback (if SHAP fails)
    st.markdown("---")
    st.subheader("Key Factors Influencing This Prediction")
    feature_importance = pd.Series(model.feature_importances_, index=input_data.columns)
    top_features = feature_importance.sort_values(ascending=False).head(7)

    # Calculate the percentage for each feature
    total_importance = top_features.sum()
    top_features_percentage = (top_features / total_importance) * 100

    # Create the horizontal bar chart
    fig = px.bar(
        x=top_features.values,
        y=top_features.index,
        labels={'x': 'Importance Score', 'y': 'Features'},
        title='Top Influencing Factors',
        text=top_features_percentage.round(1).astype(str) + '%',
        color=top_features.values,
        color_continuous_scale='Blues',
        orientation='h'
    )

    fig.update_traces(
        hovertemplate='<b>%{y}</b><br>Importance: %{x:.2f}<br>Percentage: %{text}',
    )

    fig.update_layout(
        title={'font': {'size': 25, 'color': 'white'}, 'x': 0.5, 'xanchor': 'center'},
        xaxis_title='Importance Score',
        yaxis_title='Features',
        font=dict(color='white'),
        plot_bgcolor='rgba(0,0,0,0)',
        paper_bgcolor='rgba(0,0,0,0)'
    )

    # Show the plot
    st.plotly_chart(fig, use_container_width=True)

except Exception as e:
    st.error(f"Error loading model: {str(e)}")
    st.write("Please try again or contact support")

# =====#
# Only for HR Functions #

# Recent activity
def recent_activity():
    if 'df_deployment' not in st.session_state:
        st.warning("HR data not loaded. Please log in again.")
        return

    df = st.session_state.df_deployment

    required_columns = ['Employee_ID', 'Hire_Date', 'Job_Title',
                        'Employee_Satisfaction_Score', 'Performance_Score',
                        'Years_At_Company', 'Resigned', 'Resignation_Date']
    missing_columns = [col for col in required_columns if col not in df.columns]

    if missing_columns:
        st.warning(f"Missing columns: {', '.join(missing_columns)}")
        return

    activity_data = []

```

```

# 1. New Hires (Top 5 most recent)
recent_hires = df.sort_values(by='Hire_Date', ascending=False).head(5)
for _, row in recent_hires.iterrows():
    satisfaction = row['Employee_Satisfaction_Score']
    performance = row['Performance_Score']
    years = row['Years_At_Company']

    # Priority based on satisfaction
    if satisfaction >= 4:
        priority = "Low"
    elif satisfaction == 3:
        priority = "Medium"
    else:
        priority = "High"

    activity_data.append({
        "Employee ID": str(row['Employee_ID']),
        "Date": pd.to_datetime(row['Hire_Date']).date(),
        "Job Title": row['Job_Title'],
        "Activity Status": "New Hire",
        "Performance Score": performance,
        "Employee Satisfaction": f"{satisfaction:.2f}",
        "Years At Company": years,
        "Priority": priority
    })

# 2. Resignations (Top 5 most recent)
recent_resignations = df[df['Resigned'] == True].sort_values(by='Resignation_Date', ascending=False).head(5)
for _, row in recent_resignations.iterrows():
    satisfaction = row['Employee_Satisfaction_Score']
    performance = row['Performance_Score']
    years = row['Years_At_Company']

    # Refined logic for resignations
    if performance >= 4 or years >= 5:
        priority = "High"
    elif performance == 3:
        priority = "Medium"
    else:
        priority = "Low"

    activity_data.append({
        "Employee ID": str(row['Employee_ID']),
        "Date": pd.to_datetime(row['Resignation_Date']).date(),
        "Job Title": row['Job_Title'],
        "Activity Status": "Resignation",
        "Performance Score": performance,
        "Employee Satisfaction": f"{satisfaction:.2f}",
        "Years At Company": years,
        "Priority": priority
    })

# Display section
st.subheader("Recent Activity")

if activity_data:
    activity_df = pd.DataFrame(activity_data)

    # Priority sorting order
    priority_order = pd.CategoricalDtype(['High', 'Medium', 'Low'], ordered=True)
    activity_df['Priority'] = activity_df['Priority'].astype(priority_order)

    # Sort
    activity_df = activity_df.sort_values(['Date', 'Priority'], ascending=[False, True])
    activity_df.index = np.arange(1, len(activity_df) + 1)

    color_map = {
        'High': 'background-color: rgba(255, 0, 0, 0.15); color: white;',      # very light red
        'Medium': 'background-color: rgba(255, 200, 0, 0.15); color: white;',     # very light yellow
        'Low': 'background-color: rgba(0, 200, 0, 0.15); color: white;'          # very light green
    }
}

```

```

    def highlight_priority(val):
        return color_map.get(val, '')

    styled_df = activity_df.style.applymap(highlight_priority, subset=['Priority'])

    st.dataframe(styled_df, use_container_width=True)

else:
    st.write("No recent activity found.")

# -----
# HR Dashboard Page (Home Page)
def hr_dashboard():
    if st.session_state.role != "HR":
        st.warning("You don't have permission to access this page")
        return

    if st.session_state.hr_data is None:
        st.error("HR data not loaded. Please login again.")
        return

    # Load data
    df_deployment, df_performance = st.session_state.hr_data

    # Get current date info
    today = pd.to_datetime('today')
    current_year = today.year
    current_quarter = (today.month - 1) // 3 + 1

    # Get comparison quarters
    q2_year, q2_quarter = get_previous_quarter(current_year, current_quarter)
    q1_year, q1_quarter = get_previous_quarter(q2_year, q2_quarter)

    # Get metrics
    q1 = get_quarterly_metrics(df_deployment, q1_year, q1_quarter)
    q2 = get_quarterly_metrics(df_deployment, q2_year, q2_quarter)

    # Calculate changes
    satisfaction_change = q2['satisfaction'] - q1['satisfaction']

    # Dashboard Metrics
    col1, col2, col3 = st.columns(3)

    with col1:
        employee_details_button = display_employee_metrics(q1, q2)

    with col2:
        st.metric(
            "Avg. Satisfaction",
            f"{q2['satisfaction']:.1f}/10",
            delta=f"{satisfaction_change:.1f} from Q{q1['quarter']}",
            delta_color="normal" if satisfaction_change >= 0 else "inverse"
        )
        satisfaction_details_button = st.button("View Satisfaction Details", key="satisfaction_details")

    with col3:
        churn_details_button = display_churn_metrics(q1, q2)

    st.markdown("----")

    # Conditional displays
    if employee_details_button:
        display_employee_details(df_deployment, q1, q2)

    if satisfaction_details_button:
        display_satisfaction_trend(df_deployment, current_year, current_quarter)

    if churn_details_button:
        display_churn_details(df_deployment, current_year, current_quarter)

#

```

```

# -----
# Quick actions
st.subheader("Quick Actions")
action_col1, action_col2, action_col3 = st.columns(3)
with action_col1:
    if st.button("Run Churn Prediction"):
        st.session_state.current_page = "Predict Employee Churn"
        st.rerun()
with action_col2:
    if st.button("View Performance"):
        st.session_state.current_page = "Overview of Employee Performance"
        st.rerun()
with action_col3:
    if st.button("Get Insights"):
        st.session_state.current_page = "Actionable Insights for Retention and Productivity"
        st.rerun()

st.markdown("---")

recent_activity()

# =====#
# HR Third Page #
# Employee Performance Overview
# Create department comparison charts

def create_dept_chart(data, x_col, y_col, title, company_avg, line_color="red"):

    data['Pct_Diff'] = ((data[y_col] - company_avg) / company_avg * 100)

    fig = px.bar(
        data,
        x=x_col,
        y=y_col,
        title=title,
        text=[f"val:{.1f} ({pct:+.1f}%)" for val, pct in zip(data[y_col], data['Pct_Diff'])],
        color_discrete_sequence=[ "#4878B1"]
    )

    fig.update_traces(
        textposition="outside",
        textfont_size=10,
        marker_color="#157DEC",
        textangle=0,
        opacity=0.9
    )

    fig.add_hline(
        y=company_avg,
        line_dash="dash",
        line_color=line_color,
        line_width=2,
        annotation_text=f"Company Avg: {company_avg:.1f}",
        annotation_position="top right",
        annotation_font=dict(size=12, color="black"),
        annotation_bgcolor="rgba(255,255,255,0.7)",
        annotation_y=company_avg + 1.25
    )

    y_max = max(data[y_col].max() * 1.5, company_avg * 1.5)
    fig.update_layout(
        yaxis_range=[0, y_max],
        margin=dict(t=100, b=40),
        plot_bgcolor="rgba(0,0,0,0)"
    )
    return fig

# -----
# Employee Replacement Rate

```

```

"""
# Employee Performance Page
def employee_performance():
    st.title("Employee Performance Overview")

    if 'df_deployment' not in st.session_state:
        st.warning("HR data not loaded. Please log in again.")
        return

    df = st.session_state.df_deployment

    # Calculate company-wide averages
    company_avg_perf = df['Performance_Score'].mean()
    company_avg_satisfaction = df['Employee_Satisfaction_Score'].mean()
    company_avg_overtime = df['Overtime_Hours'].mean()

    # 1. Department Performance
    st.subheader("Department Performance")

    dept_performance = df.groupby('Department').agg(
        Avg_Performance=('Performance_Score', 'mean'),
        Avg_Satisfaction=('Employee_Satisfaction_Score', 'mean'),
        Avg_Overtime=('Overtime_Hours', 'mean'),
        Employee_Count=('Employee_ID', 'count')
    ).reset_index()

    # Display metrics
    col1, col2, col3 = st.columns(3)
    with col1:
        st.metric("Total Departments", len(dept_performance))
    with col2:
        st.metric("Highest Performing Dept",
                  dept_performance.loc[dept_performance['Avg_Performance'].idxmax()]['Department'])
    with col3:
        st.metric("Most Satisfied Dept",
                  dept_performance.loc[dept_performance['Avg_Satisfaction'].idxmax()]['Department'])

    # Department comparison charts
    tab1, tab2, tab3 = st.tabs(["Performance", "Satisfaction", "Overtime"])

    with tab1:
        fig = create_dept_chart(
            dept_performance,
            'Department',
            'Avg_Performance',
            'Average Performance Score by Department',
            company_avg_perf
        )
        st.plotly_chart(fig, use_container_width=True)

    with tab2:
        fig = create_dept_chart(
            dept_performance,
            'Department',
            'Avg_Satisfaction',
            'Average Satisfaction by Department',
            company_avg_satisfaction
        )
        st.plotly_chart(fig, use_container_width=True)

    with tab3:
        fig = create_dept_chart(
            dept_performance,
            'Department',
            'Avg_Overtime',
            'Average Overtime Hours by Department',
            company_avg_overtime
        )
        st.plotly_chart(fig, use_container_width=True)

    st.markdown("---")

    # 2. Quarterly Trends
    st.subheader("Quarterly Trends")

```

```

# 2. Quarterly Trends
st.subheader("Quarterly Trends")

df['Quarter'] = df['Hire_Date'].dt.to_period('Q').astype(str)
df['Quarter'] = df['Quarter'].apply(lambda x: f'{x[:4]}{Q{x[5:]}}')
current_quarter = pd.to_datetime('today').to_period('Q')
df['Quarter_Period'] = df['Quarter'].apply(lambda x: pd.to_datetime(x[:4] + '-' + x[5:]).to_period('Q'))
df = df[df['Quarter_Period'] <= current_quarter]

# Select the latest 6 quarters
latest_quarters = df['Quarter'].sort_values(ascending=False).unique()[:6]
df_filtered = df[df['Quarter'].isin(latest_quarters)]

quarterly_trends = df_filtered.groupby('Quarter').agg(
    Avg_Performance=('Performance_Score', 'mean'),
    Avg_Satisfaction=('Employee_Satisfaction_Score', 'mean'),
    Hire_Count=('Employee_ID', 'count')
).reset_index().sort_values('Quarter')

fig = px.line(quarterly_trends,
              x='Quarter',
              y='Avg_Performance',
              title='Performance Score Trend (Last 6 Quarters)',
              markers=True,
              text=quarterly_trends['Avg_Performance'].round(2))

fig.update_traces(textposition="top center",
                   line=dict(width=2),
                   marker=dict(size=10))

fig.add_hline(y=company_avg_perf, line_dash="dash", line_color="red",
              annotation_text=f"Company Avg: {company_avg_perf:.1f}")

st.plotly_chart(fig, use_container_width=True)

st.markdown("---")

# 3. Overtime Analysis
st.subheader("Overtime Analysis")

df['Overtime_Category'] = pd.cut(df['Overtime_Hours'],
                                  bins=[-1, 5, 10, 15, 20, 100],
                                  labels=['0-5 hrs', '6-10 hrs', '11-15 hrs', '16-20 hrs', '20+ hrs'])

overtime_summary = df.groupby('Overtime_Category').agg(
    Avg_Performance=('Performance_Score', 'mean'),
    Employee_Count=('Employee_ID', 'count'),
    Avg_Satisfaction=('Employee_Satisfaction_Score', 'mean')
).reset_index()

col1, col2 = st.columns(2)
with col1:
    fig = px.bar(overtime_summary,
                  x='Overtime_Category',
                  y='Avg_Performance',
                  color='Avg_Performance',
                  title='Average Performance by Overtime Range',
                  text='Avg_Performance')
    fig.update_traces(texttemplate=' %{text:.2f}', textposition='outside')
    st.plotly_chart(fig, use_container_width=True)

with col2:
    fig = px.pie(overtime_summary,
                  names='Overtime_Category',
                  values='Employee_Count',
                  title='Distribution of Employees by Overtime Hours')
    st.plotly_chart(fig, use_container_width=True)

st.markdown("---")

```

```

# 4. Sick Days Analysis
st.subheader("Sick Days Analysis")

sick_by_dept = df.groupby('Department')['Sick_Days'].mean().reset_index()
fig = px.bar(
    sick_by_dept,
    x='Department',
    y='Sick_Days',
    title='Average Sick Days by Department',
    color='Sick_Days',
    color_continuous_scale='Blues',
    text='Sick_Days'
)
fig.update_traces(texttemplate='%{text:.2f}', textposition='outside')
fig.update_layout(yaxis_title="Average Sick Days")
st.plotly_chart(fig, use_container_width=True)

# Executive Report Section
st.markdown("----")
st.subheader("Overall Performance Report Summary Download")

# Create summary data for report
overtime_summary = df.groupby('Overtime_Hours').agg(
    Avg_Performance=('Performance_Score', 'mean'),
    Employee_Count=('Employee_ID', 'count')
).reset_index()

# Calculate sick days metrics
max_sick_dept = df.groupby('Department')['Sick_Days'].mean().idxmax()
min_sick_dept = df.groupby('Department')['Sick_Days'].mean().idxmin()

report_data = {
    "Department Performance": dept_performance,
    "Quarterly Trends": quarterly_trends,
    "Overtime Analysis": overtime_summary,
    "Performance Correlations": df[['Employee_Satisfaction_Score', 'Performance_Score', 'Department', 'Job_Title']],
    "Sick Days Analysis": df.groupby('Department').agg({
        'Sick_Days': 'mean',
        'Performance_Score': 'mean'
    }).reset_index()
}

# Convert to Excel
def create_excel_report(data_dict):
    output = BytesIO()
    with pd.ExcelWriter(output, engine='xlsxwriter') as writer:
        for sheet_name, df in data_dict.items():
            df.to_excel(writer, sheet_name=sheet_name[:31], index=False)
    return output.getvalue()

excel_report = create_excel_report(report_data)

st.download_button(
    label="Download Full Performance Report (Excel)",
    data=excel_report,
    file_name="employee_performance_report.xlsx",
    mime="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
    help="Download the full performance report in Excel format."
)

```

```

# Text summary version
text_report = """
Employee Performance Report
Generated on: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}

*** Key Metrics ***
- Total Departments: {len(dept_performance)}
- Highest Performing Department: {dept_performance.loc[dept_performance['Avg_Performance'].idxmax()]['Department']}
- Most Satisfied Department: {dept_performance.loc[dept_performance['Avg_Satisfaction'].idxmax()]['Department']}

*** Overtime Insights ***
- Highest performing overtime range: {overtime_summary.loc[overtime_summary['Avg_Performance'].idxmax()]['Overtime_Hours']}
- Most common overtime range: {overtime_summary.loc[overtime_summary['Employee_Count'].idxmax()]['Overtime_Hours']}
- Percentage in most common range: {overtime_summary['Employee_Count'].max()}/{overtime_summary['Employee_Count'].sum()}%

*** Sick Days Insights ***
- Department with most sick days: {max_sick_dept}
- Department with least sick days: {min_sick_dept}
- Average sick days across company: {df['Sick_Days'].mean():.1f} days

*** Recommendations ***
1. Investigate why {dept_performance.loc[dept_performance['Avg_Performance'].idxmax()]['Department']} performs best
2. Address satisfaction in {dept_performance.loc[dept_performance['Avg_Satisfaction'].idxmax()]['Department']}
3. Monitor departments with {overtime_summary.loc[overtime_summary['Employee_Count'].idxmax()]['Overtime_Hours']} overtime for burnout
4. Investigate high sick days in {max_sick_dept} department
5. Review wellness programs in departments with max average sick days
6. Analyze correlation between performance and sick days
"""

st.download_button(
    label="Download Quick Summary (Text)",
    data=text_report,
    file_name="performance_summary.txt",
    mime="text/plain",
    help="Download a quick summary of the performance report in text format."
)

```

```

# Generate dynamic insights based on real-time data
def generate_dynamic_insights(df, insight_type, churn_rate):

    high_performers = df[df['Performance_Score'] > 4]
    low_satisfaction = df[df['Employee_Satisfaction_Score'] < 2.5]
    high_overtime = df[df['Overtime_Hours'] > 25]
    optimal_performance = df[df['Performance_Score'] == 5]

    # Retention Insights
    if insight_type == "Retention":
        insights = []

        if churn_rate > 20:
            insights.append(f"High churn risk (churn rate: {churn_rate:.2f}%). Consider retention strategies.")
            insights.append("Suggested Actions: Consider improving engagement and career development programs, review compensation, and increase job satisfaction.")

        if len(high_performers) > 0:
            insights.append(f"High-performing employees: {len(high_performers)} found. Reward and retain them.")
            insights.append("Suggested Actions: Reward high performers with promotions, incentives, or leadership opportunities.")

        if len(low_satisfaction) > 0:
            insights.append(f"Employees with low satisfaction: {len(low_satisfaction)} need immediate attention.")
            insights.append("Suggested Actions: Schedule 1:1 meetings, understand their concerns, and offer support.")

    return insights

    # Productivity Insights
    elif insight_type == "Productivity":
        insights = []

        if len(high_overtime) > 0:
            insights.append(f"Employees working >25 overtime hours: {len(high_overtime)}. Monitor for burnout.")
            insights.append("Suggested Actions: Consider workload adjustments or introducing wellness programs.")

        if len(optimal_performance) > 0:
            insights.append(f"Optimal performers: {len(optimal_performance)} found. Foster leadership roles.")
            insights.append("Suggested Actions: Nurture these employees for future leadership roles or mentorship programs.")

    return insights

    # Engagement Insights
    elif insight_type == "Engagement":
        insights = []

        high_engaged = df[(df['Performance_Score'] > 4) & (df['Employee_Satisfaction_Score'] > 4)]
        if len(high_engaged) > 0:
            insights.append(f"Highly engaged employees: {len(high_engaged)} found. Leverage them for mentorship roles.")
            insights.append("Suggested Actions: Leverage high-engagement employees for mentoring new hires or driving culture initiatives.")

        low_engaged = df[(df['Performance_Score'] <= 3) & (df['Employee_Satisfaction_Score'] <= 3)]
        if len(low_engaged) > 0:
            insights.append(f"Low engagement risk: {len(low_engaged)} employees with both low satisfaction and performance.")
            insights.append("Suggested Actions: Address underlying issues, including performance coaching, or reassignment to more suitable roles.")

    return insights

    # Compensation Insights
    elif insight_type == "Compensation":
        insights = []

        underpaid_performers = df[(df['Performance_Score'] == 5) & (df['Monthly_Salary'] < df['Monthly_Salary'].median())]
        if len(underpaid_performers) > 0:
            insights.append(f"{len(underpaid_performers)} high performers are underpaid. Consider salary adjustments.")
            insights.append("Suggested Actions: Ensure competitive compensation packages for high performers to avoid attrition.")

        low_salary_high_satisfaction = df[(df['Monthly_Salary'] < df['Monthly_Salary'].median()) & (df['Employee_Satisfaction_Score'] > 4)]
        if len(low_salary_high_satisfaction) > 0:
            insights.append(f"{len(low_salary_high_satisfaction)} employees with low salary and high satisfaction found.")
            insights.append("Suggested Actions: Review salary adjustments to ensure satisfaction remains high.")

    return insights

return []

```

```

    # Actionable Insights Page
def actionable_insights():

    st.set_page_config(layout="centered")
    st.title("Actionable Insights for Retention and Productivity")

    # Insight Selector
    st.subheader("💡 Insight Selector", anchor=False)
    insight_type = st.selectbox(
        "What type of insights would you like to view and analyse?",
        ["Compensation", "Productivity", "Retention", "Engagement"],
        index=1,
        help="Select the HR metric you want to analyze"
    )

    st.markdown("<br>", unsafe_allow_html=True)
    generate_clicked = st.button("⚡ Generate Insights")

    if "df_deployment" not in st.session_state:
        st.error("HR data not loaded. Please log in again.")
        return

    df = st.session_state.df_deployment
    dummy_churn = 8.6

    if generate_clicked:
        today = pd.to_datetime("today")
        current_year = today.year
        current_quarter = (today.month - 1) // 3 + 1
        start_date, end_date = get_quarter_dates(current_year, current_quarter)
        churn_rate = get_churn_rate(df, start_date, end_date)

        insights = generate_dynamic_insights(df, insight_type, churn_rate)
        st.subheader(f"{insight_type} Insights")

        if insights:
            insight_text = ""
            numbered = 1
            for i in range(0, len(insights), 2):
                main = insights[i]
                action = insights[i+1] if i+1 < len(insights) else None

                st.markdown(f"**{numbered}. {main}**")
                if action:
                    for prefix in ["Suggested Actions:", "Suggested Action:"]:
                        if action.startswith(prefix):
                            action = action.replace(prefix, "").strip()
                    st.markdown(f"    🔍 Suggested Action: {action}", unsafe_allow_html=True)

                insight_text += f"{numbered}. {main}\n"
                insight_text += f"    🔍 Suggested Action: {action}\n"
            else:
                insight_text += f"{numbered}. {main}\n"

            numbered += 1

        # Action Plan and Report
        st.markdown("---")
        st.subheader("Report Summary and Action Plan")

```

```
# Action Plan and Report
st.markdown("---")
st.subheader("Report Summary and Action Plan")

action_plans = {
    "Retention": """Sample Retention Action Plan:
1. Identify at-risk employees (churn risk >40%)
2. Schedule stay interviews within 2 weeks
3. Create personalized development plans
4. Review compensation benchmarks
5. Establish mentorship pairings""",
    "Productivity": """Sample Productivity Action Plan:
1. Identify teams with overtime >50 hrs
2. Introduce flexible work schedules or task rotation
3. Launch wellness or fatigue-monitoring initiatives
4. Monitor impact of training programs on performance
5. Adjust workloads for balance and sustainability""",
    "Engagement": """Sample Engagement Action Plan:
1. Conduct anonymous engagement surveys
2. Launch mentorship or buddy programs
3. Recognize and reward contributions regularly
4. Organize quarterly engagement townhalls
5. Support career growth and learning pathways""",
    "Compensation": """Sample Compensation Action Plan:
1. Benchmark current salary against market rates
2. Identify underpaid high performers
3. Develop a transparent performance-based bonus system
4. Review compensation equity across roles and departments
5. Communicate compensation philosophy to employees"""
}

timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
report = f"""Employee {insight_type} Insights Report
Generated on: {timestamp}

--- INSIGHTS ---
{insight_text}

--- ACTION PLAN ---
{action_plans.get(insight_type, "No action plan available.")}

"""

st.download_button(
    label="Download Full Insight & Action Plan Report",
    data=report,
    file_name=f"{insight_type.lower()}_{insight_report}.txt",
    mime="text/plain",
    help="Download the full insight and action plan report in text format."
)
```

```

# Talent Acquisition Page (Talent team only) (Home Page)
def talent_dashboard():
    if st.session_state.role != "Talent":
        st.warning("You don't have permission to access this page")
        return

    if 'df_deployment' not in st.session_state:
        st.warning("HR data not loaded. Please log in again.")
        return

    df_deployment = st.session_state.df_deployment

    # Convert date columns to datetime
    df_deployment['Hire_Date'] = pd.to_datetime(df_deployment['Hire_Date'], errors='coerce')
    df_deployment['Resignation_Date'] = pd.to_datetime(df_deployment['Resignation_Date'], errors='coerce')

    # Get quarter info
    today = pd.to_datetime('today')
    current_year = today.year
    current_quarter = (today.month - 1) // 3 + 1

    # Get previous quarters
    def get_previous_quarters(year, quarter, count):
        quarters = []
        for _ in range(count):
            if quarter == 1:
                quarter = 4
                year -= 1
            else:
                quarter -= 1
            quarters.append((year, quarter))
        return quarters

    # Get last 2 quarters for comparison
    previous_quarters = get_previous_quarters(current_year, current_quarter, 2)
    q2_year, q2_quarter = previous_quarters[0]
    q1_year, q1_quarter = previous_quarters[1]

    # Get metrics
    q1 = get_quarterly_metrics(df_deployment, q1_year, q1_quarter)
    q2 = get_quarterly_metrics(df_deployment, q2_year, q2_quarter)

    # Calculate Average Tenure
    def calculate_avg_tenure(df, quarter_end_date):
        active_employees = df[(df['Hire_Date'] <= quarter_end_date) &
                               ((df['Resignation_Date'] > quarter_end_date) | (df['Resignation_Date'].isna()))]
        return active_employees['Years_At_Company'].mean() if not active_employees.empty else 0

    q1_avg_tenure = calculate_avg_tenure(df_deployment, q1['end'])
    q2_avg_tenure = calculate_avg_tenure(df_deployment, q2['end'])
    avg_tenure_change = q2_avg_tenure - q1_avg_tenure

    # Three main metrics at the top
    col1, col2, col3 = st.columns(3)

    with col1:
        employee_button = display_employee_metrics(q1, q2)

    with col2:
        st.metric(
            "Average Tenure",
            f"{q2_avg_tenure:.1f} years",
            delta=f"{avg_tenure_change:.1f} years from Q{q1['quarter']}",
            delta_color="normal" if avg_tenure_change >= 0 else "inverse"
        )
        tenure_button = st.button("View Tenure Details", key="tenure_details")

    with col3:
        churn_button = display_churn_metrics(q1, q2)

```

```

# Display detailed sections based on button clicks
if employee_button:
    st.markdown("----")
    display_employee_details(df_deployment, q1, q2)

if churn_button:
    st.markdown("----")
    display_churn_details(df_deployment, current_year, current_quarter)

if tenure_button:
    st.markdown("----")
    st.subheader("Average Tenure Trend")

    # Get last 6 quarters
    last_six_quarters = get_previous_quarters(current_year, current_quarter, 6)
    quarters = []

    for year, quarter in last_six_quarters:
        quarter_data = get_quarterly_metrics(df_deployment, year, quarter)
        quarter_data['avg_tenure'] = calculate_avg_tenure(df_deployment, quarter_data['end'])
        quarters.append(quarter_data)

    # Sort chronologically
    quarters = sorted(quarters, key=lambda x: (x['year'], x['quarter']))

    trend_df = pd.DataFrame({
        'Quarter': [f"Q{q['quarter']} {q['year']}" for q in quarters],
        'Average Tenure': [q['avg_tenure'] for q in quarters],
        'Date': [q['end'] for q in quarters]
    })

    fig = go.Figure()
    fig.add_trace(go.Scatter(
        x=trend_df['Quarter'],
        y=trend_df['Average Tenure'],
        mode='lines+markers',
        name='Average Tenure',
        line=dict(color='skyblue', width=2)
    ))
    fig.add_trace(go.Scatter(
        x=trend_df['Quarter'],
        y=[trend_df['Average Tenure'].mean() * len(trend_df)],
        mode='lines',
        name='Average',
        line=dict(color='red', width=2, dash='dot'),
        hoverlabel=dict(font=dict(color='black')),
        hoverinfo='y+name'
    ))
    fig.update_layout(
        title='Average Tenure Trend (Last 6 Quarters)',
        xaxis_title='Quarter',
        yaxis_title='Tenure (years)',
        height=400,
        autosize=True,
        margin=dict(l=20, r=20, t=60, b=20),
        plot_bgcolor='rgba(0,0,0,0)',
        paper_bgcolor='rgba(0,0,0,0)',
        hovermode='x unified'
    )
    st.plotly_chart(fig, use_container_width=True)

# Quick navigation to hiring recommendations
st.markdown("----")
if st.button("Get Hiring Recommendations", key="go_to_recommendations"):
    st.session_state.current_page = "Hiring Recommendations"
    st.rerun()

```

```

def hiring_recommendations():
    if st.session_state.role != "Talent":
        st.warning("You don't have permission to access this page")
        return

    st.title("Hiring Recommendations")

    if 'df_deployment' not in st.session_state:
        st.warning("HR data not loaded. Please log in again.")
        return

    df = st.session_state.df_deployment
    st.subheader("Education and Experience Distribution of Top Performers")

    successful_employees = df[
        (df['Performance_Score'] >= 4) &
        (df['Employee_Satisfaction_Score'] >= 4) &
        (df['Resigned'] == False)
    ]

    if not successful_employees.empty:
        col1, col2 = st.columns(2)

        with col1:
            education_dist = successful_employees['Education_Level'].value_counts()
            fig1 = px.pie(
                education_dist,
                names=education_dist.index,
                values=education_dist.values,
                title='Education Distribution'
            )
            st.plotly_chart(fig1, use_container_width=True, key=f"education_chart_{str(np.random.randint(100000))}")

        with col2:
            fig2 = px.box(
                successful_employees,
                y='Years_At_Company',
                title='Experience Distribution of Top Performers',
                labels={'Years_At_Company': 'Years of Experience When Hired'}
            )
            fig2.update_traces(
                hoverinfo='y',
                hovertemplate='<b>%{y:.1f} years</b>',
                boxpoints=False
            )
            q1, med, q3 = np.percentile(successful_employees['Years_At_Company'], [25, 50, 75])
            fig2.add_annotation(
                x=0.5,
                y=med,
                text=f"<b>Median: {med:.1f} years</b>",
                showarrow=False,
                yshift=10
            )
            st.plotly_chart(fig2, use_container_width=True, key=f"experience_chart_{str(np.random.randint(100000))}")

    # Hiring priority recommendations
    st.markdown("---")
    st.subheader("Priority Hiring Recommendations")

    # Get departments with highest churn
    resigned = df[df['Resigned'] == True]
    churn_by_dept = resigned.groupby('Department').size().reset_index(name='Resignations')
    total_by_dept = df.groupby('Department').size().reset_index(name='Total')
    churn_data = pd.merge(churn_by_dept, total_by_dept, on='Department')
    churn_data['Churn Rate'] = (churn_data['Resignations'] / churn_data['Total']) * 100

    high_churn_depts = churn_data.sort_values('Churn Rate', ascending=False).head(3)

```

```

for _, row in high_churn_depts.iterrows():
    with st.expander(f"{'{row['Department']} Department (Churn Rate: {row['Churn Rate'].values[0]:.1f}%)'}"):
        st.write(f"- Recommended Actions:")
        st.write(f"- Prioritize hiring for {'{row['Department']} roles'")
        st.write(f"- Target candidates with 2+ years experience in similar roles")
        st.write(f"- Look for evidence of stability in previous positions")
        st.write(f"- Consider offering competitive benefits for these roles")

    # Downloadable report
    st.markdown("---")
    st.subheader("Generate Hiring Strategy Report")

    report_text = f"""
Hiring Strategy Recommendations
Generated on: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}

--- Priority Departments ---
{high_churn_depts[['Department', 'Churn Rate']].to_string(index=False)}

--- Education and Experience Distribution of Top Performers ---
Education: {education_dist.index[0]} ({(education_dist.values[0]/len(successful_employees))*100:.1f}%)  

Average Experience When Hired: {successful_employees['Years_At_Company'].mean():.1f} years

--- Recommended Actions ---
1. Prioritize hiring for {high_churn_depts.iloc[0]['Department']}
2. Target candidates with 2+ years experience
3. Implement skills testing for technical roles
"""

    st.download_button(
        label="Download Hiring Strategy Report",
        data=report_text,
        file_name="hiring_recommendations.txt",
        mime="text/plain",
        key=f"download_{str(np.random.randint(100000))}"
    )

# =====
def header():
    if st.session_state.role == "HR":
        st.title("HR Analytics Dashboard")
        st.write(f"Last login: {st.session_state.last_login}")
    elif st.session_state.role == "Talent":
        st.title("Talent Acquisition Dashboard")
        st.write(f"Last login: {st.session_state.last_login}")

    # Refresh button for both HR and Talent Acquisition
    if st.button("Refresh Data"):
        st.session_state.hr_data = load_data()
        st.success("Data refreshed!")
        st.rerun()

    st.markdown("---")

# MAIN FUNCTION
def main():
    init_session_state()

    st.set_page_config(
        page_title="Employee Churn Analytics Dashboard System",
        page_icon="📊",
        layout="wide"
    )

    st.markdown("""
<style>
.main {padding-top: 1rem;}
.stButton>button {border-radius: 8px;}
.stAlert {border-radius: 8px;}
</style>
""", unsafe_allow_html=True)

```

```
# Check authentication
if not st.session_state.logged_in:
    login_page()
else:
    create_navbar()
    header()

    # Page routing based on role
    if st.session_state.role == "HR":
        if st.session_state.current_page == "HR Dashboard":
            hr_dashboard()
        elif st.session_state.current_page == "Predict Employee Churn":
            churn_prediction()
        elif st.session_state.current_page == "Overview of Employee Performance":
            employee_performance()
        elif st.session_state.current_page == "Actionable Insights for Retention and Productivity":
            actionable_insights()

    elif st.session_state.role == "Talent":
        if st.session_state.current_page == "Talent Insights":
            talent_dashboard()
        elif st.session_state.current_page == "Hiring Recommendations":
            hiring_recommendations()
        elif st.session_state.current_page == "Predict Employee Churn":
            churn_prediction()

if __name__ == "__main__":
    main()
```

## Project Gantt Chart

### Project Gantt Chart Semester 1

```

import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Project data
task_names = [
    'Project Proposal Form', 'CHAPTER 1: INTRODUCTION', '1.1 Introduction', '1.2 Problem Background', '1.3 Project Aim',
    '1.4 Objectives', '1.5 Scope', '1.6 Potential Benefit', '1.6.1 Tangible Benefit', '1.6.2 Intangible Benefit',
    '1.6.3 Target User', '1.7 Overview of IR', '1.8 Project Plan', 'CHAPTER 2: LITERATURE REVIEW', '2.1 Introduction',
    '2.2 Domain Research', '2.3 Similar Systems/Works', '2.5 Summary', 'CHAPTER 3: METHODOLOGY', '3.1 Introduction',
    '3.2 Methodology', '3.3 Data Collection', '3.4 Initial Data Understanding', '3.5 Data Preprocessing', '3.6 Data Understanding',
    '3.7 Summary', 'CHAPTER 4: CONCLUSION', '4.1 Achievements of the First Part of the Project', '4.2 Research Justification and SDG Alignment',
    '4.3 Research Gaps and Future Improvements', 'Reference', 'Appendices'
]

start_dates = [
    'Mon 16/12/24', 'Thu 13/2/25', 'Mon 13/2/25', 'Thu 13/2/25', 'Fri 14/2/25',
    'Fri 14/2/25', 'Fri 14/2/25', 'Sat 15/2/25', 'Sat 15/2/25', 'Sat 15/2/25',
    'Sun 16/2/25', 'Mon 17/2/25', 'Tue 18/2/25', 'Wed 19/2/25', 'Wed 19/2/25',
    'Thu 20/2/25', 'Wed 26/2/25', 'Fri 28/2/25', 'Sat 1/3/25', 'Sat 1/3/25',
    'Sat 1/3/25', 'Sun 2/3/25', 'Sun 2/3/25', 'Wed 5/3/25', 'Sun 9/3/25',
    'Sun 9/3/25', 'Tue 11/3/25', 'Tue 11/3/25', 'Tue 11/3/25', 'Tue 11/3/25',
    'Tue 11/3/25', 'Tue 11/3/25'
]

finish_dates = [
    'Mon 6/1/25', 'Tue 18/2/25', 'Mon 13/2/25', 'Fri 14/2/25', 'Fri 14/2/25',
    'Fri 14/2/25', 'Fri 14/2/25', 'Sat 15/2/25', 'Sat 15/2/25', 'Sat 15/2/25',
    'Sun 16/2/25', 'Mon 17/2/25', 'Tue 18/2/25', 'Fri 28/2/25', 'Wed 19/2/25',
    'Tue 25/2/25', 'Fri 28/2/25', 'Mon 10/3/25', 'Sat 1/3/25', 'Sat 1/3/25',
    'Sun 2/3/25', 'Wed 5/3/25', 'Sun 9/3/25', 'Mon 10/3/25', 'Mon 10/3/25',
    'Wed 12/3/25', 'Wed 12/3/25', 'Wed 12/3/25', 'Wed 12/3/25', 'Wed 12/3/25',
    'Wed 12/3/25', 'Wed 12/3/25'
]

df = pd.DataFrame({
    'Task Name': task_names,
    'Start Date': pd.to_datetime(start_dates, format='%a %d/%m/%y'),
    'Finish Date': pd.to_datetime(finish_dates, format='%a %d/%m/%y')
})

df['Duration'] = (df['Finish Date'] - df['Start Date']).dt.days + 1

fig, ax = plt.subplots(figsize=(12, 18))

for i, task in enumerate(reversed(df['Task Name'])):
    row = df[df['Task Name'] == task].iloc[0]
    ax.barrh(task,
              width=row['Duration'],
              left=row['Start Date'],
              height=0.6,
              color='royalblue')

ax.xaxis.set_major_locator(mdates.WeekdayLocator(byweekday=mdates.MO, interval=1))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d/%m/%y'))
ax.xaxis.set_major_locator(mdates.WeekdayLocator(byweekday=mdates.MO, interval=2))
plt.xticks(rotation=45)
ax.xaxis.set_ticks_position('top')
ax.set_xlabel('Timeline')
ax.set_ylabel('Tasks')
ax.set_title('Project Gantt Chart Semester 1', pad=20, fontsize=16)
plt.tight_layout()
plt.grid(axis='x', alpha=0.3)
plt.show()

```

[16]

Project Gantt Chart Semester 2

```

import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

task_names = [
    'CHAPTER 1: INTRODUCTION', '1.1 Introduction', '1.2 Problem Background', '1.3 Project Aim', '1.4 Objectives',
    '1.5 Scope', '1.6 Potential Benefit', '1.6.1 Tangible Benefit', '1.6.2 Intangible Benefit', '1.6.3 Target User',
    '1.7 Overview of IR', '1.8 Project Plan', 'CHAPTER 2: LITERATURE REVIEW', '2.1 Introduction', '2.2 Domain Research',
    '2.3 Similar Systems/Works', '2.5 Summary', 'CHAPTER 3: METHODOLOGY', '3.1 Introduction', '3.2 Methodology',
    '3.3 Summary', 'CHAPTER 4: DESIGN AND IMPLEMENTATION', '4.1 Introduction', '4.2 Data Collection', '4.3 Initial Data Understanding',
    '4.4 Data Preprocessing', '4.5 Data Understanding', '4.6 Model Building', '4.7 Summary', 'CHAPTER 5: RESULT AND DISCUSSIONS',
    '5.1 Introduction', '5.2 Model Evaluation and Discussions', '5.3 Feature Importance', '5.4 Model Deployment', '5.5 Summary',
    'CHAPTER 6: CONCLUSION', '6.1 Critical Evaluation', '6.2 Limitations', '6.3 Recommendations', 'Reference', 'Appendices'
]

start_dates = [
    'Fri 16/5/25', 'Fri 16/5/25', 'Sat 17/5/25', 'Tue 20/5/25', 'Tue 20/5/25',
    'Tue 20/5/25', 'Wed 21/5/25', 'Wed 21/5/25', 'Thu 22/5/25',
    'Thu 22/5/25', 'Fri 23/5/25', 'Fri 23/5/25', 'Sat 24/5/25', 'Tue 27/5/25',
    'Wed 28/5/25', 'Sun 1/6/25', 'Mon 2/6/25', 'Mon 2/6/25', 'Tue 3/6/25',
    'Sat 7/6/25', 'Sat 7/6/25', 'Sun 8/6/25', 'Mon 9/6/25', 'Thu 12/6/25',
    'Sun 15/6/25', 'Mon 16/6/25', 'Sun 22/6/25', 'Tue 8/7/25', 'Tue 8/7/25',
    'Wed 9/7/25', 'Sat 12/7/25', 'Sun 13/7/25', 'Sun 20/7/25', 'Tue 21/7/25',
    'Tue 21/7/25', 'Tue 22/7/25', 'Tue 22/7/25', 'Tue 22/7/25', 'Tue 22/7/25'
]

finish_dates = [
    'Fri 23/5/25', 'Fri 16/5/25', 'Mon 19/5/25', 'Tue 20/5/25', 'Tue 20/5/25',
    'Tue 20/5/25', 'Wed 21/5/25', 'Wed 21/5/25', 'Wed 21/5/25', 'Thu 22/5/25',
    'Thu 22/5/25', 'Fri 23/5/25', 'Sun 1/6/25', 'Fri 23/5/25', 'Tue 27/5/25', 'Mon 1/6/25',
    'Mon 1/6/25', 'Sun 1/6/25', 'Mon 2/6/25', 'Sat 7/6/25', 'Fri 6/6/25',
    'Sat 7/6/25', 'Tue 8/7/25', 'Sun 8/6/25', 'Thu 12/6/25', 'Sun 15/6/25',
    'Sat 21/6/25', 'Tue 8/7/25', 'Tue 8/7/25', 'Tue 8/7/25', 'Sun 20/7/25',
    'Fri 11/7/25', 'Sat 12/7/25', 'Sat 19/7/25', 'Sun 20/7/25', 'Mon 21/7/25',
    'Tue 22/7/25', 'Tue 22/7/25', 'Tue 22/7/25', 'Tue 22/7/25', 'Tue 22/7/25'
]

df = pd.DataFrame({
    'Task Name': task_names,
    'Start Date': pd.to_datetime(start_dates, format='%a %d/%m/%y'),
    'Finish Date': pd.to_datetime(finish_dates, format='%a %d/%m/%y')
})

df['Duration'] = (df['Finish Date'] - df['Start Date']).dt.days + 1

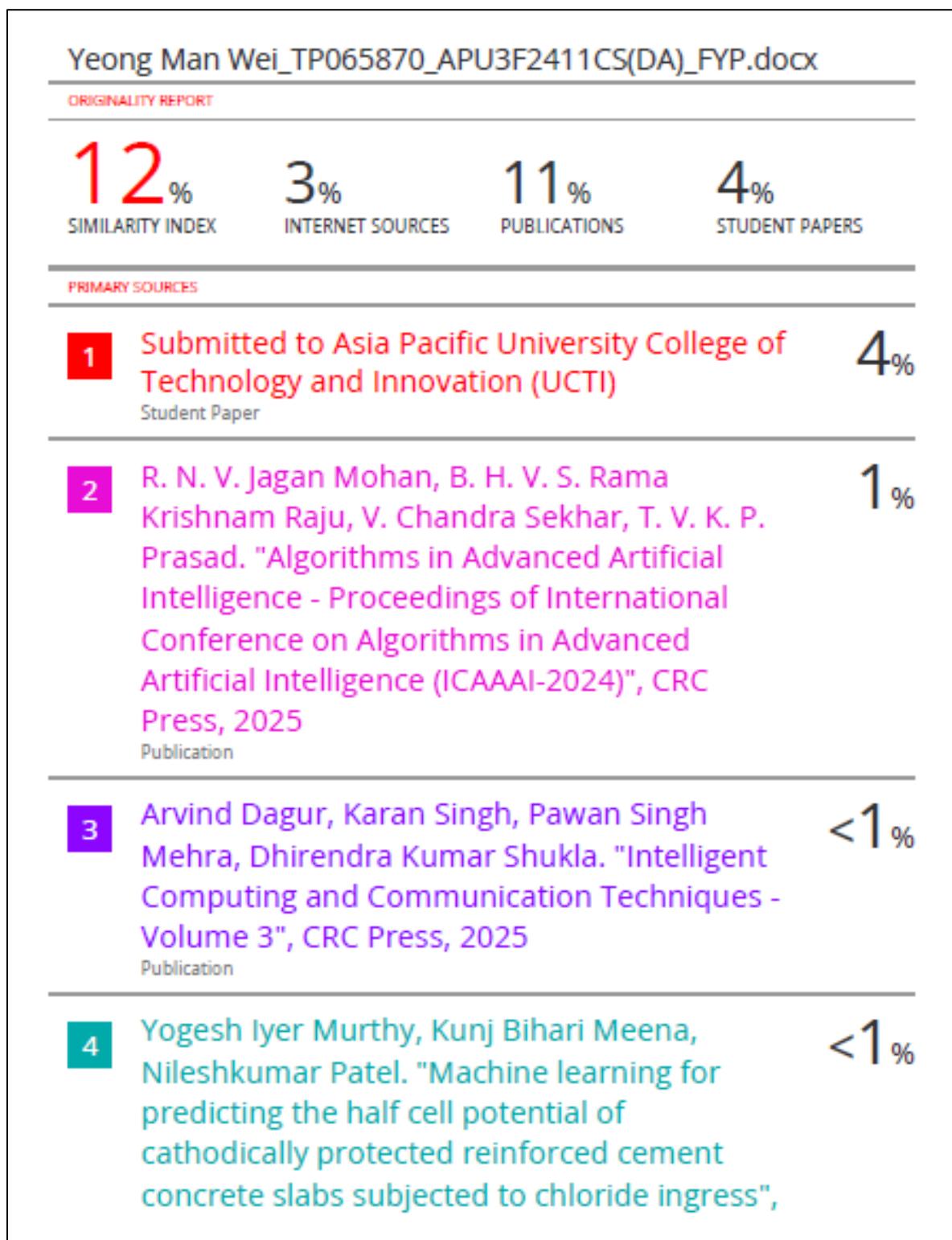
fig, ax = plt.subplots(figsize=(12, 10))

for i, task in enumerate(reversed(df['Task Name'])):
    row = df[df['Task Name'] == task].iloc[0]
    ax.bobar(task,
              width=row['Duration'],
              left=row['Start Date'],
              height=0.6,
              color='royalblue')

ax.xaxis.set_major_locator(mdates.WeekdayLocator(byweekday=mdates.MO, interval=1))
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d/%m/%y'))
plt.xticks(rotation=45)
ax.xaxis.set_ticks_position('top')
ax.set_xlabel('Timeline')
ax.set_ylabel('Tasks')
ax.set_title('Project Gantt Chart Semester 2', pad=20, fontsize=16)
plt.tight_layout()
plt.grid(axis='x', alpha=0.3)
plt.show()

```

Figure 294: Code Implementation

**Appendix G – Turnitin Similarity Report***Figure 295: Turnitin Similarity Report*

Engineering Applications of Artificial Intelligence, 2024 Publication		
5	T. Mariprasath, Kumar Reddy Cheepati, Marco Rivera. "Practical Guide to Machine Learning, NLP, and Generative AI: Libraries, Algorithms, and Applications", River Publishers, 2024 Publication	<1 %
6	Natasa Kleanthous, Abir Hussain. "Machine Learning in Farm Animal Behavior using Python", CRC Press, 2025 Publication	<1 %
7	H.L. Gururaj, Francesco Flammini, S. Srividhya, M.L. Chayadevi, Sheba Selvam. "Computer Science Engineering", CRC Press, 2024 Publication	<1 %
8	Thiele, Lukas Tom. "Uber's Scale-Up – Case Study: Introduction of Autonomous Vehicles to Scale Uber's Business Model", Universidade NOVA de Lisboa (Portugal), 2024 Publication	<1 %
9	S. Prasad Jones Christy dass, Nurhayati Nurhayati, S. Kannadhasan. "Hybrid and Advanced Technologies", CRC Press, 2025 Publication	<1 %
10	Satya Ranjan Mishra, Apul Narayan Dev, Alok Kumar Pandey, Mukesh Kumar Awasthi.	<1 %

Figure 296: Turnitin Similarity Report