**EMPLOYEE PROMOTION FORECASTING WITH ML**

**ABSTRACT**

In India, employee promotions are often tied to seniority and performance evaluations, with only 9.4% of employees receiving annual promotions, as reported by Aon in 2021. This situation is especially evident in the IT and financial services sectors, where promotion rates are higher, but manual assessment processes remain largely subjective. Existing promotion systems suffer from biases, inconsistencies, and delays, with human decision-making influencing outcomes through favoritism, gender bias, and inconsistent performance assessments, all of which hinder fair career progression.

The proposed system aims to revolutionize the employee promotion process by leveraging machine learning for accurate, unbiased promotion forecasting. Our approach begins with extensive data preprocessing, followed by the application of SMOTE (Synthetic Minority Over-sampling Technique) to balance the dataset, addressing class imbalance issues. The system also integrates Exploratory Data Analysis (EDA) techniques, including histograms, box plots, scatter plots, correlation heatmaps, violin plots, count plots, and KDE plots to gain deeper insights into the dataset. The data is then split into training and test sets, with performance models trained and evaluated using Logistic Regression Classifier (LRC) for baseline performance. The novelty of the system lies in the proposed use of a Decision Tree Classifier (DTC) with AdaBoost, a powerful ensemble method that improves the robustness and accuracy of promotion predictions by reducing bias and overfitting. This comprehensive approach not only enhances the fairness and efficiency of the promotion process but also motivates employees by providing data-driven insights into their career advancement potential. The final model leverages historical employee performance data, which includes a label indicating whether the employee was promoted (1) or not (0), to make reliable promotion predictions.

**CHAPTER 1**

**INTRODUCTION**

* 1. **Overview**

In modern organizational environments, accurate and fair employee promotion decisions are vital for maintaining workforce morale, motivation, and productivity. Traditional promotion practices often rely on manual evaluations, which are time-consuming, prone to human bias, and inconsistent across departments. These shortcomings can result in underutilization of talent, workplace dissatisfaction, and hindered organizational growth. As organizations increasingly digitize their operations, there is a growing opportunity to integrate data-driven approaches into human resource (HR) decision-making processes.

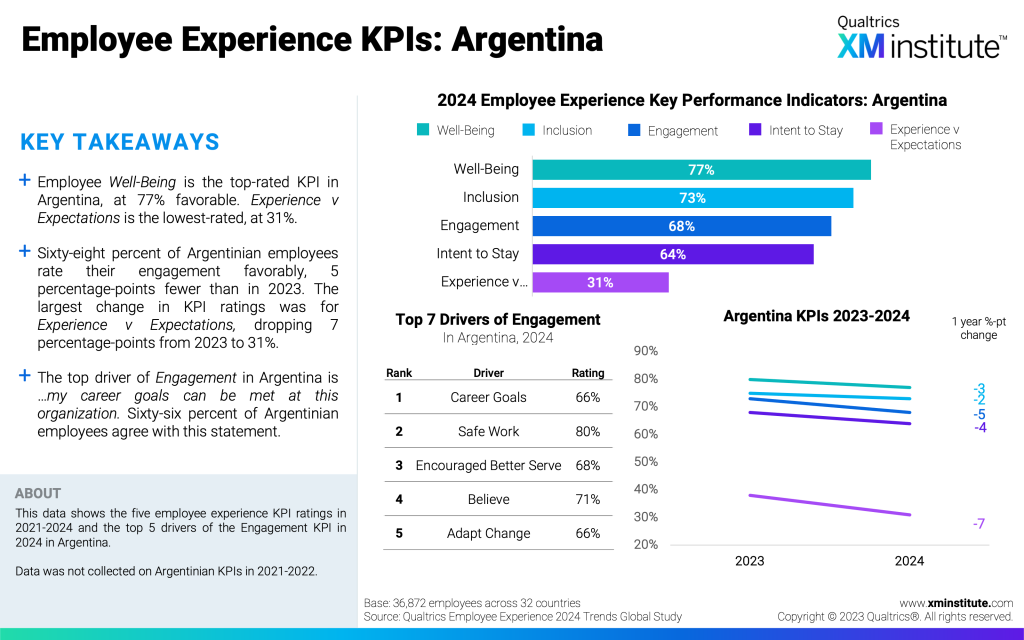


Fig. 1.1: Employee Experience Trends: Americas.

The project proposes a machine learning (ML)-based system for forecasting employee promotions using structured historical employee data. The dataset contains features such as employee performance, previous promotions, and other attributes, along with labels indicating promotion status (1 for promoted, 0 for not promoted). The system begins with data preprocessing and cleaning to handle missing or inconsistent values. Due to class imbalance, the SMOTE (Synthetic Minority Over-sampling Technique) algorithm is applied to balance the dataset and improve model learning. Comprehensive Exploratory Data Analysis (EDA) is conducted using visual tools like histograms, box plots, scatter plots, violin plots, KDEs, count plots, and correlation heatmaps to uncover trends and patterns in employee promotion behavior.

Following data preparation, the dataset is split into training and testing subsets. As a baseline, Logistic Regression Classifier (LRC) is implemented, followed by a more advanced approach involving a Decision Tree Classifier (DTC) enhanced with AdaBoost. This ensemble technique improves prediction accuracy and reduces overfitting. By adopting this ML-driven framework, organizations can make faster, more objective promotion decisions, ultimately enhancing transparency, fairness, and employee trust in HR processes.

* 1. **Research Motivation**

Employee promotion plays a critical role in shaping organizational structure, workforce motivation, and overall performance. However, in many companies, promotion decisions are still made manually based on subjective assessments, seniority, or static appraisal systems. These traditional methods often fail to consider the multifaceted performance indicators of employees and are susceptible to unconscious bias, favoritism, and inconsistencies. Such inefficiencies can demoralize deserving employees, increase attrition rates, and negatively impact productivity and organizational growth.

The increasing availability of employee data and the rise of machine learning present an opportunity to reform this decision-making process through data-driven solutions. By leveraging historical employee data and intelligent classification models, organizations can identify patterns and key indicators that correlate with successful promotions. This not only improves the transparency and fairness of promotion decisions but also supports human resource departments with evidence-based insights, ultimately fostering a meritocratic work culture.

The research is motivated by the need to bridge the gap between traditional HR practices and modern data science techniques. By incorporating data preprocessing, SMOTE to address class imbalance, and extensive EDA to visualize insights, the system is made more robust. The performance of a baseline Logistic Regression Classifier (LRC) is compared against a more refined Decision Tree Classifier (DTC) with AdaBoost to enhance prediction accuracy and reduce overfitting. The ultimate goal is to enable organizations to adopt ML-based decision-making tools that ensure promotions are granted based on comprehensive, unbiased assessments, thereby enhancing employee satisfaction, retention, and operational efficiency in the long term.

* 1. **Problem Definition**

In today's competitive corporate landscape, employee promotions are critical decisions that significantly influence workforce morale, retention, and organizational productivity. Despite their importance, most organizations still rely on traditional and manual methods for making promotion decisions, which are often influenced by subjective judgment, managerial bias, and limited performance metrics. These conventional approaches lack consistency and fail to leverage the full spectrum of employee data available. As a result, capable and deserving employees may be overlooked, leading to dissatisfaction, reduced motivation, and increased turnover.

The challenge lies in accurately identifying and forecasting which employees are most suitable for promotion based on historical and current performance data. Given the complex, multidimensional nature of employee profiles — encompassing factors such as years of experience, training hours, performance ratings, and department — manual methods are inadequate for effectively processing and analyzing such data. Furthermore, the inherent class imbalance in promotion datasets (since only a small fraction of employees are typically promoted) poses an additional hurdle for standard machine learning models.

The research aims to address the problem by developing a machine learning-based prediction system that can objectively and efficiently forecast employee promotions. Using the Employee dataset labeled with promotion status (0 = Not Promoted, 1 = Promoted), the study implements thorough data preprocessing, SMOTE for class balancing, and Exploratory Data Analysis (EDA) through various plots and correlation heatmaps. A baseline Logistic Regression Classifier (LRC) is compared with a proposed Decision Tree Classifier (DTC) integrated with AdaBoost, aiming to deliver more accurate and explainable predictions for fair and data-driven promotion decisions.

* 1. **Significance**
* **Enhances Objectivity in Promotion Decisions:** The model eliminates subjective human biases by relying on data-driven predictions, ensuring that employees are evaluated fairly based on quantifiable performance indicators, training scores, and experience levels.
* **Improves HR Efficiency and Strategic Planning:** By automating the identification of promotion-worthy candidates, HR departments can save considerable time and resources, focusing instead on career development strategies and workforce planning aligned with organizational goals.
* **Supports Talent Retention and Employee Satisfaction:** Transparent and consistent promotion practices increase trust in the system, leading to higher employee morale, lower turnover rates, and better engagement across departments, which ultimately benefits long-term organizational performance.
* **Provides Insightful Analytical Visualizations:** Through EDA techniques such as histograms, box plots, scatter plots, KDEs, and heatmaps, the system gives HR managers an interpretable overview of how variables like education, training hours, and KPIs influence promotion chances, enabling better policy formulation.
* **Combines SMOTE and Ensemble Learning for Accuracy:** The integration of SMOTE for handling class imbalance, and the proposed Decision Tree Classifier (DTC) enhanced with AdaBoost, significantly improves prediction accuracy over traditional models like Logistic Regression Classifier (LRC), especially in imbalanced datasets where promoted employees are a minority class.
  1. **Applications**
* **HR Decision Support Systems:** Integrates with Human Resource Management Systems (HRMS) to assist HR managers in making accurate, timely, and unbiased promotion decisions based on employee data analytics.
* **Workforce Planning and Succession Management:** Helps organizations identify future leaders by predicting high-performing employees suitable for promotion, enabling better succession planning and reducing leadership gaps.
* **Corporate Training and Development Programs:** Highlights key factors influencing promotions, allowing companies to design targeted upskilling or reskilling programs for employees who lack critical attributes or scores.
* **Performance Evaluation Systems:** Acts as an extension of traditional appraisal systems by quantifying promotion readiness, offering a holistic view of performance that includes historical trends, learning progress, and contribution metrics.
* **Retention and Engagement Strategies:** By predicting potential promotions, the system aids in proactively retaining top talent who might otherwise leave due to perceived stagnation or lack of growth opportunities.

**CHAPTER 2**

**LITERATURE SURVEY**

In recent times, organizations have given an increased level of attention to the human resources domain. Starting from, firstly, meticulously selecting the optimal candidates for the needed roles, followed by the process of developing their skills in alignment with the requirements of the organization, and, finally, evaluating them and assessing their performance and abilities, a process that can lead to offering these employees enhanced conditions for their work life, salary increases, or bonuses, which are offered based on performance. Consequently, it becomes highly important for companies to determine and anticipate the capabilities and the performance of their employees, factors that later will contribute to their productivity and organizational development [1].

Evaluating an employee’s performance brings challenges, as it implies offering feedback and deciding on their future career development, salary, or promotion. It also involves identifying areas that require updates or modifications. Numerous research studies have explored methodological factors such as academic credentials, technical qualifications, characteristics, and psychological aspects as indicators of employee performance in organizations. However, these factors are applicable only in certain employment domains. It is crucial to explore a multitude of other factors to gain a comprehensive understanding of employee performance [2].

Different elements can contribute to workforce attrition, including low job satisfaction, inappropriate wages, family concerns or a demanding business environment. Poor performance leads to involuntary employee attrition, which will affect, at the same time, the organization’s productivity and its progression and development [3,4].

Numerous researchers have investigated the models that can help us predict employee performance. We can identify the fact that, generally, an individual that supervises the employee or are their direct manager is the best person to conduct an employee performance evaluation [5]. It has also been determined that high levels of job satisfaction contribute to the increasing loyalty of the employee, a fact that later can reduce turnover rates [6]. Additionally, a complete evaluation should include static and dynamic elements [7].

Glinow proposes that attaining and maintaining high performance standards emerges as a predominant concern across various types of organizations, including private, public, for-profit, and non-profit entities. According to him, achieving high performance levels, accompanied by positive indicators, enhances the stability of the organization, while ensuring high levels of profitability, quality, productivity, motivation, innovation, and efficiency. Conversely, he asserts that low performance levels entail negative and dysfunctional outcomes for the organization. He argues that instances of low performance indicators are associated with specific circumstances [8].

The most classical sources of performance data generally face limitations in correctly capturing the dynamic nature of performance. Particularly, it is well known that supervisors or managers are the ones who carry out the performance appraisal, with colleagues, subordinates, or even customers providing additional feedback or notes to them [8].

Considering the dynamic nature of organizations, these sources and their ratings are often prone to rapid changes before the moment of the appraisal; therefore, these become unsuitable. Consequently, it also becomes more essential to adopt new methods for analyzing data from discrete sources and to gain a more comprehensive picture of employee performance in more organizational contexts [9].

Recent research applying data mining techniques to predict employee performance scores concludes that the most important goal is to minimize the influence of subjective factors and reduce personal biases as well [10]. Diverse performance attributes can direct the selection of appropriate data mining methods, promoting synchronization across multiple areas like business operations, technology, or information science. As a consequence, the process of performance evaluation tends to become a more scientific approach, lowering the arbitrary nature of artificial scoring. This development contributes to enhanced fairness, authority in assessments, and integrity, as well as simultaneously elevating employee engagement, productivity, and team collaboration [10].

The development of data mining technology is clear and well-known, considering the years spent undertaking research and investigating practical applications, a fact that has provided increasingly complex model types and even mining functions. This maturity enables the provision of decision support in employee performance evaluations by correctly choosing the right index systems and training models [11].

Recent efforts regarding employee performance classification have considered the implementation of various machine learning algorithms. Some studies targeted the exploration of psychological, socioeconomic and creative factors on employee performance and motivation [2,12]. One important research study considered the use of prediction model construction algorithms, such as random forest, logistic regression, support vector machine, artificial neural network or naïve Bayes [13].

Other studies underline the critical importance of employee performance in organizational operations, highlighting it as the central factor in determining survival and competitiveness. Additionally, employee performance significantly influences the rewards system within an organization. Lucy acknowledges that performance is linked to actions related to productivity, innovation, flexibility, production levels, commitment, absenteeism rates, and the overall image of the organization. He categorizes performance into high, moderate, and low levels, emphasizing that low performance is the least desirable state for any organization. He advocates for high performance levels, which correlate with increased productivity, innovation, quality, efficiency, and commitment, recognizing the potential for better prospects for the organization [14].

Liu et al. introduced a method based on artificial intelligence for predicting the employee turnover while using a dataset that was built from state enterprises. Feature extraction was undertaken as well, to determine crucial factors affecting employee performance. For classification purposes, algorithms such as random forest, support vector machine (SVM), and linear regression (LR) were used again, together with AdaBoost, also concluding that there was a direct correlation between the employees’ skills and associated performance scores [15]. Another important work focused on correctly and efficiently classifying employee job performance, based on DISC personality ((D)ominance, (i)nfluence, (S)teadiness and (C)onscientiousness). The classification for this personality test is built by comparing an individual personality with the standard personality test that the person took. DISC represents one of the most popular tests in this domain, standing for dominance, influence, steadiness and compliance. They built some models that were tested on a self-made dataset concerning the results of the DISC personality test for 2137 employees. For these models, the authors again used algorithms such as SVM, K-nearest neighbors (KNN), random forest (RF), LR, decision trees or naïve Bayes. Regarding the results, it was concluded that, for the selection chosen, decision trees provided the best performance, with the lowest Hamming loss and the highest accuracy. In addition, as feature selection techniques, the results were better when using multi-label classification with a stacking technique [16].

Jayadi et al. also investigated employee performance predictions using data mining, looking closely into the use of naïve Bayes for a dataset which was based on 310 employees [17]. Ajit et al. focused on an approach based on the eXtreme Gradient Boosting (XGBoost) classifier, again using a self-made dataset with 73,115 labeled data registries. The introduction of feature extraction can also be identified in this last research study, while underlining the impact of turnover [18].

Fallucchi et al. looked closer into machine learning approaches, like K-nearest neighbors, SVM, naïve Bayes, logistic regression, or random forest, regarding leaving the company. Their analysis included objective factors affecting worker wishes about turnover, considering both the correlation matrix for those features and running statistical analysis [19].

Hamidah et al. explored various classification methods such as decision trees (DT), neural networks (NN), and K-nearest neighbors (KNN) for predicting talent outcomes. Their research aimed to identify the most accurate technique for processing Human Resource (HR) data. The findings indicated that the decision tree method was notably effective for talent forecasting within human resource management (HRM), demonstrating the highest level of accuracy. The data utilized in this research were gathered from an academic institution’s staff database [20].

Juvitayapun et al. proposed logistic regression, random forest, gradient boosting tree and extreme gradient boosting tree classifiers to identify employees’ likelihood of turnover, while Duan et al. suggested logistic regression and XGBoost, with the latter being the better algorithm for the same purpose, outperforming logistic regression [21].

Last, but not least, Sujatha et al. introduced machine learning classifiers such as XGBoost and gradient boosting, while working with a real-time dataset [22]. Obiedat et al. tried to achieve the prediction of productivity performance in the garment sector, offering a hybrid algorithm that combines multiple algorithms for classification such as random forest, naïve Bayes, support vector machine, and multi-layer perceptron. They also tried to incorporate ensemble-learning methods such as AdaBoost and Bagging [23].

**CHAPTER 3**

**TRADITIONAL SYSTEM**

**3.1. Traditional HR Performance Reviews**

In manual systems, employee promotion decisions are often based on periodic performance reviews conducted by managers. These reviews usually occur annually or bi-annually and are based on subjective evaluations. Managers assess employees’ contributions, skills, behavior, and accomplishments during the review period. The assessments are typically presented in the form of written reports or oral evaluations. Based on these reviews, HR departments then determine promotions based on the manager's subjective views of the employee's potential.

The traditional performance review system, however, often lacks consistency due to individual biases from managers. Employees' work performance can be influenced by personal relationships, mood, or the subjective evaluation criteria used by different managers. This creates challenges in ensuring fairness and consistency across the organization, as different managers may interpret an employee’s contributions differently. Furthermore, reviews usually do not incorporate data-driven insights or quantitative factors, leading to potential oversight of important indicators of employee potential.

As a result, the manual process tends to be slow and inefficient. Managers and HR personnel spend considerable time and resources evaluating each employee, compiling reports, and deliberating on potential promotions. The time-consuming nature of this process can delay decisions and create frustration among employees, especially if the decisions are perceived as unfair or unclear.

**3.2. Employee Self-Assessment Forms**

Another manual system used for employee promotion forecasting involves self-assessment forms, where employees evaluate their own performance. This system allows employees to report on their achievements, skills, and areas of improvement. These self-assessment forms are often complemented by feedback from peers or team members, providing a 360-degree perspective of the employee’s performance.

While self-assessment can give employees a voice in their promotion process, it presents significant drawbacks. One issue is the potential for overestimation or underestimation of personal achievements, as employees may either inflate their accomplishments to position themselves for a promotion or downplay their successes due to modesty. The system lacks objectivity, leading to discrepancies in self-assessments that could be influenced by personal biases, emotions, or external pressures, ultimately skewing the promotion decisions.

Furthermore, self-assessment systems are often inconsistent across organizations and industries, with no standardized format to ensure that all employees are evaluating themselves on the same metrics or criteria. This lack of uniformity means that comparing employees across departments or teams becomes difficult, which reduces the reliability of promotion decisions based on these assessments. The process can also be highly time-consuming for both employees and HR teams to review, leading to delays in final promotion decisions.

**3.3. Managerial Nomination and Recommendation System**

In many organizations, promotions are decided through a nomination or recommendation process where managers nominate employees they believe are deserving of promotion based on their observations and overall potential. Managers submit their recommendations to HR, who then evaluates them and makes a final decision. This system relies heavily on the judgment of department heads or team leaders and is intended to ensure that employees who demonstrate leadership qualities or strong potential are recognized for career advancement.

However, this system is prone to several inefficiencies. The most notable one is the over-reliance on managerial biases. Managers’ personal preferences, their working relationship with employees, or the sheer visibility of certain individuals may influence their recommendations, leading to unfair promotions or overlooking deserving employees who may be working under less visible circumstances.

Another drawback of this approach is the lack of quantitative evaluation, which makes it harder to ensure objective and data-driven decision-making. Without concrete performance data or standardized criteria, managers' recommendations may be subjective and inconsistent, contributing to employee dissatisfaction or resentment. Additionally, this process tends to be time-consuming and often leads to delays, as managers may take considerable time to assess employees and compile their nominations, causing the entire promotion process to be slower and more prone to error.

**3.4 Limitations of Manual Systems**

* **Subjectivity and Bias:** Manual systems heavily rely on human judgment, which can lead to subjective evaluations based on personal biases. This can lead to favoritism or overlooked deserving candidates.
* **Inefficiency and Time Consumption:** Manual processes such as performance reviews, self-assessments, and nominations can be time-consuming. They require significant effort from managers, HR teams, and employees, resulting in delays in decision-making.
* **Inconsistency Across Departments:** Different managers or departments may use different criteria for evaluating employees, leading to inconsistencies in how promotions are assessed across the organization.
* **Limited Data-Driven Insights:** Manual systems often lack the ability to leverage data analytics and predictive models, which can result in a reliance on anecdotal or incomplete information for making promotion decisions.
* **Employee Frustration and Low Morale:** The lack of transparency and perceived unfairness in manual systems can lead to employee dissatisfaction. Employees may feel that their contributions are not adequately recognized, leading to decreased motivation and potential attrition.

**CHAPTER 4**

**PROPOSED ALGORITHM**

**4.1 Overview**

The proposed algorithm combines a series of novel techniques and methodologies not yet explored together in existing surveys on employee promotion forecasting. By using a data preprocessing approach that includes handling missing values, encoding categorical data, and standardizing numerical features, this method ensures that the dataset is clean and suitable for analysis. The combination of SMOTE (Synthetic Minority Over-sampling Technique) to address class imbalance and Exploratory Data Analysis (EDA) through a variety of visualizations such as histograms, box plots, scatter plots, and correlation heatmaps allows a deep understanding of the data's characteristics. Additionally, the Train-Test Split methodology is integrated to ensure proper evaluation of the model. The existing Logistic Regression Classifier (LRC) is incorporated as a baseline model, while the proposed Decision Tree Classifier (DTC) combined with AdaBoost Classifier offers an advanced, more robust alternative. The AdaBoost algorithm enhances the decision tree model by focusing on misclassified instances, thereby improving predictive performance. This combination of techniques and models ensures that the approach overcomes key issues found in existing surveys, such as subjectivity, inefficiency, and inconsistency in promotion forecasting systems. The proposed methodology emphasizes data-driven decision-making, fairness, and efficiency, which were limitations of manual methods.

**Step-1: Data Preprocessing**

The first step in the proposed methodology involves comprehensive data preprocessing. The raw employee dataset typically contains both numerical and categorical features. The numerical features, such as age, years of experience, or performance ratings, are standardized to ensure that they are on the same scale. This step prevents features with larger values from dominating the model. Categorical variables, like job titles or departments, are transformed into numerical format using techniques such as one-hot encoding or label encoding. Additionally, missing values in the dataset are imputed using appropriate methods, such as the mean or median for numerical features and the mode for categorical features, ensuring no data points are lost. This preprocessing step is critical as it cleans the data, making it ready for the subsequent stages.

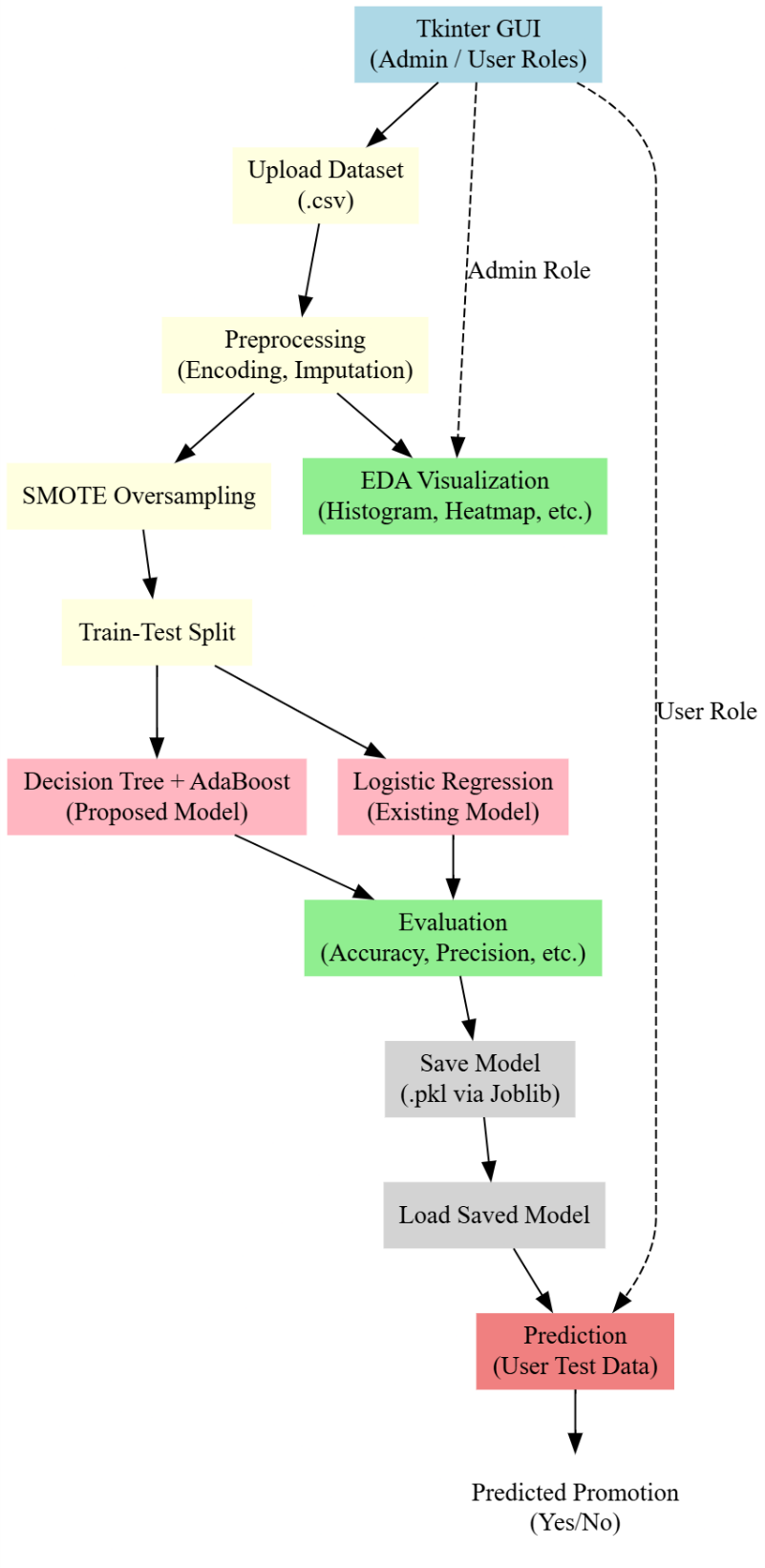


Fig. 4.1: Proposed system of forecasting employee promotion.

**Step-2: Handling Class Imbalance with SMOTE**

Employee promotion datasets typically suffer from class imbalance, where the number of employees promoted (labeled as ‘1’) is much lower than those not promoted (labeled as ‘0’). To address this imbalance, we apply SMOTE (Synthetic Minority Over-sampling Technique). SMOTE generates synthetic samples of the minority class (promoted employees) by interpolating between existing minority samples. This helps balance the dataset by ensuring that the model doesn’t become biased towards predicting the majority class (non-promoted employees). SMOTE enables the algorithm to learn better generalizations about the minority class and improves the model’s performance in predicting promotions.

**Step-3: Exploratory Data Analysis (EDA)**

Once the data is preprocessed and balanced, Exploratory Data Analysis (EDA) is conducted to gain insights into the data and understand its structure. This involves the use of various visualizations:

* **Histograms** to analyze the distribution of numerical variables like years of experience or performance scores.
* **Box plots** to detect outliers and understand the spread of data.
* **Scatter plots** to visualize relationships between two variables, such as performance score versus years of experience.
* **Correlation heatmaps** to identify relationships between different features and determine which variables are most relevant to employee promotion.
* **Violin plots and Count plots** provide a deeper understanding of class distributions for both promoted and non-promoted employees.
* **KDE (Kernel Density Estimation)** plots offer a smoothed view of data distributions. These EDA techniques help uncover hidden patterns in the data and identify which features are most important for predicting promotions.

**Step-4: Train-Test Split**

To ensure the model is properly evaluated, the dataset is divided into training and testing sets. Typically, an 80-20 or 70-30 split is used, where 80% of the data is used for training the model, and the remaining 20% is reserved for testing. This step allows for a fair assessment of the model’s performance on unseen data, ensuring that it generalizes well and does not overfit to the training data. This splitting is crucial for evaluating the robustness of the models and understanding how they perform in real-world scenarios.

**Step-5: Model Training (Logistic Regression and Decision Tree Classifier)**

The next step involves training two models: a Logistic Regression Classifier (LRC) and a Decision Tree Classifier (DTC). The LRC serves as a baseline model to evaluate its performance on the promotion dataset. Logistic regression models the probability of promotion based on employee features but may struggle with more complex relationships. On the other hand, the Decision Tree Classifier is more capable of capturing non-linear relationships in the data. However, decision trees are prone to overfitting, especially when the data is noisy.

**Step-6: AdaBoost to Improve Decision Tree Performance**

To address the overfitting issue of the decision tree and improve its predictive power, we apply the AdaBoost Classifier. AdaBoost works by training multiple weak classifiers (in this case, decision trees) sequentially. Each subsequent classifier focuses on correcting the mistakes made by the previous one. AdaBoost combines these weak classifiers into a strong ensemble model that delivers better performance than a single decision tree. This technique enhances the decision tree by giving more weight to misclassified instances, ensuring that the model improves its predictions over time.

**Step-7: Model Evaluation**

After training the models, performance is evaluated using several metrics:

* **Accuracy** to measure the overall correctness of the model.
* **Precision, Recall, and F1-Score** to assess the model's ability to identify promoted employees correctly.
* **AUC-ROC Curve** to measure the ability of the model to distinguish between the two classes (promoted vs. non-promoted employees).  
  The evaluation metrics ensure that the proposed model does not only perform well on the training data but also generalizes well to unseen data, offering a fair and unbiased decision-making process.

**Step-8: Deployment and Feedback Loop**

Once the model has been trained and evaluated, it is deployed within the organization’s HR system for real-time prediction. As the system gathers feedback from actual promotion decisions, the model is periodically retrained with updated data to ensure its accuracy and relevance over time. This dynamic feedback loop ensures the system continuously improves and adapts to changing organizational needs and employee performance trends.

**4.2 Data Preprocessing**

The preprocessing pipeline ensures that the dataset is clean, structured, and suitable for machine learning. By handling missing values, removing duplicates, encoding categorical data, visualizing data distribution, and selecting relevant features, this approach enhances data quality and improves the performance of predictive real estate models.

**Step-1: Handling Missing Values**

In real-world datasets, missing values are quite common, and they can affect the model’s performance. One of the first steps in data preprocessing is handling these missing values. There are multiple strategies to address missing data, depending on the nature of the feature. Numerical columns, such as salary or years of experience, typically use imputation techniques to fill in the missing values. Common methods include replacing missing values with the mean, median, or mode of the respective column. For categorical columns, where values represent categories (e.g., department or job role), missing values may be replaced with the mode (the most frequent category) or sometimes with a placeholder value to indicate missing data. This step ensures that no data points are excluded, maintaining the integrity of the dataset.

**Step-2: Encoding Categorical Features**

Many datasets, including the employee promotion dataset, contain categorical data (e.g., job roles, departments). Machine learning algorithms typically cannot process categorical data directly, which requires converting these features into numerical form. Label encoding is one technique where each unique category is assigned an integer value. For example, "Sales" might be assigned the value 0, "HR" might be 1, and so on. Another common method is one-hot encoding, which creates a binary column for each category. For instance, a column for "Department" could generate separate columns for "Sales," "HR," "Engineering," etc., with values of 0 or 1 to indicate the presence of a specific department for each employee. This encoding process ensures that the machine learning models can interpret categorical data effectively.

**Step-3: Feature Scaling**

Feature scaling is another essential step in preprocessing, especially for algorithms that rely on distance metrics, such as k-nearest neighbors (KNN) or gradient descent-based models. When features vary greatly in scale, for example, an employee’s age might range from 20 to 60, while years of experience could range from 1 to 30, the model may give undue importance to the feature with the larger scale. Standardization (or Z-score normalization) is one technique used for scaling features, which transforms the data to have a mean of 0 and a standard deviation of 1. Another method is min-max scaling, which transforms the data to lie between a specified range, typically 0 and 1. Scaling ensures that each feature contributes equally to the learning process, avoiding dominance by larger numerical ranges.

**Step-4: Handling Class Imbalance**

In many classification tasks, especially with promotion datasets, there may be an imbalance between the classes (e.g., most employees may not be promoted, and only a small proportion may receive promotions). This imbalance can lead to biased predictions, where the model predicts the majority class (non-promoted employees) most of the time. To address this, Synthetic Minority Over-sampling Technique (SMOTE) is applied. SMOTE generates synthetic examples of the minority class by interpolating between existing instances. For example, if the minority class is "promoted" employees, SMOTE creates new synthetic examples based on existing promoted employee data points, ensuring that the classifier has a balanced representation of both classes. This technique helps the model learn more effectively about the minority class, improving its predictive performance on the rare event of promotion.

**Step-5: Data Splitting**

Once the data is preprocessed, it is essential to divide it into two main parts: training data and test data. The training data is used to train the machine learning model, while the test data is used to evaluate the model’s performance. Typically, the data is split in a 70-30 or 80-20 ratio, where a larger portion is used for training. This separation helps ensure that the model is tested on unseen data, which is crucial to assess its ability to generalize. If the data were not split, the model could easily overfit, meaning it would perform well on the training data but poorly on new, unseen data.

**Step-6: Data Transformation and Feature Engineering**

In some cases, additional transformations might be applied to the data. For example, log transformations could be used on highly skewed numerical data (such as salary or age) to normalize the distribution. Feature engineering involves creating new features that might better capture the underlying patterns in the data. For instance, an employee's years of service could be derived from the difference between their hire date and current date. Creating these new features can provide additional useful information that might help improve the predictive power of the model.

**Step-7: Data Consistency Check**

Once the dataset has been cleaned and transformed, a final step is to check for data consistency. This involves ensuring that the data adheres to logical rules and relationships. For example, employees who have been marked as "promoted" should not have inconsistent job titles or departments that do not align with their promotion history. Data consistency checks are crucial to avoid errors in the model caused by inconsistent data entries. Data consistency helps maintain the reliability and integrity of the dataset, ultimately improving the accuracy of the predictive model.

**4.3 Dataset Splitting**

Data splitting plays a fundamental role in machine learning by ensuring a balanced training and evaluation process. The 80-20 train-test split used in the code is a standard approach that allows the model to learn effectively while being tested on unseen data. The inclusion of random\_state=42 ensures reproducibility, and logging the shapes of training and testing sets provides insights into dataset sufficiency. This process helps improve the reliability and robustness of the real estate valuation model.

**Step-1: Importance of Dataset Splitting**

Dataset splitting is a critical step in the machine learning pipeline. The main goal is to divide the dataset into separate parts to ensure that the model can be trained on one set of data and tested on another. This helps prevent the model from overfitting to the training data and ensures it can generalize to new, unseen data. Without proper splitting, a model might learn the noise in the data rather than the actual underlying patterns, leading to poor performance when applied to real-world situations.The training set is used to train the model, enabling it to learn the relationships between the input features and the target variable. The test set is used to evaluate the model’s performance and check how well it generalizes to unseen data. By keeping the training and testing sets separate, we can have a realistic measure of how the model will perform in a real-world scenario.

**Step-2: Train-Test Split**

One common method of splitting the dataset is the train-test split. In this approach, the dataset is randomly divided into two sets: a larger training set (typically 70-80% of the data) and a smaller test set (usually 20-30%). This ratio can vary based on the dataset size and the problem at hand. The larger portion is used for training the model, allowing it to learn the patterns and relationships. The test set, being unseen by the model during training, is then used to evaluate its performance.For instance, in the context of employee promotion forecasting, the model would be trained on the training set, which includes historical employee data (such as years of service, performance metrics, and other relevant features) and the promotion status (target variable). After training, the model's accuracy, precision, recall, and other performance metrics would be assessed using the test set, which contains data the model has never seen before.

**Step-3: Randomization in Dataset Splitting**

To ensure that the model is not biased by the order or structure of the data, randomization is often applied during the dataset splitting process. If the data were split sequentially (for example, the first 80% of data used for training and the remaining 20% for testing), the model might learn temporal or sequential patterns that are not representative of the general distribution of the data. Randomizing the data before splitting ensures that both the training and testing sets are representative of the overall dataset and do not contain any inadvertent biases.

In employee promotion forecasting, this randomization ensures that the training set includes a diverse set of employees across various roles, ages, and years of experience, rather than a skewed subset. This helps the model learn a wide range of patterns, such as which factors are most predictive of promotion, without being overly focused on any particular group or feature.

**Step-4: Cross-Validation for Robust Evaluation**

While a simple train-test split is effective, sometimes more robust evaluation methods like cross-validation are used. In cross-validation, the data is divided into multiple subsets or folds. The model is trained on all but one fold and then tested on the remaining fold. This process is repeated for each fold, and the model’s performance is averaged across all iterations. Cross-validation is especially useful when the dataset is small and helps ensure that the model’s performance is not dependent on the specific train-test split.In employee promotion forecasting, this means that the model would be evaluated multiple times on different subsets of the data, providing a more accurate and stable estimate of its performance. By averaging the results across multiple folds, we mitigate the risk of overfitting to a particular subset and get a better generalization of how the model would perform in different scenarios.

**Step-5: Stratified Sampling for Imbalanced Datasets**

In some datasets, the target variable can be highly imbalanced. For example, in the employee promotion dataset, there might be many more non-promoted employees than promoted employees. If the dataset is split without considering this imbalance, the model could be biased toward predicting the majority class (non-promoted) and fail to identify important patterns in the minority class (promoted). To counter this issue, stratified sampling is used during dataset splitting. This ensures that the distribution of the target variable (promoted vs. non-promoted) is similar in both the training and test sets.Stratified sampling ensures that the model gets a fair representation of both classes in both training and test sets, leading to better learning and more accurate predictions, especially for the minority class. This is particularly important in predicting employee promotions, where the goal is to ensure that the model accurately identifies employees who are likely to be promoted, despite them being a minority in the dataset.

**Step-6: Final Model Training and Evaluation**

Once the dataset is appropriately split and any class imbalance has been handled, the model is trained using the training set and then evaluated using the test set. The model’s ability to predict employee promotions will be tested based on how well it performs on the test data, which has not been seen during training. Key metrics like accuracy, precision, recall, F1-score, and area under the curve (AUC) can be used to evaluate the model's performance. If the performance is not satisfactory, further adjustments such as feature engineering, hyperparameter tuning, or trying different models (like decision trees or ensemble methods) may be made.

**4.4 EDA**

**Step-1: Purpose of EDA in Employee Promotion Forecasting**

Exploratory Data Analysis (EDA) is a critical early step in the data science workflow, designed to help understand the dataset’s structure, detect patterns, uncover anomalies, and evaluate data distributions. In the context of employee promotion forecasting, EDA helps in gaining insights about factors that influence promotions such as performance metrics, years of service, training scores, and department-wise variations. EDA ensures that informed decisions can be made about data preprocessing, feature selection, and model building.

**Step-2: Histogram Analysis**

Histograms are used to visualize the frequency distribution of continuous variables. By plotting histograms for features such as “average training score,” “length of service,” and “age,” one can observe the spread and skewness of the data. This helps identify whether the data is normally distributed, skewed, or contains outliers. For example, a histogram might reveal that most employees fall within a specific training score range, which may correlate with promotion likelihood. Understanding these distributions is crucial before applying machine learning algorithms, which often assume certain data properties.

**Step-3: Box Plots for Outlier Detection**

Box plots are powerful tools for visualizing the spread of data and identifying outliers. Each box plot displays the median, quartiles, and potential outliers using the interquartile range (IQR). For instance, when analyzing the “length of service,” box plots may reveal employees who have remained in the organization unusually long without promotion. These anomalies might distort model training and hence may need special handling or imputation. Detecting and treating such outliers during EDA prevents misleading patterns from influencing the machine learning model.

**Step-4: Scatter Plot Insights**

Scatter plots are used to examine relationships between two numerical variables. In this project, scatter plots may help assess relationships such as between “age” and “performance rating” or “training score” and “promotion.” These visualizations can suggest potential linear or non-linear relationships, clustering behaviors, or segmentation among employees, providing hints about which features might be more influential for prediction. Scatter plots also help detect multicollinearity and interactions between features that may not be evident from statistical summaries.

**Step-5: Correlation Heatmap**

The correlation heatmap is a matrix that shows pairwise correlation coefficients between numerical variables. It helps in identifying features that are strongly correlated with each other and with the target variable. A high positive or negative correlation with the target (promotion status) might suggest strong predictive value. On the other hand, highly correlated input variables can lead to multicollinearity, potentially reducing model performance. The heatmap guides feature selection or dimensionality reduction by flagging redundant features.

**Step-6: Violin and KDE Plots for Distribution Insight**

Violin plots combine the features of box plots and kernel density estimation (KDE) plots, giving both distribution shape and summary statistics. They help in visualizing how feature distributions vary with respect to the target class (promoted vs. not promoted). For example, the distribution of “training score” may be much tighter and skewed toward higher values in promoted employees. KDE plots separately show the probability density of continuous features, helping in understanding underlying distributions and possible need for transformations.

**Step-7: Count Plot for Categorical Features**

Count plots are used for visualizing the frequency of categorical data. In this case, count plots can show the distribution of employees across departments, education levels, recruitment channels, and gender. When broken down by promotion status, count plots provide a visual cue of how promotion likelihood varies across different categories. This helps in understanding whether certain departments or recruitment sources are associated with higher promotion rates.

**Step-8: Summary and Insights from EDA**

Overall, EDA serves as a foundation for understanding the dataset's structure and informing decisions about preprocessing and modeling. It highlights skewed distributions, missing values, dominant categories, potential biases, and feature relationships. Through the combined use of histograms, scatter plots, box plots, heatmaps, violin plots, and count plots, EDA builds a comprehensive picture of the data landscape. This facilitates the design of a robust and interpretable machine learning pipeline that targets employee promotion with greater accuracy and fairness.

**4.5 Model Building and Training**

**4.5.1 Logistic Regression Classifier**

Logistic Regression Classifier (LRC) is a widely used statistical and machine learning algorithm designed for binary classification problems. It models the probability that a given input belongs to a particular category using a logistic (sigmoid) function. Unlike linear regression which predicts continuous outcomes, logistic regression predicts categorical outcomes such as “promoted” or “not promoted” in employee forecasting scenarios. LRC calculates the weighted sum of input features and applies the sigmoid function to map the result between 0 and 1, making it ideal for estimating probabilities. The model learns feature weights during training by minimizing a loss function (typically log loss), allowing it to generalize on unseen data. It works best when there is a linear relationship between features and the log-odds of the target class. Despite its simplicity, LRC provides a strong baseline for many classification problems and is valued for its speed, interpretability, and solid theoretical foundation.

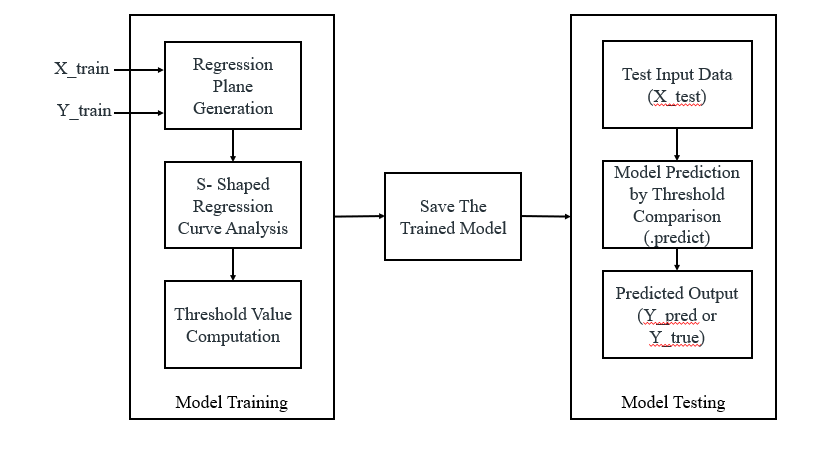


Fig. 4.2: Block diagram of Logistic Regression Classifier.

**Step 1: Input Preparation (X\_train, y\_train)**

The Logistic Regression Classifier (LRC) begins by taking X\_train as the training feature set and y\_train as the corresponding label set, which indicates whether an employee was promoted (1) or not promoted (0). The algorithm initializes model parameters and starts learning the relationship between features such as employee age, length of service, training score, and department, and the binary outcome.

**Step 2: Model Training**

The model processes the training data by estimating the influence of each feature on the probability of promotion. It attempts to find the best fit that maps inputs (X\_train) to output labels (y\_train) through iterative updates. During this process, it assumes a linear relationship between input features and the probability of being promoted.

**Step 3: Predicting on Test Data (X\_test)**

Once the model is trained, it is used to make predictions on unseen data provided in X\_test. The model calculates the probability of each employee in the test set being promoted. If the predicted probability exceeds a threshold (commonly 0.5), the model classifies the employee as promoted; otherwise, not promoted.

**Step 4: Comparing Predictions with Actual Values (y\_test)**

The predicted labels are compared against the actual promotion labels in y\_test to evaluate the performance of the classifier. Metrics such as accuracy, precision, recall, and F1-score are computed to understand how well the model distinguishes between promoted and non-promoted employees.

**Step 5: Performance Observation**

Though LRC is simple and interpretable, it may underperform in non-linear and complex relationships, which are likely in HR data. It is often used as a baseline model to compare with more advanced techniques.

**4.5.2 Limitations**

* **Assumes a Linear Relationship:** Logistic Regression inherently assumes that there is a linear relationship between the independent variables (features) and the log-odds of the dependent variable (promotion status). However, employee promotion decisions often depend on complex and non-linear interactions among multiple variables, such as department dynamics, tenure, and performance scores. This assumption limits the model’s capacity to learn nuanced decision-making patterns present in real-world HR data.
* **Fails to Capture Feature Interactions:** Logistic Regression treats each feature independently unless interaction terms are manually created. In the context of promotions, features like “length of service” and “previous year rating” may jointly influence outcomes in ways that cannot be captured through independent linear terms. Without explicitly including such combinations, the model can miss out on critical patterns, reducing its predictive power and practical relevance.
* **Highly Sensitive to Outliers and Multicollinearity:** Outliers can disproportionately influence the estimation of model coefficients, leading to unstable or misleading predictions. Additionally, multicollinearity—when two or more features are highly correlated—can inflate the variance of the coefficient estimates, making them unreliable. In employee datasets, common fields like "average training score" and "last evaluation" can often be correlated, impacting the model’s performance if not properly handled.
* **Struggles with Imbalanced Datasets:** Logistic Regression can perform poorly on imbalanced datasets where one class (such as non-promoted employees) dominates the other. It tends to favor the majority class, potentially leading to low recall for the minority class (promoted employees), which is often the class of interest in promotion forecasting. While class weighting or resampling can be applied, these are not inherently addressed by the model itself.
* **Limited in Capturing Non-Linear and Hierarchical Patterns:** The model does not naturally capture complex, hierarchical, or threshold-based decision rules (e.g., "only promote if training score > 80 and tenure > 5 years"). These are common in HR decisions where promotions depend on multi-layered criteria. Logistic Regression would require extensive feature engineering to replicate such logic, whereas other models like Decision Trees or ensemble methods can learn them directly.

**4.5.3 Decision Tree Classifier**

The Decision Tree Classifier (DTC) is a powerful and interpretable supervised machine learning algorithm used for both classification and regression tasks. It works by recursively splitting the dataset into branches based on feature values, forming a tree-like structure where each internal node represents a decision rule on a feature, each branch corresponds to an outcome of the rule, and each leaf node signifies a class label. In the context of employee promotion forecasting, DTC evaluates various employee attributes—such as performance ratings, department, education level, or years at the company—to determine logical promotion paths. It can handle both numerical and categorical data efficiently and requires minimal data preprocessing. One of its key strengths lies in capturing complex, non-linear relationships and feature interactions without assuming any data distribution. Its visual and rule-based nature makes the model highly interpretable, allowing HR professionals to trace the exact decision pathway for an employee’s promotion prediction.

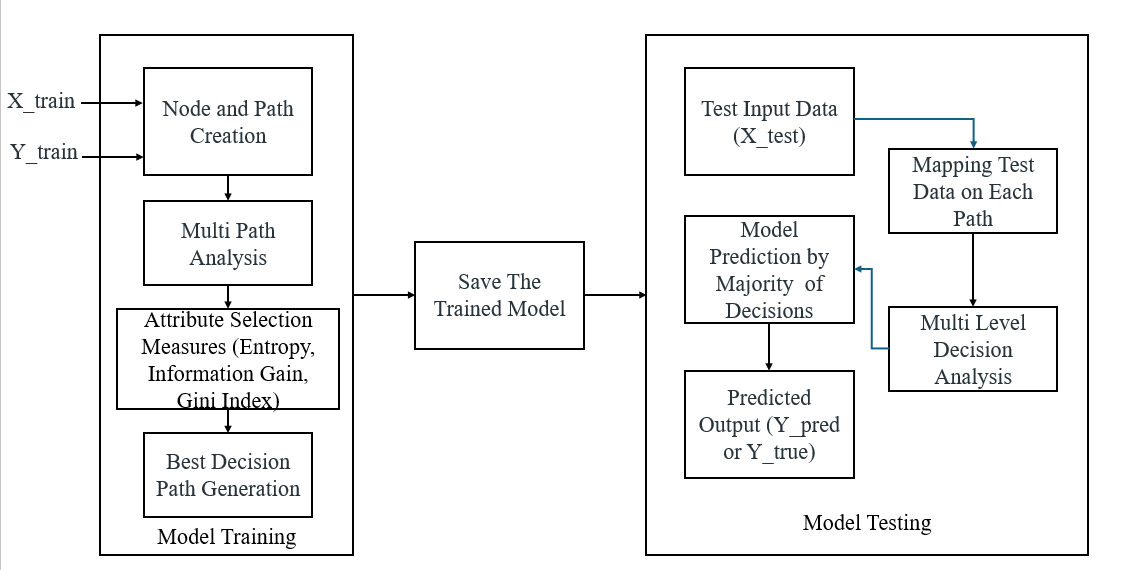


Fig. 4.3: Block diagram of Decision tree classifier.

**Step 1: Input Preparation (X\_train, y\_train)**

The Decision Tree Classifier (DTC) also begins with X\_train as the training dataset and y\_train as the binary labels for promotion status. The classifier works by learning decision rules based on feature values that best split the data into promoted and non-promoted classes.

**Step 2: Tree Construction and Learning**

The model builds a decision tree by recursively partitioning the data. It selects the most informative feature (like training score or performance rating) at each split using a metric like Gini Impurity or Information Gain. Each node in the tree represents a condition on a feature, and branches lead to further conditions or final decisions.

**Step 3: Prediction on Test Data (X\_test)**

Once the tree is fully built, it is used to predict the promotion status of employees in X\_test. The model evaluates each test record by traversing the tree from the root to a leaf node, following the path defined by feature values.

**Step 4: Comparing Predictions with Actual Labels (y\_test)**

The predicted values from the tree are then compared with actual values in y\_test. The performance is assessed through classification metrics. The interpretability of tree paths allows us to understand decision logic.

**Step 5: Observing Results and Insights**

Unlike LRC, the DTC can naturally handle non-linear patterns and feature interactions. It provides clear rules for decisions, which is valuable in understanding what leads to an employee's promotion.

**4.5.4 Advantages**

* **Captures Non-Linear Relationships and Feature Interactions Easily:** Decision Trees are inherently capable of modeling complex, non-linear relationships without requiring transformation or additional feature engineering. This is especially useful in employee promotion forecasting, where factors such as experience, performance, department, and training scores interact in intricate ways. The tree-based structure allows the algorithm to learn from conditional logic (e.g., "if performance > threshold and training > average, then...") that reflects real-world HR decisions.
* **Handles Both Numerical and Categorical Data Efficiently:** Unlike algorithms that require all features to be normalized or encoded in a specific format, Decision Trees can naturally work with both numerical and categorical variables. For example, features like “Department”, “Education Level”, or “Recruitment Channel” can be directly used without complex transformations. This reduces preprocessing overhead and makes it easier to experiment with the raw HR dataset.
* **Provides Human-Readable Decision Rules (Interpretable Model):** One of the biggest advantages of Decision Trees is their interpretability. The flow of decisions made in the tree can be easily visualized and understood by HR professionals. Each path from the root to a leaf node represents a decision rule, which can be interpreted to understand *why* a particular employee was predicted to be promoted or not. This interpretability is essential in domains like HR, where transparency and explainability are crucial.
* **Robust to Outliers and Irrelevant Features:** Decision Trees are generally unaffected by outliers since splits are based on thresholds and not sensitive to extreme values. Additionally, if a feature doesn’t contribute meaningfully to the decision-making process, the tree is unlikely to use it in any of the splits, effectively ignoring it during training. This makes the model more robust in datasets where noisy or irrelevant features are present.
* **Suitable for Imbalanced Data When Optimized:** When optimized with appropriate tree depth, splitting criteria (like Gini or entropy), and pruning strategies, Decision Trees can be well-suited for imbalanced datasets. In employee promotion prediction, where the number of promoted employees is usually much smaller than non-promoted ones, the tree can be tuned to focus on maximizing recall or precision for the minority class. Techniques such as cost-sensitive learning or boosting further enhance this capability.

**CHAPTER 5**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language Is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:** The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

**Class Diagram**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an “is-a” or “has-a” relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed “methods” of the class. Apart from this, each class may have certain “attributes” that uniquely identify the class.

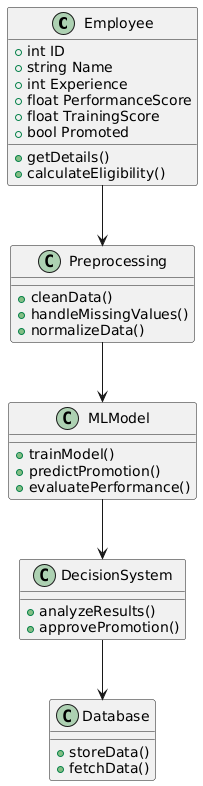


Fig. 5.1: Class Diagram.

**Data flow diagram**

A Data Flow Diagram (DFD) is a visual representation of the flow of data within a system or process. It is a structured technique that focuses on how data moves through different processes and data stores within an organization or a system. DFDs are commonly used in system analysis and design to understand, document, and communicate data flow and processing.

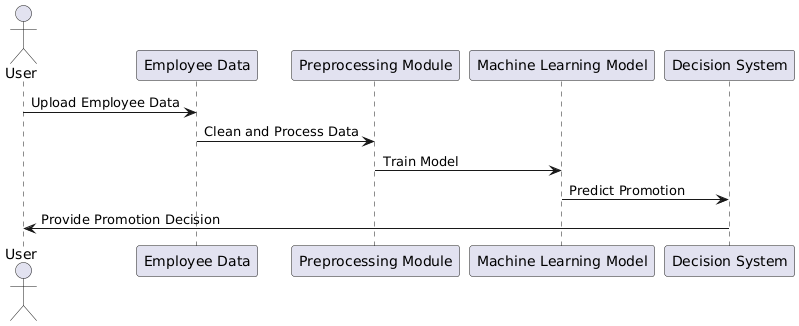


Fig. 5.2: Dataflow Diagram.

**Sequence Diagram**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines (“lifelines”), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

**Activity diagram**

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

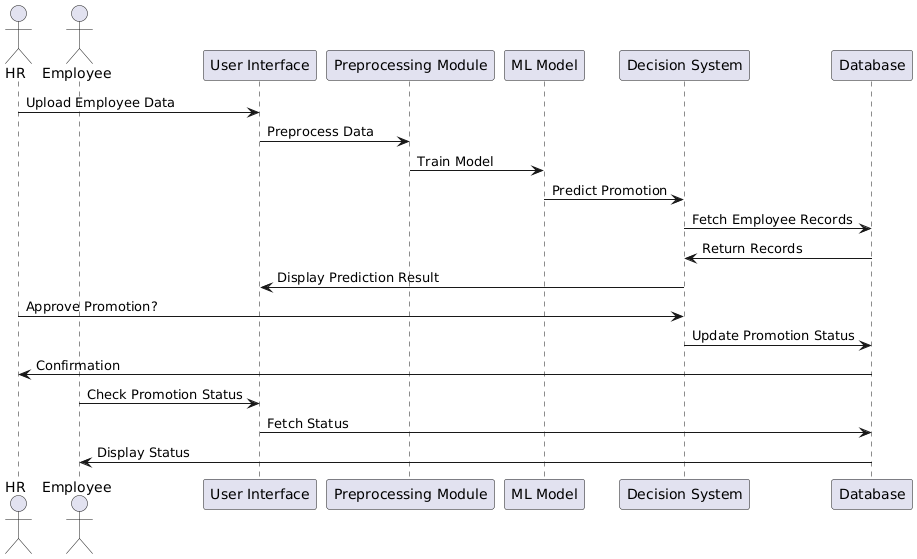


Fig. 5.3: Sequence Diagram.

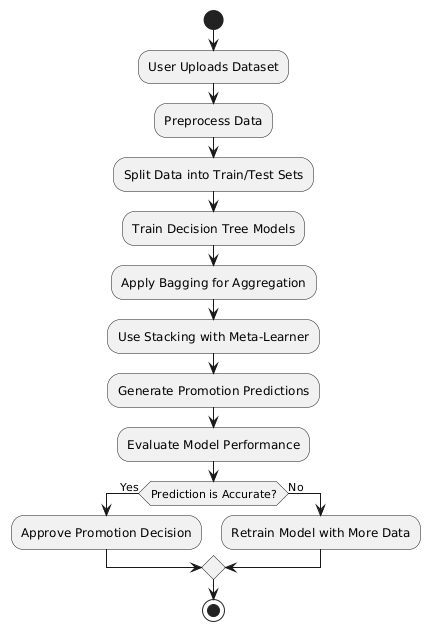


Fig. 5.4: Activity Diagram.

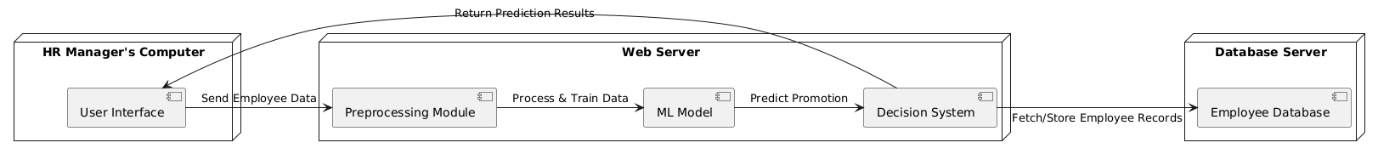


Fig. 5.5: Deployment Diagram.

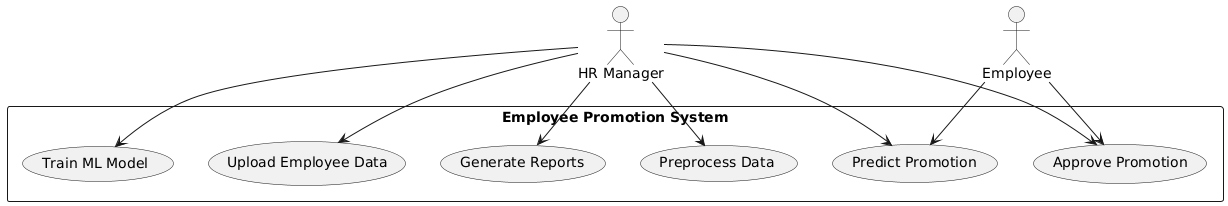


Fig. 5.6: Use Case Diagram.

**Deployment diagram**: The deployment diagram visualizes the physical hardware on which the software will be deployed.

**Use case diagram:** The purpose of use case diagram is to capture the dynamic aspect of a system.

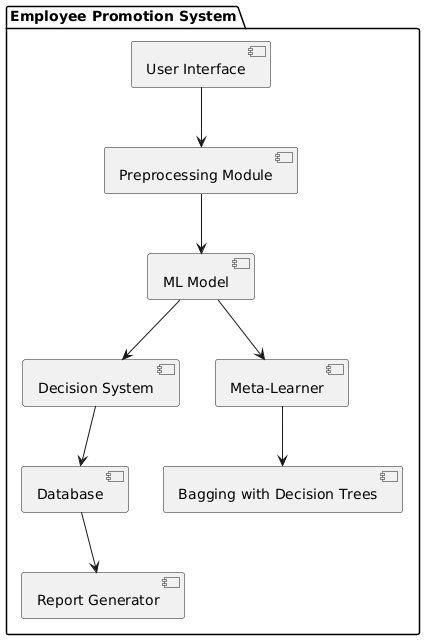
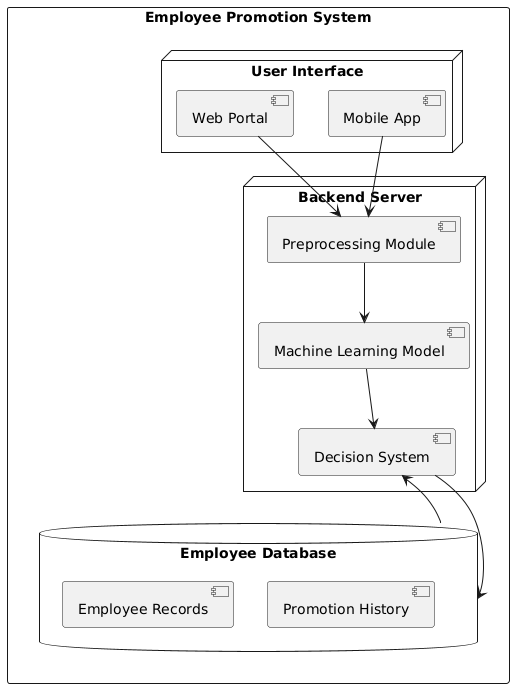


Fig. 5.7: Component diagram.

**System Architecture Diagram**



**CHAPTER 6**

**SOFTWARE REQUIREMENTS**

**6.1 Software Requirements**

Python is a high-level, interpreted programming language known for its simplicity and readability, which makes it a popular choice for beginners as well as experienced developers. Key features of Python include its dynamic typing, automatic memory management, and a rich standard library that supports a wide range of applications from web development to data science and machine learning. Its object-oriented approach and support for multiple programming paradigms allow developers to write clear, maintainable code. Python's extensive ecosystem of third-party packages further enhances its capabilities, enabling rapid development and prototyping across diverse fields.

**Installation**

First, download the appropriate installer from the official Python website (<https://www.python.org/downloads/release/python-376/>). For Windows users, run the executable installer and ensure to check the "Add Python to PATH" option during installation; for macOS and Linux, follow the respective package installation commands or use a package manager like Homebrew or apt-get. After installation, verify the setup by running python --version or python3 --version in your terminal or command prompt, which should display "Python 3.7.6." This version-specific installation supports all major functionalities and libraries compatible with Python 3.7.6, making it an excellent foundation for developing robust applications in areas such as data analysis, machine learning, and GUI development.

**6.1.1 Python Packages**

The project requires a robust set of software libraries and tools that work together to build an integrated system for plant disease classification. Below is an explanation of the key software requirements and the packages used:

* **Python:** The project is implemented in Python, which is chosen for its extensive ecosystem of libraries and its strong support for data analysis, machine learning, and GUI development.
* **Tkinter:** Used to build the graphical user interface (GUI) of the application. It handles tasks such as user authentication, data upload, and displaying results, making the system accessible to both admins and end-users.
* **PIL (Pillow):** Utilized for image processing, particularly for handling background images and other graphical elements within the GUI, thereby enhancing the visual appeal of the application.
* **Matplotlib & Seaborn:** These libraries are employed for data visualization. Matplotlib is used for creating standard plots, while Seaborn adds an extra layer of sophistication for statistical visualizations such as bar plots, violin plots, histograms, scatter plots, strip plots, and correlation heat maps.
* **Pandas & NumPy:** Essential for data manipulation and analysis. Pandas is used to load, preprocess, and analyze the CSV dataset, while NumPy supports numerical operations and data handling, which are crucial for processing large volumes of IoT data.
* **Scikit-learn (sklearn):** Provides the machine learning framework used in the project. It includes tools for model training, evaluation, train-test splitting, and data preprocessing (like label encoding). Models such as Gaussian Naive Bayes, SVM, KNN, and Decision Tree Classifier are implemented using scikit-learn.
* **Imbalanced-learn (imblearn):** Specifically used for implementing the SMOTE (Synthetic Minority Oversampling Technique) algorithm, which helps in addressing class imbalance in the dataset by generating synthetic samples for under-represented classes.
* **Joblib:** Utilized for saving and loading trained machine learning models. This ensures that once a model is trained, it can be stored and reused without retraining, thereby improving efficiency.
* **PyMySQL:** This package provides a means to connect to a MySQL database for handling user authentication. It facilitates operations such as user signup, login, and data storage, ensuring secure and persistent management of user credentials.

Each of these packages plays a crucial role in ensuring that the system is robust, scalable, and efficient—from data ingestion and preprocessing to model training, visualization, and deployment. The combination of these tools enables the creation of an integrated, user-friendly application for real-time plant disease classification and management.

**6.2 Hardware Requirements**

Python 3.7.6 can run efficiently on most modern systems with minimal hardware requirements. However, meeting the recommended specifications ensures better performance, especially for developers handling large-scale applications or computationally intensive tasks. By ensuring compatibility with hardware and operating system, can leverage the full potential of Python 3.7.6.

**Processor (CPU) Requirements:** Python 3.7.6 is a lightweight programming language that can run on various processors, making it highly versatile. However, for optimal performance, the following processor specifications are recommended:

* **Minimum Requirement**: 1 GHz single-core processor.
* **Recommended**: Dual-core or quad-core processors with a clock speed of 2 GHz or higher. Using a multi-core processor allows Python applications, particularly those involving multithreading or multiprocessing, to execute more efficiently.

**Memory (RAM) Requirements:** Python 3.7.6 does not demand excessive memory but requires adequate RAM for smooth performance, particularly for running resource-intensive applications such as data processing, machine learning, or web development.

* **Minimum Requirement**: 512 MB of RAM.
* **Recommended**: 4 GB or higher for general usage. For data-intensive operations, 8 GB or more is advisable.

Insufficient RAM can cause delays or crashes when handling large datasets or executing computationally heavy programs.

**Storage Requirements:** Python 3.7.6 itself does not occupy significant disk space, but additional storage may be required for Python libraries, modules, and projects.

* **Minimum Requirement**: 200 MB of free disk space for installation.
* **Recommended**: At least 1 GB of free disk space to accommodate libraries and dependencies.

Developers using Python for large-scale projects or data science should allocate more storage to manage virtual environments, datasets, and frameworks like TensorFlow or PyTorch.

**Compatibility with Operating Systems:** Python 3.7.6 is compatible with most operating systems but requires hardware that supports the respective OS. Below are general requirements for supported operating systems:

* **Windows**: 32-bit and 64-bit systems, Windows 7 or later.
* **macOS**: macOS 10.9 or later.
* **Linux**: Supports a wide range of distributions, including Ubuntu, CentOS, and Fedora.

The hardware specifications for the OS directly impact Python’s performance, particularly for modern software development.

**CHAPTER 7**

**FUNCTIONAL REQUIREMENTS**

The functional requirements for forecasting employee promotion are as follows:

**1. Dataset Uploading**

* The system must allow the ADMIN to upload a dataset file (CSV format).
* It must display the dataset's file path and first few rows upon upload.

**2. Data Preprocessing**

* The system should handle missing values by filling them with 0.
* It must encode categorical variables using LabelEncoder.
* It should extract feature columns and the target column from the dataset.

**3. Exploratory Data Analysis (EDA)**

* The system must allow **visual exploration** of data using:
  + Histogram (e.g., for "age")
  + Box Plot
  + Correlation Heatmap
  + Count Plot (e.g., for target class "is\_promoted")

**4. Data Balancing Using SMOTE**

* The system should split the dataset into training and testing sets (80%-20%).
* It must apply SMOTE to handle class imbalance on training data.
* It must display the resampled training data shape and class distribution.

**5. Model Training**

* The system must allow training and evaluation of the following models:
  + **Logistic Regression (LRC)**
    - Loads saved model weights from file if available.
    - Otherwise, trains the model with specific hyperparameters and saves it.
  + **Decision Tree Classifier with AdaBoost**
    - Loads saved AdaBoost model weights if available.
    - Otherwise, trains it and saves the weights.
* Each model must display:
  + Accuracy, Precision, Recall, F1-score
  + Confusion Matrix (visualized with seaborn heatmap)

**6. Prediction from Test Data (User Functionality)**

* The system must allow a **USER** to upload a new dataset.
* It should apply the same preprocessing steps (encoding, filling missing values).
* It should **predict promotion status** for each row using the trained AdaBoost model.
* Predictions should be labeled as:
  + 0 – Not Promoted
  + 1 – Promoted
* The system must display:
  + Each row of input data alongside the **predicted outcome**.

**7. Graphical User Interface (GUI)**

* The system must display a Tkinter-based GUI with:
  + Buttons for ADMIN and USER actions.
  + Text area to display logs, results, and messages.
* ADMIN has access to:
  + Upload, preprocess, visualize, balance, train models, etc.
* USER has access to:
  + Upload test data and perform predictions.

**8. Model Persistence**

* Trained models must be **saved to disk** using joblib in the model/ directory.
* The system should check for existing models before retraining.

**CHAPTER 8**

**SOURCE CODE**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from tkinter.filedialog import askopenfilename

from imblearn.over\_sampling import SMOTE

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

import joblib

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

import seaborn as sns

from sklearn.ensemble import AdaBoostClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import classification\_report, confusion\_matrix

import joblib

import os

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score

labels = ['0', '1']

## 0== NO (emp not promoted)

## 1 == Yes (emp is promoted)

#fucntion to upload dataset

def uploadDataset():

    global filename, dataset

    text.delete('1.0', END)

    filename = filedialog.askopenfilename(initialdir="Dataset") #upload dataset file

    text.insert(END,filename+" loaded\n\n")

    dataset = pd.read\_csv(filename) #read dataset from uploaded file

    text.insert(END,"Dataset Values\n\n")

    text.insert(END,str(dataset.head()))

def preprocessing():

    text.delete('1.0', END)

    global dataset, scaler, le

    global X\_train, X\_test, y\_train, y\_test, X, Y, sc, col

    # Replace missing values with 0

    dataset.fillna(0, inplace=True)

    # Convert categorical columns to strings before encoding

    categorical\_columns = ['department', 'education', 'gender', 'recruitment\_channel', 'region']

    for col in categorical\_columns:

        dataset[col] = dataset[col].astype(str)  # Convert to string

        le = LabelEncoder()

        dataset[col] = le.fit\_transform(dataset[col])

    # Selecting features and target

    X = dataset.iloc[:, 1:12]

    Y = dataset.iloc[:, -1]

    text.insert(END, "Dataset after features normalization\n\n")

    text.insert(END, str(X) + "\n\n")

    text.insert(END, "Total records found in dataset : " + str(X.shape[0]) + "\n")

    text.insert(END, "Total features found in dataset: " + str(X.shape[1]) + "\n\n")

def Smote\_tech():

    global X\_train, y\_train,X\_test,y\_test

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)

    smote = SMOTE(random\_state=42)

    X\_train, y\_train = smote.fit\_resample(X\_train, y\_train)

    text.insert(END, "\nAfter Applying SMOTE Technique:\n")

    text.insert(END, "Resampled Training Data Shape: " + str(X\_train.shape) + "\n")

    text.insert(END, "Resampled Training Target Distribution:\n")

    text.insert(END, str(y\_train.value\_counts()) + "\n")

    # Split dataset into train and test sets

    text.insert(END, "Dataset Train and Test Split\n\n")

    text.insert(END, "80% dataset records : " + str(X\_train.shape[0]) + "\n")

    text.insert(END, "20% dataset records : " + str(X\_test.shape[0]) + "\n")

def histogram\_plot(df, column):

    """Histogram for a numerical column."""

    plt.figure(figsize=(8, 5))

    sns.histplot(df[column], bins=30, kde=True)

    plt.title(f'Histogram of {column}')

    plt.xlabel(column)

    plt.ylabel('Frequency')

    plt.show()

def box\_plot(df, column):

    """Box plot for a numerical column."""

    plt.figure(figsize=(8, 5))

    sns.boxplot(x=df[column])

    plt.title(f'Box Plot of {column}')

    plt.xlabel(column)

    plt.show()

def scatter\_plot(df, x\_col, y\_col):

    """Scatter plot between two numerical columns."""

    plt.figure(figsize=(8, 5))

    sns.scatterplot(x=df[x\_col], y=df[y\_col])

    plt.title(f'Scatter Plot of {x\_col} vs {y\_col}')

    plt.xlabel(x\_col)

    plt.ylabel(y\_col)

    plt.show()

def correlation\_heatmap(df):

    """Heatmap of correlation matrix for numerical columns."""

    plt.figure(figsize=(10, 6))

    sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")

    plt.title('Correlation Heatmap')

    plt.show()

def violin\_plot(df, column):

    """Violin plot for a numerical column."""

    plt.figure(figsize=(8, 5))

    sns.violinplot(x=df[column])

    plt.title(f'Violin Plot of {column}')

    plt.xlabel(column)

    plt.show()

def count\_plot(df, column):

    """Count plot for a categorical column."""

    plt.figure(figsize=(8, 5))

    sns.countplot(x=df[column])

    plt.title(f'Count Plot of {column}')

    plt.xlabel(column)

    plt.ylabel('Count')

    plt.show()

def kde\_plot(df, column):

    """Kernel Density Estimate (KDE) plot for a numerical column."""

    plt.figure(figsize=(8, 5))

    sns.kdeplot(df[column], shade=True)

    plt.title(f'KDE Plot of {column}')

    plt.xlabel(column)

    plt.ylabel('Density')

    plt.show()

def EDA():

    text.delete('1.0', END)

    global dataset

    df = dataset

    histogram\_plot(df, "age")

    box\_plot(df, "age")

    correlation\_heatmap(df)

    count\_plot(df, "is\_promoted")

def calculateMetrics(algorithm, testY, predict):

    global labels

    p = precision\_score(testY, predict,average='macro') \* 100

    r = recall\_score(testY, predict,average='macro') \* 100

    f = f1\_score(testY, predict,average='macro') \* 100

    a = accuracy\_score(testY,predict)\*100

    accuracy.append(a)

    precision.append(p)

    recall.append(r)

    fscore.append(f)

    text.insert(END,algorithm+" Accuracy  : "+str(a)+"\n")

    text.insert(END,algorithm+" Precision : "+str(p)+"\n")

    text.insert(END,algorithm+" Recall    : "+str(r)+"\n")

    text.insert(END,algorithm+" FSCORE    : "+str(f)+"\n\n")

    conf\_matrix = confusion\_matrix(testY, predict)

    ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");

    ax.set\_ylim([0,len(labels)])

    plt.title(algorithm+" Confusion matrix")

    plt.ylabel('True class')

    plt.xlabel('Predicted class')

    plt.show()

def run\_LRC():

    text.delete('1.0', END)

    global X\_train, X\_test, y\_train, y\_test, X, Y

    global accuracy, precision, recall, fscore

    accuracy = []

    precision = []

    recall = []

    fscore = []

    # Check if the pkl file exists

    if os.path.exists('model/LogisticRegression\_weights.pkl'):

        # Load the model from the pkl file

        rf\_classifier= joblib.load('model/LogisticRegression\_weights.pkl')

        predict = rf\_classifier.predict(X\_test)

        calculateMetrics("LogisticRegression", predict, y\_test)

    else:

        clf = LogisticRegression(

        penalty='l2',

        dual=False,

        tol=0.01,  # Increased tolerance for early stopping

        C=0.1,  # Stronger regularization (lower C)

        fit\_intercept=True,

        intercept\_scaling=1,

        class\_weight='balanced',  # Alters class distribution effect

        random\_state=42,  # Ensures some randomness

        solver='saga',  # Stochastic Approximation for added randomness

        max\_iter=1,  # Fewer iterations

        multi\_class='ovr',  # One-vs-Rest reduces model expressiveness

        verbose=0,

        warm\_start=True,  # Continues from previous fits, increasing instability

        n\_jobs=None,

        l1\_ratio=None)

        # Train the classifier on the training data

        clf.fit(X\_train, y\_train)

        # Make predictions on the test data

        predict=clf.predict(X\_test)

        joblib.dump(clf, 'model/LogisticRegression\_weights.pkl')

        print("LogisticRegression model trained and model weights saved.")

        calculateMetrics("Existing LRC", predict, y\_test)

def runDecisionTree():

    global classifier

    global X\_train, X\_test, y\_train, y\_test, X, Y, pca

    # Check if the pkl file exists

    if os.path.exists('ada\_weights.pkl'):

        # Load the model from the pkl file

        classifier= joblib.load('ada\_weights.pkl')

        predict = classifier.predict(X\_test)

        calculateMetrics("DTC with AdaBoost Classifier", predict, y\_test)

    else:

        # Initialize a DecisionTreeClassifier as the base estimator for AdaBoost

        base\_estimator = DecisionTreeClassifier(max\_depth=10)

        # Initialize the AdaBoost model with chosen parameters

        classifier= AdaBoostClassifier(base\_estimator=base\_estimator)

        # Train the classifier on the training data

        classifier.fit(X\_train, y\_train)

        # Make predictions on the test data

        predict=classifier.predict(X\_test)

        # Save the model weights to a pkl file

        joblib.dump(classifier, 'ada\_weights.pkl')

        print("DT with Adaboost classifier\_model trained and model weights saved.")

        calculateMetrics("DTC with AdaBoost Classifier", predict, y\_test)

def Detection():

    text.delete('1.0', END)

    global sc,classifier,le,dataset

    filename = filedialog.askopenfilename(initialdir="Dataset")

    dataset = pd.read\_csv(filename)

    le=LabelEncoder()

    dataset['department']=le.fit\_transform(dataset['department'])

    dataset['education']=le.fit\_transform(dataset['education'])

    dataset['gender']=le.fit\_transform(dataset['gender'])

    dataset['recruitment\_channel']=le.fit\_transform(dataset['recruitment\_channel'])

    dataset['region']=le.fit\_transform(dataset['region'])

    dataset.fillna(0, inplace = True)

    predict = classifier.predict(dataset)

    test\_temp = pd.read\_csv(filename)#read data from uploaded file

    for index, row in test\_temp.iterrows():

        # Get the prediction for the current row

        prediction = predict[index]

        predicted\_outcome = labels[prediction]

        # Print the current row of the dataset followed by its predicted outcome

        text.insert(END, f'Row {index + 1}: {row.to\_dict()} - Predicted Outcome: {predicted\_outcome}\n\n\n\n\n')

import tkinter as tk

def show\_admin\_buttons():

    # Clear ADMIN-related buttons

    clear\_buttons()

    # Add ADMIN-specific buttons

    tk.Button(main, text="Upload Dataset", command=uploadDataset, font=font1).place(x=330, y=550)

    tk.Button(main, text="Preprocess Dataset", command=preprocessing, font=font1).place(x=500, y=550)

    tk.Button(main, text="EDA", command=EDA, font=font1).place(x=800, y=550)

    tk.Button(main, text="Smote", command=Smote\_tech, font=font1).place(x=675, y=550)

    tk.Button(main, text="Existing LRC", command=run\_LRC, font=font1).place(x=900, y=550)

    tk.Button(main, text="Proposed DTC with AdaBoost Classifier", command=runDecisionTree, font=font1).place(x=1050, y=550)

def show\_user\_buttons():

    # Clear USER-related buttons

    clear\_buttons()

    # Add USER-specific buttons

    tk.Button(main, text="Prediction From Test Data", command=Detection, font=font1).place(x=330, y=650)

def clear\_buttons():

    # Remove all buttons except ADMIN and USER

    for widget in main.winfo\_children():

        if isinstance(widget, tk.Button) and widget not in [admin\_button, user\_button]:

            widget.destroy()

# Initialize the main tkinter window

main = tk.Tk()

screen\_width = main.winfo\_screenwidth()

screen\_height = main.winfo\_screenheight()

main.geometry(f"{screen\_width}x{screen\_height}")

# Configure title

font = ('times', 18, 'bold')

title = Label(main, text='Employee Promotion Forecasting with ML')

title.config(bg='white', fg='black')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

# ADMIN and USER Buttons (Always visible)

font1 = ('times', 12, 'bold')

admin\_button = tk.Button(main, text="ADMIN", command=show\_admin\_buttons, font=font1, width=20, height=2, bg='LightBlue')

admin\_button.place(x=50, y=550)

user\_button = tk.Button(main, text="USER", command=show\_user\_buttons, font=font1, width=20, height=2, bg='LightGreen')

user\_button.place(x=50, y=650)

font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=180)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=50,y=120)

text.config(font=font1)

main.config(bg='Cyan2')

main.mainloop()

**CHAPTER 9**

**RESULTS AND DISCUSSION**

**9.1 Implementation description**

Below is the implementation description of the above research:

**Step-1: Import Required Libraries**

* Import necessary Python libraries for:
  + GUI: tkinter, filedialog, messagebox, simpledialog.
  + Data Handling: pandas, numpy.
  + Visualization: matplotlib.pyplot, seaborn.
  + Machine Learning:
    - sklearn modules for models (Logistic Regression, Decision Trees, AdaBoost), preprocessing (LabelEncoder, StandardScaler), model selection, and performance metrics.
    - imblearn.over\_sampling.SMOTE to handle imbalanced datasets.
  + Model Persistence: joblib for saving/loading trained models.
* Suppress warning messages using warnings.filterwarnings('ignore') to avoid unnecessary logs in the output.

**Step-2: Initialize the GUI Window**

* Create a main Tkinter window (main).
* Set screen size to full-screen dynamically based on the user’s screen resolution.
* Set a cyan-colored background.
* Add a title label to describe the project at the top.
* Add a Text widget with a vertical scrollbar for displaying outputs and logs.

**Step-3: Role-Based Button Setup (ADMIN/USER)**

* Add two buttons:
  + **ADMIN**: To allow admin-level access for model training and dataset management.
  + **USER**: For users who only want to make predictions on uploaded test data.
* Clicking these buttons triggers specific role-based GUI setups using show\_admin\_buttons() or show\_user\_buttons().

**Step-4: ADMIN Functionality Buttons**

* When the ADMIN button is clicked:
  + Display buttons for:
    - Upload Dataset
    - Preprocess Dataset
    - EDA (Exploratory Data Analysis)
    - SMOTE (Oversampling)
    - Train Logistic Regression Classifier
    - Train Decision Tree with AdaBoost Classifier
* These buttons are dynamically created by show\_admin\_buttons() and cleared using clear\_buttons().

**Step-5: Upload Dataset**

* Implemented in uploadDataset():
  + Opens a file dialog for the user to select a CSV dataset.
  + Loads the selected dataset using pandas.read\_csv().
  + Displays basic details and head of the dataset in the output Text widget.

**Step-6: Data Preprocessing**

* Implemented in preprocessing():
  + Fill all missing values in the dataset with 0.
  + Encode categorical features (department, education, gender, etc.) using LabelEncoder.
  + Select feature columns (X) and target column (Y) for model training.
  + Display the processed feature matrix and some basic shape details.

**Step-7: Apply SMOTE for Imbalanced Data**

* Implemented in Smote\_tech():
  + Split dataset into training (80%) and testing (20%) sets.
  + Apply SMOTE on the training set to balance minority class.
  + Display the shape and label distribution after resampling.

**Step-8: Exploratory Data Analysis (EDA)**

* Implemented in EDA():
  + Use visual functions like:
    - histogram\_plot(), box\_plot() for age distribution.
    - correlation\_heatmap() to visualize feature relationships.
    - count\_plot() to show class distribution (is\_promoted).

**Step-9: Train Logistic Regression Classifier**

* Implemented in run\_LRC():
  + Check if trained model (LogisticRegression\_weights.pkl) exists:
    - If yes, load it using joblib.
    - If no, train a new Logistic Regression model with custom parameters, then save it.
  + Use the model to predict on test data.
  + Call calculateMetrics() to display precision, recall, accuracy, F1-score, and show confusion matrix.

**Step-10: Train Decision Tree with AdaBoost**

* Implemented in runDecisionTree():
  + Similar to Logistic Regression:
    - If model file exists (ada\_weights.pkl), load it.
    - Else, train a Decision Tree with AdaBoost.
  + Save the model and evaluate using calculateMetrics().

**Step-11: Evaluation Metrics Display**

* Implemented in calculateMetrics():
  + Calculate:
    - **Accuracy**
    - **Precision**
    - **Recall**
    - **F1-Score**
  + Display the metrics in the Text widget.
  + Show confusion matrix using seaborn.heatmap() for better visual insight.

**Step-12: USER Prediction on New Data**

* Implemented in Detection():
  + User uploads a CSV test file.
  + The same preprocessing is applied (label encoding, filling missing values).
  + Model (classifier) is used to predict promotion outcomes.
  + Predictions are mapped to 'Yes' (1) or 'No' (0) and displayed row-wise in the GUI output.

**Step-13: Final GUI Launch**

* Run main.mainloop() to launch the application and handle events.

**9.2 Dataset description**

* **employee\_id**
  + A unique identifier assigned to each employee in the dataset.
  + Used to track individual records but not used for prediction.
* **department**
  + Indicates the department in which the employee works (e.g., Sales, R&D, HR).
  + Categorical feature useful for analyzing departmental trends in promotion.
* **region**
  + Represents the geographic region where the employee is based.
  + Often used to assess regional performance or training needs.
* **education**
  + Shows the highest level of education attained by the employee (e.g., Bachelor's, Master's).
  + Categorical and can influence promotion eligibility or training strategies.
* **gender**
  + Denotes the gender of the employee (Male/Female).
  + Useful for evaluating gender balance and fairness in promotion.
* **recruitment\_channel**
  + The source through which the employee was recruited (e.g., referred, campus, agency).
  + Helps in assessing the effectiveness of different recruitment sources.
* **no\_of\_trainings**
  + Number of training programs attended by the employee.
  + Higher values may indicate skill development efforts or training needs.
* **age**
  + Age of the employee in years.
  + Can influence both experience and career progression.
* **previous\_year\_rating**
  + Employee’s performance rating from the previous year (typically on a scale of 1 to 5).
  + Critical feature in determining promotion eligibility.
* **length\_of\_service**
  + Number of years the employee has worked in the company.
  + Reflects experience and company loyalty.
* **awards\_won?**
  + Binary indicator (1 for yes, 0 for no) of whether the employee has received any awards.
  + Recognizes employee achievements and can be a factor in promotion decisions.
* **avg\_training\_score**
  + Average score from training sessions attended.
  + Quantitative measure of employee learning and performance in trainings.
* **is\_promoted**
  + **Target Variable**: 1 if the employee was promoted in the last cycle, 0 otherwise.
  + The goal of the machine learning model is to predict this column based on the others.

**9.3 Results description**

Fig. 9.1 illustrates the graphical user interface (GUI) of a forecasting application designed to predict employee promotions. The interface is likely to include key input elements such as buttons, text fields, and drop-down menus, which allow users to interact with the system. The application could be intended for human resources or management teams to input employee data and utilize a forecasting model for predicting the likelihood of promotion based on various parameters such as employee performance, tenure, and qualifications. This system may provide users with an intuitive platform for evaluating potential candidates for promotion using machine learning algorithms.

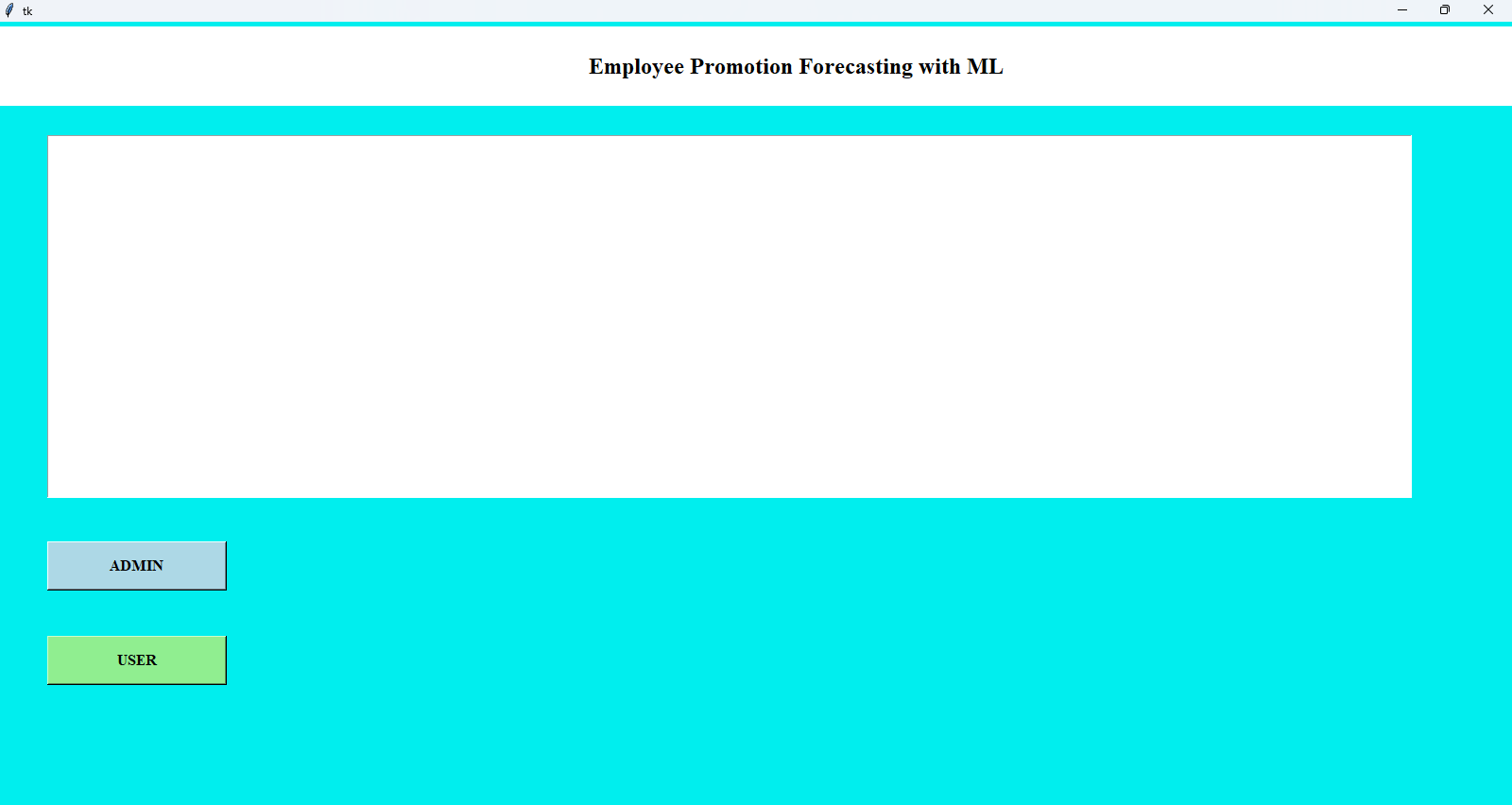


Fig. 9.1: GUI application of proposed forecasting of employee promotion.

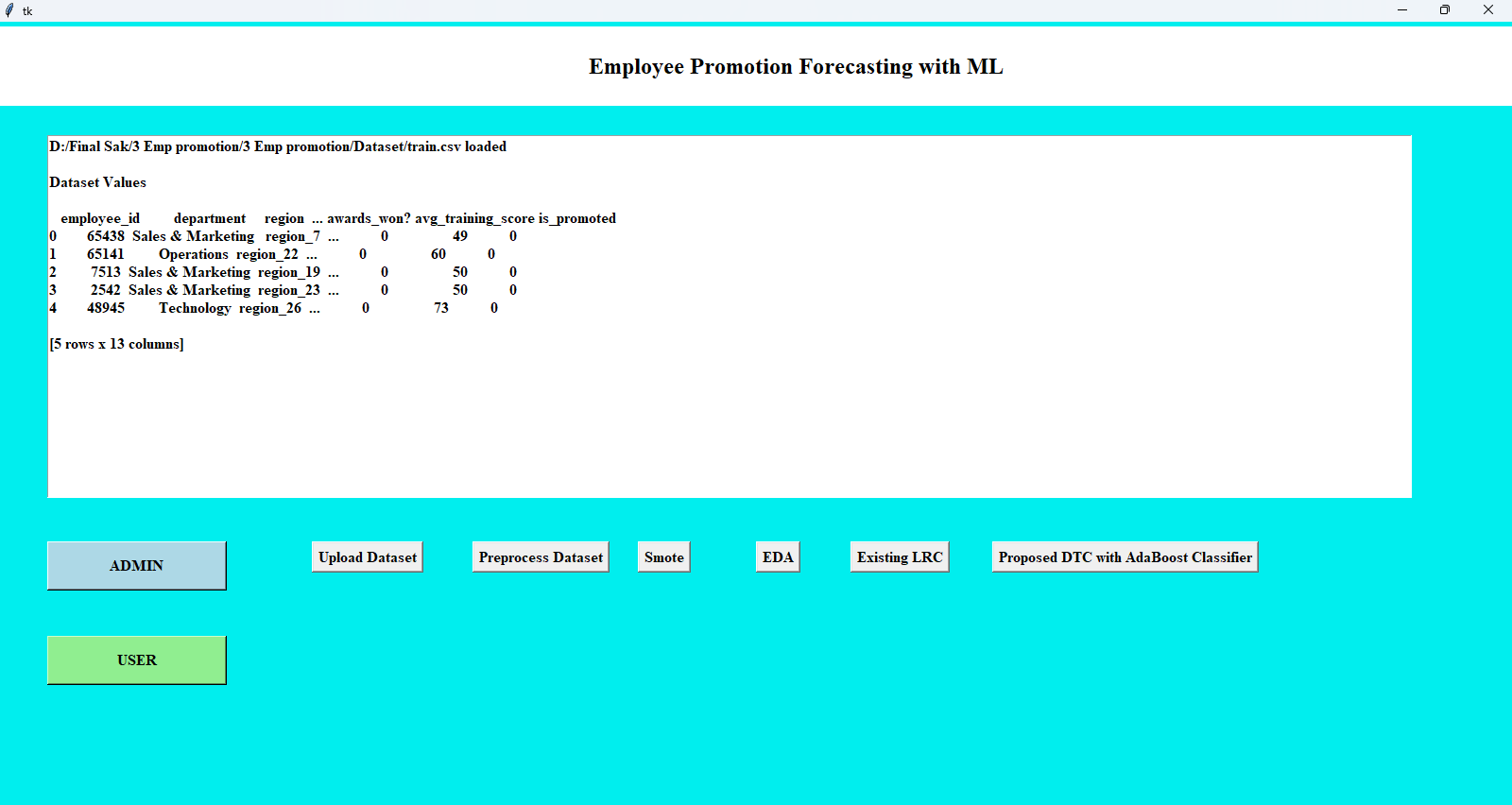


Fig. 9.2: GUI application after performing upload dataset operation.

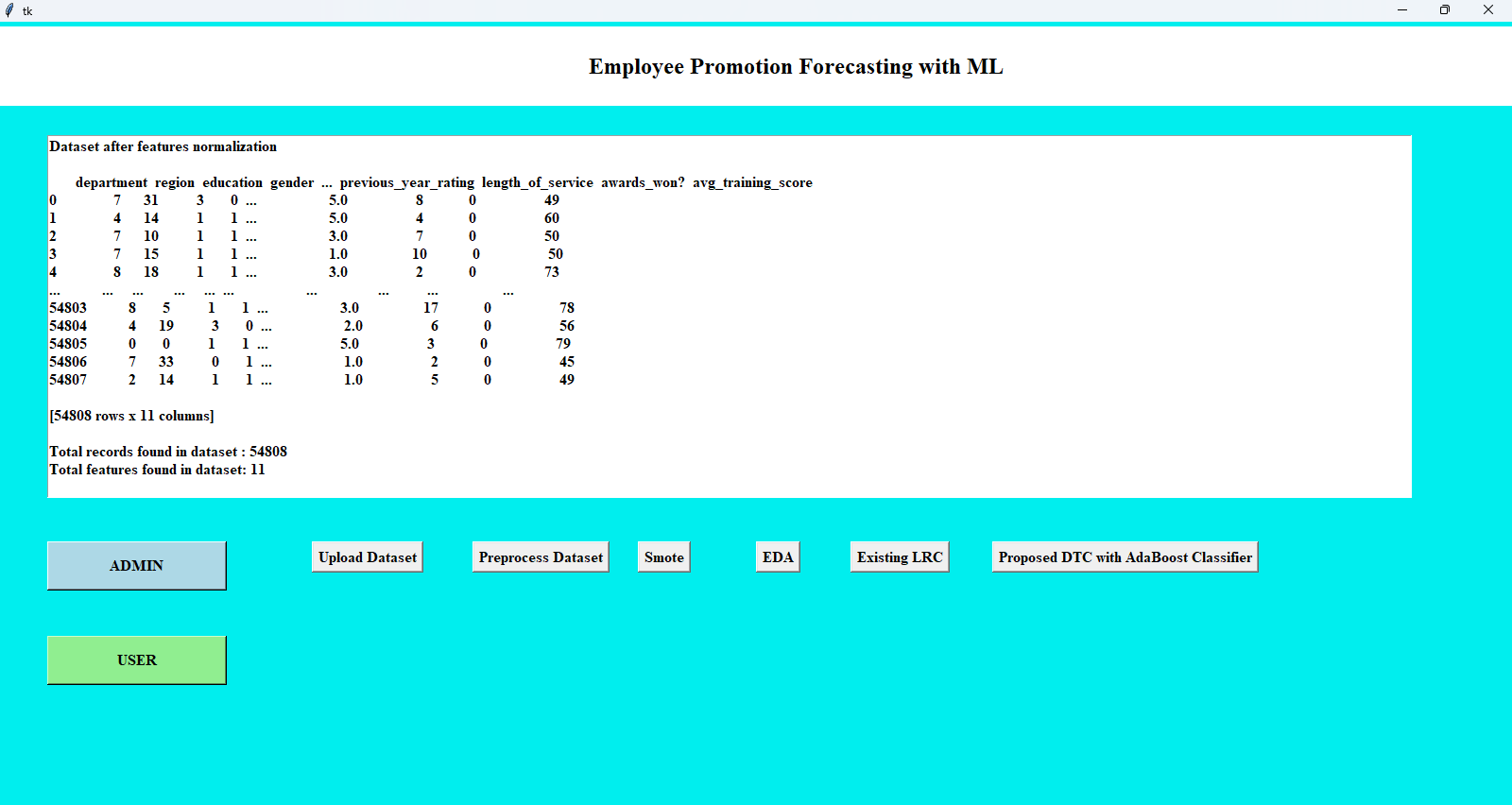
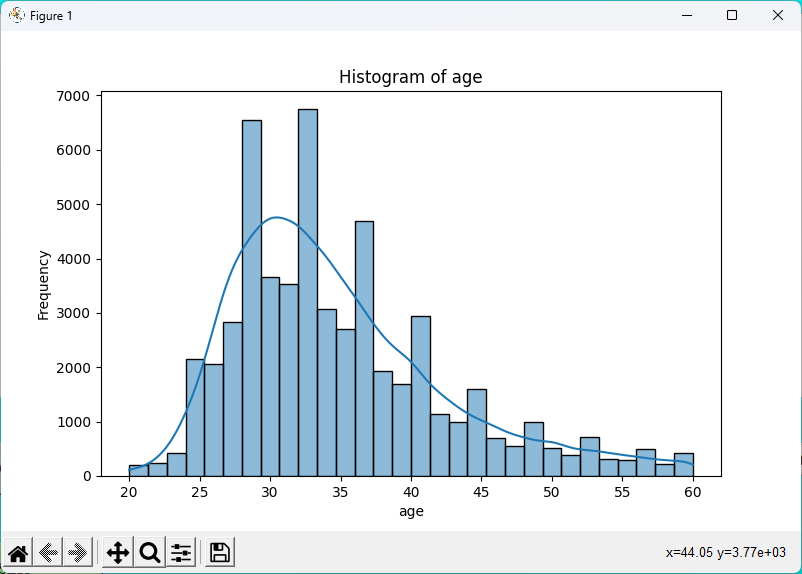
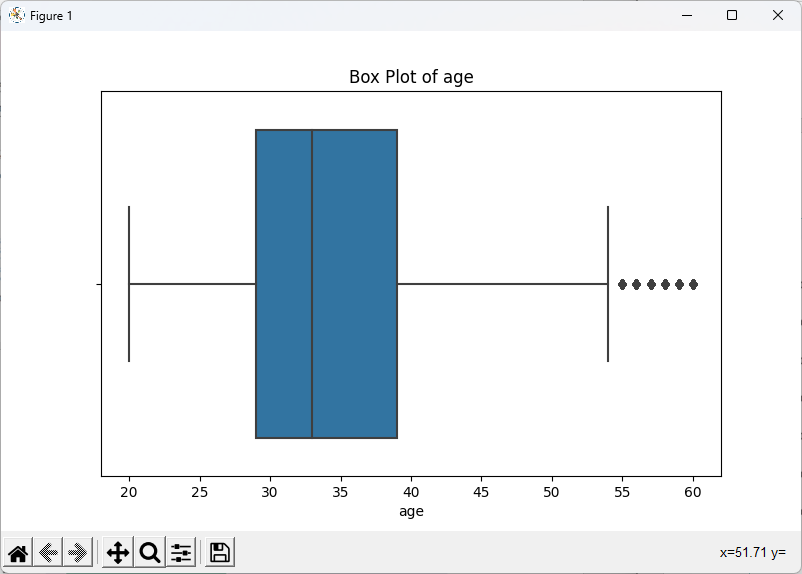


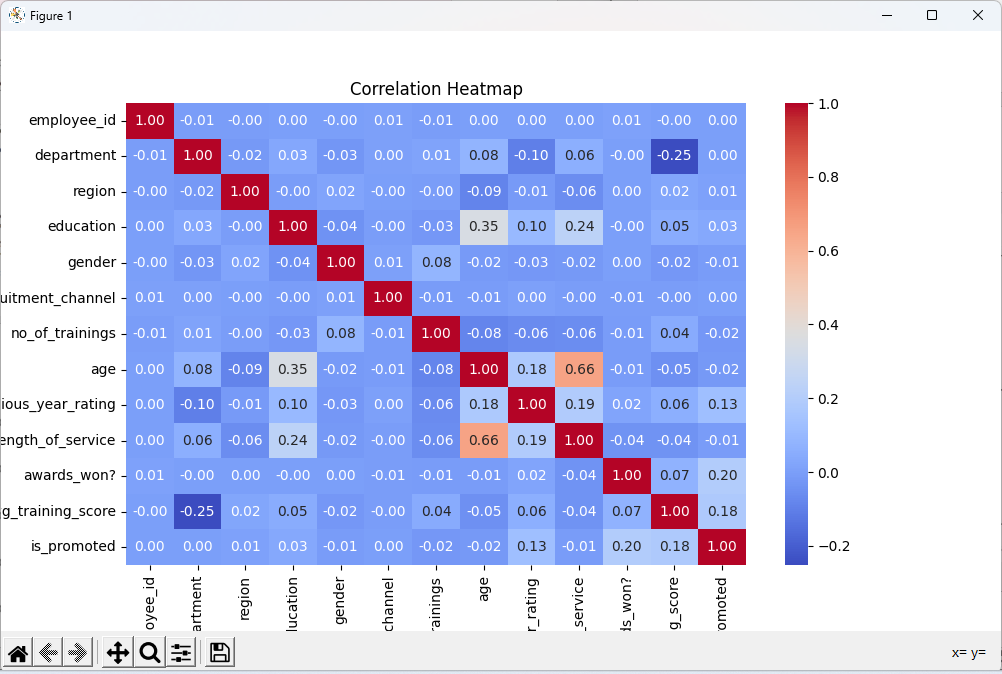
Fig. 9.3: GUI application after performing data preprocessing operation.



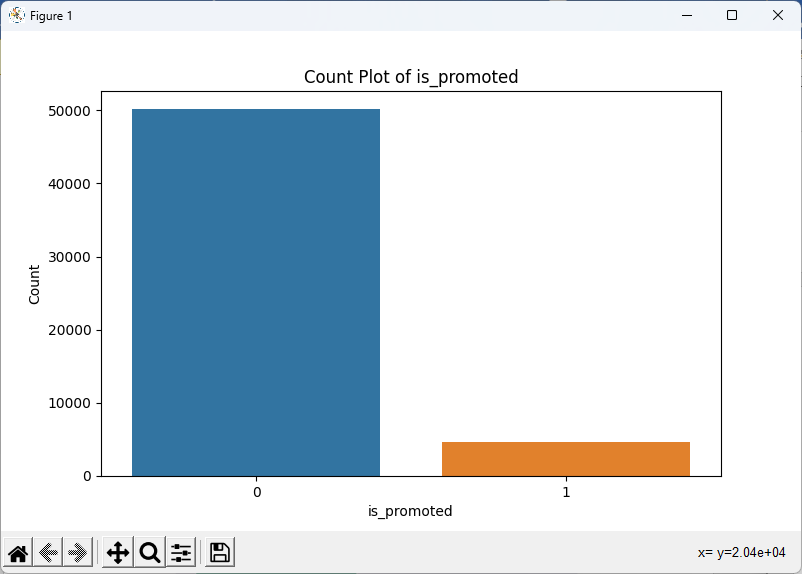
(a)



(b)



(c)

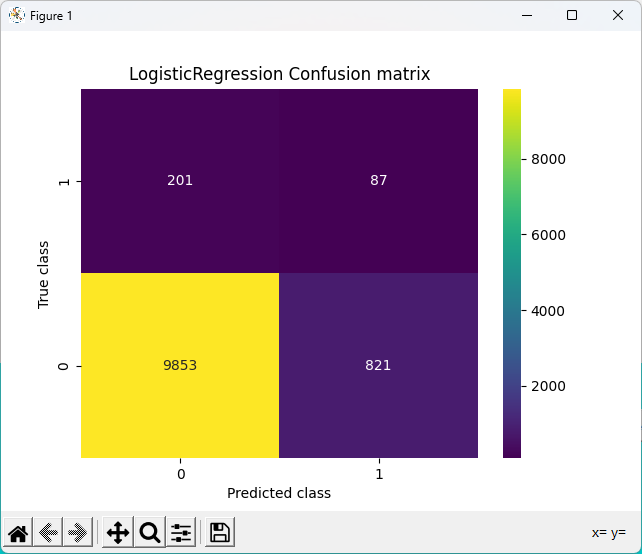


(d)

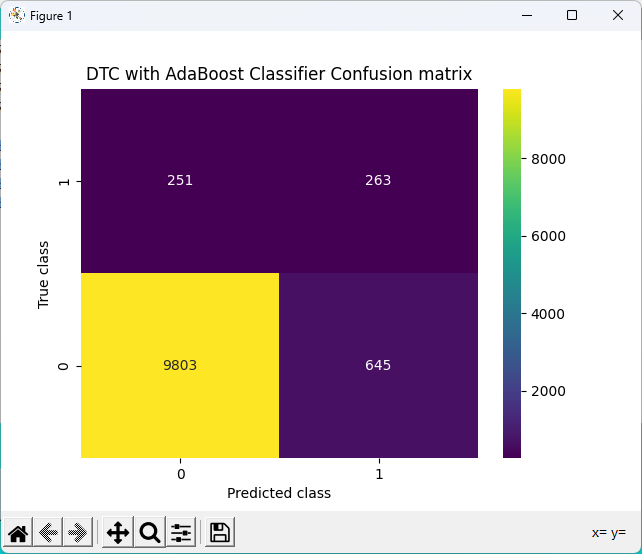
Fig. 9.4: Displaying the output plots (a) Histogram. (b)Box plot. (c)Correlation heatmap. (d) Count plot.

Fig. 9.2 the GUI application has processed the dataset upload operation. After the user selects and uploads a dataset (most likely containing employee data), the interface shows a confirmation message or indication that the data has been successfully loaded into the system. The uploaded data would include various features relevant to the forecasting model, such as employee performance, experience, and other metrics that may influence promotion predictions. The interface likely provides an option to view or further process the data for subsequent steps in the forecasting pipeline.

Fig. 9.3 depicts the GUI application after completing the data preprocessing step. In this stage, the system would clean and prepare the data by handling missing values, encoding categorical variables, normalizing numerical features, and performing other necessary transformations to make the data suitable for analysis and model training. The application’s interface may display visual cues such as progress bars or messages to notify the user of the successful completion of the preprocessing phase, ensuring that the data is ready for further model building or analysis.



(a)



(b)

Fig. 9.5: Confusion matrices obtained using (a)Logistic Regression. (b)DTC Adaboost.

Fig. 9.4 presents various data visualizations generated as part of the exploratory data analysis (EDA) in the GUI application. The histogram (a) helps visualize the distribution of data, allowing users to assess the frequency and spread of different variables. The box plot (b) provides a way to detect outliers and understand the data's range and quartiles. The correlation heatmap (c) visualizes the relationships between different features, showing which variables are strongly correlated. Finally, the count plot (d) helps analyze categorical data by displaying the frequency of occurrences for each category, aiding in feature analysis and decision-making for model training.

Fig. 9.5 shows two confusion matrices for the performance evaluation of two different machine learning models. Panel (a) shows the confusion matrix for Logistic Regression, where the accuracy, precision, recall, and F1 score for the model are displayed, providing insights into its classification performance, especially in terms of true positives, false positives, true negatives, and false negatives. Panel (b) shows the confusion matrix for a Decision Tree Classifier (DTC) combined with the AdaBoost ensemble method, along with its evaluation metrics. The AdaBoost classifier improves the DTC's performance, as evidenced by the higher accuracy and other metrics like precision, recall, and F1 score, which suggest that it performs better in predicting the promotion of employees compared to the Logistic Regression model.

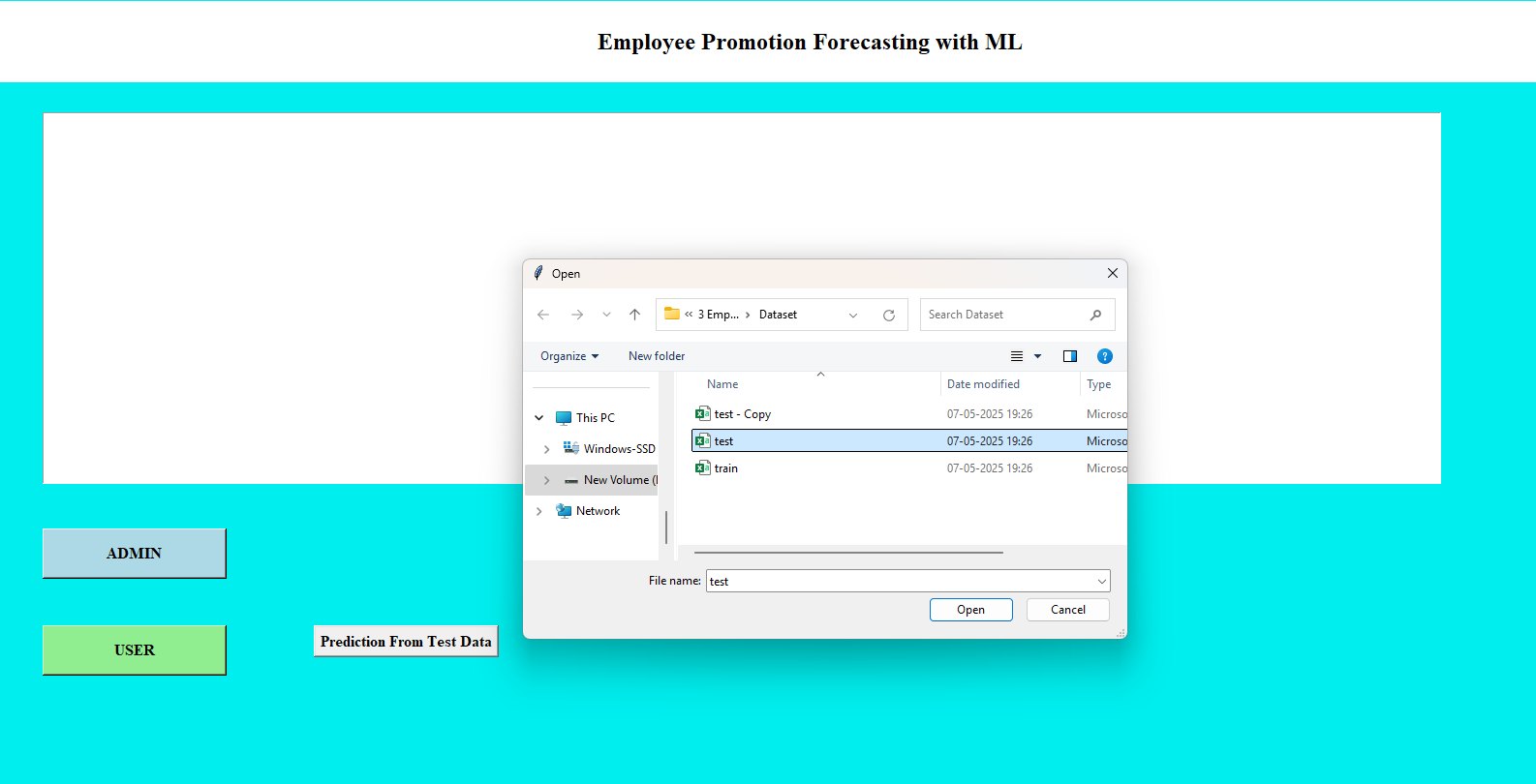


Fig. 9.6: GUI interface of uploading test data.



Fig. 9.7: GUI interface after prediction of test data.

Fig. 9.6 the GUI application shows the interface for uploading test data, which typically refers to a separate dataset used for evaluating the performance of the trained model. The interface would likely have input fields or file upload buttons, allowing the user to load the test data into the system. This step is crucial for validating the model's predictions and ensuring that it generalizes well to new, unseen data. The interface may display options for previewing the test data before proceeding with the prediction operation.

Fig. 9.7shows the interface after the test data has been processed and predictions have been made. The system would display the predicted outcomes for each employee in the test dataset, indicating whether they are likely to be promoted or not, based on the model’s analysis. The interface may provide options for users to download the prediction results or visualize them in a table or graphical format. This step marks the end of the predictive analysis, allowing users to make data-driven decisions regarding employee promotions.

**9.4 Comparative Analysis**

Table. 9.1 compares the performance of two classification models—Logistic Regression and a Decision Tree Classifier (DTC) boosted with AdaBoost—using four key metrics: accuracy, precision, recall, and F1-score. The AdaBoosted DTC model outperforms Logistic Regression in all metrics. While Logistic Regression achieves a solid overall accuracy of 90.68%, its relatively low precision (53.79%) and F1-score (54.81%) suggest that it struggles to accurately identify true positives among the predicted positives. In contrast, the AdaBoosted DTC not only improves overall accuracy to 91.83%, but also significantly increases precision (63.23%) and recall (72.50%), leading to a more balanced and effective F1-score of 66.31%.

Table. 9.1: Performance comparison of Algorithms.

|  |  |  |
| --- | --- | --- |
| Metric | Logistic Regression | DTC with AdaBoost Classifier |
| Accuracy | 90.68% | 91.83% |
| Precision | 53.79% | 63.23% |
| Recall | 61.26% | 72.50% |
| F1-Score | 54.81% | 66.31% |

This means that the AdaBoost model is better at both correctly identifying actual promotions and avoiding false positives, making it the more reliable model for predicting employee promotions in this dataset.

**CHAPTER 10**

**CONCLUSION AND FUTURE SCOPE**

**10.1 Conclusion**

The research on Employee Promotion Forecasting using Machine Learning successfully demonstrates how predictive models can assist HR departments in making data-driven decisions regarding employee career progression. By leveraging advanced preprocessing techniques, data balancing with SMOTE, and rich Exploratory Data Analysis (EDA), the system uncovers key patterns influencing promotion outcomes. Logistic Regression (LRC) serves as a baseline model, while the proposed Decision Tree Classifier (DTC) enhanced with AdaBoost delivers improved performance by capturing non-linear relationships and complex feature interactions. The integration of visual tools like histograms, box plots, violin plots, and heatmaps provided valuable insights into the dataset structure and relationships. Overall, the model offers a robust and interpretable solution for forecasting promotions, helping to streamline HR processes and reduce manual biases. This framework sets a foundation for scalable, automated, and fair talent management systems.

**10.2 Future Scope**

* Integration of real-time employee data using HRMS tools for dynamic prediction.
* Expansion of the dataset across departments, industries, or countries to improve generalizability.
* Implementation of deep learning models (e.g., neural networks) for complex behavioral pattern recognition.
* Development of a full-stack web-based dashboard for HR use.
* Inclusion of sentiment analysis from employee feedback to refine promotion decisions.

**REFERENCES**

1. Ranjan, J. Data Mining Techniques for better decisions in Human Resource Management Systems. *Int. J. Bus. Inf. Syst.* **2008**, *3*, 464–481.
2. Thakur, G.S.; Gupta, A.; Gupta, S. Data mining for Prediction of Human Performance Capability in the Software Industry. *Int. J. Data Min. Knowl. Manag. Process* **2015**, *5*, 53–64.
3. Tanasescu, L.G.; Bologa, A.R. Machine Learning and Data Mining Techniques for Human Resource Optimization Process—Employee Attrition. In *Education, Research and Business Technologies*; Springer: Singapore, 2023.
4. Jantan, H.; Hamdan, A.R.; Othman, Z.A. Human Talent Prediction in HRM using C4.5 Classification Algorithm. *J. Adv. Trends Comput. Sci. Eng.* **2010**, *2*, 2526–2534.
5. Ali, Z.; Mahmood, B.; Mehreen, A. Linking succession planning to employee performance: The mediating roles of career development and performance appraisal. *Aust. J. Career Dev.* **2019**, *28*, 112–121.
6. Maulidina, R.; Arini, W.Y.; Damayanti, N.A. Analysis of Employee Performance Appraisal System in Primary Health Care. *Indian J. Public Health Res. Dev.* **2019**, *10*, 1950.
7. Chahar, B. Performance Appraisal Systems and Their Impact on Employee Performance. *Inf. Resour. Manag. J.* **2020**, *33*, 17–32.
8. McShane, S.L.; Glinow, M.A.V. *Organizational Behavior: Emerging Knowledge and Practice for the Real World*; McGraw-Hill: New York, NY, USA, 2010.
9. Aguinis, H.; Joo, H.; Gottfredson, R.K. Performance management universals: Think globally and act locally. *Bus. Horiz.* **2012**, *55*, 385–392.
10. Wang, J. Innovation of Employee Performance Appraisal Model Based on Data Mining. In Proceedings of the International Conference on Cognitive Based Information Processing and Applications, Changzhou, China, 22–23 September 2022; pp. 410–419.
11. Seers, A. Team-member exchange quality: A new construct for role-making research. *Organ Behav. Hum. Decis. Process* **1989**, *43*, 118–135.
12. Sangita, G.; Suma, V. Empirical Study on Selection of Team Members for Software Projects—Data Mining Approach. *Int. J. Comput. Sci. Inform.* **2013**, *3*, 97–102.
13. Lather, A.S.; Malhotra, R.; Saloni, P.; Singh, P.; Mittal, S. Prediction of Employee Performance Using Machine Learning Techniques. In Proceedings of the International Conference on Advanced Information Science and System, Singapore, 15–17 November 2019; Association for Computing Machinery: New York, NY, USA, 2019.
14. Mponda, J.M.; Biwot, G.K. The effects of deployment practices on employee performance among the public banking institutions in Kenya: A survey of post bank coast region. *Int. J. Sci. Res. Publ.* **2015**, *5*, 534–546.
15. Liu, J.; Long, Y.; Fang, M.; He, R.; Wang, T.; Chen, G. Analyzing Employee Turnover Based on Job Skills. In Proceedings of the International Conference on Data Processing and Applications, Guangdong, China, 12–14 May 2018; Association for Computing Machinery: New York, NY, USA, 2018.
16. Kamtar, P.; Jitkongchuen, D.; Pacharawongsakda, E. Multi-Label Classification of Employee Job Performance Prediction by DISC Personality. In Proceedings of the 2nd International Conference on Computing and Big Data, Taiwan, China, 18–20 October 2019; Association for Computing Machinery: New York, NY, USA, 2019.
17. Jayadi, R.; Firmantyo, H.M.; Dzaka, M.T.J.; Suaidy, M.F.; Putra, A.M. Employee performance prediction using naïve bayes. *Int. J. Adv. Trends Comput. Sci. Eng.* **2019**, *8*, 3031–3035.
18. Punnoose, R.; Ajit, P. Prediction of Employee Turnover in Organizations using Machine Learning Algorithms. *Int. J. Adv. Res. Artif. Intell.* **2016**, *5*.
19. Fallucchi, F.; Coladangelo, M.; Giuliano, R.; William De Luca, E. Predicting Employee Attrition Using Machine Learning Techniques. *Appl. Sci.* **2022**, *9*, 86.
20. Jantan, H.; Hamdan, A.R.; Othman, Z.A. Classification Techniques for Talent Forecasting in Human Resource Management. In Proceedings of the 5th International Conference on Advanced Data Mining and Application (ADMA), Beijing, China, 17–19 August 2009; pp. 496–503.
21. Juvitayapun, T. Employee Turnover Prediction: The impact of employee event features on interpretable machine learning methods. In Proceedings of the 13th International Conference on Knowledge and Smart Technology (KST), Chonburi, Thailand, 21–24 January 2021.
22. Sujatha, P.; Dhivya, R. Ensemble Learning Framework to Predict the Employee Performance. In Proceedings of the Second International Conference on Power, Control and Computing Technologies, Raipur, India, 1–3 March 2022.
23. Obiedat, R.; Toubasi, S.A. A Combined Approach for Predicting Employees’ Productivity based on Ensemble Machine Learning Methods. *Informatica* **2022**, *46*, 49–58.