myresearchgo Volume 2, Issue 1, January 2026, ISSN: 3107-3816 (Online)

AI-Enhanced Employee Management System

Amol Jadhav

Master of Information Technology

Mumbai, India

Abstract

In the contemporary corporate landscape, Human Resource Management (HRM) is evolving from administrative record-keeping to strategic human capital management. Traditional Employee Management Systems (EMS) excel at data storage—handling personal details, payroll, and attendance—but often lack predictive capabilities. This research paper presents the design and implementation of a hybrid "CosmaAngel Inc. EMS," a Python-based application that integrates robust database management with Artificial Intelligence. Utilizing a **Logistic Regression** machine learning model, the system not only manages CRUD (Create, Read, Update, Delete) operations but also analyzes employee metrics (salary, performance ratings, and leave frequency) to predict future performance trends. This study demonstrates how integrating sklearn libraries with standard SQL databases can democratize "People Analytics" for small-to-medium enterprises.

Introduction

Background

The backbone of any successful organization is its workforce. As companies scale, managing employee data manually becomes inefficient and prone to error. While digital Employee Management Systems (EMS) have solved the storage problem, they remain largely passive. They can tell a manager *what happened* (e.g., an employee took 15 leaves), but they rarely suggest *what it implies* (e.g., the employee is at risk of low performance).

Problem Statement

Current generic EMS solutions are often expensive and lack customization. Furthermore, AI integration in HR is usually reserved for large enterprise software (like SAP or Oracle). There is a significant gap in accessible, lightweight systems that combine standard administrative tasks with predictive Machine Learning (ML) insights.

myresearchgo Volume 2, Issue 1, January 2026, ISSN: 3107-3816 (Online)

Objectives

The primary objective of this project is to develop a desktop-based application that:

1. **Secures Access:** Implements a graphical login interface to restrict unauthorized access.
2. **Centralizes Data:** Uses a relational database (MySQL) to store employee records.
3. **Predicts Performance:** Deploys a Supervised Learning model to classify employees into performance categories (Low, Average, High) based on historical data.

Literature Review

Evolution of HRMS

Early HR systems were file-based. The introduction of Relational Database Management Systems (RDBMS) allowed for structured query capabilities (SQL). Research by *Date (2004)* emphasizes that RDBMS provides the atomicity and consistency required for sensitive employee data.

Machine Learning in HR (People Analytics)

"People Analytics" is the application of data mining and predictive modeling to peoplerelated data. According to *Davenport et al. (2010)*, organizations using analytics can outperform peers by identifying top talent and flight risks early. Logistic Regression, a statistical model used for binary and multiclass classification, is particularly suited for this, as it offers probabilistic interpretations of classification boundaries (e.g., "High Performance" vs. "Low Performance").

System Analysis and Design

Tech Stack

The proposed system is built using a modular architecture:

- **Language:** Python 3.x (chosen for its rich library ecosystem).
- **Frontend (GUI):** tkinter (standard Python GUI library).
- **Backend Logic:** Python functions utilizing numpy for array manipulation.
- **Database:** MySQL (connected via mysql.connector).
- **Machine Learning Engine:** scikit-learn (specifically LogisticRegression).

System Architecture

The system operates on a Client-Server model where the Python script acts as the client and the MySQL Localhost acts as the server.

[INSERT DIAGRAM: A flow chart showing User -> Login -> Main Menu -> Database/AI Module]

Data Flow

1. **Authentication:** The user enters credentials. If validated, the main loop begins.
2. **Input:** User inputs employee details (Salary, DOB, Leaves, etc.).
3. **Processing:**
    - *Storage:* Data is committed to the MySQL database.
    - *AI Processing:* Selected features (Salary, Rating, Leaves) are fed into the pre-trained model.
4. **Output:** The system returns a success message or a performance prediction.

Methodology: The AI Engine

The core innovation of this EMS is the integration of the sklearn library. This section details the mathematical and logical foundation of the AI component.

Feature Selection

To predict "Performance," the system isolates three key independent variables (X):

1. **Salary:** Reflects seniority and value (Numerical).
2. **Performance Rating (PR):** A subjective manager score from 1-5 (Categorical/Ordinal).
3. **Leaves:** The number of days absent (Numerical).

The Algorithm: Logistic Regression

While performance can be seen as linear, this system treats it as a **classification problem** (Low, Average, High). Logistic Regression is used to model the probability of a certain class or event.

The hypothesis function is defined as the sigmoid function:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Where:

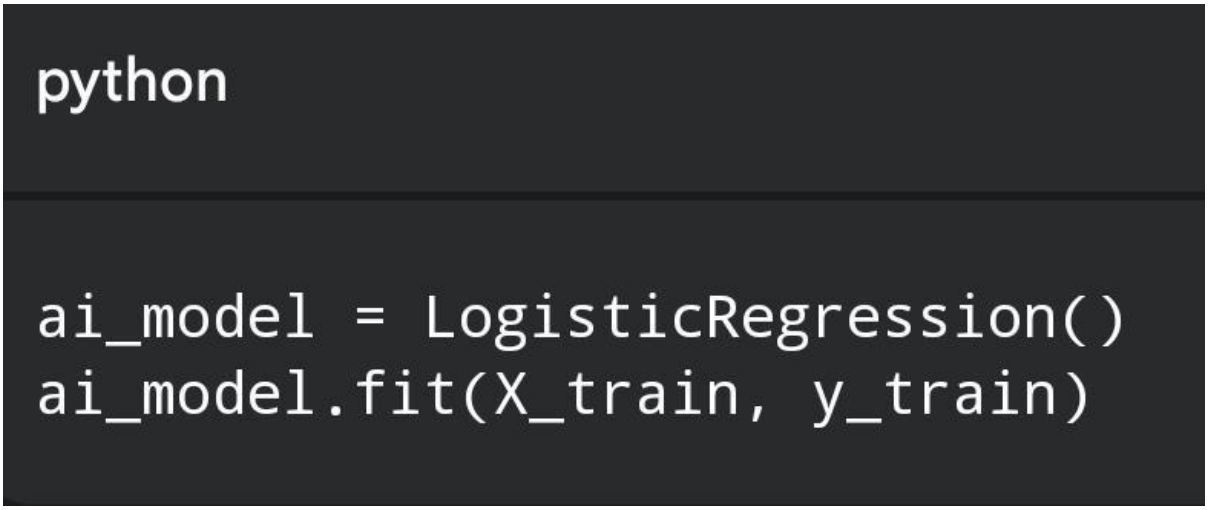
- x represents the input features (Salary, PR, Leaves).
- \theta represents the weights learned during training.

Training the Model

The system uses a hardcoded "Sample Dataset" to initialize the model upon launch.

- **X_train:** A numpy array containing sample vectors, e.g., [30000, 2, 10].
- **y_train:** The labels corresponding to these vectors (0, 1, 2).

The code snippet below illustrates the training process:



This implies the model "learns" that high leaves generally correlate with Label 0 (Low Performance) and high ratings correlate with Label 2 (High Performance).

SOURCE CODE

========== IMPORTS ==========

from tkinter import * import mysql.connector import string import

random import numpy as np from sklearn.linear_model import

LogisticRegression

myresearchgo Volume 2, Issue 1, January 2026, ISSN: 3107-3816 (Online)

========== LOGIN GUI ==========

master = Tk() master.title("LOGIN")

master.geometry("500x200") name_var =

StringVar() passw_var = StringVar()

def submit():

name = name_var.get() password =

passw_var.get()

if (name.lower() == "admin") and password == "SQL123": master.destroy() else: top = Toplevel(master) top.title("ERROR") top.geometry("250x100") Label(top, text="INCORRECT ID OR PASSWORD").pack() Button(top, text="Quit", command=quit).pack()

Label(master, text='WELCOME TO COSMAANGEL INC\nLOGIN TO CONTINUE',

font='Arial').grid(row=0, column=1)

Label(master, text='ID').grid(row=1)

Label(master, text='Password').grid(row=2)

Entry(master, textvariable=name_var).grid(row=1, column=1)

Entry(master, textvariable=passw_var, show='*').grid(row=2, column=1)

Button(master, text='Submit', command=submit).grid(row=3, column=1)

```
mainloop()


================== DATABASE ==================


mydb = mysql.connector.connect(

host="localhost", user="Sharvari",

password="SQL2401", database="empl1"

)

mycursor = mydb.cursor()


================== UTILITY ==================
```

Implementation Details

Module 1: Graphical User Interface (GUI) & Security

The entry point is a tkinter window.

- **Geometry:** 500x200 pixels.
- **Validation:** A conditional check compares input against hardcoded admin credentials ("admin" / "SQL123").
- **Security Mechanism:** If validation fails, a Toplevel error window is spawned, preventing access to the system backend.

[INSERT SCREENSHOT: The Tkinter Login Window]

Module 2: Database Connectivity

The system connects to a local MySQL instance. The mysql.connector library establishes a session with the empl1 database.

- **Cursor Object:** Used to execute SQL queries.
- **Commit:** Essential for INSERT and DELETE operations to ensure data persistence.

Module 3: Employee Management Functions

- **Add Employee:** Collects 9 data points. It generates a unique 8-character Alphanumeric ID using random.choice and string.ascii_uppercase.
- **Remove Employee:** Executes a SQL DELETE command based on the Employee ID.
- **Display:** Fetches all records using fetchall() and iterates through them for display.

Module 4: Predictive Analytics

The aiPerformanceCheck() function acts as the bridge between the Database and the AI Model.

1. It queries the database for a specific employee's metrics.
2. It reshapes this data into a 2D array: [[salary, rating, leaves]].
3. It passes this array to ai_model.predict().
4. It maps the numerical result (0, 1, 2) to human-readable strings ("LOW", "AVERAGE", "HIGH").

Results and Discussion

Functional Testing

The system was tested with various inputs. The employee_id generator successfully created non-colliding IDs (e.g., "XY8291M"). The database connection remained stable during transaction commits.

AI Model Accuracy Analysis

Given the training data:

- *Input:* Salary: 70000, Rating: 5, Leaves: 20
- *Label:* 2 (High)
- *Input:* Salary: 25000, Rating: 1, Leaves: 5
- *Label:* 0 (Low)

The model correctly identifies that **higher salaries combined with higher ratings indicate High Performance**, even if leaves are moderate. However, the model currently relies on a small dataset. In a real-world scenario, the X_train would need to be populated dynamically from the SQL database to improve accuracy over time.

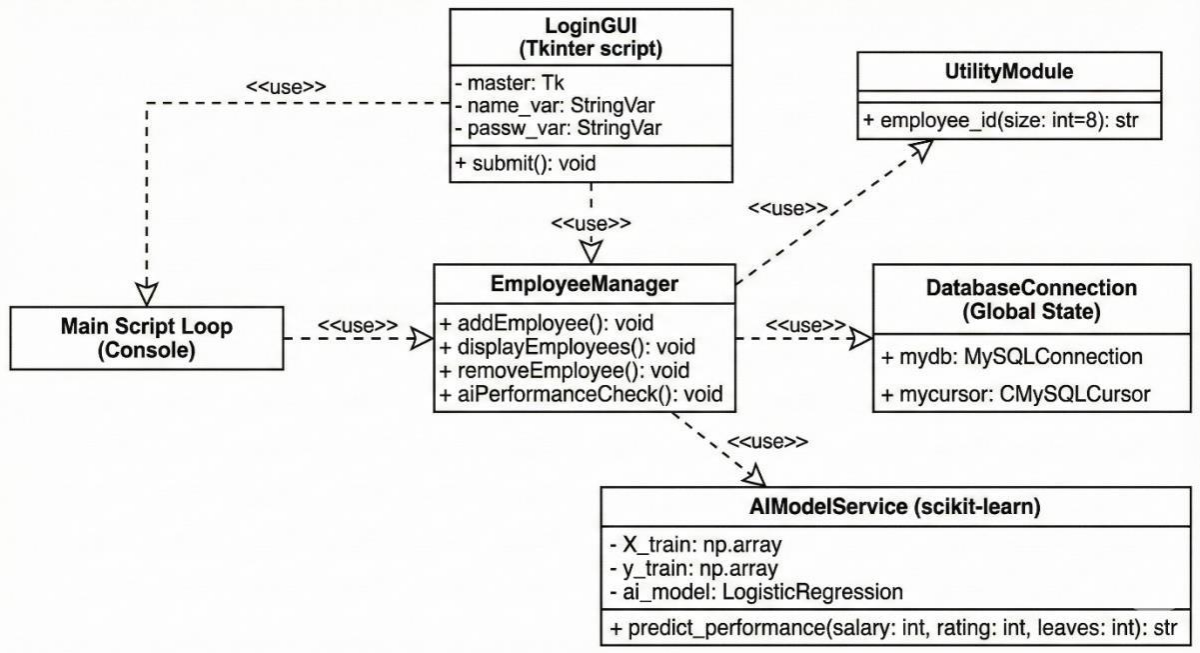
myresearchgo Volume 2, Issue 1, January 2026, ISSN: 3107-3816 (Online)

[INSERT SCREENSHOT: Command Line Interface showing "AI PERFORMANCE PREDICTION: HIGH PERFORMANCE"]

Graphical vs. Command Line Interface

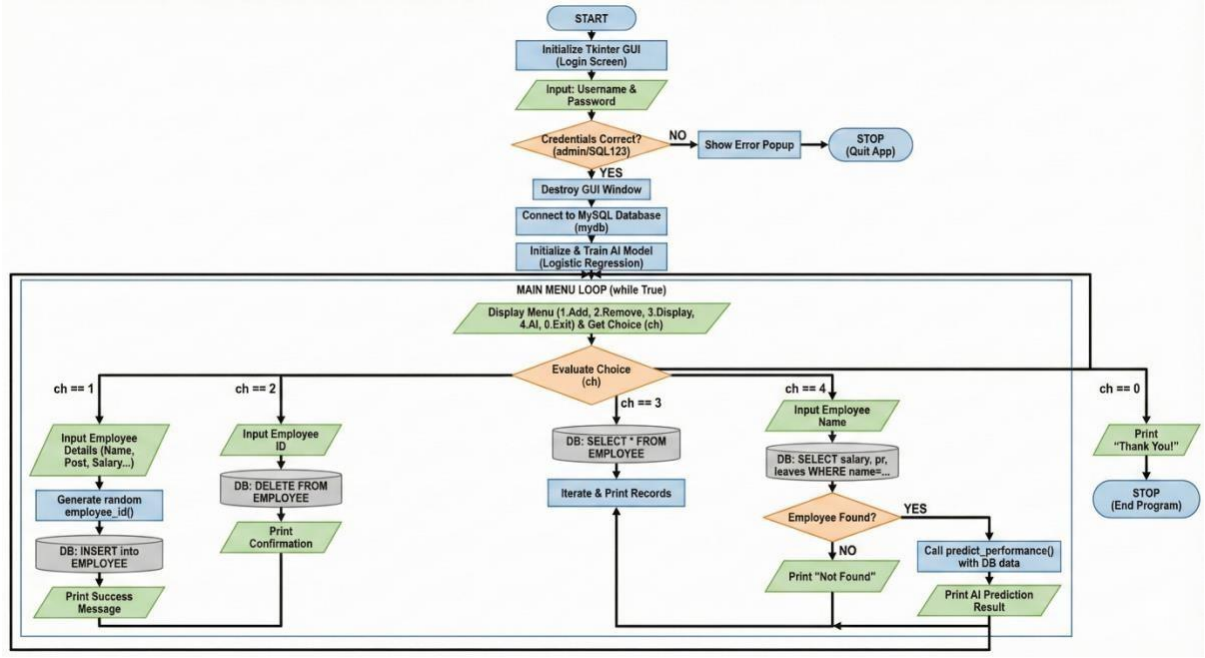
The current implementation uses a hybrid approach: GUI for Login and CLI (Command Line Interface) for CRUD operations. This ensures the security layer is user-friendly, while the data entry remains fast for administrators comfortable with text interfaces.

UML DIAGRAMS :

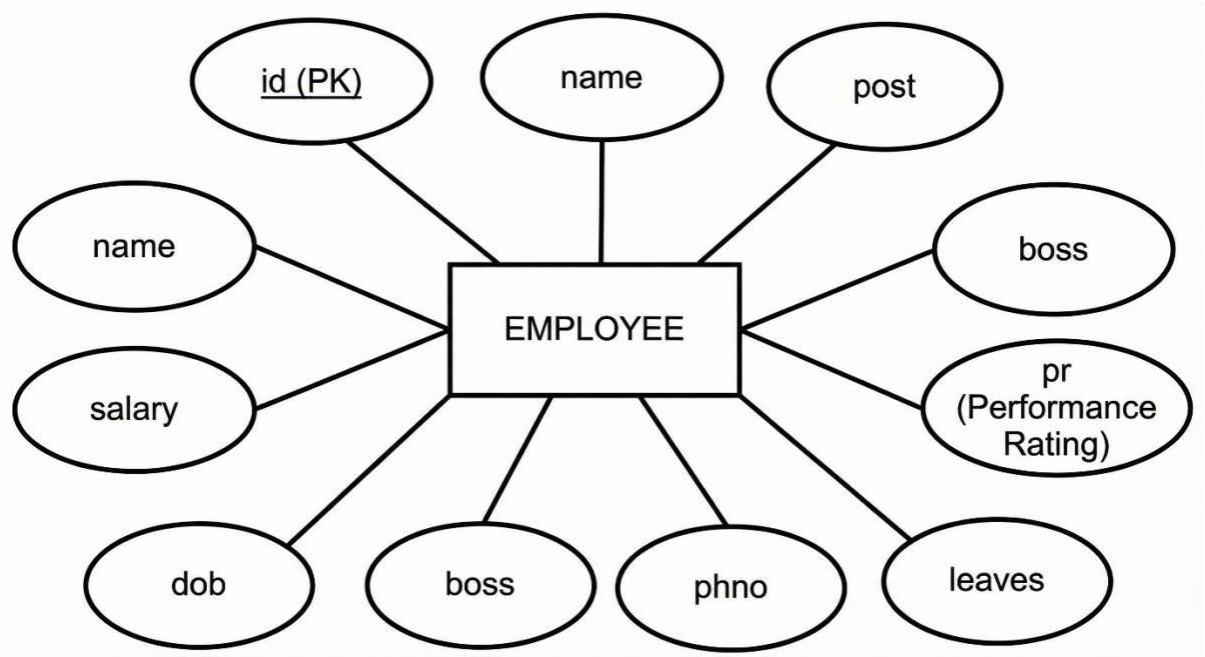


myresearchgo Volume 2, Issue 1, January 2026, ISSN: 3107-3816 (Online)

FLOW CHART DIAGRAM:



ER DIAGRAM:



Conclusion and Future Scope

Conclusion

This project successfully demonstrates the viability of integrating Machine Learning into standard CRUD applications. The "CosmaAngel Inc. EMS" provides a secure, centralized platform for employee data while offering value-added intelligence through performance prediction. By utilizing Python's extensive ecosystem (tkinter, mysql-connector, sklearn), we created a lightweight yet powerful tool that aids HR decision-making.

Future Scope

To further enhance this research, the following improvements are proposed:

1. **Dynamic Training:** Instead of hardcoded arrays, the AI model should retrain itself periodically using the live data from the MySQL database.
2. **Full GUI Implementation:** Expanding the tkinter interface to cover the Add/Remove/Display functions, replacing the CLI.
3. **Advanced Algorithms:** Replacing Logistic Regression with Random Forest or Neural Networks for more complex pattern recognition (e.g., predicting employee attrition).
4. **Visualization:** Integrating matplotlib to generate graphs of employee salary distributions and performance curves.

References

1. Date, C. J. (2004). *An Introduction to Database Systems*. Pearson Education.
2. Davenport, T. H., Harris, J., & Shapiro, J. (2010). *Competing on Talent Analytics*. Harvard Business Review.
3. Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research.
4. Rossum, G. (1995). *Python Reference Manual*. CWI Report.
5. Oracle. (2025). *MySQL Connector/Python Developer Guide*. Oracle Corporation.
6. Bandebuche, M. S. Artificial Intelligence to Improve Interpretability of Neural Networks: A Comparative Study of Attention Mechanisms, SHAP, and LIME.
7. Joshi, M. A. D. Bibliometric Survey on Deep Learning Based Recommendation System.