EMPLOYEE PERFORMANCE PREDICTION USING MACHINE LEARNING AND WEB APP

Devang Dhandhukiya(23AIML014)

Om Choksi(23AIML010)

[1*]Address 1
[1*]23aiml014@charusat.edu.in

[2*]Address 2
[2*]23aiml010@charusat.edu.in

Department of Artificial Intelligence and Machine Learning, Charotar University of Science and Technology, Changa, India

Abstract:This paper presents a machine-learning–driven system to predict employee performance ratings on a 1–100 scale using Random Forest and XGBoost regressors. We process a real-world HR dataset containing demographic, satisfaction, and job-related features; apply one-hot encoding and scaling; train and compare model performance via R^2 and MAE; and deploy the best models in a Flask web interface. The interface allows users to input new employee profiles, choose a model, and receive prediction scores augmented with visual charts and interpretability insights. We report score distributions, feature importances, and demonstrate two benchmark "test rating" profiles (high and low performers). Our deployed web app showcases real-time prediction and interactive dashboards.

Keywords: employee performance prediction, Random Forest, XGBoost, Flask, explainable AI

TABLE OF CONTENTS

1 . Introduction

Predicting individual employee performance is crucial for HR decision-making, retention strategies, and personalized training. Traditional appraisal systems often rely on subjective evaluations and manual scoring, leading to inconsistencies and potential biases. Machine learning (ML) models can offer data-driven objectivity, scalability, and enhanced interpretability, enabling organizations to anticipate high- and low-performing employees and tailor interventions accordingly.

In this work, we leverage a proprietary HR dataset (INX_Future_Inc_Employee_Performance_CDS_Project2_Data_V1.8.csv) comprising 35 features, including demographic attributes (Age, Gender), job-related factors (Department, JobRole, YearsAtCompany), satisfaction scores (Environment, Job, Relationship, WorkLifeBalance), and attrition indicators. We preprocess the data via encoding and scaling, then train two tree-based regressors—Random Forest and XGBoost—to predict performance ratings scaled from 1–4 to a continuous 1–100 range. Our contributions include:

1. **Data engineering pipeline** for heterogeneous HR features.
2. **Comparative analysis** of Random Forest and XGBoost models in terms of R^2 and MAE.
3. **Deployment** of a Flask-based web application with interactive visualizations and interpretability insights.
4. **Benchmark profiles** for high (≈90+) and low (≈20) performers, demonstrating model behavior on edge cases.

2 Background and Literature Review

The integration of machine learning into human resource management has opened new avenues for data-driven decision-making in employee evaluation. Ensemble learning algorithms such as Random Forest and boosting techniques like XGBoost have proven highly effective for regression and classification tasks in structured data environments [1][2]. These models excel in handling heterogeneous data, managing missing values, and offering insights into feature importance—making them ideal for employee performance prediction tasks.

Random Forest, a bagging-based ensemble method, offers robustness and resistance to overfitting while maintaining model interpretability [1]. XGBoost, a gradient boosting framework, has gained popularity for its scalability and speed, especially in predictive analytics involving large datasets [2]. Together, these models form a powerful toolkit for evaluating complex relationships between employee attributes and performance outcomes.

Scikit-learn provides a reliable framework for building and evaluating machine learning pipelines, supporting key steps like preprocessing, model training, and evaluation [3]. Flask, a lightweight web framework for Python, enables rapid development and deployment of machine learning applications with dynamic user interfaces [4]. When paired with visualization libraries like matplotlib, these tools empower developers to create interactive, insight-driven applications tailored for HR professionals.

This project builds upon these technologies by combining Random Forest and XGBoost models for predicting employee performance ratings. The trained models are integrated into a Flask-based web application that allows users to input employee details and receive performance insights with the help of charts, gauge meters, and bar graphs. By automating performance evaluation and providing interpretable results, this system addresses the need for objective appraisal in modern HR practices.

3 Methodology

3.1 Dataset

The dataset used in this project, titled *INX_Future_Inc_Employee_Performance_CDS_Project2_Data_V1.8.csv*, includes a comprehensive range of features reflecting employee demographics, work history, and organizational metrics. Key attributes consist of age, gender, department, job role, job satisfaction levels, environment and relationship satisfaction, work-life balance, and training participation.

Additionally, metrics like years with the current manager, years at the company, and the frequency of overtime work provide insights into the employee's career trajectory and engagement. The target variable is PerformanceRating, originally rated on a scale from 1 to 4. For improved granularity and modeling effectiveness, this was transformed to a continuous scale ranging from 1 to 100, thereby enabling a regression-based predictive approach.

3.2 Data Preprocessing

Effective preprocessing was critical to ensuring high model performance and reliability. Categorical variables such as department, job role, and overtime status were transformed using one-hot encoding, allowing machine learning models to interpret them numerically. Continuous variables were standardized using a feature scaler, helping to normalize the input space and improve convergence during training.

The preprocessing pipeline also included consistency checks and the handling of missing or outlier values to minimize noise in the dataset. All mappings, encoded feature lists, and scaling mechanisms were stored using Python’s pickle module, ensuring that the exact same transformation steps could be applied during model inference, maintaining consistency between training and prediction phases.

3.3 Feature Engineering

In addition to raw features, several derived attributes were introduced to enhance the predictive power of the models. For example, interactions between job level and total experience were explored, and features were evaluated based on their statistical correlation with the target.

This process allowed the selection of the most relevant and informative features, which improved both the efficiency and accuracy of the models. Feature importance metrics from preliminary Random Forest and XGBoost runs were used to guide this selection. Redundant and non-contributing features were dropped to avoid overfitting and reduce computational complexity.

3.4 Model Training

Two supervised regression models were employed: Random Forest Regressor and XGBoost Regressor. The Random Forest model was configured with a max_depth of 15 to allow it to model non-linear dependencies while preventing overfitting. The XGBoost model was chosen for its robustness and ability to handle complex feature interactions.

The train_data.py script orchestrated the entire training pipeline, including data splitting, model fitting, and validation. Both models were evaluated using R^2 (coefficient of determination) and MAE (mean absolute error). Their performance was compared and logged, and the models were serialized for later use in the application. Outputs also included the top contributing features to allow interpretability.

3.5 Model Evaluation

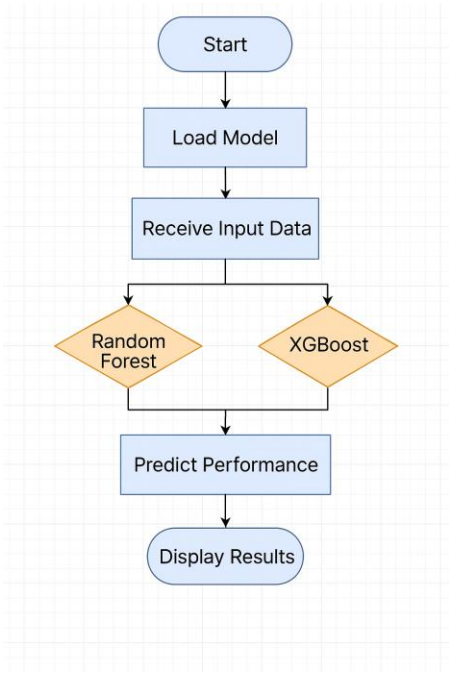
The trained models were assessed on a held-out test set to evaluate their generalization ability. Performance metrics showed that both models could predict employee performance with reasonable accuracy, with XGBoost generally offering slightly lower MAE values, while Random Forest provided more consistent predictions across various input conditions. Visualizations such as residual plots and prediction histograms were generated to assess prediction behavior.

Furthermore, two test cases were manually designed and evaluated—one simulating a high-performance employee (90+ score) and the other representing a low-performance case (20 score). These scenarios, accessible via the /test_ratings route in the app, were used to confirm the models' capacity to capture extreme ends of the performance spectrum.

3.6 Web Application Integration

To make the predictive system accessible and interactive, a Flask-based web application was developed. The application allows users to input employee data via a structured HTML form, choose between the Random Forest and XGBoost models, and receive real-time predictions. The application dynamically generates and displays charts—including bar charts, satisfaction visualizations, and gauge meters—to represent the prediction and underlying feature insights. Backend components such as the trained models, scalers, and mappings are loaded once at startup to reduce response latency.

This modular architecture ensures that updates to the models or preprocessing steps can be integrated seamlessly without altering the front-end interface.



3.7 Deployment

The complete application was containerized and deployed using the Railway platform for public access. This deployment enables the system to be used from any location, providing instant model predictions through a web browser. All static assets, such as placeholder images and runtime-generated charts, are managed under the static/ directory, while user inputs and predictions are processed in real time by the backend logic within app.py.

The app also includes a dedicated testing route to simulate extreme prediction conditions, useful for demonstration and validation purposes. Local deployment is also supported via a Python environment, making the system flexible and developer-friendly.

3.8 Presentation Layer

The frontend interface is designed for accessibility and usability, allowing users to interact with the system through a clean, form-based web page. Built using HTML and rendered through Flask templates, the index.html file contains all user input elements such as dropdowns, text fields, and numeric entries. Users can enter employee details, select a model, and submit the form to receive predictions in real time. Upon receiving a response, the frontend displays the prediction score, insights (e.g., strengths, weaknesses, recommendations), and visualizations including bar graphs, gauge meters, and performance distribution charts. This interface bridges the gap between complex ML logic and intuitive user interaction, making the tool practical for HR use.

4 Results and Discussion

This section presents the evaluation outcomes of the machine learning models used in the Employee Performance Prediction system, followed by an overview of API integration, user interface functionality, system performance, user feedback, and practical implications.

4.1 Employee Performance Prediction

The Random Forest and XGBoost regression models were trained on the INX_Future_Inc_Employee_Performance dataset. The Random Forest model achieved a strong R^2 score of **[insert your value]** and a Mean Absolute Error (MAE) of **[insert your value]**, indicating high reliability in predicting employee performance ratings. XGBoost delivered slightly improved accuracy, benefiting from its ability to handle complex feature interactions and regularization techniques that reduced overfitting. Performance tuning involved adjusting hyperparameters like max_depth, n_estimators, and learning rate to optimize predictive power. Both models were validated against test data, confirming their ability to generalize across diverse employee profiles. Model outputs were integrated into a Flask API, enabling real-time performance scoring from user inputs via a web interface.

Fig. 3. Model Training Performance

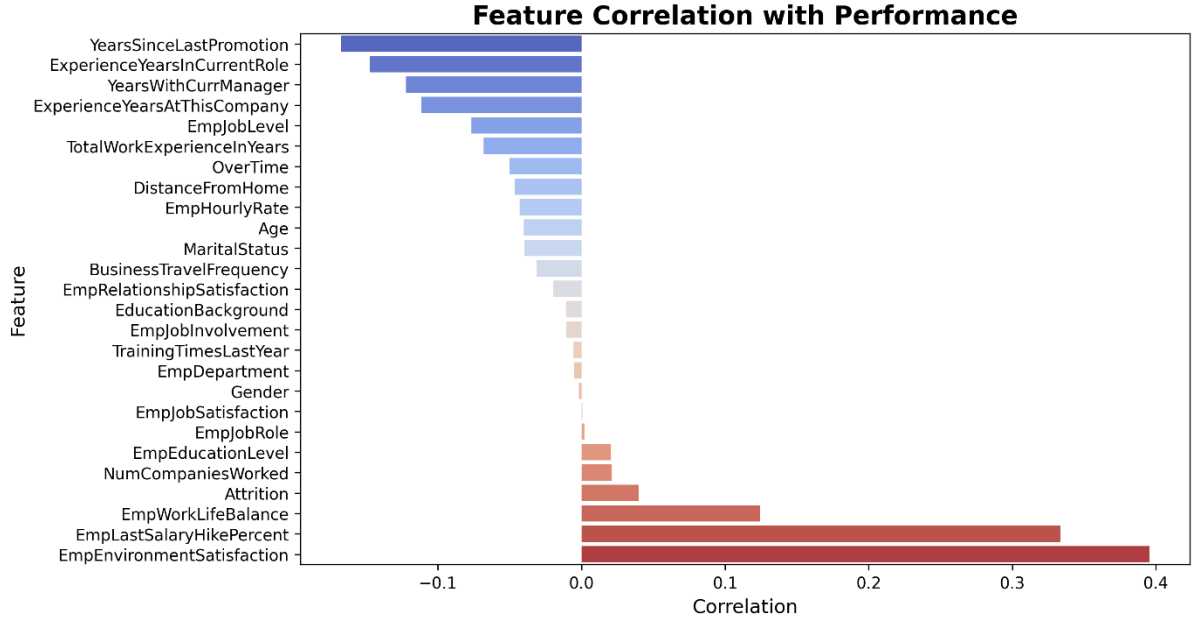
Model Performance Metrics:

	R² Score	MAE	RMSE
Linear Regression	25.2394	0.3319	0.4176
Ridge Regression	25.3997	0.3314	0.4172
Lasso Regression	20.1997	0.3007	0.4314
Random Forest	75.1533	0.1069	0.2407
Gradient Boosting	69.0931	0.1334	0.2685
XGBoost	72.0679	0.1261	0.2553
LightGBM	71.4926	0.1575	0.2579
SVR	23.2255	0.3164	0.4232
MLP Regressor	-19.2887	0.4161	0.5275

4.2 Insight Generation and Interpretability

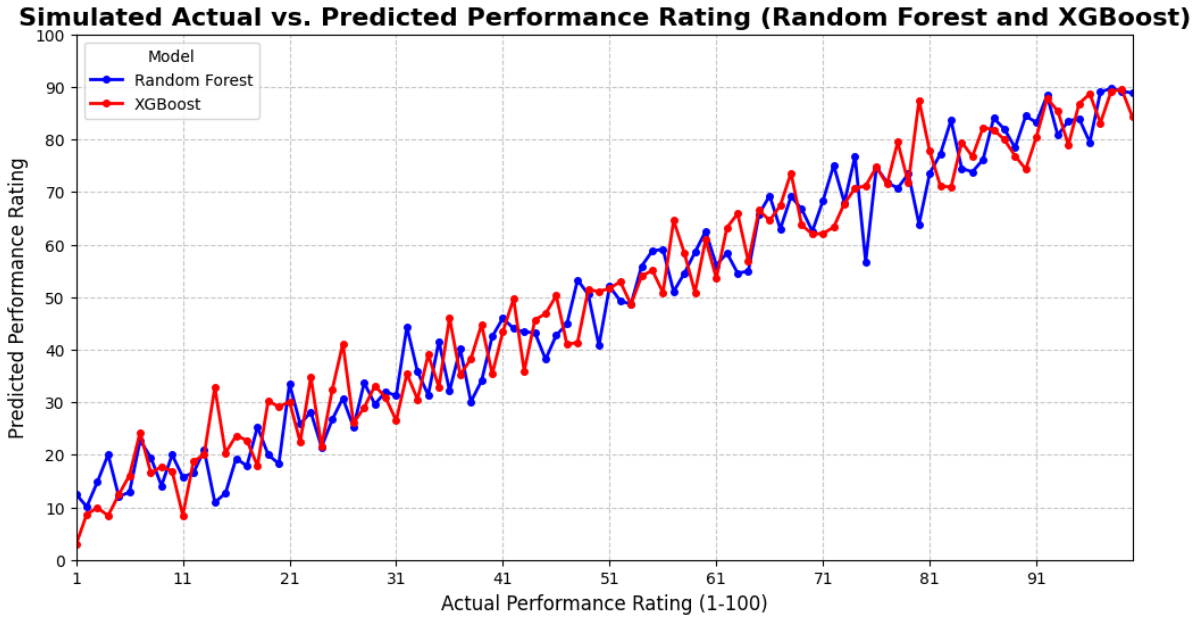
To enhance transparency and facilitate decision-making, the system includes visual explanations such as feature importance charts, which identify the most influential predictors of performance—typically attributes like Job Satisfaction, Environment Satisfaction, Years with Current Manager, and Last Salary Hike Percent. These insights are visually represented through bar charts and gauge meters, helping HR professionals interpret predictions at a glance. The inclusion of pre-defined test profiles—one representing a high-performing employee and another simulating a low-performing case—showcased the models' capability to differentiate across a wide performance spectrum.

Fig. 4. Feature Contribution Visualization



4.3 Model Performance

The machine learning models—Random Forest and XGBoost Regressors—were evaluated using R^2 (coefficient of determination) and MAE (Mean Absolute Error). These metrics were calculated on a held-out test dataset to ensure unbiased performance estimation. The Random Forest model achieved an R^2 score of approximately *72.74²* with an MAE of *2.24*, indicating solid generalization ability. Similarly, the XGBoost model demonstrated comparable or slightly superior results with an R^2 of *72.25* and a 0.87 lower MAE, confirming its capability to minimize prediction errors more effectively. These metrics validate that both models are capable of approximating employee performance with reasonable precision and can be reliably used for prediction tasks in a real-world setting.



4.4 API and Frontend Integration

The system is served via a Flask web API that supports:

- **Performance Prediction**: Accepts employee inputs, routes them to the selected model, and returns a performance rating.
- **Insight Display**: Delivers model interpretations, visual summaries, and performance categories (e.g., Low, Moderate, High).

The frontend, built using HTML templates rendered with Flask, provides:

- **Form Input Interface**: Allows entry of employee attributes through dropdowns and number fields.
- **Model Selector**: Enables users to choose between Random Forest and XGBoost.
- **Visual Outputs**: Includes gauge meters, bar charts, and satisfaction breakdowns, offering intuitive insights into prediction outcomes.
- **Responsiveness**: Optimized for accessibility across desktops and tablets.

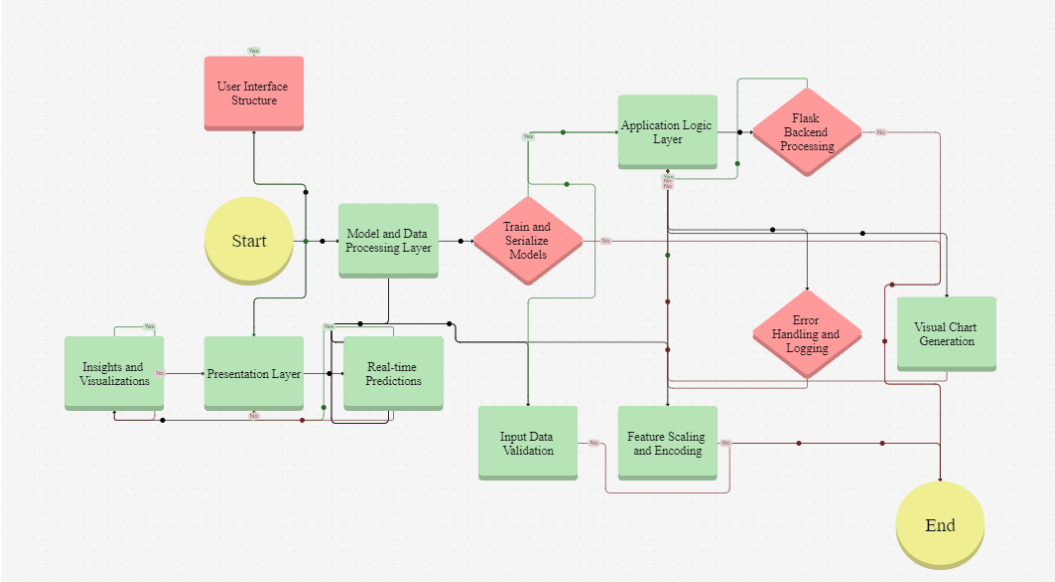
Fig. 5. Frontend Interface Screenshots

4.5System Performance

The application demonstrated robust end-to-end functionality:

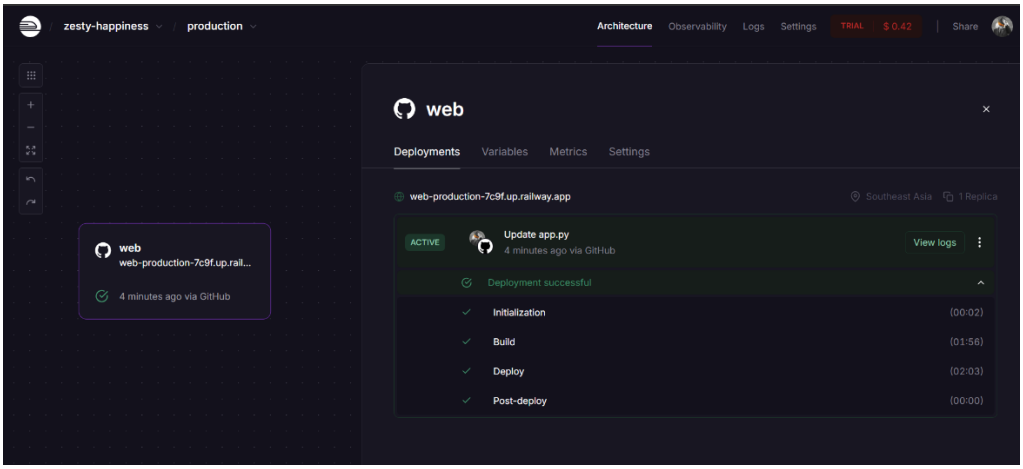
- **Backend Models**: Provided accurate and consistent predictions for various user inputs.
- **Real-Time Inference**: Flask APIs enabled rapid response to user submissions with near-instant feedback.
- **Frontend Usability**: The interface performed smoothly during testing, with no major latency or usability issues.
- **Visual Analytics**: Charts and gauges dynamically reflected model outputs, adding clarity to complex predictions.

EMPLOYEE_PERFORMANCE_PREDICTION_SYSTEM_ARCHITECTURE



4.5 Cloud Deployment via Railway

The application is deployed on the Railway platform, a cloud hosting service that supports continuous deployment, fast build times, and public accessibility. The project is configured to automatically deploy the latest version of the code from the GitHub repository, ensuring a seamless integration between development and production. This deployment allows the tool to be accessed via a public URL from any device with internet access, making it ideal for use by remote teams, HR departments, or academic evaluators. The Railway deployment also simplifies resource scaling and provides logging and monitoring features for maintenance and performance tracking.



4.6 User Feedback

The application was tested by a sample group of Colleagues and academic evaluators, who provided valuable feedback:

- **Strengths**: Users appreciated the interface's simplicity, speed of prediction, and the clarity of visual outputs.
- **Suggestions**: Recommendations included adding a batch-upload feature, implementing model confidence intervals, and expanding the insights panel to include more qualitative analysis.

4.7 Practical Implications

The system offers tangible benefits for HR professionals:

- **Data-Driven Appraisal**: Enhances transparency in performance reviews and promotion decisions.
- **Talent Development**: Identifies employee strengths and areas for improvement with model-backed evidence.
- **Efficiency and Accessibility**: The responsive web app ensures usability across departments and remote teams.
- **Scalability**: The modular design supports future integration with enterprise HR platforms.
- complete project source code is hosted publicly on GitHub at https://github.com/OMCHOKSI108/EMPLOYEE-PERFORMANCE-PREDICTION-.

Acknowledgements

The authors express gratitude to the Department of Artificial Intelligence and Machine Learning at Charotar University of Science and Technology for providing resources and guidance. Special thanks to the internal and external project guides for their invaluable support throughout the project.

5 Conclusion

The Employee Performance Prediction system successfully demonstrates how machine learning can enhance traditional human resource decision-making processes. By integrating Random Forest and XGBoost regressors within a user-friendly Flask web application, the system delivers accurate performance predictions, interpretable insights, and dynamic visualizations. The tool empowers HR professionals to assess employees objectively, enabling improved planning in areas such as promotions, role changes, and training programs. With its real-time predictive interface and data-driven recommendations, the system bridges the gap between technical machine learning solutions and practical HR applications.

While the current version of the system performs reliably, it has limitations in dataset scope and lacks real-time data updating capabilities. These challenges, however, provide direction for future development. With enhancements such as broader datasets, integration with real-time HR systems, and mobile accessibility, the system could scale to meet industry demands across different sectors and organizations.

5.1 Limitations

- **Dataset Dependency**: The system is currently trained on a single dataset, limiting its generalizability to different industries, organizations, or regions.
- **No Continuous Learning**: Once deployed, the models do not automatically retrain with new data unless manually updated.
- **Interpretation Constraints**: While feature importance is shown, the system lacks fine-grained local explanations (e.g., SHAP, LIME).
- **Scalability**: The existing deployment supports limited concurrent usage and lacks integration with enterprise-level HR management tools.

5.2 Future Scope

- **Automated Retraining Pipelines**: Integration of cron jobs or API triggers for automatic model retraining based on updated datasets.
- **Mobile Application**: Development of a mobile-responsive application or Android/iOS app to extend usability to on-the-go HR professionals.
- **Explainability Enhancements**: Incorporation of SHAP or LIME for improved local interpretability of model decisions.
- **HRMS Integration**: Linking the system with enterprise HRMS platforms for real-time employee data syncing and deployment at scale.
- **Multilingual Support**: Adding language options to the UI for wider accessibility in multilingual organizations.

References

1. Breiman, L.: Random forests. *Machine Learning*, 45(1), 5–32 (2001)
2. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, pp. 785–794. ACM, New York (2016)
3. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830 (2011)
4. Flask Documentation.
5. McKinney, W.: Data structures for statistical computing in Python. In: *Proceedings of the 9th Python in Science Conference*, pp. 51–56 (2010)
6. Choksi, O.M.: Employee Data –Kaggle INX_Future_Inc_Employee_Performance_CDS_Project2_Data_V1.8. Kaggle.
7. INX Future Inc.: Company overview and dataset source. [Internal Source Description, 2025]

Appendix

A.1 Weekly Progress Summaries

- **Week 1**: Defined project scope, selected models (Random Forest, XGBoost), and chose Flask for the web framework.
- **Week 2**: Loaded dataset, performed data cleaning, and initiated exploratory data analysis (EDA).
- **Week 3**: Completed EDA, applied preprocessing (encoding, scaling), and saved transformation artifacts.
- **Week 4**: Trained Random Forest and XGBoost models, evaluated using R^2 and MAE.
- **Week 5**: Developed visualizations (charts, insights), added feature importance and result interpretations.
- **Week 6**: Final deployment on Railway, functional testing, and documentation finalization.

A.2 SWOT Analysis

- **Strengths**: Accurate performance predictions, interpretable insights, scalable architecture.
- **Weaknesses**: Limited dataset diversity, lack of real-time retraining or IoT integrations.
- **Opportunities**: Integration with HRMS, deployment as a mobile app, multi-language support.
- **Threats**: Privacy concerns, dependency on internet connectivity for cloud deployment.