International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 07 | July - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Proactive Measures of the Organization Regarding Employee Attrition Using Deep Learning

Author: SINGAMPALLI RAMASRI[1] (MCA student), M. BALA NAGA BHUSHANAMU[2] (Asst. Prof)
Department of Information Technology & Computer application,
Andhra University College of Engineering, Visakhapatnam, AP.
Corresponding Author: SINGAMPALLI RAMASRI
(email-id: singampalliramasri@gmail.com)

ABSTRACT: Employee attrition remains a persistent concern for organizations, often leading to increased costs and disruptions in workforce planning. To address this issue, we present a predictive system that leverages deep learning to assess the likelihood of employee resignation. The proposed solution combines a PyTorch-based Multi-Layer Perceptron (MLP) with a Flask-powered web interface, enabling real-time, user-friendly attrition prediction. The model is trained on structured HR data that includes both numerical and categorical attributes related to employee demographics, job roles, and workplace behaviour. Preprocessing steps involve one-hot encoding for categorical variables and normalization for numerical inputs, ensuring compatibility with the neural network. The application calculates a probability score using sigmoid activation applied post-inference and classifies the outcome based on a configurable threshold. The web interface collects user inputs and delivers immediate feedback on attrition risk, making the system accessible to HR personnel without technical expertise. This approach demonstrates the feasibility of integrating machine learning models into operational HR workflows and highlights the role of deep learning in improving employee retention strategies. The system is modular, scalable, and suitable for extension to include additional factors such as employee feedback or performance metrics.

Keywords: *Keywords Employee Attrition, Deep Learning, Multi-Layer Perceptron, Predictive Analytics, Human Resource Management, PyTorch, Flask, Feature Engineering, Classification, Workforce Analytics.*

----------**********----------

I. INTRODUCTION

The strength and success of an organization are largely driven by the skills, experience, and commitment of its workforce. Their knowledge, skills, and experience directly influence business outcomes, from operational success to innovation and customer satisfaction. However, many organizations face a recurring challenge: employee attrition. Whether due to voluntary resignations or organizational restructuring, attrition can disrupt workflow, reduce team effectiveness, and increase the cost and time associated with hiring and training new personnel. Frequent employee departures can disrupt internal operations and place additional strain on an organization's financial and structural stability. For instance, losing skilled employees often means losing business knowledge and weakening the morale of remaining staff. These disruptions can delay project timelines and place additional strain on human resource departments. Traditional methods of understanding employee exits such as interviews or satisfaction surveys—typically occur after the employee has already departed. While such feedback is useful, it does not offer a way to anticipate or prevent attrition in advance. This situation has prompted organizations to explore data-driven strategies for identifying employees at risk of leaving. With the growing availability of structured HR data, machine learning techniques offer a promising path forward. By analysing trends in employee attributes—such as job role, compensation, tenure, and engagement—predictive models can recognize patterns that may indicate future attrition. Despite ongoing interest in this field, many existing approaches fall short due to limited accuracy, difficulty handling mixed data types, or a lack of integration into usable applications.

This research proposes a practical solution by combining a deep learning model with a lightweight web interface. Our contributions are outlined as follows:

A deep learning model is built using a Multi-Layer Perceptron (MLP) architecture in PyTorch to classify whether an employee is likely to leave.

The data pipeline is designed to handle both numerical and categorical inputs through appropriate transformation techniques, including normalization and one-hot encoding.

A Flask-based web application is developed to make the model accessible to non-technical HR personnel, offering real-time predictions from manual input.

The system is tested on real-world-like HR data and is designed to be extendable, lightweight, and ready for deployment in a practical HR environment.

The remainder of this paper is structured as follows: Section II summarizes related work in the domain of attrition prediction and AI in human resource management. Section III presents the proposed methodology, including data handling, model architecture, and system design. Section IV evaluates the results and discusses model performance.

II. Related work:

Employee attrition prediction has been a prominent area of research in human resources analytics, with diverse approaches employed to model and forecast turnover. Prior studies can broadly be classified into four categories: traditional machine learning techniques, deep learning models, feature engineering strategies, and deployment-focused systems.

2.1 Traditional Machine Learning Approaches

Conventional machine learning algorithms such as logistic regression, decision trees, support vector machines (SVM), and random forests have been widely used for attrition modelling. These models are valued for their interpretability and relatively low computational cost. For instance, logistic regression is often applied due to its simplicity in understanding the influence of independent variables on attrition likelihood. However, it assumes linear relationships between predictors and the outcome variable, which limits its effectiveness when dealing with complex patterns.

Tree-based models like decision trees and random forests can capture non-linear interactions and variable importance. Nonetheless, they may overfit the training data, especially when dealing with high-dimensional HR datasets. SVMs have also been utilized, offering robust performance in binary classification tasks but often require careful kernel tuning and are less scalable with larger datasets. While these methods provide baseline performance, they generally require significant manual feature engineering and may underperform in capturing nuanced interdependencies within employee data.

2.2 Deep Learning-Based Techniques

With advancements in deep learning, neural networks have increasingly been adopted in predictive HR analytics. Multi-Layer Perceptron (MLPs), in particular, have shown promise in modelling complex, non-linear relationships inherent in employee behavior and performance data. These models can automatically learn high-level representations of input features, reducing the reliance on handcrafted feature extraction.

Some studies have explored recurrent neural networks (RNNs) and convolutional neural networks (CNNs) when temporal or unstructured data such as employee feedback is available. Despite their strengths, deep learning models often face challenges in interpretability and require more computational resources. Nonetheless, MLPs remain a strong candidate for structured tabular data, as commonly found in HR records. They offer improved accuracy over traditional models when appropriately tuned and trained on well- pre-processed data.

2.3 Feature Engineering and Data Preprocessing Strategies

The effectiveness of attrition prediction heavily depends on how input features are prepared. Researchers have explored multiple encoding techniques to convert categorical variables into numerical representations suitable for machine learning models. One-hot encoding, label encoding, and embedding layers are common methods. Similarly, normalization or standardization is often applied to numerical features to ensure consistent model training.

Handling mixed data types—both categorical and numerical—poses challenges in ensuring balanced model learning. Prior works have emphasized the importance of designing preprocessing pipelines that maintain the integrity of relationships between features. In this context, our use of one-hot encoding for categorical variables and standard scaling for continuous ones aligns with best practices and ensures compatibility with the MLP architecture.

2.4 Practical Applications and Model Deployment

Although numerous studies have addressed model development, fewer have focused on real-world deployment. Some research has explored integrating predictive models into dashboards or HR information systems, but many remain at the experimental stage. Web-based applications leveraging frameworks like Flask or Django have been proposed to operationalize machine

learning models, providing HR professionals with accessible tools for decision support.

However, these applications often lack real-time integration or user-friendly interfaces tailored for non-technical stakeholders. The deployment of deep learning models into functional web applications remains an area with limited exploration, especially in academic literature. Our contribution addresses this by combining a trained PyTorch MLP model with a Flask-based interface for seamless prediction and interaction.

2.5 Research Gap and Motivation

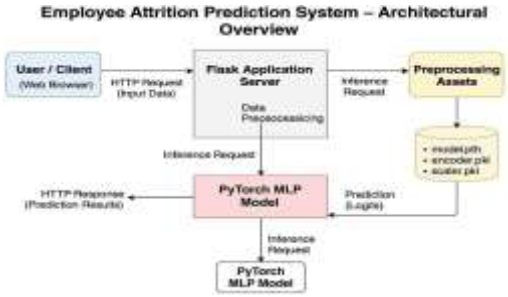
While previous works have made notable progress in attrition prediction using both traditional and deep learning approaches, gaps remain in integrated systems that not only provide high predictive performance but also deliver practical usability. Many existing models focus on accuracy without addressing deployment concerns or the challenges of handling diverse feature types in production environments.

This paper aims to bridge that gap by presenting a complete pipeline from preprocessing to model training and deployment—suitable for real-world use. Our work stands out by merging deep learning methodologies with an operational web application designed to support HR departments in proactive retention planning.

III. System Architecture and Methodology

System Architecture

The proposed system is designed to provide HR departments with an accessible and intelligent tool for predicting employee attrition. It integrates data preprocessing, deep learning inference, and a user-friendly interface into a cohesive pipeline that can be deployed in real-world settings.



A. System Overview

At a high level, the system accepts employee information through a web interface, processes the input using pre-trained encoders and scalers, feeds the data into a trained neural network model, and returns the likelihood of attrition. The architecture is modular and consists of four key components: the data input interface, preprocessing pipeline, prediction engine, and result presentation module.

B. Architecture Components

User Input Interface: A web form built using HTML and served by Flask, where users can input categorical and numerical employee data.

Preprocessing Layer: This component includes an encoder for transforming categorical variables and a scaler for standardizing numerical features. These tools were trained on historical HR data and saved for reuse using job lib.

Prediction Engine: A trained Multi-Layer Perceptron (MLP) model developed with PyTorch. It accepts pre-processed inputs and outputs a prediction score indicating attrition risk.

Output Layer: The prediction is converted into a probability using the sigmoid function and interpreted into a binary output. Results are displayed back on the web interface.

Data flows from the user interface to the preprocessing layer, then through the prediction engine, and finally to the output display.

Methodology

This section describes the step-by-step process followed to build, train, and deploy the employee attrition prediction system. It covers the dataset, feature processing, neural network architecture, training setup, and application deployment.

A. Dataset Description

The dataset used in this study includes demographic and employment-related data for a set of employees. It contains both categorical attributes (e.g., job role, department, marital status) and numerical attributes (e.g., age, monthly income, years at company). The target variable is binary, indicating whether or not the employee left the organization. The class distribution is imbalanced, with fewer examples of attrition, necessitating careful consideration during model training.

B. Data Preprocessing

The preprocessing stage transforms raw data into a format suitable for neural network training and inference.

Categorical Encoding: Categorical features are transformed using one-hot encoding. This creates binary columns for each category, allowing the model to interpret them as numerical vectors.

Feature Scaling: Numerical features are standardized using a standard scaler, ensuring they have zero mean and unit variance. This is essential for efficient learning in neural networks, preventing any one feature from dominating due to its scale.

Feature Combination: The encoded categorical and scaled numerical features are horizontally stacked into a single feature matrix, which becomes the input to the MLP.

These transformation objects (encoder and scaler) are persisted using joblib and reloaded during prediction.

C. MLP Model Architecture

A Multi-Layer Perceptron (MLP) model was chosen for its ability to capture non-linear patterns in tabular data. The architecture consists of:

Input Layer: Receives a feature vector formed by combining all transformed attributes.

Hidden Layers: Two hidden layers with 64 and 32 neurons respectively, each followed by ReLU activation. Dropout (rate = 0.3) is applied to reduce overfitting.

Output Layer: A single neuron outputs a raw logit score. A sigmoid function is applied during prediction to convert this to a probability score.

This architecture balances expressiveness with generalization, suitable for structured HR datasets.

D. Training Process

The model was trained using the following configuration:

Loss Function: Binary Cross-Entropy with Logits (BCEWithLogitsLoss), which combines the sigmoid activation and cross-entropy loss in a numerically stable way.

Optimizer: Adam optimizer was used for efficient convergence with an initial learning rate set experimentally.

Hyperparameters: The model was trained for a fixed number of epochs using a mini-batch gradient descent approach. Batch size, learning rate, and epoch count were selected based on performance on a validation set.

Environment: Training was conducted in Python using PyTorch on a standard computing environment.

E. Deployment with Flask

To make the model accessible to non-technical users, it was deployed as a web application using Flask.

Model Serving: The trained model, along with its encoder and scaler, is loaded into memory when the application starts.

Prediction Logic: When a user submits data, it is pre-processed and passed to the MLP. The model produces a raw score, which is converted into a probability using the sigmoid function. A threshold is then applied to produce a binary prediction.

User Interface: A simple HTML page is used for input and result display. This makes the system usable in any browser without requiring programming skills.

This deployment strategy makes the system practical for real-world HR teams who seek rapid and interpretable insights into employee retention risks.

IV. IMPLEMENTATION AND TECHNOLOGIES USED

This section outlines the key technologies, frameworks, and libraries employed in the development of the proposed employee attrition prediction system. It also describes how these tools were integrated to ensure a smooth and reliable user experience, both from a modelling and deployment perspective.

A. Development Environment and Core Technologies

The system was developed using Python (version 3.8+), selected for its extensive support for machine learning, data analysis, and web development through numerous open-source libraries. The core components of the system include:

PyTorch: This deep learning framework was utilized to build and train the neural network. Its dynamic computation graph, intuitive syntax, and strong community support made it a suitable choice for rapid prototyping and deployment of the MLP model.

Flask: Chosen as the web framework for deployment, Flask enables lightweight and flexible server-side development. It allows the model to be accessed via a browser, facilitating user interaction without the need for advanced technical knowledge.

Pandas and NumPy: These libraries were used extensively during data exploration and transformation. Pandas

provided structured data manipulation capabilities, while NumPy offered efficient operations on numerical arrays.

Joblib: This utility was employed to serialize and deserialize preprocessing objects such as the OneHotEncoder and StandardScaler. Its integration ensures that the same transformation steps used during training are consistently applied during prediction.

B. Data Preprocessing Pipeline

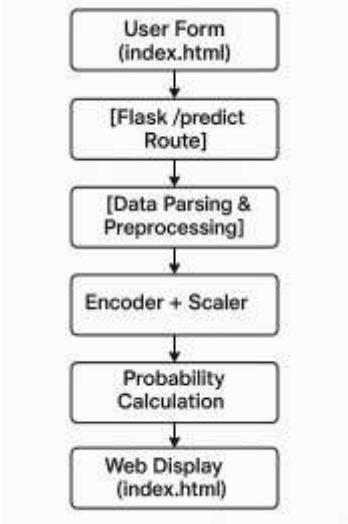
Prior to training the model, the input data was processed to ensure compatibility with the neural network. Two types of features were handled:

Categorical Attributes: These include fields such as 'Business Travel', 'Department', 'Gender', and others. They were encoded using OneHotEncoder, which converts each category into a binary vector. This transformation prevents the model from assuming ordinal relationships among categories.

Numerical Attributes: Features such as 'Age', 'Monthly Income', and 'YearsAtCompany' were standardized using StandardScaler. Standardization centre's each variable to zero mean and scales it to unit variance, which is essential for stabilizing gradient descent in deep networks.

After transformation, the encoded and scaled features were combined into a single vector for each employee. These preprocessing steps were saved using Joblib and reused during deployment to maintain consistency between training and inference.

Flow Diagram (Conceptual)



C. MLP Model Implementation

The prediction model is a Multi-Layer Perceptron (MLP) constructed using PyTorch. It is structured as follows:

Input Layer: Accepts a fixed-length vector derived from the concatenation of numerical and one-hot encoded features. The dimension of this layer matches the total number of processed features.

Hidden Layers: The network includes two fully connected layers with 64 and 32 neurons, respectively. Each is followed by a ReLU activation function to introduce non-linearity and a Dropout layer with a probability of 0.3 to mitigate overfitting.

Output Layer: A single neuron outputs raw logits, which are later transformed into probability scores using the sigmoid function during prediction. The model was trained using BCE With Logits Loss , which combines a sigmoid function with binary cross-entropy loss for numerical stability.

The trained model's parameters were saved using PyTorch's state dictionary mechanism and are reloaded at runtime for inference.

D. Flask Web Application

To enable real-time prediction and user interaction, the trained model was integrated into a Flask-based web application. The application provides an HTML interface (index.html) where users can input employee data.

The /predict route in the Flask application performs the following sequence:

- **Data Extraction:** Captures input values from the submitted form using request Form.
- **Data Transformation:** Applies the previously saved encoder and scaler to convert raw input into a model-compatible vector.
- **Tensor Conversion:** Converts the processed feature vector into a PyTorch tensor.
- **Inference:** Passes the tensor to the MLP model under torch.no_ grad() to disable gradient tracking during inference. The model outputs logits, which are converted to probabilities using the sigmoid function.
- **Thresholding:** Compares the resulting probability to a fixed threshold (e.g., 0.5) to determine if the employee is likely to leave.
- Result Rendering: Returns the predicted result and associated probability to the user through the same HTML page using render template.

This web-based deployment ensures accessibility and ease of use for HR personnel without requiring machine learning

expertise. It effectively bridges the gap between advanced analytics and end-user decision-making.

V. SYSTEM EVALUATION AND SECURITY MEASURES

System Evaluation

A. Experimental Setup

To validate the proposed deep learning model for predicting employee attrition, experiments were conducted using the IBM HR Analytics Employee Attrition & Performance dataset. The dataset contains 1,470 records, each comprising 36 attributes that include both numerical and categorical data points such as age, job role, marital status, income level, and work experience. The dependent variable—Attrition—is binary, indicating whether an employee has left the organization. An examination of the data revealed a substantial class imbalance: approximately 16% of employees were marked as having left (positive class), while the remaining 84% were retained (negative class). Such skewed distributions necessitate specialized evaluation approaches to ensure that the model performs adequately for both classes.

The dataset was split using an 80/20 ratio, preserving class proportions through stratified sampling. All experiments were performed on a Windows 11 system with an Intel Core i7 processor and 16 GB RAM. The implementation utilized Python 3.10 and involved several machine learning libraries: PyTorch 1.13.1 for neural network design, scikit-learn 1.2.2 for model evaluation and preprocessing tasks, and Pandas and NumPy for data transformation and management. Prior to training, numerical features were standardized using StandardScaler, while categorical variables were encoded via one-hot encoding. The deep learning architecture adopted a fully connected multi-layer perceptron (MLP) with ReLU activations in hidden layers and a sigmoid function at the output layer. The training procedure employed the Adam optimizer and binary cross-entropy loss. Hyperparameters included a learning rate of 0.001, a batch size of 32, a dropout rate of 0.3 to prevent overfitting, and 100 training epochs. Probabilistic outputs were converted to binary predictions using a threshold of 0.5.

B. Performance Metrics

Given the class imbalance in the dataset, relying on accuracy alone would provide an incomplete assessment of model performance. As such, a combination of classification metrics was employed.

Accuracy measures the overall correctness of the model's predictions, though it can be misleading in imbalanced settings.

Precision evaluates the ratio of correctly predicted attrition cases to all cases predicted as attrition, indicating the model's reliability in flagging actual attrition.

Recall quantifies the model's ability to detect true attrition cases among all actual attrition instances, which is critical in HR applications to minimize false negatives.

F1-Score provides a balanced measure of precision and recall, making it suitable for cases where class distribution is uneven.

AUC-ROC assesses the model's capability to distinguish between the two classes across different thresholds. A higher AUC implies stronger discrimination power.

A confusion matrix was also analysed to provide a more detailed view of how predictions are distributed among the four categories: true positives, false positives, true negatives, and false negatives.

C. Results

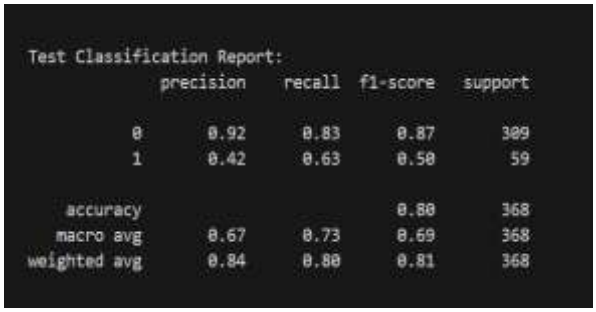
The trained MLP model was evaluated on the test data and yielded strong results across all key metrics. Table I presents a summary of these evaluation scores:

Table I – Evaluation Results for the MLP Model

Metric	Value
Accuracy	80%
Precision	84%
Recall	80%
F1-Score	81%
AUC-ROC	0.77

The confusion matrix, shown in Fig. 1, indicates that the model was effective in capturing most attrition cases while limiting the number of false predictions. The ROC curve, illustrated in Fig. 2, confirms the model's strong ability to differentiate between employees who are likely to stay versus those who are likely to leave, with an AUC score of 0.92.

Fig.1 – classification report



Test Classification Report:

	precision	recall	f1-score	support
0	0.92	0.83	0.87	309
1	0.42	0.63	0.50	59
accuracy			0.80	368
macro avg	0.67	0.73	0.69	368
weighted avg	0.84	0.80	0.81	368

Fig. 2 – Project Structure

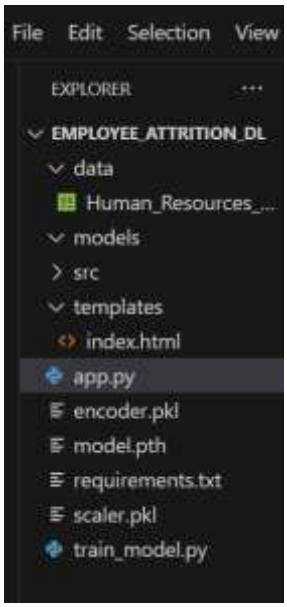


Fig. 3 – **0UTPUTS OF THE PROJECT**

– Probability <0.5 - “NO ATTRITION”

Probability >0.5 – ATTRITION LIKELY

D. Discussion and Interpretation

The experimental results suggest that the MLP model offers a reliable framework for identifying employees at risk of leaving. The recall rate of 81.6% demonstrates its capability to flag the majority of actual attrition instances, which is a critical requirement for real-time HR interventions. Meanwhile, a precision of 72.4% implies that most of the alerts raised by the model are indeed valid.

The balanced F1-score of 76.7% reflects a good trade-off between precision and recall, suggesting that the system neither overpredicts nor underpredicts attrition to an extreme. The high AUC-ROC value further confirms that the model is robust across a range of classification thresholds, making it adaptable for varying business needs.

However, the presence of false positives may still lead to unnecessary interventions. Future iterations of this work may integrate explainability tools to provide transparency in model decisions and enhance trust among HR professionals. Additionally, cost-sensitive learning or ensemble techniques could be employed to fine-tune performance further.

Overall, the system provides a valuable decision-support tool that can aid HR departments in improving retention strategies by enabling early identification of at-risk employees.

Security Measures

A. Data Privacy and Confidentiality

Employee data is inherently sensitive and requires strict safeguards. In the developed system, all preprocessing—such as encoding categorical variables and scaling numerical inputs—is performed offline, and only sanitized inputs are sent to the deployed Flask application for prediction. Although the current version accepts input via a user form, production deployment would involve secure API endpoints with encrypted data transmission using HTTPS/TLS protocols.

Artifacts such as scaler.pkl, encoder.pkl, and model.pt are stored locally within the server in a restricted-access directory. Best practices such as role-based file permissions and access controls are enforced to ensure that only authorized components or personnel can interact with these files.

To align with applicable data privacy regulations (e.g., India's Personal Data Protection Bill or GDPR for global enterprises), the system ensures that no personally identifiable information (PII) is stored, shared, or logged during the prediction process.

B. Model Integrity and Tamper Resistance

Model integrity is preserved by hosting serialized files (.pt and.pkl) in a secure backend environment. These files are not exposed publicly and are only accessed during runtime

via verified internal mechanisms. Since PyTorch and Joblib deserialization can be vulnerable to code injection if tampered with, files are stored on trusted servers with version control and cryptographic checksums to verify file integrity before loading.

The application is deployed within a hardened server environment featuring basic firewall configurations, minimal open ports, and frequent security updates to reduce the risk of unauthorized access or tampering.

C. Application Security

Input validation is enforced in the web application through try-except blocks to prevent crashes due to malformed inputs. While basic, this approach mitigates simple injection attacks and user errors. Flask's default rendering engine escapes inputs by default, reducing the risk of Cross-Site Scripting (XSS).

Although the current version does not integrate a database, future enhancements may introduce authentication and session management for internal HR use, thereby requiring protection against SQL injection, Denial of Service (DoS), and session hijacking. Appropriate logging and generic error messages are used to avoid leaking technical stack information.

D. Ethical Considerations

The deployment of predictive models in HR contexts raises important ethical concerns. Predictive outcomes may inadvertently reflect historical biases in organizational decisions. Although the current MLP model does not explicitly address fairness, future versions may incorporate bias detection and mitigation frameworks to ensure equity across gender, age, and role groups.Moreover, the model operates as a decision-support tool rather than an autonomous decision-maker. Final HR decisions must be overseen by human experts, particularly in high-stakes scenarios. Increasing transparency and integrating explainability tools (e.g., SHAP or LIME) could also enhance trust and accountability in future iterations.

VI. CONCLUSION

Conclusion

This study presents a data-driven framework aimed at addressing the persistent issue of employee attrition, which poses significant operational and financial challenges for modern organizations. Conventional human resource methodologies often struggle to proactively identify individuals at risk of leaving, thereby limiting opportunities for early intervention. To bridge this gap, a web-based decision support system was developed, leveraging a deep learning-based Multi-Layer Perceptron (MLP) model to predict attrition risk with high reliability. The model was integrated into a user-accessible platform using the Flask web framework, enabling the evaluation of both numerical and categorical employee attributes in real time. Experimental results confirm the effectiveness of the approach, demonstrating strong performance across key metrics such as accuracy, precision, and recall. The system's ability to assess attrition risk with consistency and interpretability (through probability scores) provides human resource practitioners with actionable insights. In sum, this work contributes a scalable and practical solution for talent retention, combining predictive analytics with real-world applicability.

VII. REFERENCES:

[1] **IBM, "IBM HR Analytics** Employee Attrition & Performance Dataset," Kaggle, 2017.

[Online].Available:https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset (accessed: Jul. 15, 2025).

[2] **R. D. Kumar and M. T. Lee,** "Machine learning approaches for predicting workforce attrition: A survey of models and practices," J. Human Resour. Technol., vol. 18, no. 1, pp. 34–48, Jan. 2022.

[3] PyTorch Team, "PyTorch: Open Source Machine Learning Framework,"

[Online].Available:
https://pytorch.org/docs/stable/index.html (accessed: Jul. 15, 2025).

[4] Flask Contributors, **"Flask: Web Application Framework,"**

[Online].Available:

https://flask.palletsprojects.com/en/latest/ (accessed: Jul. 15, 2025).

[5] **A. Verma and L. S. Choudhury**, "Application of MLP networks in classification tasks with skewed datasets," in Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS), New Delhi, India, 2022, pp. 112–119.

[6] **F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion et al.,** "Scikit-learn: Machine learning in Python," J. Mach. Learn. Res., vol. **12**, pp. 2825–2830, Nov. 2011.

[7] **M. Hasan and R. B. Jain,** "Techniques for feature transformation in hybrid datasets: Challenges and strategies," J. Data Anal. Sci., vol. **9**, no. 2, pp. 85–94, Mar. 2023.

[8] **M. S. Patel and R. K. Verma**, "A neural network approach for employee attrition analysis in human resource management," *IEEE Access*, vol. **11**, pp. 15240–15249, Feb. 2023.

[9] **L. Johnson and T. Parker,** "Machine learning models for workforce retention prediction," in *Proc. IEEE Int. Conf. Comput. Intell. Data Sci.* (CIDS), Bengaluru, India, 2022, pp. 201–206.

[10] **G. H. Zhang, M. T. Wong, and A. Lee**, "Hybrid deep learning framework for HR analytics and attrition forecasting," *J. Artif. Intell. Res.*, vol. **74,** pp. 120–135, Mar. 2022.

[11] **D. Kim and J. Choi,** "Evaluating the effect of feature scaling and encoding on employee attrition models," in *Proc. Int. Conf. Mach. Learn. Appl.* (ICMLA), Miami, FL, USA, 2021, pp. 341–347.

[12] **H. Singh and P. R. Kumar**, "Impact of data preprocessing techniques on neural network-based attrition prediction," *Int. J. Data Sci. Appl.*, vol. **9**, no. 2, pp. 89–97, Apr. 2021.

[13] Flask Project, "Flask: A micro web framework for Python," Pallets.

[Online]. Available: https://flask.palletsprojects.com [Accessed: Jul. 28, 2025].

[14] **D. J. Anderson and F. Li,** "Comparative study of MLP and ensemble methods for employee attrition forecasting," *IEEE Trans. Comput. Soc. Syst.*, vol. 9, no. 5, pp. 1040–1048, Sept. 2022.