Employee Attrition Prediction Pipeline

This project outlines a comprehensive **Employee Attrition Prediction Pipeline** using Python. It covers essential steps from data preparation to model deployment, leveraging advanced machine learning techniques and MLflow for experiment tracking and model registration. The system includes a **Streamlit frontend** for user input and a **Flask backend** for serving predictions, forming a complete full-stack application for real-time attrition prediction.

Data Loading and Initial Inspection

Load Dataset

The HR attrition dataset is loaded using pandas, initiating the data pipeline.

Data Overview

Initial checks include shape, data types, and descriptive statistics to understand the dataset's structure.

Duplicate Check

Ensuring no duplicate entries exist, guaranteeing a clean starting point for analysis.

Dataset Overview

The dataset contains **1,470 employee records** with a mix of **numerical, categorical, and engineered features**. It is structured to capture factors that influence employee attrition and includes both original HR attributes and derived metrics for predictive modeling.

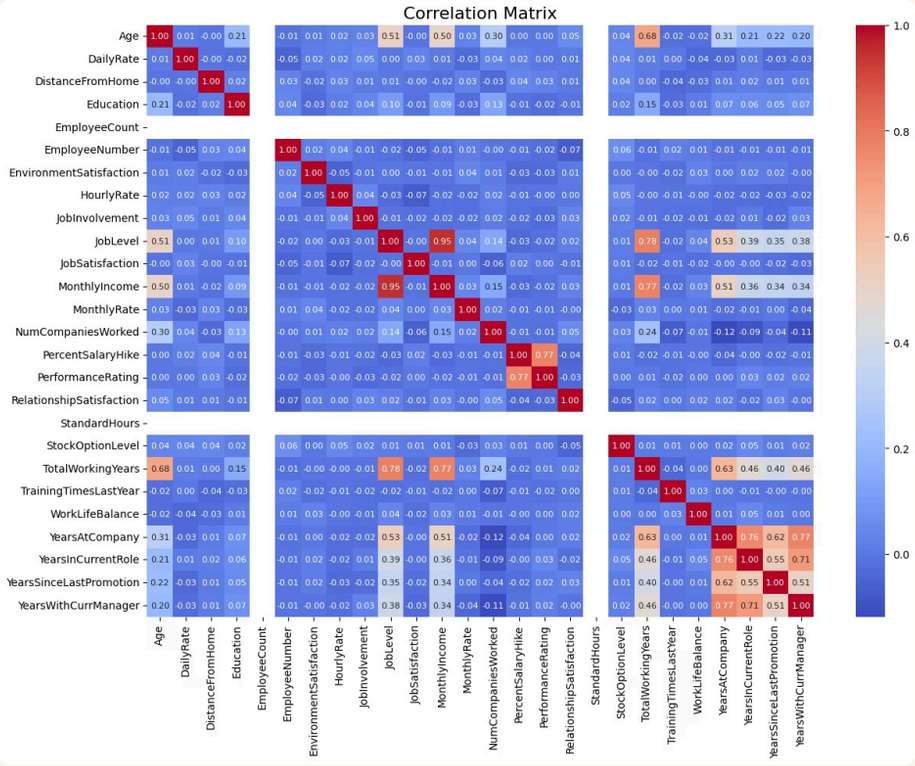
Key Features

- **Demographics & Personal Info:**
  `Age`, `Gender_Male`, `MaritalStatus_Married`, `MaritalStatus_Single`
- **Employment & Job Details:**
  `JobRole_*` (multiple roles), `Department_*`, `BusinessTravel_*`, `JobLevel`, `YearsAtCompany`, `YearsInCurrentRole`, `YearsWithCurrManager`, `OverTime_Yes`
- **Compensation & Performance:**
  `DailyRate`, `HourlyRate`, `MonthlyIncome`, `MonthlyRate`, `PercentSalaryHike`, `StockOptionLevel`, `PerformanceRating`
- **Work & Satisfaction Metrics:**
  `EnvironmentSatisfaction`, `JobInvolvement`, `JobSatisfaction`, `RelationshipSatisfaction`, `WorkLifeBalance`, `TrainingTimesLastYear`
- **Career Progression & Experience:**
  `TotalWorkingYears`, `NumCompaniesWorked`, `YearsSinceLastPromotion`
- **Engineered Features:**
  `TenureRatio`, `IncomePerYear`, `IncomeToAge`, `YearsSincePromotionRatio`, `WorkLifeScore`, `IncomePerYearExperience`
- **Target Variable:**
  `Attrition` (1 = Employee left, 0 = Employee stayed)

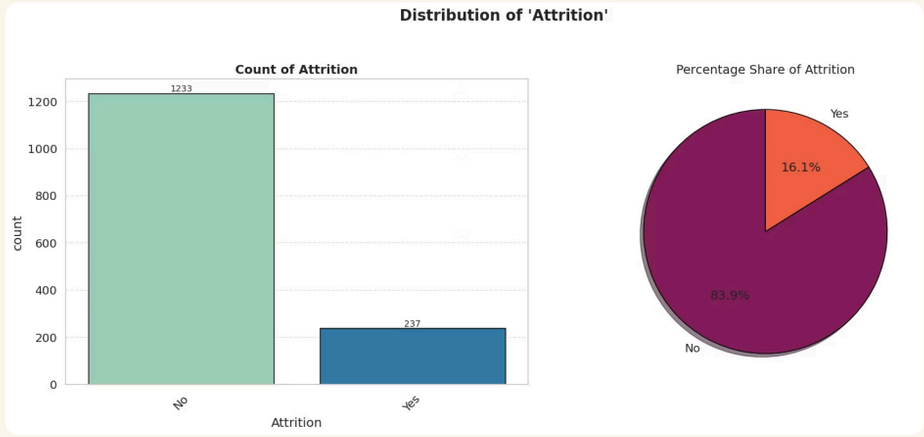
This combination of **raw HR features** and **engineered metrics** enables the XGBoost model to effectively predict employee attrition.

Some visualizations of the data

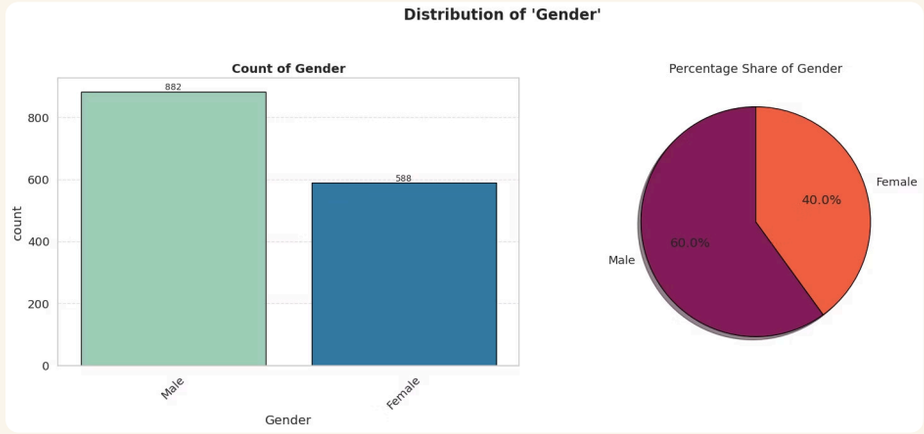
Correlation Matrix



Distribution of Attrition



Distribution of Gender



Exploratory Data Analysis (EDA)

Categorical Analysis

Examines distributions of categorical features to identify class imbalances and rare categories, crucial for model training.

Numerical Feature Visualization

Visualizes numerical data to detect skewness, outliers, and understand variable distributions.

Correlation Analysis

Utilizes heatmaps and pairplots to uncover highly correlated features and relationships with attrition.

Feature Preprocessing: Cleaning and Outlier Handling

Removing Useless Features

Columns with zero predictive power, such as 'EmployeeCount', 'Over18', 'StandardHours', and 'EmployeeNumber', are removed to streamline the dataset.

String Normalization

Ensures consistency by stripping extra spaces from string values, preventing incorrect duplicates during encoding.

Outlier Detection & Capping

IQR and Z-score methods identify extreme values. Instead of removal, outliers are capped using the 3-sigma rule to stabilize model training and reduce variance.

Advanced Feature Engineering

New features are engineered to help the ML model learn complex patterns, significantly boosting performance.

Income Per Year Experience

Ratio of monthly income to total working years.

Tenure Ratio

Years at company relative to total working years.

WorkLifeScore

Interaction between WorkLifeBalance and JobSatisfaction.

Years Since Promotion Ratio

Years since last promotion relative to years at company.

Encoding and Target Transformation

One-Hot Encoding

Categorical features are converted into a numeric format using one-hot encoding, essential for machine learning models.

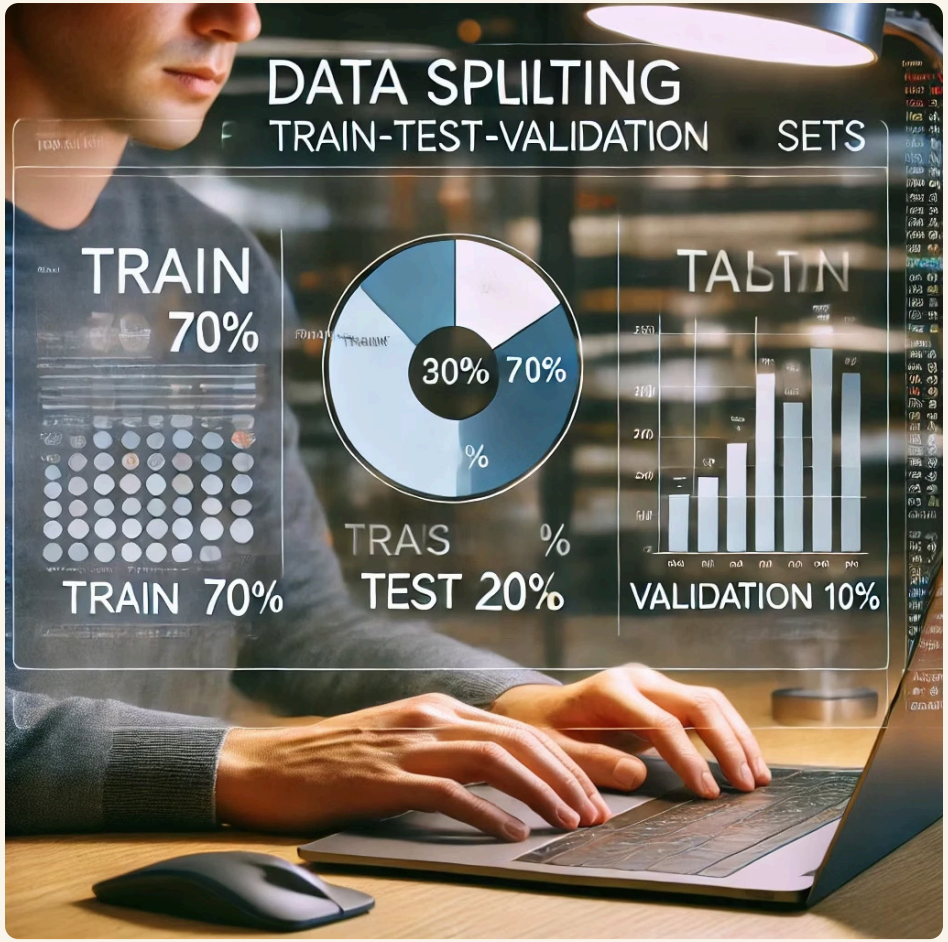
Ordinal Encoding

Specific ordinal columns are transformed into integer representations, preserving their inherent order.

Target Encoding

The 'Attrition' target variable is mapped to binary labels ('Yes': 1, 'No': 0) for classification.

Train-Test Split and Imbalance Handling



Stratified Split

The dataset is split into training and testing sets while preserving the original attrition ratio, ensuring representative samples.

SMOTE for Imbalance

Synthetic Minority Over-sampling Technique (SMOTE) is applied to generate synthetic samples for the minority class ('Yes' attrition), effectively handling class imbalance and improving model fairness.

XGBoost Model Training and Evaluation

Model Initialization

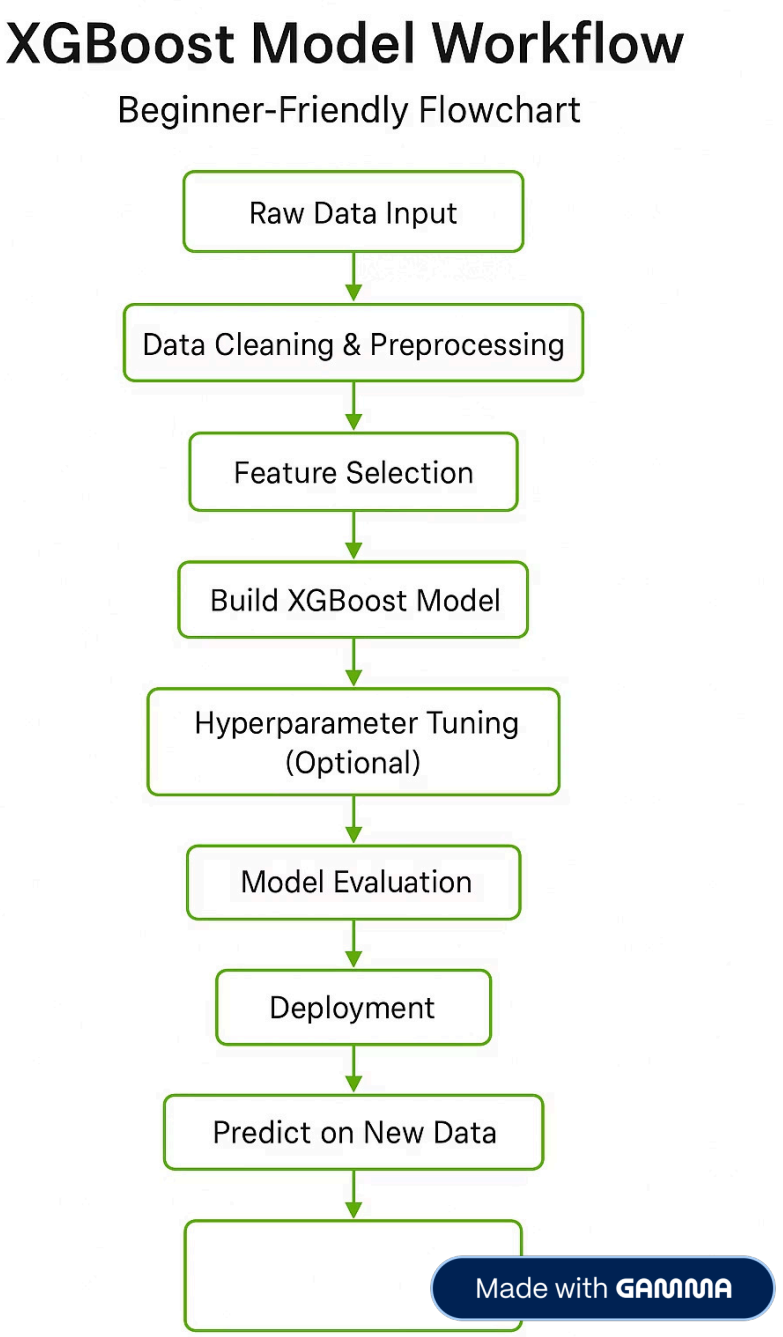
An XGBoost Classifier is initialized with optimized hyperparameters, including 500 trees, max_depth=20, and a learning rate of 0.05.

Balanced Training

The model is trained on the resampled data, using `scale_pos_weight` to address class imbalance directly during training.

Performance Metrics

Model performance is evaluated using accuracy, precision, recall, and F1-score, providing a comprehensive view of its effectiveness.



MLflow Experiment Tracking and Model Registration

Experiment Tracking

MLflow is used to log key parameters, metrics (accuracy, recall, F1, AUC), and artifacts like confusion matrices and classification reports.

Model Saving

The trained XGBoost model is saved using `joblib`, creating a production-ready artifact.

Model Registration

The model is registered in the MLflow Model Registry and transitioned to the "Production" stage, ready for deployment.

Deployment Readiness

Saving Cleaned Dataset

The preprocessed and cleaned dataset is saved to a CSV file, ensuring it can be reused for future deployment and inference without repeating preprocessing steps.

This final step ensures that the entire pipeline is robust and ready for operational use, providing a seamless transition from development to production.

Frontend: Streamlit App

This section details the interactive user interface built using Streamlit, designed to empower HR managers with real-time employee attrition predictions. It serves as the primary point of interaction for business users to leverage the insights generated by the machine learning pipeline.

01

Collect Employee Details

The Streamlit application presents an intuitive form where HR managers or users can easily input essential employee data points. This includes critical numeric attributes like age, monthly income, job level, job satisfaction, and work-life balance, as well as categorical factors such as business travel and overtime status.

- Numeric: Age, Monthly Income, Job Level, Job Satisfaction, Work Life Balance
- Categorical: Business Travel, OverTime

02

Prepare and Send Data to Backend

Upon user submission, the app efficiently converts the collected input data into a standardized JSON payload. This structured data is then securely transmitted as a POST request to the Flask backend API's `/predict` endpoint, ensuring seamless communication with the trained machine learning model.

03

Display Prediction Results

After receiving the prediction from the backend, the Streamlit app clearly displays the model's output to the user. The primary result is a direct prediction: either "Will Leave" or "Will Stay." Additionally, a confidence score (probability of attrition) is provided, offering greater transparency into the model's certainty.

- Prediction: “Will Leave” or “Will Stay”
- Probability of attrition (confidence score)

This user-friendly interface provides a powerful, interactive dashboard for HR professionals to quickly assess employee attrition risk, facilitating proactive talent management and retention strategies.

Backend: Flask API (app.py)

The Flask API acts as the robust prediction engine, serving the trained XGBoost model to external applications. It is designed to receive employee data, process it according to the model's requirements, and return accurate attrition predictions. This backend service is critical for transforming our machine learning model into a functional, deployable asset.

`/predict` Endpoint

This `POST` endpoint accepts employee data in JSON format, constructs the necessary feature vector, and invokes the trained model to generate predictions and associated probabilities. It returns results as JSON.

`/` Health Check

A simple `GET` endpoint that confirms the API is operational, providing essential service availability monitoring for the deployment.

Prediction Workflow

01

Receive JSON Input

The API receives employee demographic and work-related data as a JSON payload from the Streamlit frontend.

02

Feature Vector Construction

User-provided fields are transformed and expanded into the comprehensive 57-feature vector expected by the XGBoost model. Missing fields are intelligently populated with default values.

03

Model Inference

The constructed feature vector is fed into the loaded XGBoost model to predict attrition (0 for 'Will Stay', 1 for 'Will Leave') along with a probability score.

04

Return JSON Response

The prediction and confidence score are packaged into a JSON response and sent back to the frontend for display to the user.

Key Features

- **Robust Data Handling**: Automatically manages missing input fields by filling in pre-defined defaults, ensuring smooth model inference even with incomplete user data.
- **Categorical Encoding**: Seamlessly supports one-hot encoding for categorical features within the incoming data, aligning with the model's preprocessing requirements.
- **CORS Enabled**: Configured with Cross-Origin Resource Sharing (CORS) to facilitate secure communication and data exchange with the Streamlit frontend.

Together, the Streamlit frontend and Flask API create a powerful full-stack prediction system, empowering HR managers with real-time, actionable insights into employee attrition risks.