Capstone Project - HR Analytics

Submitted in partial fulfilment of the requirements for the degree of

Post Graduate Diploma in Data Engineering

Group 10

Gaurav Ranjan(G23AI1011)

Swetlina Nayak(G23AI1047)

Somalaya Mondal(G23AI1042)

Vishal Kumar(G23AI1071)

Pooja Yadav(G23AI1026)

Indian Institute of Technology Jodhpur

Capstone Project

Trimester-3 (July 2024)

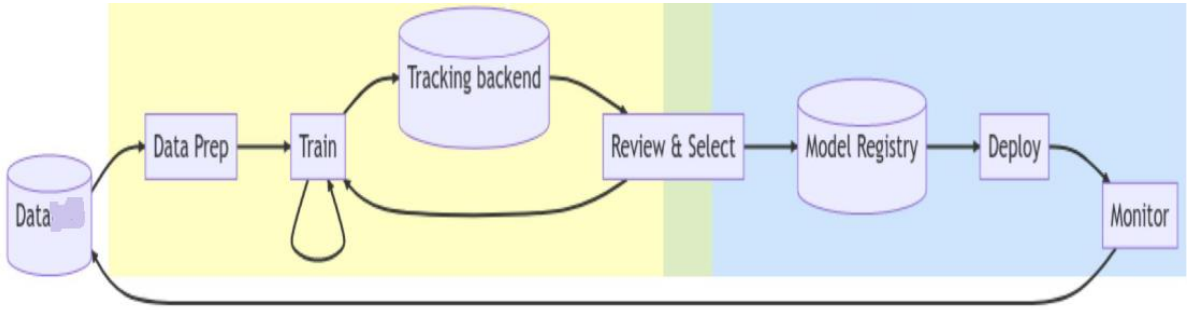
Scope of Document

This document outlines the high-level design for implementing a machine learning pipeline to predict employee salaries based on various features in an HR dataset. The focus is on reading data from CSV files, performing exploratory data analysis (EDA), feature engineering, training and evaluating machine learning models, and managing the entire machine learning lifecycle using MLOps principles.

Problem Statement

Employee salary prediction is a crucial task for HR departments to ensure fair compensation practices, budget planning, and employee satisfaction. The goal of this project is to develop a machine learning model that can accurately predict the salary of employees based on features such as age, gender, department, years of experience, education level, performance rating, work-life balance, and other relevant attributes**.**

Architectural Diagram



High Level Design

1. **Data Ingestion**

   1.1 Read Data from CSV: Implement mechanisms to read and load data from CSV files into the system.

2. **Exploratory Data Analysis (EDA) and Data Preprocessing**

   2.1 Data Exploration: Perform initial data exploration to understand the data structure, distribution, and patterns.

   2.2 Check for Missing Values: Identify any missing values in the dataset.

   2.3 Fill or Drop Missing Values: Decide on strategies to handle missing values, either by filling them with appropriate values or dropping the affected rows/columns.

   2.4 Encode Categorical Variables: Convert categorical variables into numerical format using techniques like one-hot encoding or label encoding.

3. **Feature Engineering**

   3.1 Separate Features and Target: Distinguish between feature variables (independent variables) and the target variable (dependent variable).

4. **Data Splitting**

   4.1 Separate Train and Test Data: Split the dataset into training and testing sets to evaluate model performance.

5. **Model Development**

   5.1 **Hyperparameter Tuning**: Optimize the hyperparameters of machine learning models to improve performance.

6. **Model Evaluation**

   6.1 Model Evaluation: Assess the performance of different models using appropriate metrics.

   6.2 Choose the Best Model: Select the best-performing model based on evaluation metrics.

7. **Model Management**

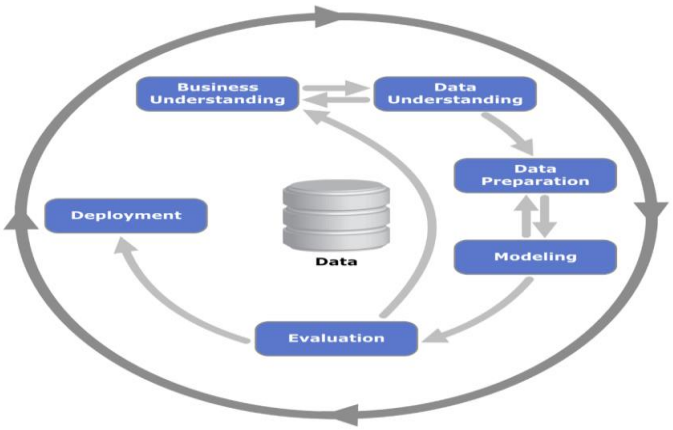
   7.1 ML Tracking: Track experiments, model versions, and performance metrics to manage the machine learning lifecycle.

   7.2 Register the Best Model: Register the best model in a model registry for deployment and future reference.

8. **Model Deployment**

   8.1 Create a Streamlit app: To interact with the trained model. The app should allow users to input features and get salary predictions. Display relevant model performance metrics **and** visualizations in the app

   8.2 Deploy the model : Deploy the best model using Hugging Face Spaces. Ensure the deployed model is accessible via an API for inference.



Technical Stacks:

1. **Data Ingestion and EDA**
    - **Dependencies**: Libraries for data handling (e.g., pandas for Python), visualization tools.
    - **Integration Points**: Functions for reading data, handling missing values, and encoding variables.
2. **Feature Engineering and Data Splitting**
    - **Dependencies**: Machine learning libraries (e.g., scikit-learn).
    - **Integration Points**: Methods for feature extraction, train-test splitting.
3. **Model Development and Evaluation**
    - **Dependencies**: Machine learning frameworks (e.g., TensorFlow, PyTorch, scikit-learn).
    - **Integration Points**: Hyperparameter tuning tools, model evaluation metrics.
4. **Model Management**
    - **Dependencies**: MLOps tools (e.g., MLflow, DVC).
    - **Integration Points**: Experiment tracking, model registry systems.
5. Deployment
    - **Dependencies**: Streamlit (e.g., MLflow).
    - **Integration Points**: Hugging Face.

Future Considerations

1. **Scalability**: Ensure the MLOps pipeline can scale with increasing data volumes and complexity.
2. **Performance Optimization**: Continuously optimize the performance of the entire pipeline, from data ingestion to model deployment.
3. **Security and Compliance**: Implement robust security measures and ensure compliance with relevant regulations for data handling and model deployment.