Project Report - Phase 3: Model Development

Author: Aryan Rai **Date:** August 6, 2025 **Project:** Employee Performance Prediction using a Machine Learning Approach

3.1. Model Selection and Rationale

Model Selection

- **Classification Algorithms: The project will use classification models to predict performance categories (e.g., high, medium, low).**
- **Algorithms Considered:**
    - **Random Forest: Selected for its high accuracy and robustness.**
    - **Logistic Regression: Used as a simple and effective baseline model.**
    - **Gradient Boosting (e.g., XGBoost): Considered for its potential for top-tier performance.**

Rationale

- **Jupyter Notebook (Employee_Productivity_Analysis.ipynb): This file is where various models will be trained and evaluated.**
- **Evaluation Metrics: The best model will be chosen based on its performance on key metrics like accuracy, precision, and recall.**
- **Final Model (best_model.pkl): The best-performing model is saved as a .pkl file. This file is then loaded by the app.py code for making predictions in the web application.**
- **Dependencies (requirements.txt): The required libraries like scikit-learn are listed here, ensuring the models can be trained and used correctly.**
- **The final selection ensures the project has a reliable and well-performing model ready for deployment.**

3.2. Training and Testing Methodology

The preprocessed dataset was split into a training set (80% of the data) and a testing set (20% of the data). This separation is crucial to ensure that the models are evaluated on data they have not seen before, providing an unbiased assessment of their performance. Each of the three models was trained on the same training data.

3.3. Model Performance Evaluation

The following metrics, commonly used for classification problems, will be used to evaluate the model:

- **Accuracy: This metric measures the percentage of correct predictions the model makes.**
- **Precision: This metric focuses on the positive predictions made by the model. It answers the question: "Of all the employees the model predicted as high-performers, how many were actually high-performers?"**
- **Recall: This metric focuses on the actual positive cases. It answers the question: "Of all the actual high-performing employees, how many did the model correctly identify?"**
- **F1-Score: This is the harmonic mean of Precision and Recall. It provides a balanced measure of the model's performance.**
- **Confusion Matrix: A table that visualizes the performance of the model by showing the number of true positive, true negative, false positive, and false negative predictions.**

The model with the best overall performance, considering these metrics, will be chosen as the final model and saved as best_model.pkl for deployment. This robust evaluation process ensures that the model is not only accurate but also reliable in a real-world scenario.

3.4. Model Selection and Finalization

The final step in the model development process, as documented in the **Employee_Productivity_Analysis.ipynb** notebook, is to select and finalize the best-performing model. This phase is crucial for transitioning the project from an experimental stage to a deployable solution.

- **Selection:** Based on the performance metrics (accuracy, precision, recall, and F1-score) from the model evaluation phase, the algorithm that demonstrated the highest and most balanced performance on the testing dataset is chosen. This is the model that is best suited to make reliable predictions in a real-world scenario.
- **Finalization:** Once selected, the final model is serialized and saved to a file using Python's pickle library. This is the **best_model.pkl** file that is mentioned in the previous errors and is required for the web application to function. This process converts the trained model object into a binary file that can be easily loaded later without needing to retrain it.
- **Integration with Application:** The **app.py** file is coded to load this best_model.pkl file at startup. This ensures that the Flask web application has direct access to the trained model's logic and can use it to process new employee data entered by a user and provide an instant performance prediction. The requirements.txt file guarantees

that the necessary libraries to load and use this model are present on the deployment server.

This finalization process ensures that the project has a stable, well-evaluated model ready for deployment, making it the core component of the employee performance prediction application.