INTELLIGENT JOB RECOMMENDATION ENGINE

Phase 3 - Supervised Model Training and Persistence

Member 1 : Ashfaq Ali Anees Ali (40406560)
Member 2 : Naveen Kumar Raju (87474061)
University of Europe for Applied Sciences
Date: 30-11-2025

Objective

Use the Hybrid Score and detailed skill data from Phase 2 to train a supervised model that predicts how likely a candidate's resume is to match a given job, then save the top-performing model so it is ready for deployment in production.

---

1. Supervised Training Methodology: Synthetic Labeling

The process required creating training data by using the reliable Hybrid Score (the output of Phase 2) as a synthetic label generator.

- **Training Data Source:** Synthetic labels were generated from the Hybrid Score Matrix. A specific threshold was applied to this score:
    - Pairs scoring above the threshold on the Hybrid Score were labeled 1 (Good Match).
    - Pairs scoring below the threshold were labeled 0 (Bad Match).
- **Data Volume:** The training process utilized a balanced dataset, ensuring unbiased learning.

- **Input Features: 70 Binary Features** These 70 yes/no features—35 tracking specific skills on the resume and 35 for the job requirements—come straight from the explicit skill codes. They give the classifier the raw data it needs to spot patterns in skill overlaps that signal a "Good Match."

---

2. Model Evaluation and Selection

Two common machine-learning models, Logistic Regression and Random Forest, were tested and compared to the original Hybrid Similarity Model used as a baseline. The comparison focused on two key metrics: F1-score, which balances precision and recall, and ROC-AUC, which shows how well each model can rank good matches higher than bad ones.

Model Performance Metrics

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Logistic Regression	1.0000	1.0000	1.0000	1.0000	1.0000
Random Forest	1.0000	1.0000	1.0000	1.0000	1.0000
Hybrid Similarity	0.4737	0.0000	0.0000	0.0000	1.0000

Note: The Hybrid Similarity Model is included as a ranking baseline (ROC-AUC 1.0000) but its classification metrics (F1-Score 0.0000) are not directly comparable to the optimized classifiers.

Selection Rationale

The Logistic Regression model was selected as the final predictive engine due to its perfect performance and superior efficiency:

- **Perfect Scores (1.0000 F1-Score and ROC-AUC):** The results show that “Good Match” and “Bad Match” pairs can be separated perfectly using just the 70 explicit skill features, with a straight-line decision boundary in feature space. This means the skill features and the synthetic labels work extremely well together and strongly validate the quality of both the feature engineering and the automatic labeling approach.
- **Efficiency and Simplicity:** Because Logistic Regression, the simpler linear model, reached the same perfect scores as the more complex Random Forest, it is the most practical model to put into production. It runs much faster, is cheaper to train and serve, and uses fewer computational resources than an ensemble of decision trees, while delivering identical predictive performance in this case.

---

3. Model Persistence for Deployment

The optimal model was re-trained on the full dataset and serialized to ensure a robust, portable object ready for the final application (Phase 4).

- **Final Training:** The Logistic Regression model was retrained on **100% of the available training data**.
- **Serialization:** The Python joblib library was used to save two crucial components for deployment:
    1. **final_logistic_regression_model.joblib:** The complete trained Logistic Regression model is saved as a single object that can be easily loaded into the recommendation system. This model object is then used directly to generate probability predictions for new resume-job pairs, making it simple and efficient to integrate into the final application.
    2. **feature_columns.joblib:**The ordered list of all 70 feature names (like R_ACCT for resume accounting skill, J_ACCT for job accounting requirement, and so on) is a must-have file to keep the model working correctly. When the live application processes new resumes and jobs, it has to feed those features into the saved model in exactly the same column order used during training. Otherwise, you risk the classic "feature mismatch" error that breaks ML deployments and leads to garbage predictions.

Conclusion

Phase 3 proved the supervised learning approach works perfectly. The Logistic Regression classifier nailed a flawless F1-score of 1.0000, making it the heart of the prediction system. With the final trained model and all key files saved, everything is set for Phase 4: deployment and integration into the live application.