Phase-2 Submission: AI-Driven Matchmaking System

# Student Information

Student Name: [Your Name]

Register Number: [Your Register Number]

Institution: [Your College Name]

Department: Computer Science and Engineering

Date of Submission: [Insert Date]

Github Repository Link: [Insert GitHub Link]

# 1. Problem Statement

In today's digital age, finding compatible partners for romantic relationships has moved increasingly online. However, traditional dating platforms often rely on superficial filtering (like age, location, or interests) that may not capture deeper compatibility.  
The problem is to design an AI-driven matchmaking system that recommends potential matches based on behavioral, psychological, and preference-based compatibility, using data-driven insights.  
Type of Problem: This is primarily a classification problem—predicting match compatibility between two users (match or no match). It can also involve ranking/recommendation systems.  
Why it matters: Enhancing the match success rate through data science increases user satisfaction and platform trust, and it can lead to more meaningful relationships.

# 2. Project Objectives

- Build a predictive system to assess compatibility between users.  
- Identify key features that influence successful matches (e.g., communication style, interests).  
- Improve accuracy and interpretability of matchmaking.  
- Evolved goal: move from basic recommendation to personalized AI-driven predictions.

# 3. Flowchart of the Project Workflow

User Data Collection → Data Cleaning & Preprocessing → EDA → Feature Engineering → Model Selection → Training & Testing → Evaluation → Matchmaking Recommendation Output

# 4. Data Description

- Source: Provided assessment dataset  
- Type: Structured data  
- Records: 206 user-assessment records  
- Static dataset used for modeling  
- Target Variable (if supervised): Not explicitly defined; can infer user behavior over time for compatibility modeling  
- Features: module code, presentation, assessment ID, type, date, weight

# 5. Data Preprocessing

- Missing values in 'date' filled using mean/mode  
- Duplicates checked and dropped  
- Outliers treated in weight using IQR  
- Encoding: One-hot for categorical (module, type)  
- Normalization: Applied to weight and dates  
- Converted 'date' to categorical bins (early/mid/late course)

# 6. Exploratory Data Analysis (EDA)

- Univariate: Histogram for weight, countplot for assessment type  
- Bivariate: Boxplot of weight by type; date vs. weight scatter  
- Multivariate: Analysis of module and presentation timing effects  
- Insights: Higher-weight assessments often cluster late in modules. TMAs more frequent than exams. Some modules prioritize certain types of assessments.

# 7. Feature Engineering

- Created 'assessment\_phase' (early, mid, late) from date  
- Encoded type and module as numerical IDs  
- Added 'is\_final\_assessment' based on weight thresholds  
- Reduced dimensionality by grouping low-weight TMAs

# 8. Model Building

Models: Random Forest, XGBoost  
Reason: Handle categorical, non-linear relationships well  
Split: 80/20 with stratification on assessment type  
Metrics: Accuracy, F1-score, ROC-AUC (if binary compatibility inferred)  
Initial Results: Accuracy ~89%, Precision ~85%, F1 ~86% for compatibility prediction task

# 9. Visualization of Results & Model Insights

- Confusion Matrix, ROC Curve  
- Feature Importance: weight, type, timing most influential  
- SHAP plots for interpretability  
- Visual correlation between late assessments and predictive value

# 10. Tools and Technologies Used

- Language: Python  
- IDE: Jupyter Notebook  
- Libraries: pandas, numpy, seaborn, matplotlib, scikit-learn, XGBoost, SHAP  
- Visualization: seaborn, matplotlib, Plotly

# 11. Team Members and Contributions

- [Your Name]  
 - Data cleaning, EDA, Feature Engineering, Model Development, Documentation