**Match Prediction Project**

**1. Data Collection**

**Google Form Creation**

To gather training data, we designed a Google Form consisting of 12 multiple-choice questions. Each question had predefined options to ensure uniformity and prevent inconsistent or noisy inputs.

**Data Gathering Strategy**

The form was shared across various platforms including personal contacts, peer circles, and social media networks. Our objective was to gather responses from a wide and diverse group of participants to create a balanced dataset.

**Challenges and Resolutions**

Initially, our target was to collect at least 200 responses. However, we received only 149 within the first phase due to a limited outreach. To resolve this, we extended the submission period and re-promoted the form to gain more responses. Although we fell slightly short of our goal, the final dataset provided sufficient variety for training purposes.

**2. Data Labeling**

**Techniques Evaluated**

Two primary strategies were explored for labeling:

* **Rule-Based Assignment**
* **Unsupervised Clustering**

**Chosen Strategy: Unsupervised Learning**

We proceeded with an unsupervised learning approach.

**Rationale**

A rule-based system requires explicit logic and pre-defined rules, which can be restrictive, especially with a varied user-generated dataset. Unsupervised learning, such as clustering, allowed us to discover natural groupings and patterns in the data without human bias. This approach provided greater adaptability and aligned better with our dynamic dataset.

**3. Label Encoding**

**Encoding Method Used**

To transform categorical features into a format suitable for machine learning models, we implemented **Label Encoding**.

**Why Label Encoding?**

* **Compact Feature Space**: Given the small dataset (149 records), one-hot encoding would unnecessarily increase dimensionality.
* **Ordered Categories**: Some features had inherent ordering, which label encoding preserved.
* **Computational Efficiency**: Label encoding minimized memory usage and computational load, reducing the chance of overfitting.

**Why Not Advanced Techniques?**

We avoided techniques like TF-IDF and embeddings, as our dataset didn't include complex text-based inputs, and such methods were beyond the scope of this project.

**4. Model Architecture**

**Initial Neural Network Design**

We built an Artificial Neural Network (ANN) consisting of:

* Input Layer
* Two Hidden Layers
* Output Layer

The model used ReLU activation functions and dense connections to learn feature patterns.

**Overfitting Issue & Enhancements**

The limited and imbalanced dataset led to overfitting. Initially, we simplified the architecture by removing one hidden layer, but overfitting persisted. To improve generalization, we applied the following techniques:

* **L2 Regularization**: Penalized large weights to prevent complexity.
* **Batch Normalization**: Stabilized and accelerated training.
* **Dropout**: Introduced randomness during training to avoid neuron dependency.

**5. Model Saving**

**Saving Strategy**

To preserve the most optimal version of our model, we implemented a **ModelCheckpoint** callback using Keras.

**Why This Approach?**

Rather than saving just the final state of the model, we configured the checkpoint to monitor validation accuracy. This ensured that the best-performing model on unseen data was automatically saved for deployment.

**6. Inference Script**

**Functionality**

The inference script allows users to make predictions using the trained model directly from the command line.

**Command-Line Arguments with argparse**

The script uses the argparse library to accept:

* weights\_path: Location of the saved model.
* data\_path: Path to the input dataset.
* num\_preds: Number of predictions to perform.

**Process Overview**

1. **Load Model**: Using the specified weights.
2. **Read & Preprocess Data**: Load the dataset and apply label encoding.
3. **Predict**: Generate predictions for the requested number of samples.
4. **Time Measurement**: Calculate and display prediction time for performance monitoring.