***“Genetic for LSTM Hyperparameter Tuning”***

To optimize hyperparameters of an LSTM model we need:

1. A set of population (or Generation) each chromosome representing a combination of hyperparameters.
2. Fitness Function to evaluates how well a set of hyperparameters performs
3. Chooses parents for reproduction based on fitness (roulette wheel selection)
4. Uses Crossover & Mutation to creates new solutions by combining and randomly altering parent chromosomes.

The genetic algorithm tunes the following LSTM model hyperparameters:

* LSTM layer sizes:
  + lstm1\_units (64–512 neurons)
  + lstm2\_units (32–256 neurons)
* Dense layer size (dense\_units: 16–64 neurons)
* Dropout rates:
  + dropout1 (0.1–0.5)
  + dropout2 (0.1–0.5)
* Learning rate (learning\_rate: 0.0001–0.01)

Key Functions:

* encode\_params(params): Converts hyperparameter values into a binary string (chromosome).
* decode\_chromosome(chromosome): Converts a binary string back into a hyperparameter dictionary.
* fitness(chromosome, X\_train, y\_train, X\_val, y\_val): Builds and trains an LSTM model using the hyperparameters from the chromosome, then returns the validation accuracy (fitness score).
* roulette\_wheel(pop, fitnesses): Selects parents probabilistically (higher fitness = better chance of selection).
* crossover(parent1, parent2): Combines two parents' chromosomes to create two children
* mutate(chromosome): Randomly flips bits in the chromosome to introduce diversity
* optimize\_hyperparameters(X\_train, y\_train, X\_val, y\_val): The main optimization function that runs the genetic algorithm for GENERATIONS and returns the best hyperparameters found.

What encode\_params() function doing:

1. Normalizes Values: Converts each hyperparameter value to a range of [0, 1] based on its predefined min/max (PARAM\_RANGES).
2. Scales to 5-Bit Precision: Multiplies the normalized value by 31 (since 2^5 − 1= 31) and rounds to an integer, enabling 32 discrete levels per parameter.
3. Binary Conversion: Encodes each scaled integer as a 5-bit binary string (e.g., 13 → 01101), ensuring fixed-width representation.
4. Concatenates Bits: Combines all 5-bit binary strings into a single chromosome (e.g., 0110101101... for 6 parameters × 5 bits = 30 bits total).
5. Lossy but Efficient: Tradeoff: Loses some precision (due to 5-bit discretization) but enables efficient genetic algorithm operations (crossover/mutation).

What fitness() function doing:

1. Hyperparameter Decoding: Converts the binary chromosome (genetic representation) back into a dictionary of hyperparameters (e.g., LSTM units, dropout rates) using decode\_chromosome().
2. LSTM Model Construction: Builds a Keras LSTM model dynamically using the decoded hyperparameters, including:
   1. Embedding layer (fixed dimensions).
   2. Two LSTM layers (units tuned by GA).
   3. Dropout layers (rates tuned by GA).
   4. Dense layers (units/activation tuned by GA).
3. Model Training: Trains the model for 5 epochs (reduced for speed) on X\_train/y\_train, with validation data (X\_val, y\_val).
   1. Uses Adam optimizer with GA-tuned learning rate.
   2. Tracks sparse\_categorical\_crossentropy loss and accuracy.
4. Fitness Evaluation: Returns the final validation accuracy (from the last epoch) as the fitness score.
5. Higher accuracy → Better fitness → Higher chance of selection

What roulette\_wheel() function doing:

1. **Calculates Probabilities**:
   * Each individual’s selection probability is proportional to its fitness (e.g., higher fitness = higher chance of being selected).
   * Example: If fitnesses are [0.5, 0.3, 0.15, 0.05], probabilities become [50% , 30%, 15%, 5%].
2. **Randomly Selects 2 Individuals**:
   * Uses random.choices with the computed probabilities to pick 2 parents.
   * Fitter individuals are more likely to be chosen, but weaker ones still have a chance.