📌 Machine Learning Class Fall 2025 Assignment 1 Q5 Specification:

📅 Deadline: 2025-10-10 23:59

📌 Assignment Type

Individual Assignment - Each student must submit their own work

📜 Submission Policy

⏳ You have a total of two extra days available for late submissions across all assignments.

- If you want to use late days for an assignment, please send an email to ml-elsalab@googlegroups.com specifying the number of late days you intend to use (in units of days).

🚫 No copying or cheating; otherwise, you will receive zero points for this assignment.

⚙️ Rules

- You may use any ML algorithm (linear models, tree ensembles, etc.)
- You may use any standard packages or libraries (e.g., numpy, pandas, scikit-learn, tensorflow, pytorch, lightgbm, etc.) to implement your solution. (Only for Q5)
- The work must be your own implementation of the method, even if the algorithm or approach itself is described in a paper or other resource.
    - You are not allowed to use any open-source implementations of this task that are publicly available, such as:
        - Copying code from GitHub repositories, blogs, or other students.
        - Directly using implementations released with research papers.
    - Otherwise, you will receive zero points for this assignment.

Go Rank Prediction

🎯 Objective

The goal of this assignment is to implement a Machine Learning model that can predict the playing strength (rank) of a Go player from their game records. The dataset contains Go games annotated with player rank labels.
Your task is to build a model that uses extracted features from the games to accurately estimate the player's rank.

🔘 Core Challenge: Generalization Ability

- Your model should not rely on memorizing specific games.
- The training set and the test set are not identical: test samples are aggregated from different sets of games.
- Your model must demonstrate the ability to generalize to unseen game records.

☑ Task Requirements

1. Build a ML model (e.g. regression or classification mode) to predict the player's rank.
2. Evaluate your model's performance in terms of:
    - Accuracy (exact rank prediction)

🗁 Dataset

- Training Data
    - The training set consists of nine files, each corresponding to one rank (1D–9D).
    - For example: log_9D_policy_train.txt contains data from 9-dan players, log_8D_policy_train.txt from 8-dan, and so on.
    - In every game, both players belong to the same rank, ensuring clean labels.

    - Each file records all moves from multiple games played at that rank.
    - Each move includes several types of numerical features, such as:
        - Policy values
        - Rank model outputs
        - Strength score
        - Lead information
- Test Data
    - The test set consists of multiple files, each identified by a random ID (e.g., 1.txt, 2.txt) rather than rank.
    - Each file contains multiple games, all from the same rank.
    - In every game, both players belong to the same rank, ensuring clean labels.
    - Test samples are aggregated from several games per file.
    - The internal structure of each move in the test set is identical to that of the training set.
- Output Label
    - The task is to predict the correct rank for the entire file.
- Data Format

  Example snippet from log_1D_policy_train.txt:

```
B[Q16]
0.149007 0.212477 0.192526 ... 0.127816
50.146% 51.0391% 49.2588% ... 48.8448%
0.0915034 0.110794 0.108707 ... 0.135415
-2.84864
41.2% -0.6 17.2
```

  Each move (e.g., B[Q16], W[D4]) is represented by **six parts**:

---

1. Move

- The player's color and coordinate, e.g., B[Q16].
- The color (B or W) is important, since it affects how winrate and lead differences are computed (both are black's perspective).

---

2. Policy Probabilities (9 values)

- These come from **nine different SL models, each trained on games of a specific rank (1D–9D)**.
- They represent the probability of this move according to each model.
- **Meaning**: If the probability is high for a given model, the move is highly consistent with how players of that rank tend to play.

---

3. Value Predictions (9 values)

- Also from the **nine SL models (1D–9D)**, but these are the predicted winrates of the position.

---

4. Rank Model Outputs (9 values)

- Produced by a dedicated **SL rank_model**.
- The outputs are probabilities that the position belongs to each of the 1D–9D ranks.
- **Reference**:
    - *Determining Player Skill in the Game of Go with Deep Neural Networks*
    - In the paper, the outputs were “strong / median / weak.” Here, they are expanded to **nine ranks**.

---

5. Strength Score (1 value)

- A relative strength score.

- **Meaning**: Moves from players of similar strength will have similar values.
- **Reference**:
    - *Strength Estimation and Human-Like Strength Adjustment in Games*

---

6. KataGo Analysis (3 values)

These values are computed by **KataGo**, currently the **strongest open-source Go engine**, widely used in research and analysis:

- **Winrate (black's perspective)**: The predicted winning probability for black, directly from the network's value head.
- **Lead (black's perspective)**: The estimated territory lead (in points).
- **Uncertainty**: How uncertain the model is about this position; higher values mean the prediction is more likely to be wrong.

⚠ Both winrate and lead are given from black's perspective. For moves played by white, the signs must be inverted.

🏆 Evaluation

Your submission will be scored based on:

- Exact Rank Accuracy: fraction of test samples where predicted rank matches true rank.

🛠 What You Need to Implement

📌 You must submit your prediction result (e.g. submission.csv) to Kaggle. (https://www.kaggle.com/competitions/machine-learning-class-fall-2025-assignment-1-q-5)

- You are only responsible for submitting the predicted results of your model.
- Summit Rule
    - Team Name
        - Please use your student ID as the team name.
        - If you are an auditor, add _a after your student ID (e.g., b12345678_a).
    - Submission Limit
        - Each team may submit up to 10 times per day.

📝 Grading Policy (25 Points + Up to 20 Bonus Points)

🎯 Base Score (based on private leaderboard accuracy)

- ≥ 70% → 5 points
- ≥ 73% → 10 points
- ≥ 77% → 15 points
- ≥ 80% → 20 points
- ≥ 83% → 25 points
- ≥ 85% → eligible for leaderboard bonus (see below)

🏆 Leaderboard Bonus

- Only participants with private leaderboard accuracy ≥ 85% are eligible for extra bonus points.
- Bonus is based on private leaderboard ranking:
    - Top 5%: +20 points
    - Next 5%: +19 points
    - Next 5%: +18 points
    - … decreasing by 1 point for every additional 5%
    - Until bonus reaches 0

Note:

- The private leaderboard will be revealed on **2025/10/17**.
- ⚠ If you do **not** make a Kaggle submission, you will receive **0 points** for this question, regardless of other files you submit.

Submission Requirements (To NTU Cool)

Your submission should be compressed into a .zip file named after your student ID.

- Example: If your student ID is D13922036, your submission file should be named D13922036.zip.

Submission Requirements

- Required Files:
  - Q5.py
    - Must be executable and directly produce submission.csv (in the correct Kaggle format).
    - This is to ensure that your Kaggle submission is indeed generated by your own code.
    - ⚠ If Q5.py cannot generate a valid submission.csv, your score for Q5 will be 0, regardless of your Kaggle leaderboard performance.
- Training Code:
  - You must also submit your training script. This can be:
    - A separate training.py, or
    - Integrated into Q5.py (in which case, Q5.py must default to evaluation mode, and training must only be enabled via an additional switch/flag).
    - Please provide a readme.txt with instructions for running training
  - ⚠ If no training code is submitted, or if the training code cannot be executed, your score for Q5 will be 0, regardless of leaderboard results.

- Weight Files:
  - If Q5.py loads a pre-trained weight file (e.g., .pkl, .txt, .npy), you must include this file in your .zip submission.
- Important Note:
  - ❌ Do not include your Q1–Q4 .ipynb files in the .zip submission. Only files related to Q5 should be included.
- Example
  - <student_id>.zip

    ├── Q5.py # Required. Must generate submission.csv when run.

    ├── training files # (Optional) If training is separated from Q5.py

    ├── weight files # any required pretrained weights

    ├── readme.txt # Instructions (How to run your training?)

⚠ Any submission format errors will result in score deductions.

🎮 Enjoy the assignment! 😀