# Decision Trees for RISK: Learning winning states

---

This assignment involves learning a classifier for the game of RISK that will return a probability of a player winning given the territory assignments at the end of the territory selection phase at the beginning of the game. Naturally, such a classifier could be used by an AI for the game to decide which territories to select. We will do this by implementing a decision tree to do the classification. You should first download the `RISK-AI.zip` file from Canvas and unzip in on your computer (If you already did this for a previous assignment you don't need to do it again). The provided code has been tested with Python 3.

The folder you are provided contains (in addition to files for the final RISK AI assignment, which are described in that handout):

- `RISK_Decision_Tree_Handout.pdf` - This file that you are reading
- `generate_dt_data.py` - This file is used to generate labeled data instances that we will feed into our decision tree algorithm. This has already been run for the attacker AI, and data is provided, but if you want to generate your own data from your own AI, you can do this. It generates data by playing games between the specified AIs and saving the state at the end of the assignment phase, as well as who wins the game at the end. It can be run by typing:

  ```
  python generate_dt_data.py ai_1.py ai_1_name ai_2.py ai_2_name
                             number_of_games
  ```

  from the command line.
- `learn_d_tree.py` - This is the file you will be editing for this assignment. It learns a decision tree, given a data file. It then saves the learned decision tree to a file.

  It can be run by typing:

  ```
  python learn_d_tree.py datafile.dat depth_of_decision_tree
  ```

  from the command line.
- `evaluate_d_tree.py` - This is the script that you will run to evaluate your decision tree accuracy.

  It can be run by typing:

```
python evaluate_d_tree.py decision_tree.tree datafile.dat
```

from the command line.

- **decision_trees, dt_data** - these are folders where the scripts save the decision trees and dataset for this programming assignment, respectively. The dataset folder already contains the datasets you will use, which are:
  - `dt_10.dat`
  - `dt_1000.dat`
  - `dt_10000.dat`
  - `dt_100000.dat`
  - `dt_test.dat`

  These datasets differ in size, and each contains the results from as many games as the name implies. (There are, however, twice this many examples, because each single game provides us a winning example and a losing example.) You will use them to see how extra data effects the quality of the decision tree, using the dataset labeled "test" for evaluation.

# Part 1 [60 points]

The first part of this assignment is to complete the code so that you can learn a decision tree from data. In this task, you will be required to implement two functions to learn decision trees. Starter code has been provided with the assignment

- **Task 1 [10 points]** This task requires you to look over the provided code and implement the `compute_entropy` function within the `learn_d_tree.py` file. This function should return the entropy value for a given probability $p$. Look for the place in the code marked "BEGIN TASK 1".
- **Task 2 [50 points]** This task requires you to implement the `determine_info_gain` function within the `learn_d_tree.py` file. This function should return the information gain that would occur if the decision tree node were to split on the given feature. This code should use the `compute_entropy` function that you completed earlier.

  You can learn a decision tree on the `dt_10.dat` dataset to make sure this works right. The printed screen output that you should get after learning a depth 3 tree is stored in "example_tree.txt". If your output does not match this is exactly, that is ok. Apparently some systems have some floating point differences with the entropy calculation that impact the tiebreaking. If your output splits the data into similarly sized and labeled subsets at each level (same entropy gains), then your implementation is correct.

## Part 2 [40 points]

Now that you have the decision tree learning code working, you will experimentally see how well the tree compares with varying depth and input dataset size. For each depth = {1,3,5,7,10} and each supplied dataset (mentioned above), report the accuracy of the resulting decision tree on the testset provided.

Comment on how well you think the decision tree performed.

You are required to hand in the following for this assignment:

1. Your code with the required parts implemented or changed.
2. Your evaluation of the different decision trees, learned to different depths and on different datasets.
3. At least one substantial paragraph describing what you learned from these experiments.

# 1 Extra credit [20 points]

Modify the `attacker_ai.py` RISK AI to use the best decision tree you learned to help it select countries at the beginning. Turn in your code and a report of how well this new AI performs against the original attacker ai, with random initial country selection.