CS221 Project README
Predicting Winning Lineups For Daily Fantasy Leagues

File Organization and Descriptions

Baseline/

- This folder contains the source for our original baseline algorithm -- a basic greedy algorithm that goes off a given FanDuel player list and creates a roster based on average fantasy points earned per week.

Data/

- This folder contains the raw data files, and data structure files used to manage said data.
    - FantasyDB.py -- This data structure loads all historical fantasy data and provides methods for accessing the data to train ML models, as well as evaluating model results. This is primarily used by our prediction models.
    - FantasyProbabilityDB.py -- This data structure loads the predicted probability output of a classification model for a given week to be used by our MDP.
    - LineupDB.py -- This data structure loads the predicted output of a regression model or the expectation of a classification model for a given week to be used by our CSP.

Evaluations/

- This folder contains the output of our evaluation scripts for both our CS229 models (under Players/) and CS221 models.

Graphs/

- This folder contains the output of our plotting scripts for our CS221 models.

Lineups/

- This folder contains the output of our MDP and CSP models.
    - The CSP lineups are generated and listed under CSP_classifcation_lineup.csv for a CSP ran with input from a classification model, CSP_regression_lineup.csv for a CSP ran with input from a regression model, and CSP_oracle_lineup.csv for a CSP ran with input from an Oracle (perfect input).
    - MDP lineups are found within the MDP/ folder and are broken up in the following format:
        - Week_[ALGORITHM]_[GREEDY]_[WEEK].csv -- wherein Algorithm refers to 'RF', 'GDBT', or 'LReg' depending on the classification algorithm used to generate the input, Greedy refers to whether it was greedy or not, and week refers to the week it is for.

Poster/ Progress/ Proposal/

- These folders contain materials for our poster, progress report, and proposal.

Utility/

- This folder contains files sourced from CS221 homework assignments that we leveraged in our own implementations: namely CSP files from Scheduling, and MDP util file from Blackjack.

CompareOneLineup.py -- Script used to evaluate a single line-up a time. Loads historical data from FantasyDB to compare to output from MDP/CSP.
EvaluateLineups.py -- Script used to evaluate line-ups in bulk.
EvaluatePlayers.py -- Script used to evaluate player predictions from CS229 models.
FantasyCSP.py -- Script used to run CSP to generate line-ups.
FantasyMDP.py -- Script used to run MDP to generate line-ups.
FantasyOracle.py -- Script used to generate input files for CSP/MDP from historical data (not predictions).
FantasyPlayerPredictions.py -- Script used to train, evaluate, and run predictions of player models.
fitParameters.py -- Script used to fit parameters for CS229 models.
Graph.py -- Script used to generate Matplotlib graphs for reports.
MLParams.py -- File containing model parameters for CS229 models.

Running CSP:

python FantasyCSP.py
Update START_WEEK and END_WEEK to change range of weeks to run CSP for
Update MAX_PLAYERS to change number of players to consider for each position (based on descending expected production)
Update ALGO to change what algorithm was used to generate input files.
Update MODEL to decide if CSP should be run on predictions from regression, classification, or oracle.

Running oracle:

python FantasyOracle.py
Then run the CSP with ALGO = ‘’ and MODEL = ‘oracle’ (also updating FILENAME to reflect the desired output file).

Running MDP:

python FantasyMDP.py --week [WEEK] --greed [True/False] --all [True/False] --algo [RF/GDBT/LReg]
The --week argument specifies the week to run predictions for.
The --greed argument specifies whether to use a greedy algorithm or a standard MDP. Note this flag is case sensitive -- to use greedy one must enter “True”.
The --all argument specifies whether to run predictions for just the given week, or all weeks up to and including the given week. Like --greed, this is case sensitive.
The --algo argument specifies what algorithm was used to generate the input -- RF (for Random Forest), GDBT (for Gradient Boosted Trees) or LReg (for Logistic Regression). This is case sensitive.
Update MAX_DB_POS in FantasyMDP.py -- this dictionary specifies how many of each position to look at for generating the line-up. Note the individuals are looked at in descending sorted order by expectation.

Running Evaluations:

```
python EvaluateLineups.py
```

This will run evaluations on both the CSP and MDP, and output to file the lineup’s predicted score and actual score.

To change the input files for the CSP or MDP evaluations adjust the CSP_LINEUP_FILE or MDP_LINEUP_FILE constants. Note that the ‘input files for the CSP or MDP evaluations’ refer to the output of the CSP or MDP -- namely the predicted line-ups.

To change the output files for the CSP or MDP evaluations, adjust the CSP_OUTPUT_FILE or MDP_OUTPUT_FILE constants.