NFL Spread Classifier

https://github.com/vatsalbhargava/NFL-Spread-Classifier

Vatsal Bhargava, Coel Morcott, Will Pattie,
Aidan Villasenor, Alex Romanenko

About the Team

Vatsal Bhargava, BS
Computer Science, Math

Coel Morcott, BS
Computer Science &
Cognitive Science

Will Pattie, BA MMSS,
Math, Econ

Alex Romanenko, BS IEMS
& MS Computer Science

Aidan Villasenor, BS,
Computer Science,
Statistics

Task Description

- Develop a predictive model to determine if a given NFL team will cover their spread in a given game. Will the favorite cover the spread? (T - bet on favorite, F - bet on underdog)
- This task is particularly intriguing due to the challenge it presents in outperforming bookmakers and the team's strong interest in the NFL.
- Predicting games for Week 10-Super Bowl based on team statistics from Week 1-9.
- Goal: 52.4% model accuracy (threshold of profitability in spread betting)

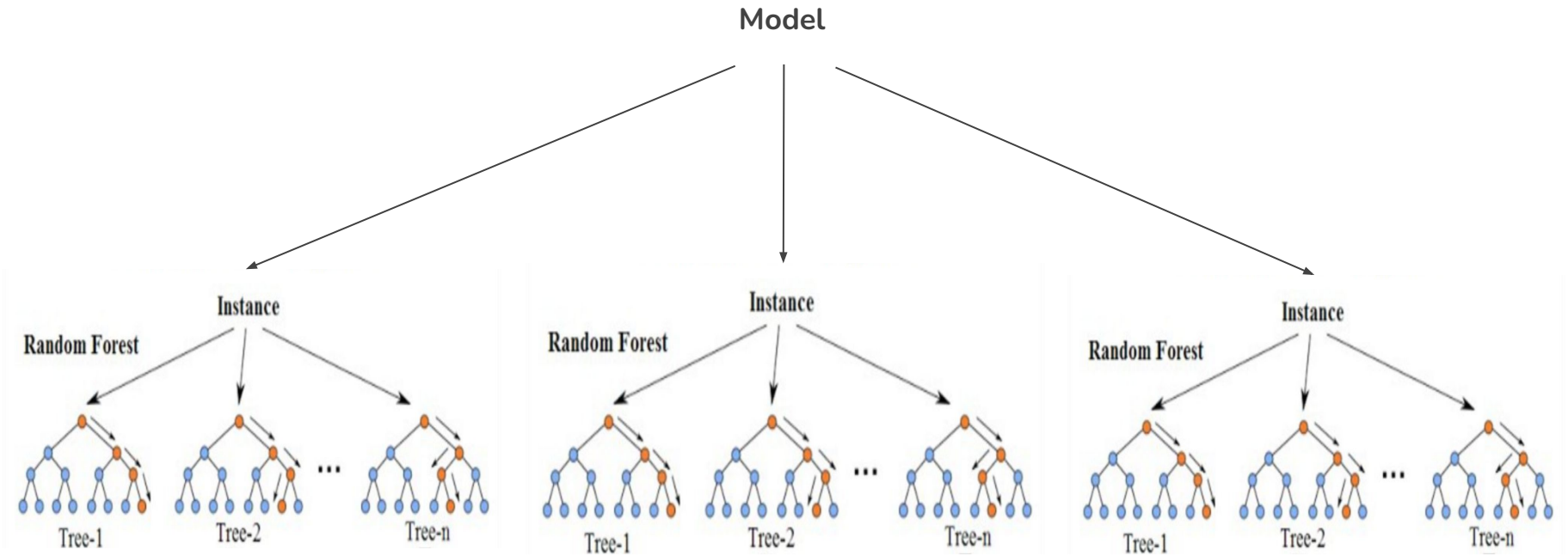
Dataset Description

- Sources: Sports Odds History (historical spread data) and Team Rankings (team statistics)
- Features: 30 total advanced offensive and defensive metrics per team that we deemed most relevant to the spread, based on intuition. (Ex. yards per play, rushing attempts, completions, and passer ratings)
- Labels: T or F based on if favorite covered (50/50 in dataset)
- 2517 rows dating back to 2002
- Summary: Comprehensive statistics on NFL games, focusing on both favorite and underdog teams' performance in various aspects of the game, along with betting spread information.

Model Specification - Random Forest

- Split data into 85% training/validation, and set aside 15% of data for test set
- Utilized sklearn's random forest classifier to generate 11 random forests
  - Each random forest is trained on a different subset of the 85% training/validation
  - Each random forest used majority-class voting from 20 decision trees
- Found hyperparameters that resulted in the best avg validation accuracy across the 11 random forests
  - Criterion: gini and entropy
  - Min_samples_split: 2-100 in increments of 5
  - Max_fetaures: 10, 20, 30, 40, 50, sqrt
- Utilized ensembling (bagging) to make predictions based on the majority classification from the predictions of the 11 random forests

Model Specification - Random Forest



Results - Random Forest

- Achieved average test accuracy of 53.4%
  - Tested with 5 different train/validation versus test splits
  - Best results were with the gini criterion
- Hoping to improve our results even further, we wanted to analyze the accuracy on the test set if we make less frequent, yet more confident predictions

Number of Random Forests Predicting Majority Class >=	% of Test Data We Bet On	Test Accuracy
6	100%	53.4%
7	86.1%	53.6%
8	68.9%	54.2%
9	52.2%	55.9%
10	34.5%	56.3%

Analysis of Results - Random Forest

- Model appears to provide significant value as the test accuracy is 56.3% when betting on the most confident results
- Better than the 50% threshold that we would expect from random sampling
    - Exceeds 52.4% threshold of expected profitability in sports betting
- Demonstrated we could improve our results even further by making less frequent, yet more confident predictions.
    - Betting only on games with at least 8/11 predictions being True or False, yields a test accuracy of 56.3%.
    - With this approach, we would still be able to bet on 34.5% of the games in the test set which continues to be a relatively large sample size.

Model- Neural Net (Binary Classification)

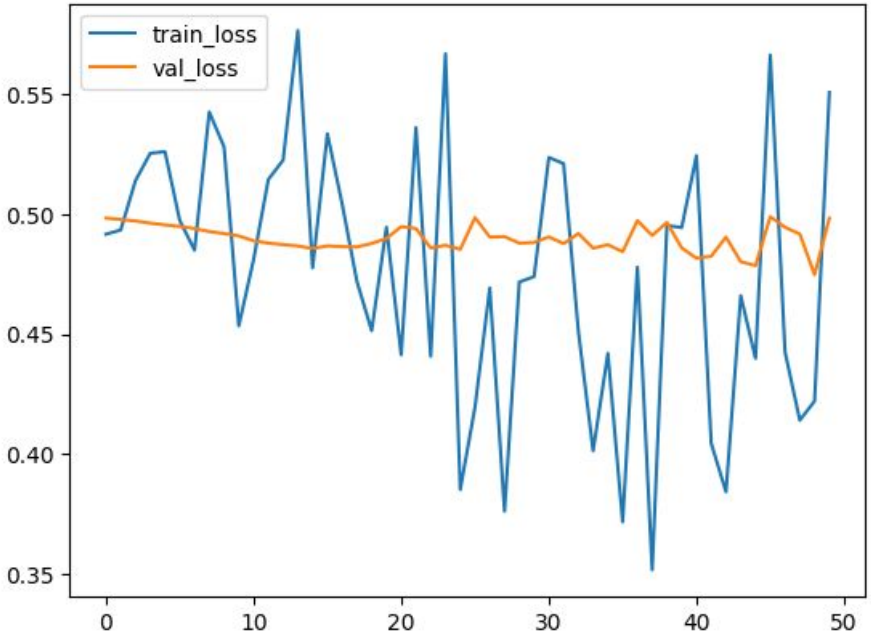
- Feed Forward Neural Net, layer organization: [61 -> 256 -> 64 -> 32 -> 2]
- Stochastic Gradient Descent
- Leaky relu activation function between each hidden layer
- L1 Loss function and softmax activation
- Torch Adam optimization function
- Learning rate of 3e-5 with 50 epochs of training
- Binary classification - output is probabilities of True and False on if a team will cover or not
- 70% training data, 15% test, 15% validation

Neural Network Equation

$$\hat{y}_k = g\left(\sum_{l=0}^{L} w_{lk}^{(4)} f_3\left(\sum_{j=0}^{M} w_{jl}^{(3)} f_2\left(\sum_{h=0}^{H} w_{hj}^{(2)} f_1\left(\sum_{i=0}^{N} w_{ih}^{(1)} x_i + b_h^{(1)}\right) + b_j^{(2)}\right) + b_l^{(3)}\right) + b_k^{(4)}\right)$$

- Generalized equation for a Neural Network with 3 hidden layers
- Optimizing weight and bias terms to minimize loss function

Results - Neural Net



Results after 50 Epochs of training (it did not seem to improve after that)

confusion matrix on test data (x is predicted, y is true)
[[110 65]
[108 91]]
Accuracy on test data: 54%

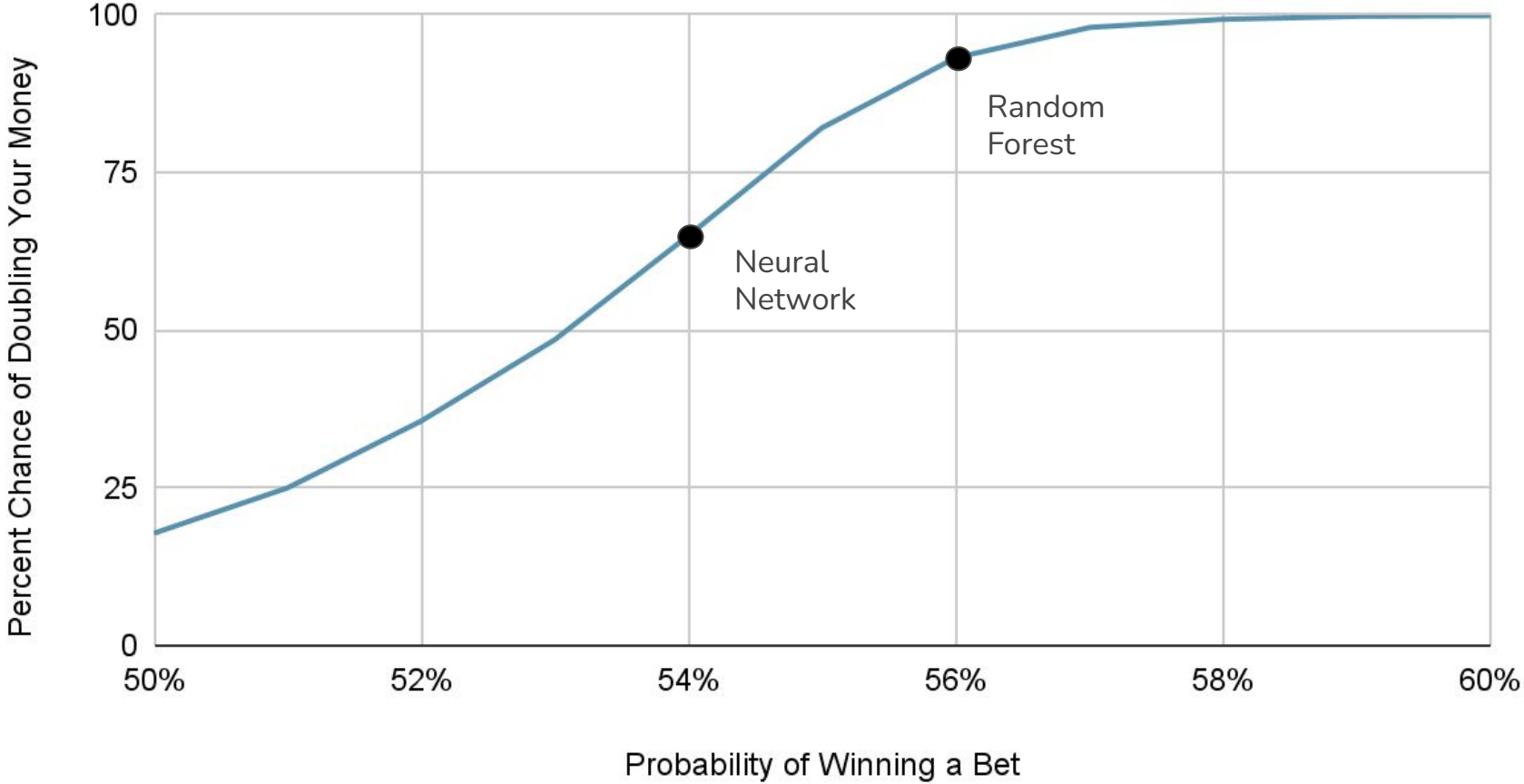
Analysis of Results - Neural Net

- During tuning phase, when we had a higher learning rate,
  - The network learned that it could get a pretty good accuracy predicting only T (as the dataset had slightly more T's).
- Lowering the learning rate gave us a model where it predicted relatively even T's and F's and also had much better than 50% accuracy.
  - 53-55% accuracy persisted even after shuffling the training, testing, and validation data.
- The validation loss has more variance with more epochs
  - Likely due to the model straying away from 50/50 probabilities on T and F with more epochs
- L1 loss >> L2 loss as with L2 loss predicting 0.5 for each minimized loss.
  - Any deviation didn't increase prediction accuracy much so squared error increased more.
- The 54% accuracy that the model consistently gets is enough to turn a consistent profit even with vegas scraping some money
  - Could improve accuracy more by only betting on confident games.

Assessing Profitability of Our Models

- Assume we start with $100 and our strategy is to bet on 10 games each week in the NFL in equal dollar amounts (10 bets/week where each bet = account value/10)
- We assume that sportsbooks (Ex. DraftKings) take a 5% fee on all payouts (including both the initial investment and money won in the bet).
- Ran simulations with 50,000 iterations for each tested probability threshold

Probability of Doubling Your Money Before Losing It All



Expected Number of Weeks to Double Your Money Given That You Double Your Money Before You Lose It All