EVERGLADES Analytics

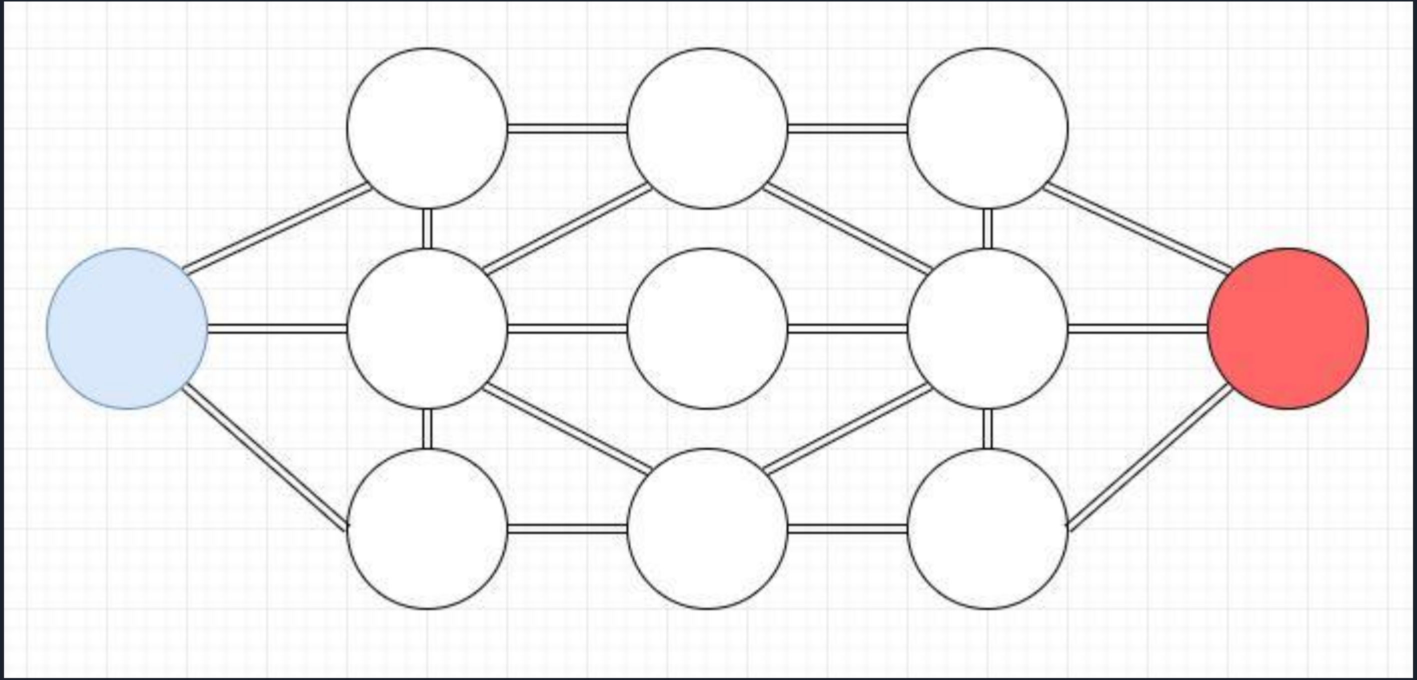
Group 9

Brian Catrett
Chandler Epes
Sebastian Krupa
Shauna Hyppolite
Read O’Quinn

Overview (Administrative Intro)

EVERGLADES is a strategy game that pits two AI agents against each other. It produces extensive telemetry files about what happens in the game. The objective of our group is to take these statistics and analyze them to draw conclusions about the behaviors of the AI agents.

Game Description



- 11 Nodes
- 24 Groups
- 7 Groups move for each player at a time
- 1 Watchtower node
- 1 Fortress node
- Win by Base Capture or Score at End

Project Goals

01 Characterize Everglades match output to determine behaviors that most often lead to wins

02 Create AI scripts of targeting behavior to test hypothesis generated from characterization activity

03 Propose new metrics to more accurately predict agent performance, similar to sabermetrics

04 Implement a supervised machine learning algorithm that predicts the outcome of a match given the current game state

05 Develop a real-time analysis engine that runs simultaneously with Everglades match playback. The engine should at minimum predict odds of winning and describe the agent behaviors

Budget

- DigitalOcean Database Hosting Server: $5 a month * 8 months = $40
- Developing on our own hardware: $0
- Total Cost: $40

Division of Labor

Database	Chandler
Data Gathering & Preparation	Chandler, Shauna, Brian
Analytics	Shauna
AI Scripts	Read & Sebastian
Real Time Analytic Engine	Brian
Real Time Analytic Engine Performance Analysis	Brian

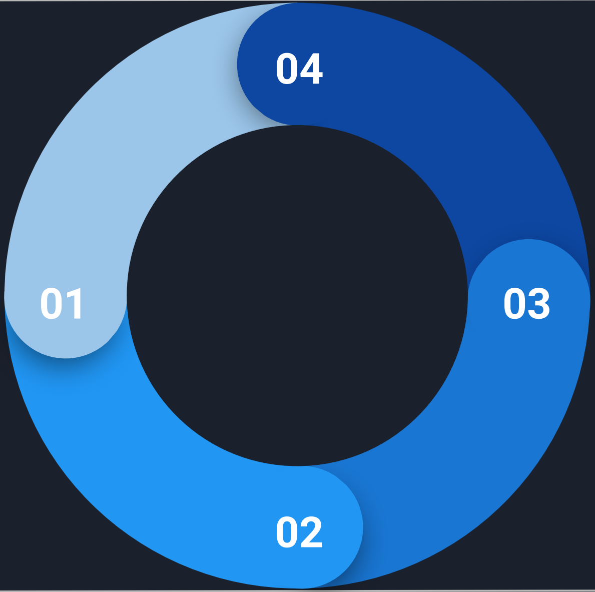
Research Cycle Diagram

Hypothesize

First, we brainstorm metrics and characteristics that we believe will be impactful

Create Scripts

Next, we create scripts to test our hypotheses



Review

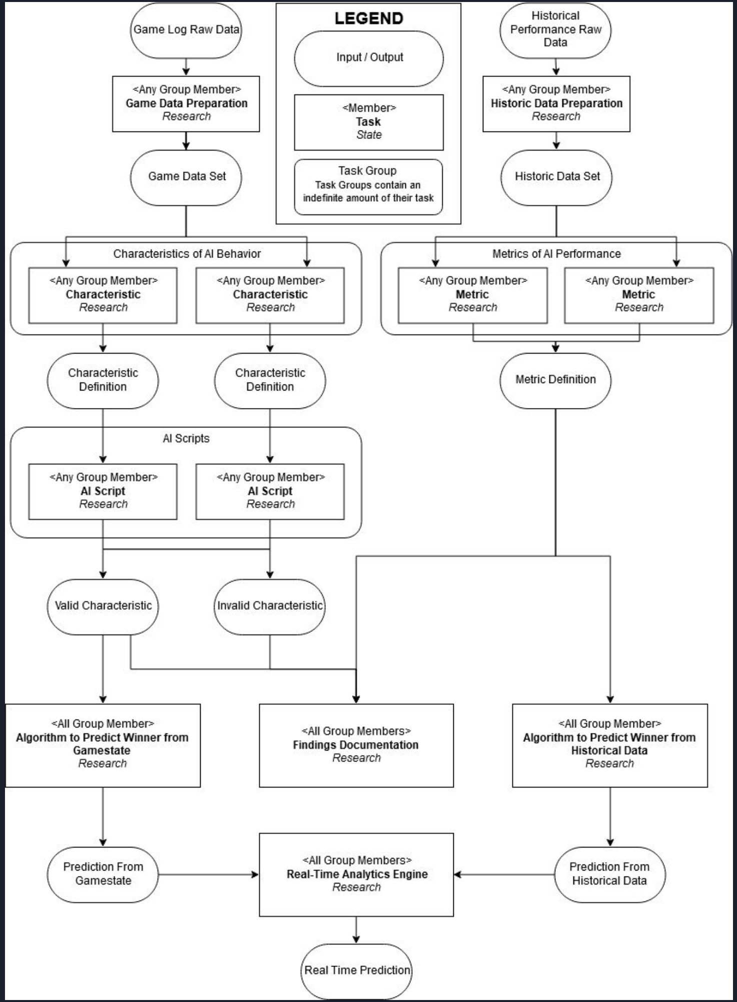
Lastly, we review results of testing and determine if our hypotheses were accurate

Test

Furthermore, we test agents against the default set of agents provided in EVERGLADES

Block Diagram

- Heavy Emphasis on Research and Data Analytics portion, Perform several cycles of script generation and Agent testing to produce “sabermetrics”
- Manage a database of game telemetry data to more easily perform analysis and data visualization
- Use data collected to develop the machine learning algorithm to predict match outcome


Game Log Raw Data
LEGEND
Input / Output
<Member> Task State
Task Group
Task Groups contain an indefinite amount of their task
Historical Performance Raw Data
<Any Group Member> Game Data Preparation Research
<Any Group Member> Historic Data Preparation Research
Game Data Set
Historic Data Set
Characteristics of AI Behavior
<Any Group Member> Characteristic Research
<Any Group Member> Characteristic Research
Metrics of AI Performance
<Any Group Member> Metric Research
<Any Group Member> Metric Research
Characteristic Definition
Characteristic Definition
Metric Definition
AI Scripts
<Any Group Member> AI Script Research
<Any Group Member> AI Script Research
Valid Characteristic
Invalid Characteristic
<All Group Member> Algorithm to Predict Winner from Gamestate Research
<All Group Members> Findings Documentation Research
<All Group Member> Algorithm to Predict Winner from Historical Data Research
Prediction From Gamestate
<All Group Members> Real-Time Analytics Engine Research
Prediction From Historical Data
Real Time Prediction

Database

SQL Database designed around telemetry files generated by EVERGLADES gameplay. Primary goal is to make the database 1:1 with the raw data as much as possible in order to preserve data integrity and make it easier to collaborate with other groups.

- Tracks both gamestate and events that occur over the course of a game
- Tracking of events allows for analysis on effectiveness of specific behaviors and play patterns such as capturing watchtowers and defeating enemy groups early
- Tracks historical performance of AI agents to allow for analysis of successful behaviors and habits over time

Analytics Process - Overview

- Gathering Data
- Preparing Data Sets
- Algorithms for Model Implementation
    - Classification
        - K- Nearest Neighbors
        - Random Forest
        - Logistic Regression
        - Support Vector Machine
- Hyperparameter Tuning
    - Grid Search CV
    - Random Search CV
- Model Evaluation
    - Model Accuracy
    - Confusion Matrix
- Feature Analysis & Conclusion

Data Gathering & Preparation

- Data Gathering
  - Data Gathering is simple and inefficient
  - (A-1)! * n = Total Games, where A is number of agents and n is games per matchup
  - Want to make it faster and more efficient to eliminate downtime
- Data Preparation
  - Some information is not included explicitly in the telemetry
  - Information is almost always needed from multiple tables in the database
  - Make queries as fast as possible to reduce turnaround time for analysis members to resume work

K Nearest Neighbors

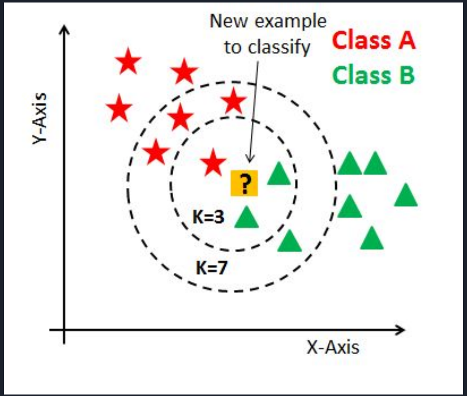
1. Compute the distance between the item that is being classified to every other item in the data set
2. Pick the K closest data points
3. It is then classified based on the majority data points from the data set within K neighbors

Advantages:

- Simple implementation
- Easy to understand
- Few parameters to tune

Disadvantages:

- Calculating distance can become expensive with large data sets
- Sensitive to irrelevant features
- Sensitive to unbalanced data sets when most entities belong to a single class



Random Forest Classifier

Uses a large number of decision trees to make predictions

1. Select random samples from the data set
2. For each sample, construct a decision tree and obtain a prediction from each tree
3. Tally the predictions from all of the trees
4. The final decision is the prediction made by the most trees

Advantages:

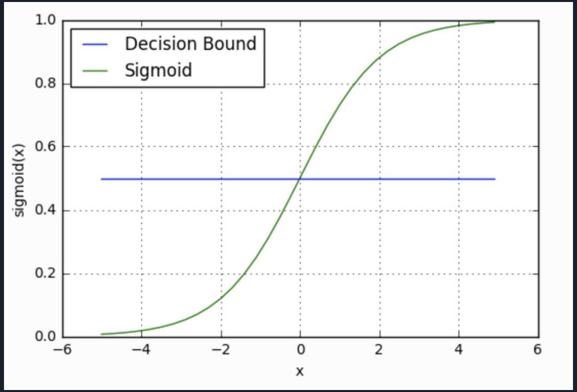
- Highly accurate
- It does not suffer from overfitting
- Provides details on feature importance

Disadvantages:

- It can be slow when generating predictions
- Difficult to interpret

Logistic Regression

A predictive analysis algorithm that maps predictions to be between 0 and 1 and is based on the concept of probability using the logistic sigmoid function



Advantages:

- Relatively easy to implement, since responses are binary
- Shows a relationship between features and the probability of particular outcomes

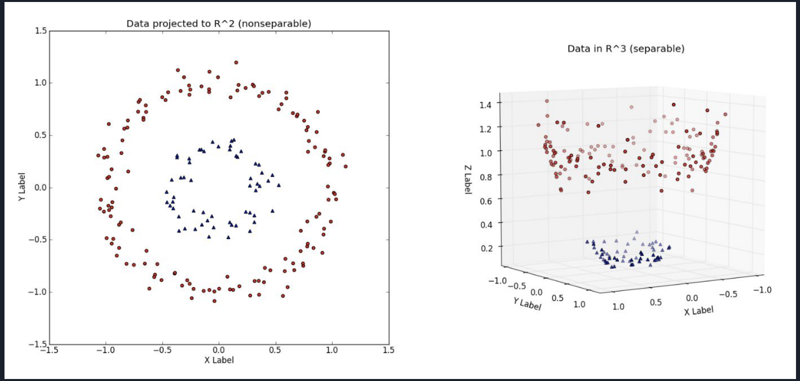
Disadvantages:

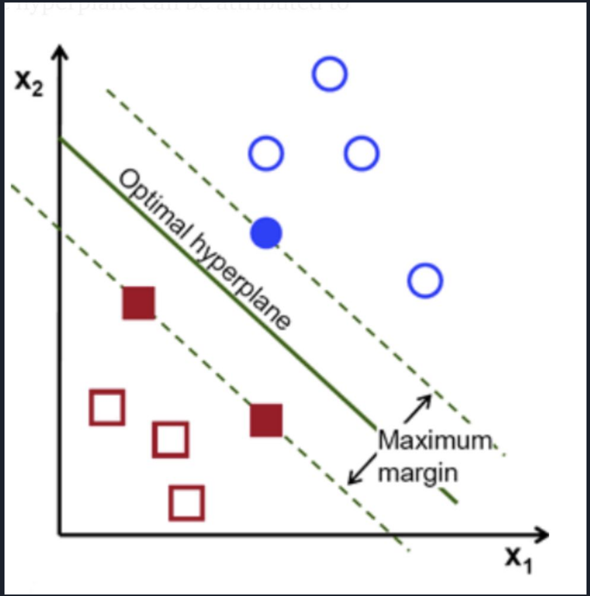
- Underperforms when there are multiple or non-linear decision boundaries
- Not flexible enough for more complex relationships

Support Vector Machine (SVM)

Uses support vectors to find the hyperplane that produces the maximum margin in an N-dimensional space to classify the data points

- Helps generalize our predictions
- Uses kernels to transform the data into a required form





Support Vector Machine

Advantages:

- Performs well in high dimensional spaces
- Memory efficient (only uses a subset of the training points as decisive factors for classification)
- Versatile: can use different kernels for different decision functions as long as produce correct results

Disadvantages:

- Does not perform well when the data set has more noise (target overlap)
- Does not provide probability estimates

Hyperparameter Tuning

Process of selecting the values for the ideal model architecture to maximize the accuracy of the model

Grid Search CV : Methodically builds and evaluates a model for each combination of parameters in a specified grid

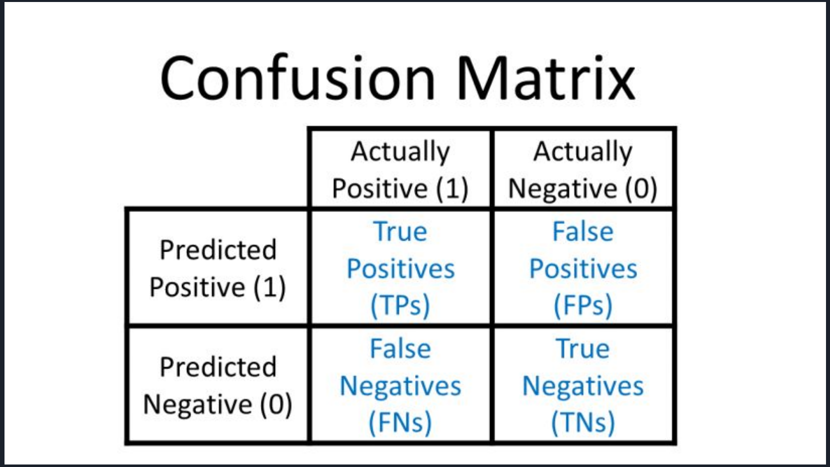
Random Search CV : Samples algorithm parameters from a random distribution for a set number of iterations. A model is then constructed and evaluated for each combination of parameters

Random Search CV sometimes produces results close to Grid Search CV

The main trade off between the two is **time** vs **quality**

Model Evaluation

- Model Accuracy
- Confusion Matrix : A matrix of predictions that summarizes correct and incorrect predictions by class. It gives insight into when our classification models are making errors and which type of errors.
    - A 2 x 2 matrix that consists of:
        - True Positive
        - False Positive
        - True Negative
        - False Negative



Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Sci-Kit Learn

- Free Open Source Library in Python that implements a range of machine learning, preprocessing, cross-validation, and visualization algorithms using a unified interface
- Features various classification, regression, and clustering algorithms
- Built on NumPy, SciPy, and Matplotlib
- Simple and efficient tools for data mining and analysis. Accessible to everybody and reusable

Experiment 1 - First Group Disband

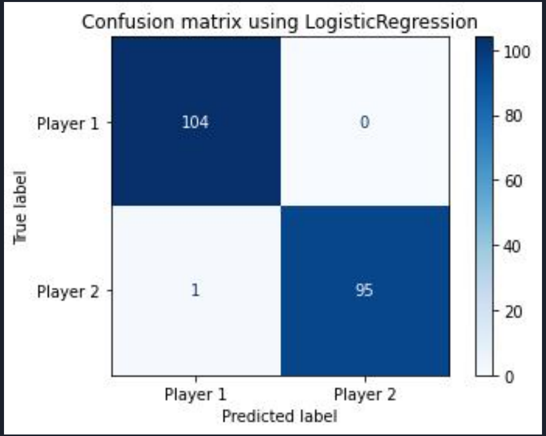
Hypothesis: The player with the first group disband has a direct effect on the outcome of the game and the types of units of that group correlates to the win/loss outcome

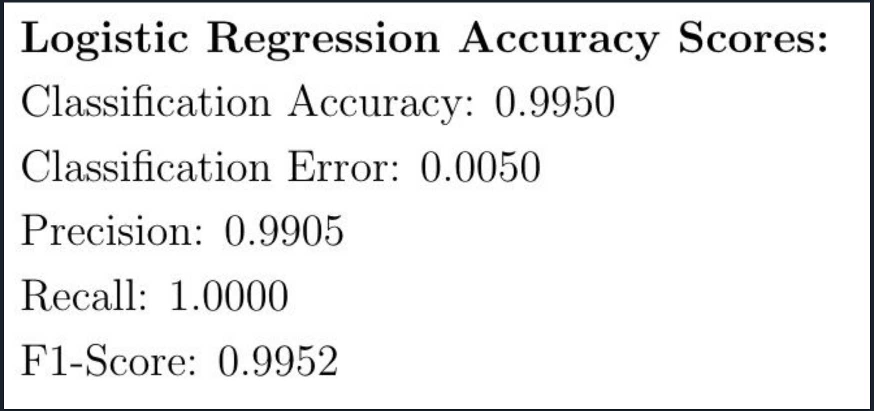
Agents:
Player 1: Random Action
Player 2: Random Action

Overall Games Outcome:
Player 1: 52% Win Rate
Player 2 lost the first group **53%** of the time and lost the game **58%** of those times
54% of the groups in the lost games were Strikers units

First Group Disband

F1-Score and Accuracy of tuned model:





Logistic Regression Accuracy Scores:

Classification Accuracy: 0.9950

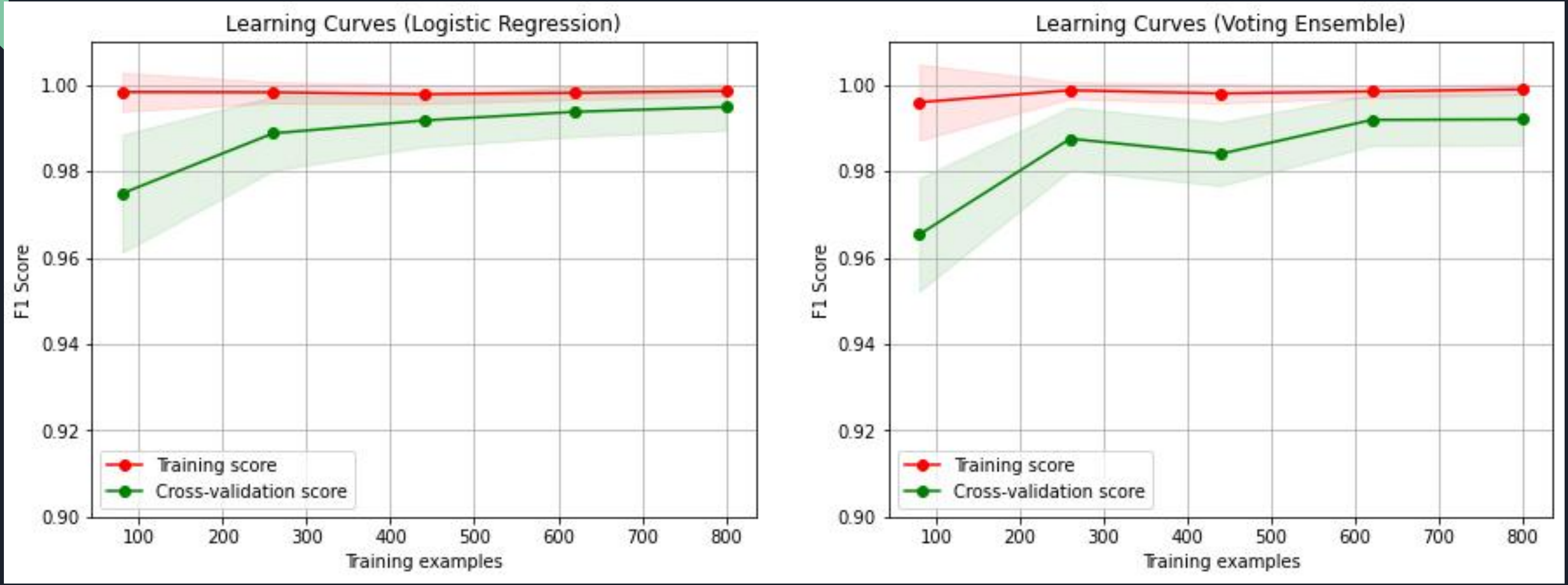
Classification Error: 0.0050

Precision: 0.9905

Recall: 1.0000

F1-Score: 0.9952

First Group Disband



Experiment 2 - First Group Disband

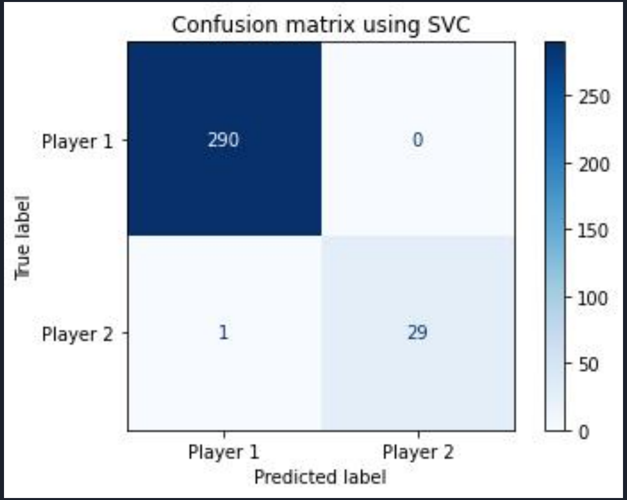
Hypothesis: The player with the first group disband has a direct effect on the outcome of the game and the types of units of that group correlates to the win/loss outcome

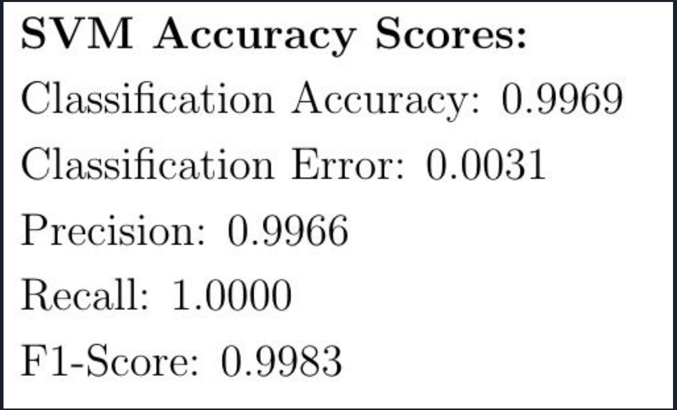
Agents:
Player 1: Base Rush
Player 2: Random Action

Overall Games Outcome:
Player 1: **90.69%** Win Rate (Increase with Agent)
Player 2 lost the first group increased by **14%** and lost the game increased by **34%**
78% of the groups in the lost games were Strikers units

First Group Disband - V2

Increase in F1-Score and Accuracy with new tuned model





SVM Accuracy Scores:

Classification Accuracy: 0.9969

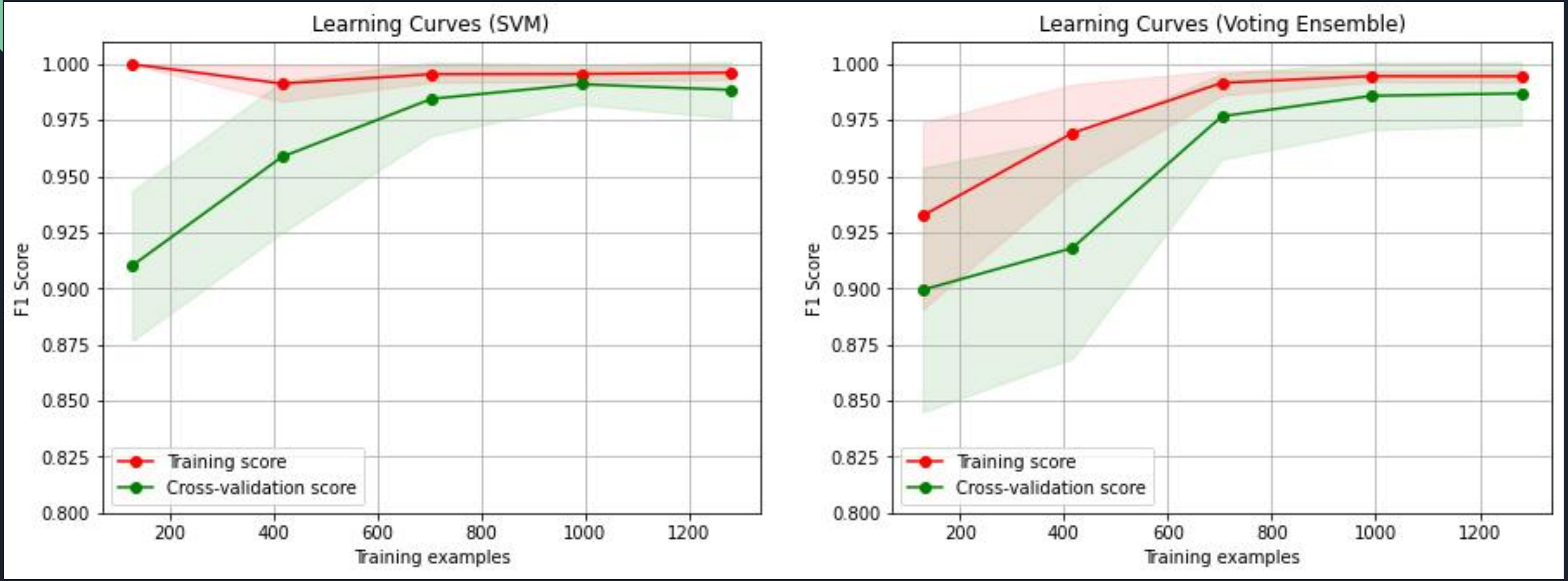
Classification Error: 0.0031

Precision: 0.9966

Recall: 1.0000

F1-Score: 0.9983

First Group Disband - V2



Models could possibly benefit from more features and training examples

Experiment 3 - Fortress Control: Node 4

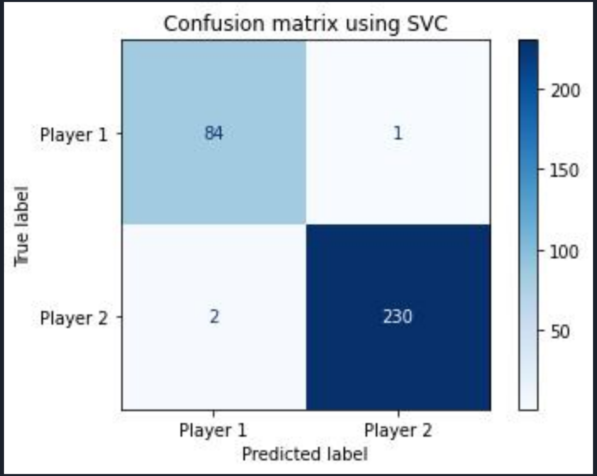
Hypothesis: The player that controls the fortress node 4 last has a direct effect on the outcome of the game

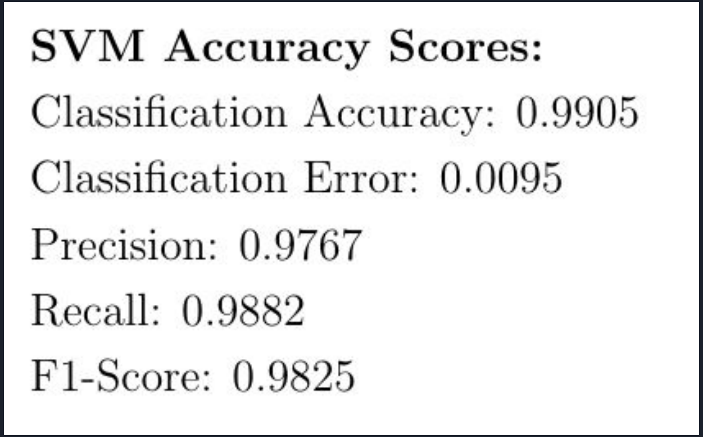
Agents:
Player 1: Random Action
Player 2: Base Rush

Overall Games Outcome:
Player 2: 73.25% Win Rate
Player 2 controlled Node 4 last 7% of the time

Fortress Control: Node 4

F1-Score and Accuracy Score





SVM Accuracy Scores:

Classification Accuracy: 0.9905

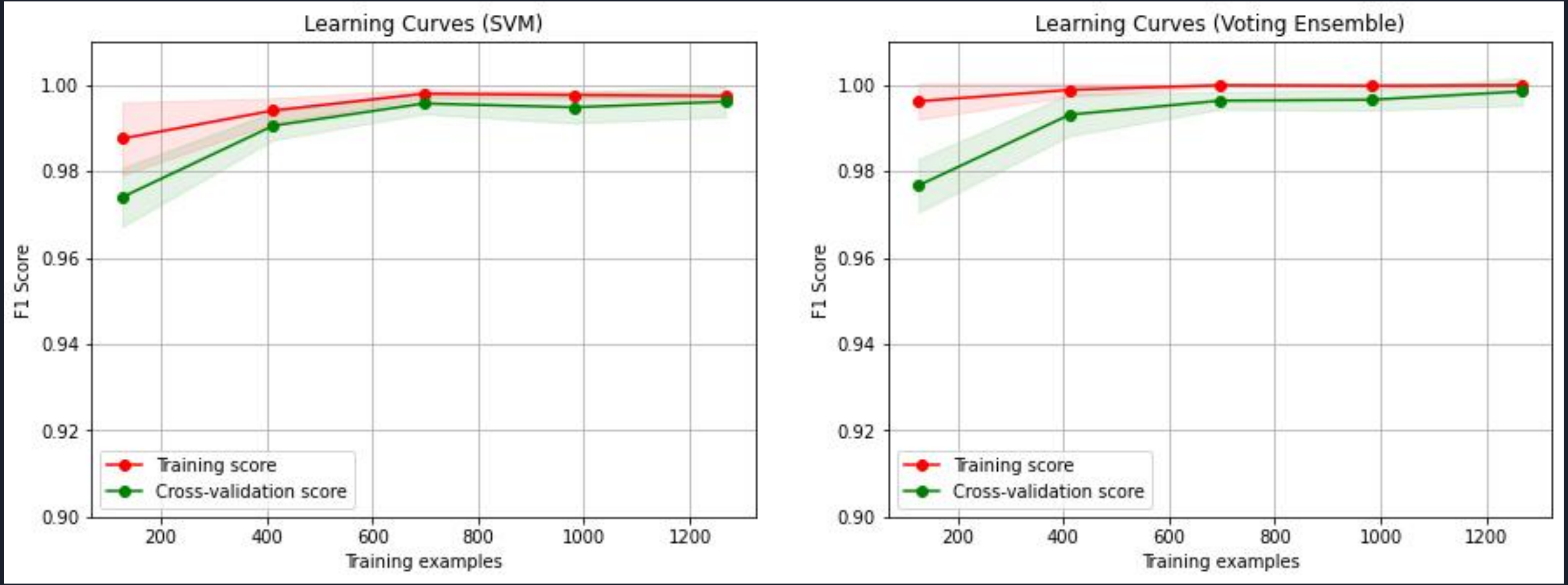
Classification Error: 0.0095

Precision: 0.9767

Recall: 0.9882

F1-Score: 0.9825

Fortress Control: Node 4



Models seem to be performing well, although could be indication of model being too simple.

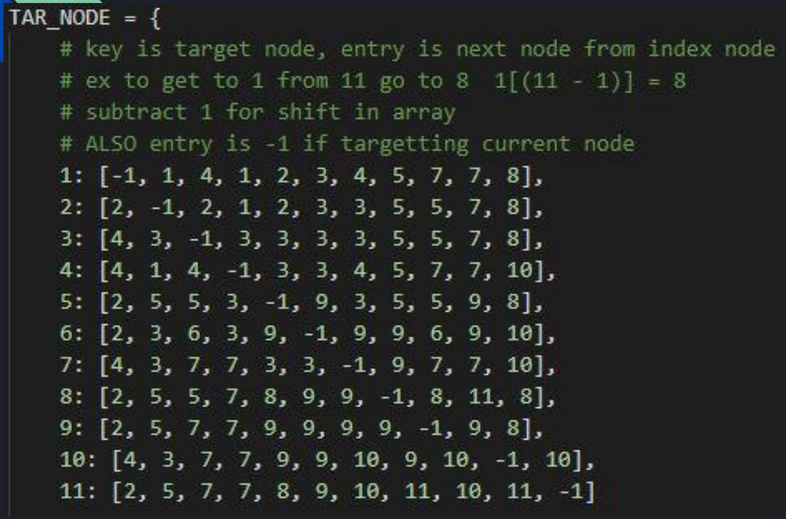
AI Script Development and Evaluation Process

- Initially planned on developing RL agents but switched to finite-state machines
- Process starts with finding specific game statistic that needs further testing
- We then take this statistic and looks into ways to develop a hyper-specific agent script based on the original agents given to us to test the statistic
    - In the case of the base rush script we took the all cycle agent and made it focus on trying to take the base of the opponent player
- We then take this parameter-specific agent and run further testing to see if the script tests the statistic properly and if the statistic is valid

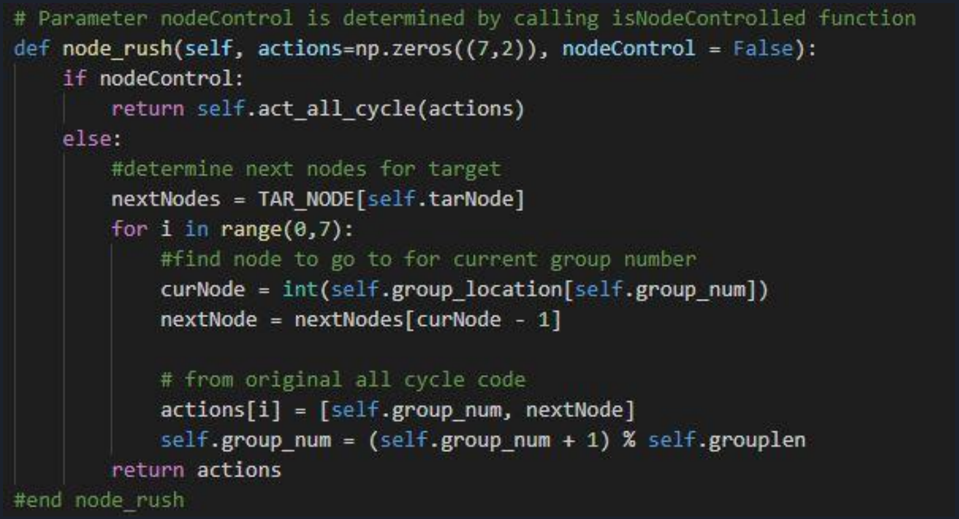
Finite-State Agent Development

- Agents receive feedback from the game environment through an observation array which provides information about node control, opponent location, and group health
- Agents provided to us were improved upon so they could be used for base functionality
- Each agent seeks to accomplish the target metric and then returns to base functionality (random action or all cycle)
- Allowed us to generate data while changing as few variables as possible
- Required creative problem solving to create scripts for certain metrics
- Agents “see” the game map the same way whether they are player 1 or player 2 but node control value changes sign
- “Dead” groups can still be moved but have no impact on combat or node capture

Example Agent: Node 11 Capture

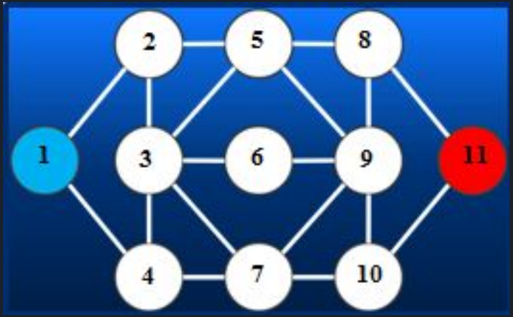


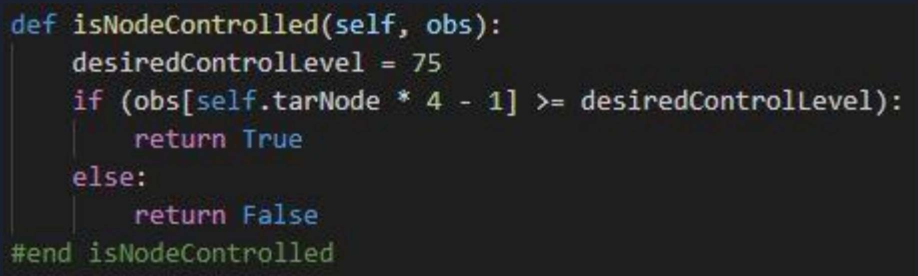
```
TAR_NODE = {
    # key is target node, entry is next node from index node
    # ex to get to 1 from 11 go to 8  1[(11 - 1)] = 8
    # subtract 1 for shift in array
    # ALSO entry is -1 if targetting current node
    1: [-1, 1, 4, 1, 2, 3, 4, 5, 7, 7, 8],
    2: [2, -1, 2, 1, 2, 3, 3, 5, 5, 7, 8],
    3: [4, 3, -1, 3, 3, 3, 3, 5, 5, 7, 8],
    4: [4, 1, 4, -1, 3, 3, 4, 5, 7, 7, 10],
    5: [2, 5, 5, 3, -1, 9, 3, 5, 5, 9, 8],
    6: [2, 3, 6, 3, 9, -1, 9, 9, 6, 9, 10],
    7: [4, 3, 7, 7, 3, 3, -1, 9, 7, 7, 10],
    8: [2, 5, 5, 7, 8, 9, 9, -1, 8, 11, 8],
    9: [2, 5, 7, 7, 9, 9, 9, 9, -1, 9, 8],
    10: [4, 3, 7, 7, 9, 9, 10, 9, 10, -1, 10],
    11: [2, 5, 7, 7, 8, 9, 10, 11, 10, 11, -1]
```



```
# Parameter nodeControl is determined by calling isNodeControlled function
def node_rush(self, actions=np.zeros((7,2)), nodeControl = False):
    if nodeControl:
        return self.act_all_cycle(actions)
    else:
        #determine next nodes for target
        nextNodes = TAR_NODE[self.tarNode]
        for i in range(0,7):
            #find node to go to for current group number
            curNode = int(self.group_location[self.group_num])
            nextNode = nextNodes[curNode - 1]

            # from original all cycle code
            actions[i] = [self.group_num, nextNode]
            self.group_num = (self.group_num + 1) % self.grouplen
        return actions
#end node_rush
```





```
def isNodeControlled(self, obs):
    desiredControlLevel = 75
    if (obs[self.tarNode * 4 - 1] >= desiredControlLevel):
        return True
    else:
        return False
#end isNodeControlled
```

Real Time Analytic Engine and Predictive Algorithm

Goals:

- To run in real time with the game and the match playback system
- Provide a summary of agent behaviors or board states
- Predict the odds of each player winning turn by turn

Primary Issues:

- CSV telemetry files were not generated in real time
- Match playback system was led by an entirely different team

Resolution:

- Pipeline the telemetry data from the client to the engine
- Implement a design that could be utilized by the match playback team in the future

Classification Algorithms and Game State

With the telemetry data pipelined it enabled the ability to provide game state updates turn by turn that summarized player board position, points, health and unit counts. It also provided the capability to analyze features turn by turn using classification models

Classification Algorithms Tested:

- Logistic Regression
- Random Forest Classifier
- K-Nearest Neighbors

Early in development Random Forest Classifier was chosen as the primary focus

Feature Selection

What features should be analyzed to determine the probability of a win?

- Turn Number
- Player points
- Base Capture Values
- Remaining units
- Average unit health

These features will be represented by:

- Player One points minus Player Two points
- Player One base capture value (as a percent of node control)
- Player Two base capture value (as a percent of node control)
- Player One remaining units minus Player Two remaining units
- Player One average unit health minus Player Two average unit health

What about turn number?

Classification Model Implementation

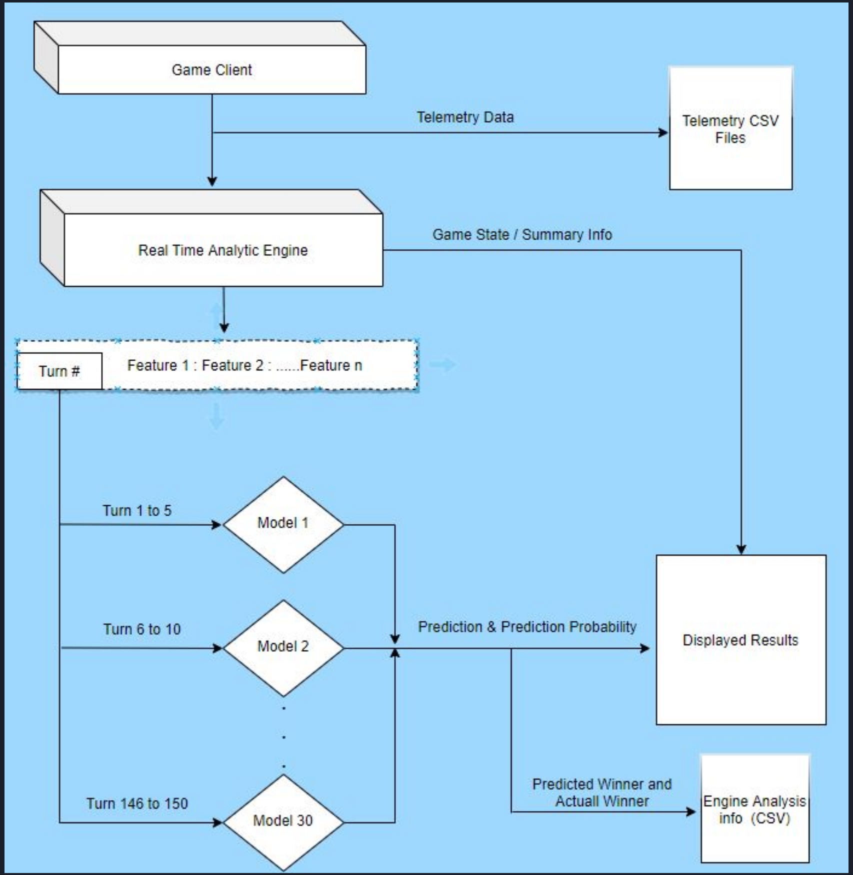
Because turn number is highly correlated with every other feature it was a concern that a single model trained on a single data set would struggle with accurately capturing the strong correlation

- Initial implementation: A multi-model system trained on game data that is split into 2 turn increments
  - 75 different models trained on pairs of turns (Model 1: Turn 1 & 2, Model 2: Turn 3 & 4....Model 75: Turn 149 & 150)
  - Instead of passing turn number as a feature, it would use turn number to select the appropriate model and make a prediction with respect to the given turn

Initial implementation looked promising! Unfortunately this system had issues with predictions with uncommon or rare game states or more specifically, agents that captured the base

The fix, expand the data sets from 2 turn increments to 5 turn increments, this would also reduce the models from 75 to 30

- Final implementation:
  - 30 different models trained on 5 turn increments (Model 1: Turn 1 -5, Model 2: Turn 6-10....Model 30: Turn 146 & 150)


Game Client
Telemetry Data
Telemetry CSV Files
Real Time Analytic Engine
Game State / Summary Info
Turn #
Feature 1 : Feature 2 : ......Feature n
Turn 1 to 5
Model 1
Turn 6 to 10
Model 2
Prediction & Prediction Probability
Displayed Results
Turn 146 to 150
Model 30
Predicted Winner and Actuall Winner
Engine Analysis info (CSV)

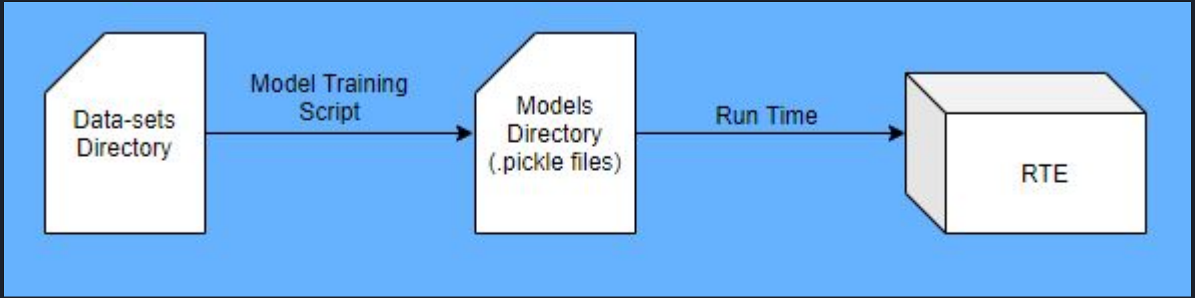
Data-sets and Agents

Data-set Generation:

- Historical data from the database
- Historical data generated by The Real-Time Analytic Engine*

The final models were trained on 2500 games from the following agents:

- Random Action Agent
- Early Game Base Capture Agent
- Mid Game Base Capture Agent
- Late Game Base Capture Agent



Analysis

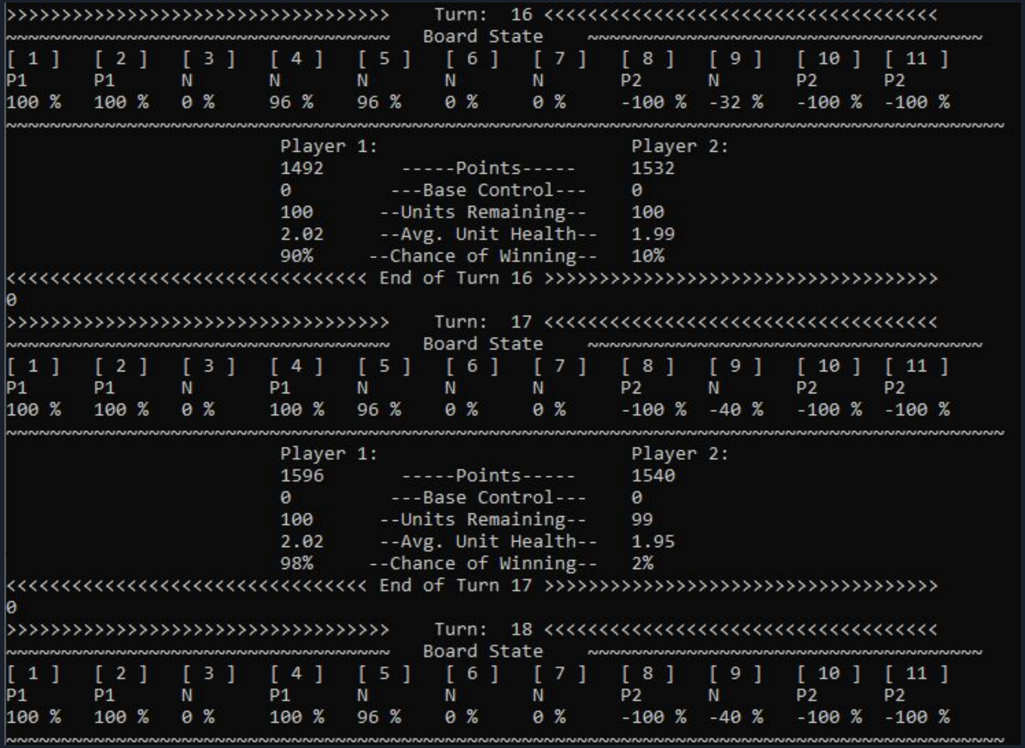
An intuitive approach:

- Given a game state, does the prediction and the confidence in that prediction make sense
  - ie. If player 2 is losing in all aspects of the game that player should not be predicted to win
- Given each player has an equal chance of winning, it was expected to see the predictions start at 50-50 and increase steadily (in most situation)
- There are no drastic or over confident predictions early in the game
  - ie. Any player being predicted to win with 100% confidence on turn 70
    - This was resolved using CalibratedClassifierCV. This recalibrates the model so that the predict probability functionality does not take extremes of 0 or 1

Statistical Analysis: Used confusion matrix metrics to analyze the performance over the course of the game.

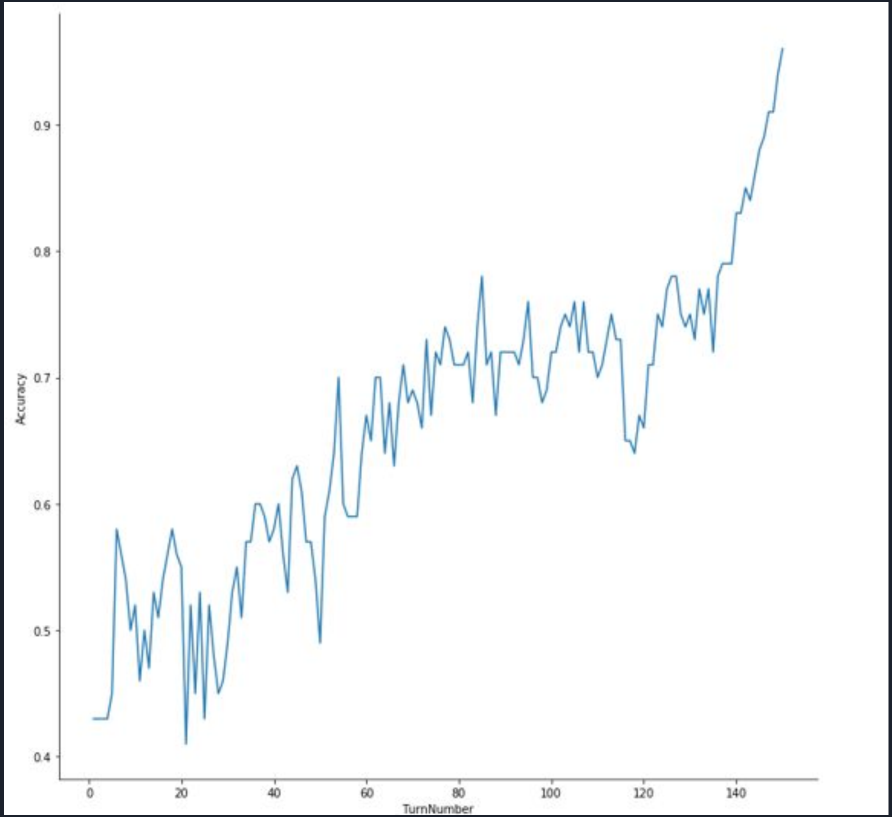
- Accuracy
- Sensitivity (Recall)
- Specificity

CalibratedClassiferCV Fix

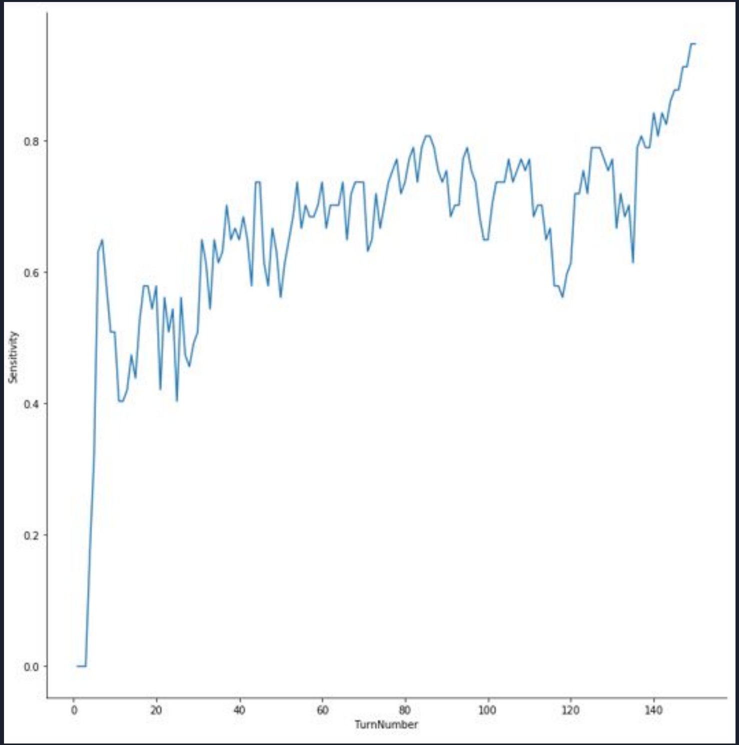


```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>    Turn:  16 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   Board State    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[ 1 ]   [ 2 ]   [ 3 ]   [ 4 ]   [ 5 ]   [ 6 ]   [ 7 ]   [ 8 ]   [ 9 ]   [ 10 ]  [ 11 ]
P1      P1      N       N       N       N       N       P2      N       P2      P2
100 %   100 %   0 %     96 %    96 %    0 %     0 %     -100 %  -32 %   -100 %  -100 %
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                         Player 1:                      Player 2:
                         1492      -----Points-----     1532
                         0        ---Base Control---    0
                         100     --Units Remaining--    100
                         2.02    --Avg. Unit Health--   1.99
                         90%    --Chance of Winning--   10%
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< End of Turn 16 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>    Turn:  17 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   Board State    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[ 1 ]   [ 2 ]   [ 3 ]   [ 4 ]   [ 5 ]   [ 6 ]   [ 7 ]   [ 8 ]   [ 9 ]   [ 10 ]  [ 11 ]
P1      P1      N       P1      N       N       N       P2      N       P2      P2
100 %   100 %   0 %     100 %   96 %    0 %     0 %     -100 %  -40 %   -100 %  -100 %
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                         Player 1:                      Player 2:
                         1596      -----Points-----     1540
                         0        ---Base Control---    0
                         100     --Units Remaining--    99
                         2.02    --Avg. Unit Health--   1.95
                         98%    --Chance of Winning--   2%
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< End of Turn 17 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
0
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>    Turn:  18 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   Board State    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[ 1 ]   [ 2 ]   [ 3 ]   [ 4 ]   [ 5 ]   [ 6 ]   [ 7 ]   [ 8 ]   [ 9 ]   [ 10 ]  [ 11 ]
P1      P1      N       P1      N       N       N       P2      N       P2      P2
100 %   100 %   0 %     100 %   96 %    0 %     0 %     -100 %  -40 %   -100 %  -100 %
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

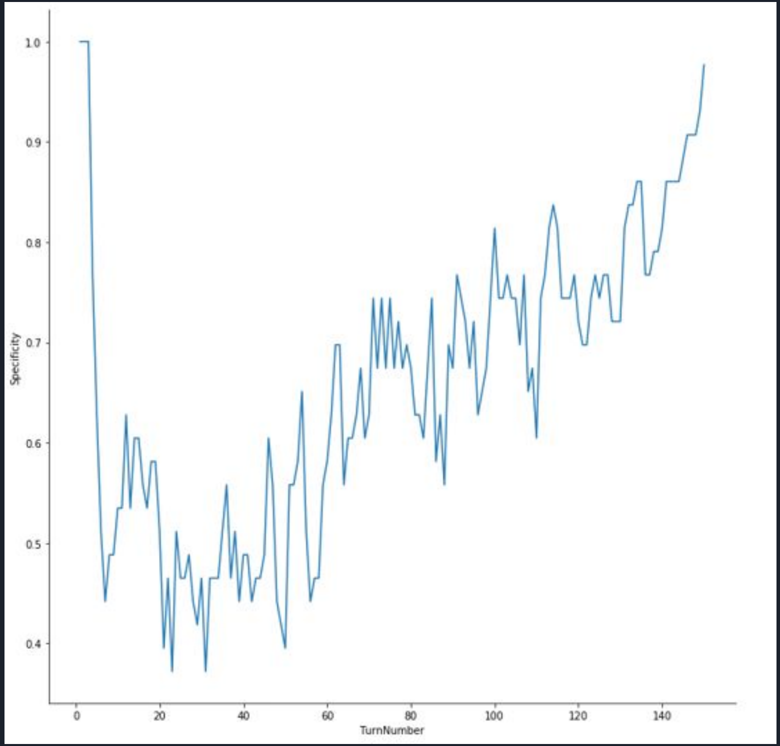
Accuracy: Random Action vs Random Action



Sensitivity: Random Action Vs Random Action



Specificity: Random Action Vs Random Action



Final Real Time Analytic Engine and Predictive Algorithm:

Conclusion and Issues

The final models used were a drastic improvement but still had issues:

- The models predicted games that were determined by points on turn 150 accurately but suffered as more base capture agent data was incorporated in the data sets
- The prediction for wins via base capture had lower than expected probabilities
- The models will struggle with making proper predictions against any agent that could reliably capture the base before turn 30 and between turns 100 and 120
- It does not account for ties. Ties are rare but they can happen