The NEWMAC Score Predictor

Our project aimed to use historical NEWMAC soccer data to train machine learning models to predict the outcome of an upcoming Clark Men's Soccer Conference run. Specifically, we implemented three models that incorporated Logistic Regression, Decision Tree, and Random Forest Algorithms, comparing the accuracies and conclusions drawn from the different models.

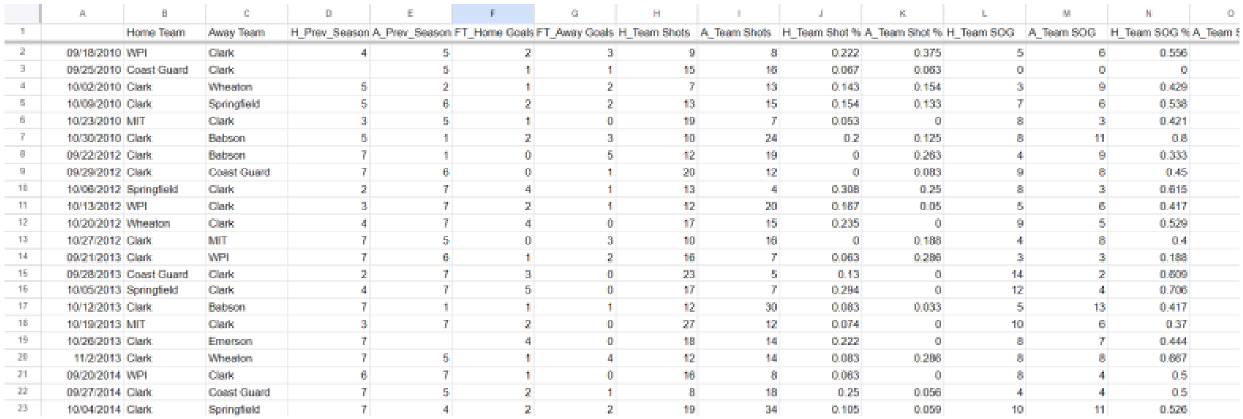
A big part of our motivation for choosing this project was that we are all part of the Men's Soccer Team at Clark and have noticed firsthand how statistics are having more of an impact on the beautiful game than ever before.

Implementation Process

Our first step towards the implementation of this idea was data collection. While tedious, the process of manually inputting NEWMAC game-to-game data was essential in ensuring that we had a sufficient database to draw conclusions from. Below is the format of our .csv file containing said data:

The full scope of data does not fit in the screenshot, but the included data points are as follows:

{Date, Home Team, Away Team, H_Prev_Season, A_Prev_Season, FT_Home Goals, FT_Away Goals, H_Team Shots, A_Team Shots, H_Team Shot %, A_Team Shot %, H_Team SOG, A_Team SOG, H_Team SOG %, A_Team SOG %, HT_Corners, AT_Corners, HT_Red, AT_Red, HT_Yellow, AT_Yellow, HT_Fouls, AT_Fouls, Result(Clark)}

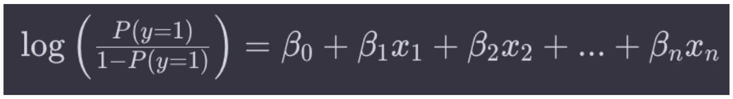


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		Home Team	Away Team	H_Prev_Season	A_Prev_Season	FT_Home Goals	FT_Away Goals	H_Team Shots	A_Team Shots	H_Team Shot %	A_Team Shot %	H_Team SOG	A_Team SOG	H_Team SOG %	A_Team S
2	09/18/2010	WPI	Clark	4	5	2	3	9	8	0.222	0.375	5	6	0.556	
3	09/25/2010	Coast Guard	Clark		5	1	1	15	16	0.067	0.063	0	0	0	
4	10/02/2010	Clark	Wheaton	5	2	1	2	7	13	0.143	0.154	3	9	0.429	
5	10/09/2010	Clark	Springfield	5	6	2	2	13	15	0.154	0.133	7	6	0.538	
6	10/23/2010	MIT	Clark	3	5	1	0	19	7	0.053	0	8	3	0.421	
7	10/30/2010	Clark	Babson	5	1	2	3	10	24	0.2	0.125	8	11	0.8	
8	09/22/2012	Clark	Babson	7	1	0	5	12	19	0	0.263	4	9	0.333	
9	09/29/2012	Clark	Coast Guard	7	6	0	1	20	12	0	0.083	9	8	0.45	
10	10/06/2012	Springfield	Clark	2	7	4	1	13	4	0.308	0.25	8	3	0.615	
11	10/13/2012	WPI	Clark	3	7	2	1	12	20	0.167	0.05	5	6	0.417	
12	10/20/2012	Wheaton	Clark	4	7	4	0	17	15	0.235	0	9	5	0.529	
13	10/27/2012	Clark	MIT	7	5	0	3	10	16	0	0.188	4	8	0.4	
14	09/21/2013	Clark	WPI	7	6	1	2	16	7	0.063	0.286	3	3	0.188	
15	09/28/2013	Coast Guard	Clark	2	7	3	0	23	5	0.13	0	14	2	0.609	
16	10/05/2013	Springfield	Clark	4	7	5	0	17	7	0.294	0	12	4	0.706	
17	10/12/2013	Clark	Babson	7	1	1	1	12	30	0.083	0.033	5	13	0.417	
18	10/19/2013	MIT	Clark	3	7	2	0	27	12	0.074	0	10	6	0.37	
19	10/26/2013	Clark	Emerson	7		4	0	18	14	0.222	0	8	7	0.444	
20	11/2/2013	Clark	Wheaton	7	5	1	4	12	14	0.083	0.286	8	8	0.667	
21	09/20/2014	WPI	Clark	6	7	1	0	16	8	0.063	0	8	4	0.5	
22	09/27/2014	Clark	Coast Guard	7	5	2	1	8	18	0.25	0.056	4	4	0.5	
23	10/04/2014	Clark	Springfield	7	4	2	2	19	34	0.105	0.059	10	11	0.526	

The next step of our project was data processing, primarily using pythons, ***pandas,*** and ***numpy*** libraries. Part one of data processing was ensuring that we had a data matrix that was compatible with python's ***sklearn*** library (All values must be numerical and some are unnecessary for prediction). In doing this, we used strategies like column dropping and one-hot encoding. Column dropping is as simple as dropping unnecessary columns for decision-making, and encoding is a tool used to convert categorical information (such as home/away team) into numeric form. The second part of data processing was data optimization, where we looked to tweak and adjust our data matrix to create data trends that were more compatible with our models, thus improving accuracy. In doing this we restructured the data so that statistics were only taken from the perspective of the home team, as well as implementing a strategy called rolling averages, where an average data point is calculated from the prior *x* games (*x* can be any number, in our case it was 3). The next step of our project was the setup and training of the ML models. From the ***sklearn*** library, we got the necessary tools to train the backbone of our Logistic Regression, Decision Tree, and Random Forest Models. Below is an in depth description of how each algorithm functions:

Logistic Regression:

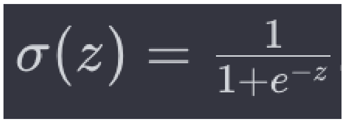
A logistic model is a method used for modeling the probability of a particular event occurring. In a logistic model the log(odds), or the ratio between the probability of the event occurring and the probability of the event not occurring, are determined by a linear function of independent variables, or predictors, in our case the NEWMAC game-to-game data.



$$\log\left(\frac{P(y=1)}{1-P(y=1)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

The linear function is a sum of the predictors (xn) multiplied by their weights. The weight (βn) is determined through the fitting/training process, during fitting the model was given the training set of data which included each predictor as well as the result of the match. To fit the model an optimization algorithm minimizes the difference between the predicted probabilities and the actual outcomes. Sk-learn provides many different optimization algorithms (solvers=), we utilized a ParameterGrid to try multiple algorithms and return the most efficient one. The solver that calculates the weight (βn) that results in a line that best describes our training data is chosen, this process is called maximum likelihood estimation.

After training we move on to testing, the model is provided with the testing data, in this case the game by game statistics of our 2023 NEWMAC season leaving out the game results. The model then plots a line for each instance (individual game) in the data set based on the sum of the data points multiplied by their weights which are calculated during training. The logistic function below is then applied to the linear function of that line.

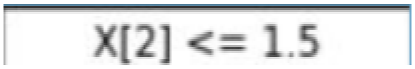


$$\sigma(z) = \frac{1}{1+e^{-z}}$$

This function results in a probability that the instance belongs in the target category, win or tie/loss. Our threshold was set to .5 so if the probability was greater than 50% that the instance should be categorized as a win, the model will predict a win.

Decision Tree:

At its core the decision tree recursively splits a data set until we are left with pure leaf nodes. Pure leaf nodes are nodes that contain only one class of data. Decision trees are composed of two types of nodes: decision nodes and as mentioned earlier leaf nodes. Decision nodes contain a condition that is used to split the data. For example, in our project the root node had the following condition:



X[2] <= 1.5

This means that the data set was split on the condition of goals scored being less than or equal to 1.5. If the condition in the node is satisfied the data point is passed to the left child, if not to the right. This process is repeated until we are left with an optimized amount of pure nodes.

In classifying a new data point, it traverses the tree until it reaches a leaf node. If the leaf node is pure the new data point is assigned the class of the pure node. If the leaf node is not pure a majority vote is taken, and the new data point is assigned to the class of the winner.

But with a new dataset how do we determine how to split the data? How are the conditions for the decision nodes chosen? Data can be split in many ways, so we look to choose the split that maximizes information gain. To calculate information gain we must understand the information in a given state. A state has higher impurity/uncertainty the lower the chance of correctly predicting the class of a random datapoint in the dataset is. There are many ways to measure this level of impurity. With our project we dealt with entropy and gini:

$$Gini = 1 - \sum_j p_j^2$$

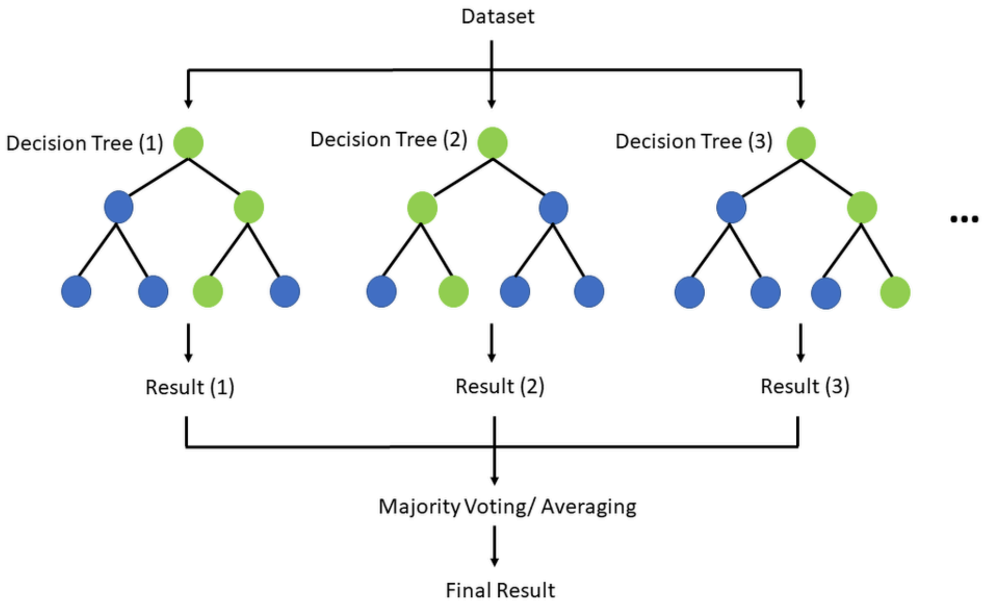
$$Entropy = \sum - p_i \ \log(p_i)$$

$$p_i = \text{probability of class i}$$

The higher the measure of entropy/gini, the higher the uncertainty in a node is. With this we can calculate the uncertainty of every split. To find information gain of a given split, we need to subtract combined impurity of child nodes from that of the parent node. All the splits are tested, and the one that yields the highest information gain is chosen.

Random Forest:

The Random Forest tree is an algorithm aimed to produce results combining multiple models to try and improve the overall accuracy of the result. The Random Forest tree utilizes various decision model trees using random subsets of data.

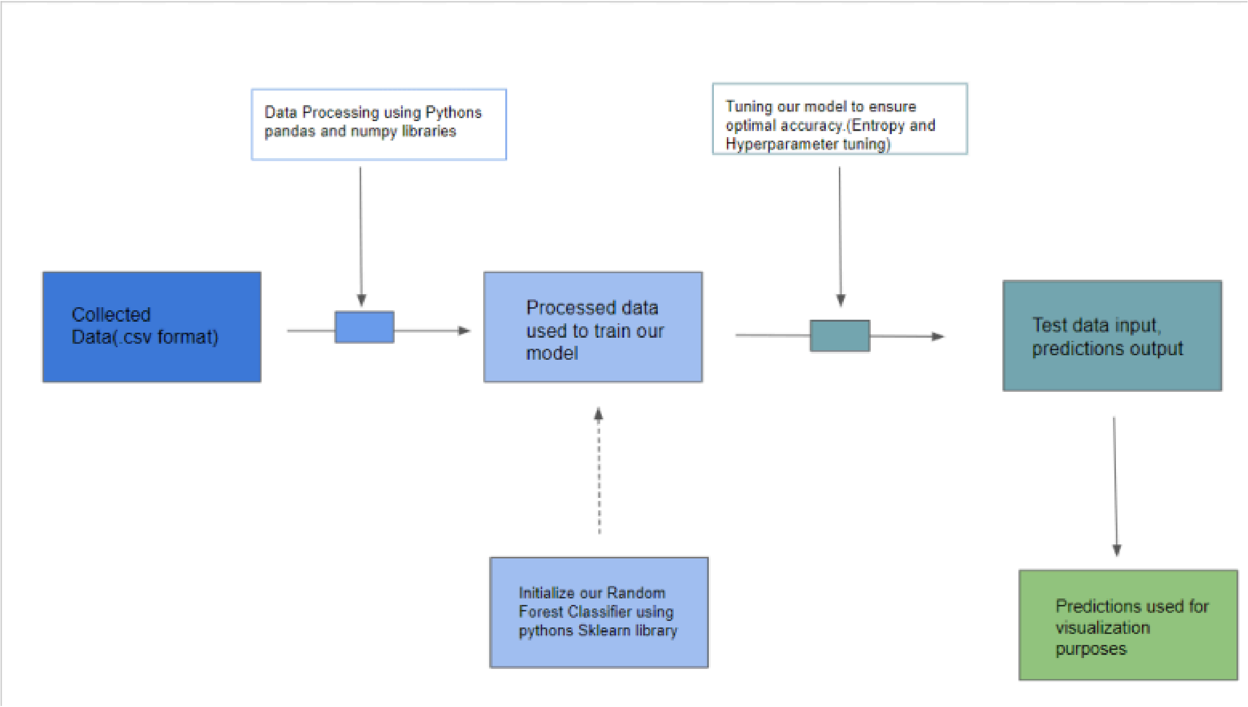


Random Forest

Each decision included in the forest is made up of a bootstrap sample taken from the data set. For each node a subset of characteristics is chosen at random, using gini and entropy (explained by the decision model above) a best split is found between the two characteristics, (in this case goals, SOG%, fouls, Shots, shot percentage, etc.) out of the best chosen a child node is made with this characteristic. The model continuously repeats these steps, and in the end the model decides by vote to see which part of the tree most accurately predicted the outcome, and that prediction is the model's output.

The Random Forest tree introduces "controlled randomness", taking it a step further from the decision tree as it uses multiple decision trees to improve upon the accuracy and results. As the randomness of the algorithm helps us lower bias results, and introduces many variations of the result.

After the setup of the models came the final step, which was the training and tuning of the actual models to ensure higher quality predictions. In tuning we used a strategy called Hyperparameter Tuning, where the parameters of our models(such as depth, gini/entropy, etc) were optimized. Below is a map that highlights what the basic architectural plan for our project was:


Data Processing using Pythons pandas and numpy libraries
Tuning our model to ensure optimal accuracy.(Entropy and Hyperparameter tuning)
Collected Data(.csv format)
Processed data used to train our model
Test data input, predictions output
Initialize our Random Forest Classifier using pythons Sklearn library
Predictions used for visualization purposes

Problems/Challenges

While the process was relatively smooth we did face some challenges along the way. In the data collection period we had to manually input data into our .csv file, which ended up being a very tedious and time consuming process, as the sources for NEWMAC athletics statistics were inconsistent and scattered. Human error also led to some issues with misspellings and spacing, which created some hiccups in the data processing portion. Apart from that there was a learning curve regarding the machine learning algorithms, but that curve was not as steep as we had anticipated.

Milestones

In our initial report we planned to train and get predictions from only one machine learning model. After being told that our scope needed to be bigger we shifted this milestone to training three machine learning models. We met this milestone with our final product, training, tuning, and getting predictions from all three ML models.

Conclusion/Takeaways

The process of creating these algorithms was tough but exciting for us. This project was completely new to us since we had never worked in such a big group, worked with machine learning, or developed this big of a project from scratch. Because of this we were unorganized and everybody was trying to do everything which led to us not progressing fast enough. It was when we separated into our own groups that we truly became efficient and were able to do more than what we thought was possible given the time constraints. We believe that in the future there is room for expansion in this project. We can get into other sports or maybe pitching this idea to our Coach and even the NEWMAC. All in all the project was fun and we would do it all over again.