The NEWMAC Score Predictor

Luke Sheldon, Amechi Aduba, Henry Kuerbis, Diego Marin, Mauro Perez

Men's Soccer Secures NEWMAC Tournament Berth With 2-1 Win Over Salve Regina

1	**Salve Regina** (6-10-2, 0-6-2)	AT	**Springfield** (5-6-5, 2-3-3)	2

Springfield keeper **Noah Pote (Washington, Conn.)** single-handedly saved the Pride's season as it needed a win to make it to the postseason today. With 1:04 to play in the game, a penalty kick was called and Jordan Borges lined up for the kick. Pote dove to his left to stop the initial attempt and then dove again to save Borges' follow up shot to stop Salve Regina from equalizing the match at 2-2.

Introduction

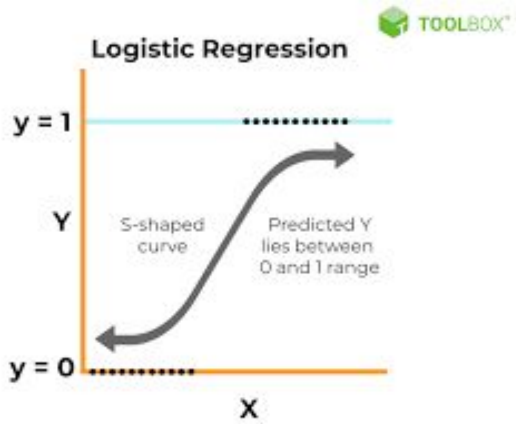
- We aimed to predict the current season's NEWMAC results using three different algorithms, leveraging historical data.
- By comparing the outcomes of each algorithm we can determine which of them is the best or if any of them really work.

Algorithms, and Languages

- Random Forest Algorithm
- Logistic Regression Algorithm
- Decision Tree Algorithm
- Python
- Pandas
- SKLearn

Logistic Regression

Logistic regression utilizes the sigmoid (logistic) function to model the relationship between the independent variables and the probability of a binary outcome to make classifications

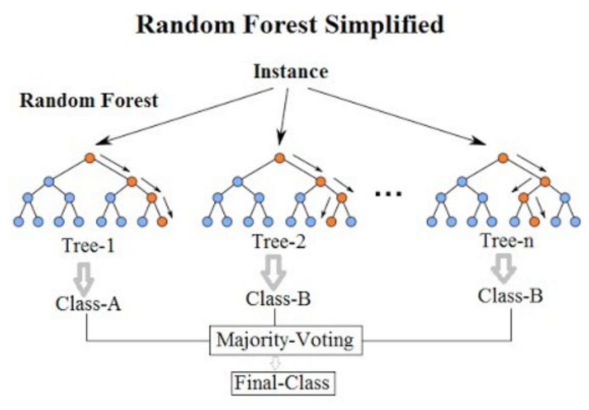


Decision Tree

- A decision based tree where each node acts like an attribute and based on that attribute the model branches off in a particular way based of the decision it took.

Random Forest

- Combines multiple decision tree models using a voting mechanism, with the best possible outputs for every input there is a designated output for that model.



Tuning and Data Processing

Label Encoding

- Encoding is a technique used to convert categorical information into the numerical form necessary for most ML models.
- With Label Encoding, a unique integer or alphabetical ordering represents each label.

Home Team	Away Team
WPI	Clark
Coast Guard	Clark
Clark	Wheaton
Clark	Springfield
MIT	Clark

Home Team	Away Team
0	1
0	1
1	0
1	0
0	1

```
0.5384615384615384
```

One Hot Encoding

- With One Hot Encoding, additional features are created based on the number of unique values in the categorical feature.

Home Team	Away Team
WPI	Clark
Coast Guard	Clark
Clark	Wheaton
Clark	Springfield
MIT	Clark

Home Team_WPI	Home Team_Wheaton	Away Team_Babson	Away Team_Clark
1	0	0	1
0	0	0	1
0	0	0	0
0	0	0	0
0	0	0	1

```
0.6538461538461539
```

Data Trial and Error

- This is not an official term and is pretty straight forward. This is simply the act of messing with the data to try to achieve better accuracy/compatibility.

```
df = df.drop(columns = "Date")
```

```
####MESSING WITH DATA POINTS####
y2 = df_one_hot_t["Result(Clark)"]
specified_cols = ["A_Team SOG %"]
X2 = df_one_hot_t[specified_cols]
```

MinMax Scaling

- Subtracts the minimum value of each feature and then divides by the range of that feature
- Ensures that features with larger scales do not dominate the model's decision-making process

	Venue	Prev_Season	Goals	Shots	Shot %	SOG	SOG %	Corners	Red	Yellow	Fouls	Result
0	0	5	3	8	0.375	6	0.750	7	0	1	6	2
1	0	5	1	16	0.063	0	0.000	2	0	1	12	1
2	1	5	1	7	0.143	3	0.429	1	0	2	0	0
3	1	5	2	13	0.154	7	0.538	6	0	2	0	1
4	0	5	0	7	0.000	3	0.429	2	0	0	9	0
...	...	...	...	...	...	...	...	...	...	...	...	...
87	0	8	0	3	0.000	2	0.667	2	0	3	11	0
88	1	8	0	8	0.000	1	0.125	5	0	0	16	0
89	1	8	1	20	0.050	8	0.400	5	0	1	13	0
90	1	8	0	9	0.000	1	0.111	2	0	0	15	1
91	1	8	2	16	0.125	11	0.688	4	0	0	14	2

	Venue	Prev_Season	Goals	Shots	Shot %	SOG	SOG %	Corners	Red	Yellow	Fouls	Result
0	0.0	0.625	0.75	0.250000	1.000000	0.315789	0.93750	0.500000	0.0	0.25	0.315789	1.0
1	0.0	0.625	0.25	0.583333	0.168000	0.000000	0.00000	0.142857	0.0	0.25	0.631579	0.5
2	1.0	0.625	0.25	0.208333	0.381333	0.157895	0.53625	0.071429	0.0	0.50	0.000000	0.0
3	1.0	0.625	0.50	0.458333	0.410667	0.368421	0.67250	0.428571	0.0	0.50	0.000000	0.5
4	0.0	0.625	0.00	0.208333	0.000000	0.157895	0.53625	0.142857	0.0	0.00	0.473684	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
87	0.0	1.000	0.00	0.041667	0.000000	0.105263	0.83375	0.142857	0.0	0.75	0.578947	0.0
88	1.0	1.000	0.00	0.250000	0.000000	0.052632	0.15625	0.357143	0.0	0.00	0.842105	0.0
89	1.0	1.000	0.25	0.750000	0.133333	0.421053	0.50000	0.357143	0.0	0.25	0.684211	0.0
90	1.0	1.000	0.00	0.291667	0.000000	0.052632	0.13875	0.142857	0.0	0.00	0.789474	0.5
91	1.0	1.000	0.50	0.583333	0.333333	0.578947	0.86000	0.285714	0.0	0.00	0.736842	1.0

Rolling Averages/Data Restructuring

- Data Restructuring is pretty straight forward, but was an essential step before any rolling averages could be taken.
- With rolling averages we take statistics from the previous three games and find the mean with the goal of creating a more defined trend within the data.
- There were some questions regarding rolling averages, but they were resolved.

Home Team	Away Team	H_Prev_Season
WPI	Clark	4
Coast Guard	Clark	7
Clark	Wheaton	5
Clark	Springfield	5
MIT	Clark	3
...	...	...
Babson	Clark	2
WPI	Clark	4
Emerson	Clark	8
Springfield	Clark	5
Clark	Wheaton	7

	Team	Opponent	Venue
0	Clark	WPI	Away
1	Clark	Coast Guard	Away
2	Clark	Wheaton	Home
3	Clark	Springfield	Home
4	Clark	MIT	Away
...	...	...	...
80	Clark	Babson	Away
81	Clark	WPI	Away
82	Clark	Emerson	Away

Goals_rolling	Shots_rolling	Shot %_rolling	SOG_rolling	SOG %_rolling	Corners_rolling	Fouls_rolling
1.666667	10.333333	0.193667	3.000000	0.393000	3.333333	6.000000
1.333333	12.000000	0.120000	3.333333	0.322333	3.000000	4.000000
1.000000	9.000000	0.099000	4.333333	0.465333	3.000000	3.000000
1.333333	10.000000	0.118000	6.000000	0.589000	3.666667	6.000000
1.333333	10.666667	0.111000	6.000000	0.565333	3.666667	8.000000
...	...	...	...	...	...	...
1.000000	7.000000	0.116667	3.333333	0.514000	1.333333	10.333333
0.666667	5.000000	0.083333	2.000000	0.430667	0.666667	10.000000
0.000000	7.333333	0.000000	3.000000	0.439000	1.333333	4.666667
0.000000	10.000000	0.000000	3.666667	0.338000	2.666667	7.333333
0.333333	14.333333	0.019667	5.333333	0.372333	3.666667	5.666667

Hyperparameter Tuning

- Tweaking the parameters of the model itself to ensure optimal performance

```
Best Hyperparameters: {'criterion': 'gini', 'max_depth': None, 'min_samples_split': 2}
```

```
Best Hyperparameters: {'bootstrap': True, 'criterion': 'entropy', 'max_depth': 50, 'max_features': None, 'min_samples_leaf': 2,
'min_samples_split': 7, 'n_estimators': 150}
```

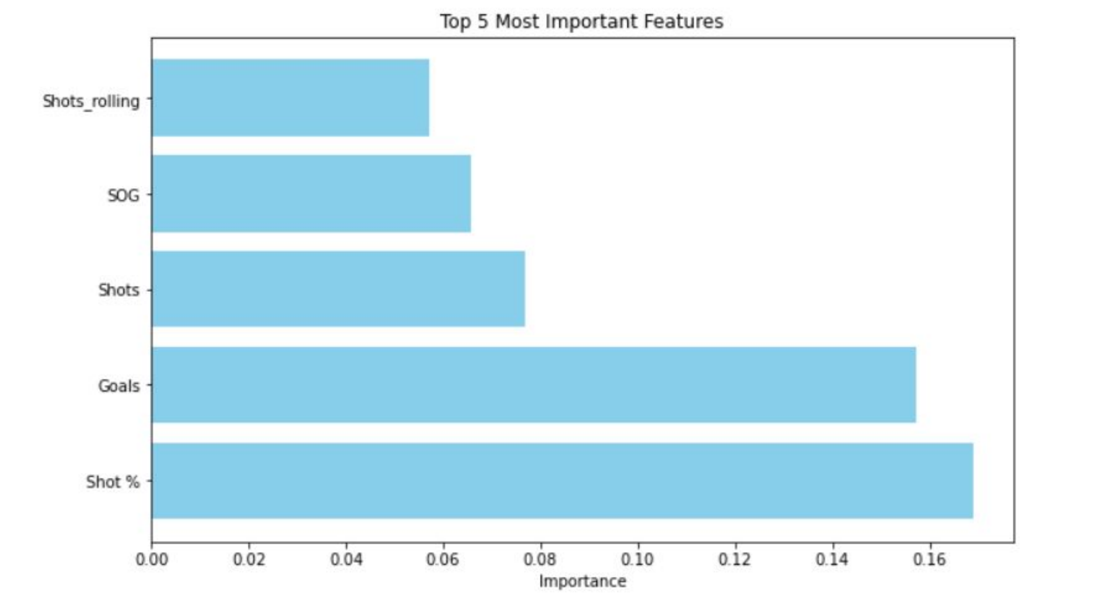
Results and Data Analysis

Date	Home Team	Away Team
09/17/2023	Wheaton	Clark
09/23/2023	Clark	Springfield
09/30/2023	MIT	Clark
10/07/2023	Clark	Coast Guard
10/18/2023	Clark	WPI
10/25/2023	Clark	Babson
10/28/2023	Clark	Emerson

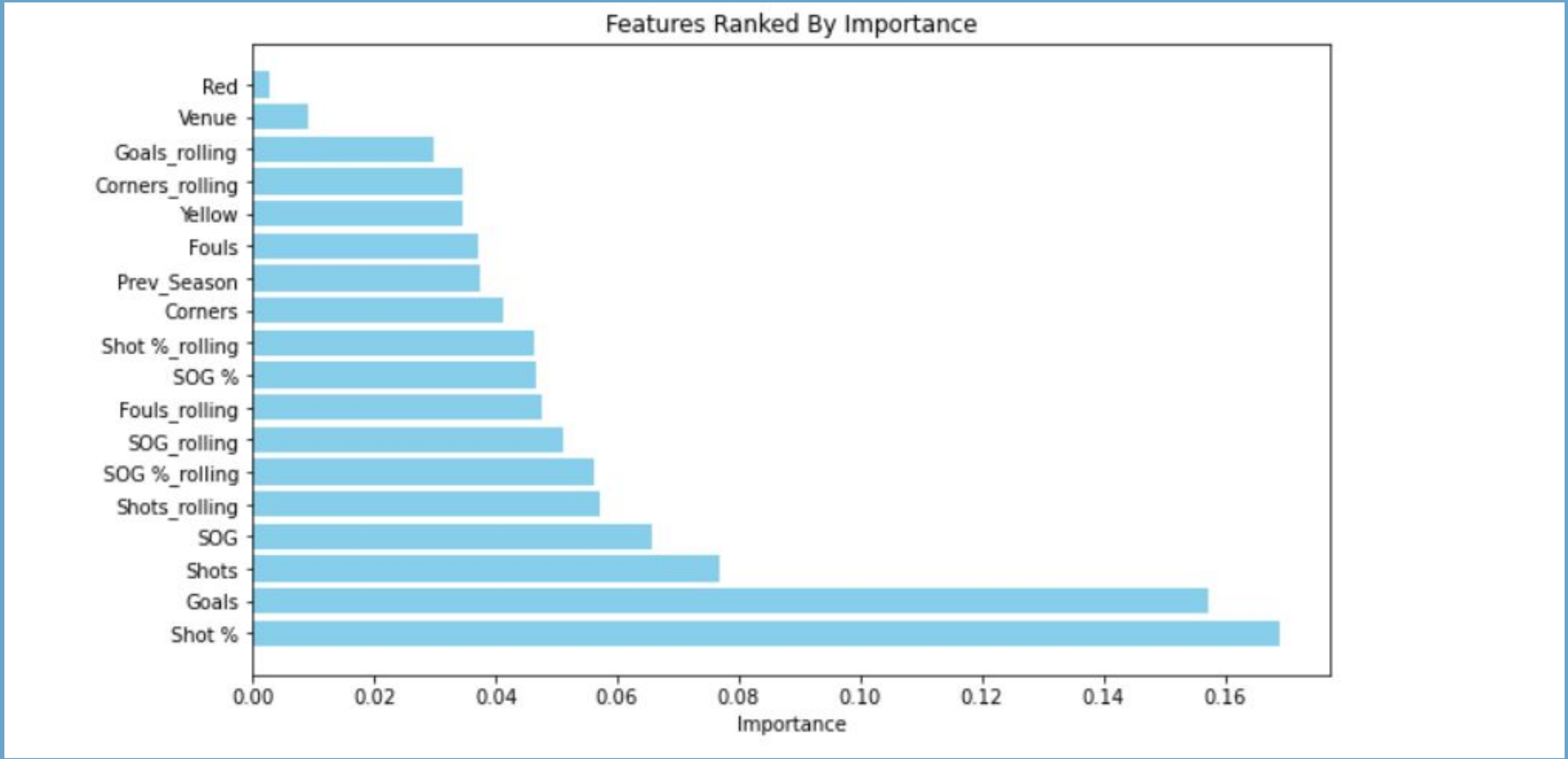
Random Forest

	actual	predicted
85	0	0
86	1	0
87	0	0
88	0	0
89	0	0
90	1	0
91	2	0

```
Final Accuracy: 0.6818181818181818
```



Features Ranked By Importance



Logistic Regression

Multi-class classification- Win vs loss vs tie

	actual	predicted
85	0	0
86	1	0
87	0	0
88	0	0
89	0	0
90	1	0
91	2	2

```
predictions_test=logreg1.predict(X_test1)
accuracy_score(y_test1, predictions_test)
```

✓ 0.0s

```
0.5769230769230769
```

```
from sklearn.metrics import precision_score

precision_score(y_test1, predictions_test, average='micro')
```

✓ 0.0s

```
0.5769230769230769
```

Logistic Regression

Binary classification- Win vs loss or tie

	actual	predicted
85	0	0
86	0	0
87	0	0
88	0	0
89	0	0
90	0	0
91	1	1

```
predictions_test=logreg1.predict(X_test1)
accuracy_score(y_test1, predictions_test)
```

✓ 0.0s

```
0.8928571428571429
```

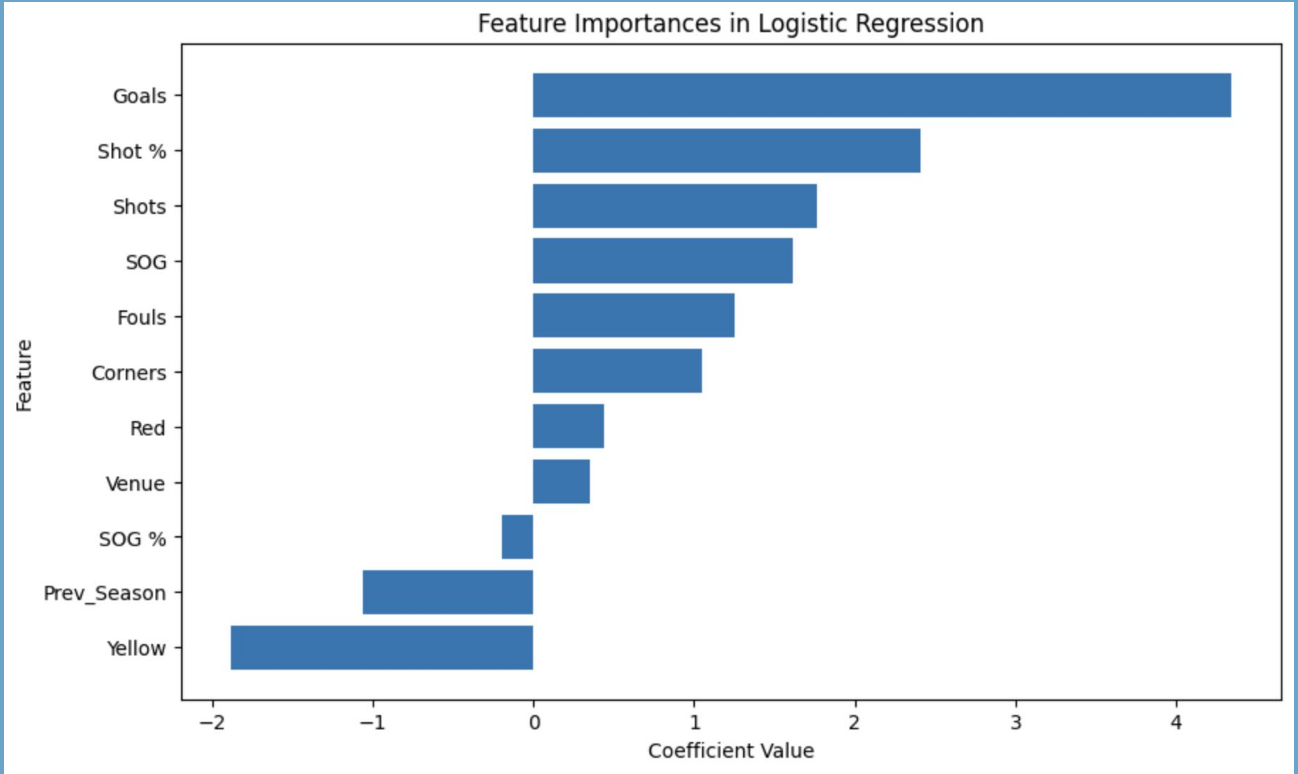
```
from sklearn.metrics import precision_score

precision_score(y_test1, predictions_test, average='micro')
```

✓ 0.0s

```
0.8571428571428571
```

Logistic Regression

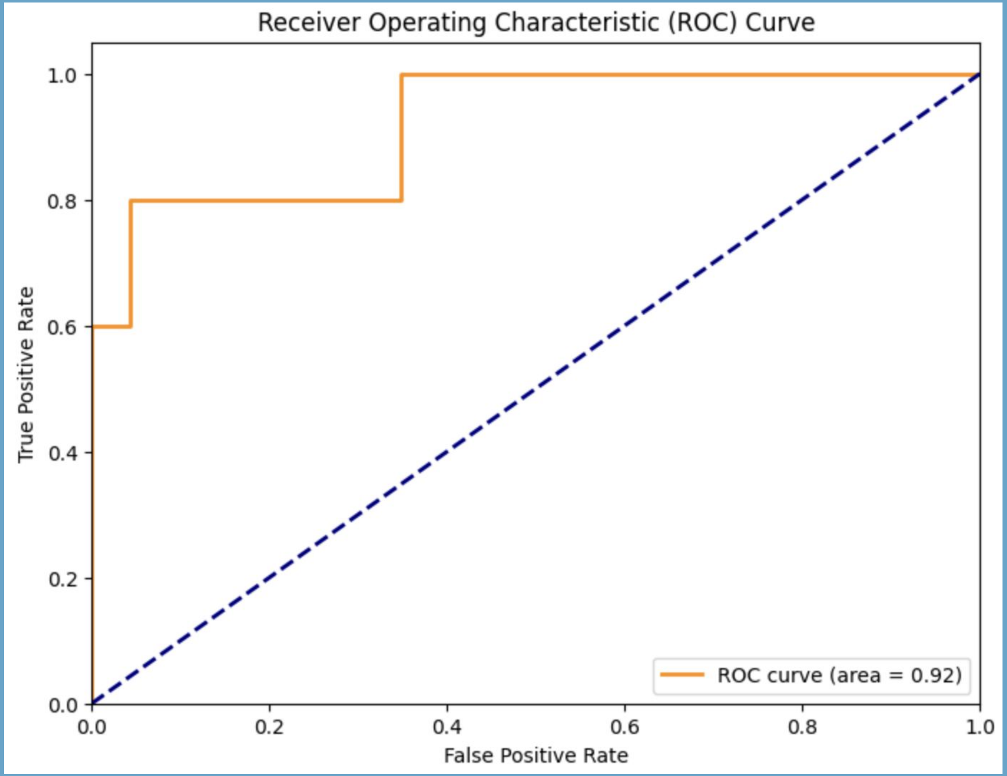


Larger value– more influence

Positive value– as value increase odds of win increase

Negative value– as value increase odds of loss or tie increase

Logistic Regression



True positive – predicted win and we actually won

False positive – predicted win and we actually lost

TPR is high while FPR is low, model generally good at predicting positive and negative cases

Decision Tree

	actual	predicted
85	0	0
86	1	0
87	0	0
88	0	0
89	0	0
90	1	0
91	2	2

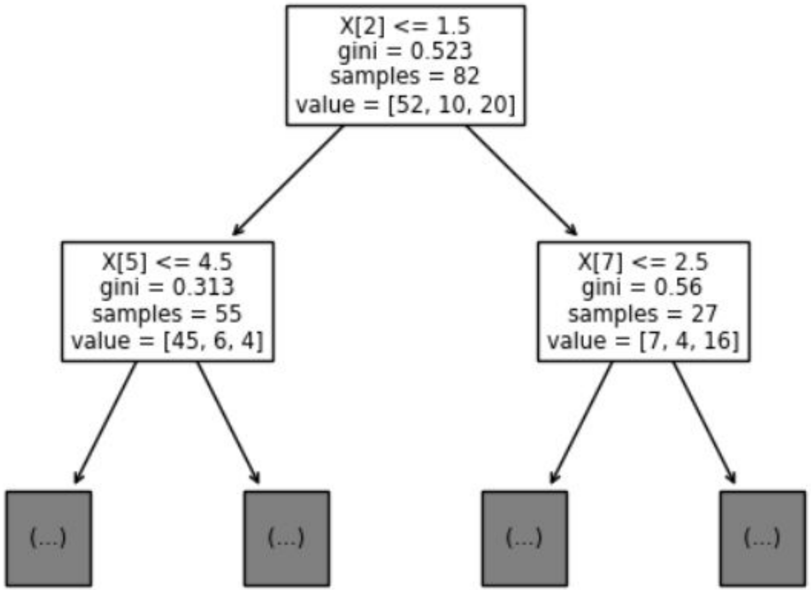
```
predictions_test=FINAL.predict(X_test_)
accuracy_score(y_test_, predictions_test)
```

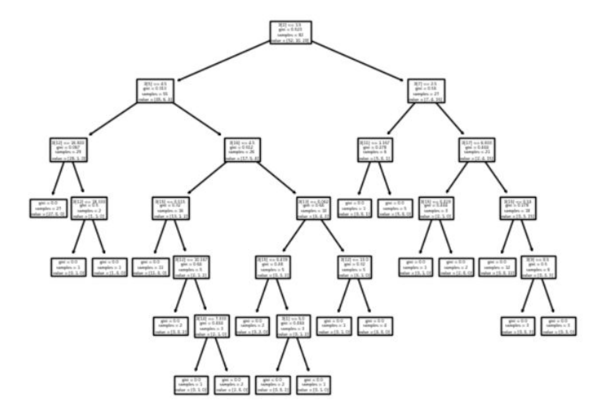
```
0.68
```

```
precision_score(y_test_, predictions_test, average = 'micro')
```

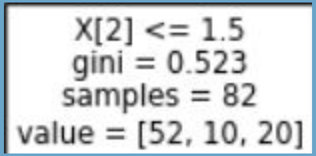
```
0.68
```

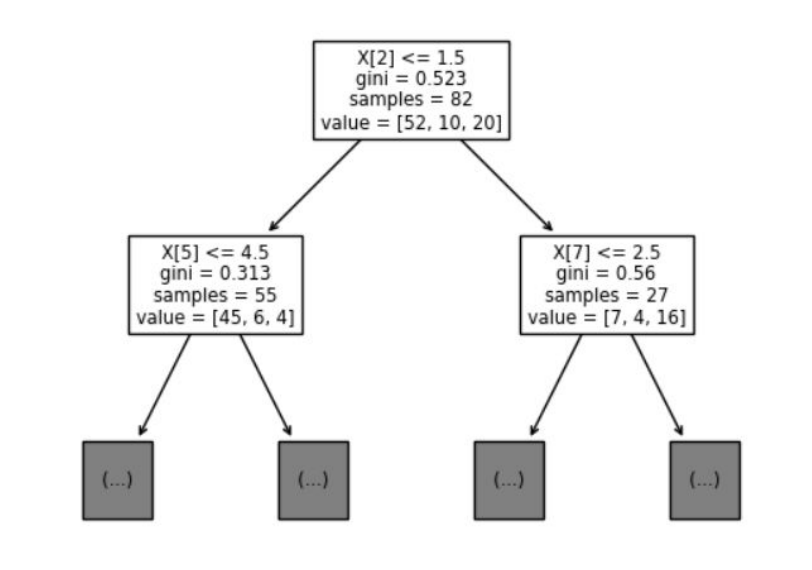
Decision Tree





Decision Tree

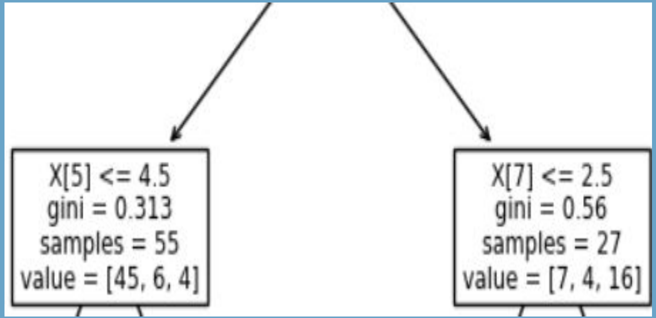


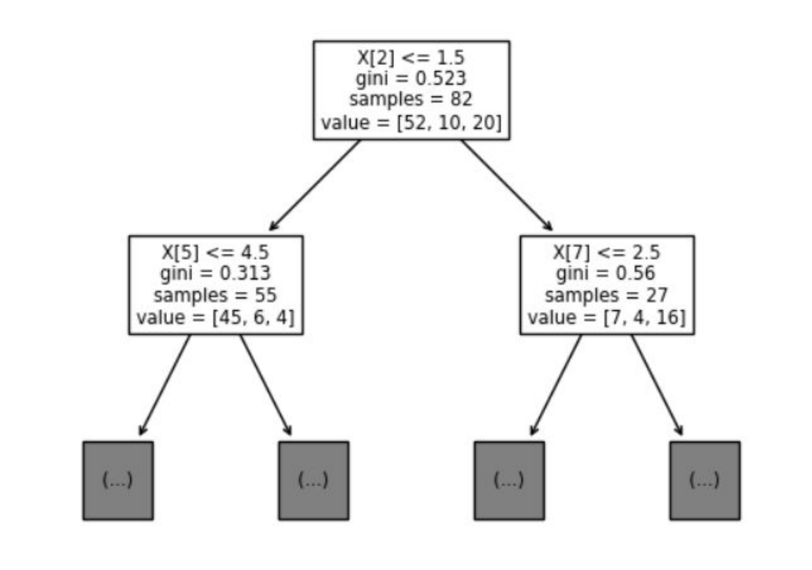


Root Node

- The decision to split is based on goals being less than or equal to 1.5.
- Gini of 0.523
- [52,10,20] class distribution

Decision Tree





Left Child

- The decision to split is based on SOG being less than or equal to 4.5.
- Gini of 0.313
- [45,6,4] class distribution

Right Child

- The decision to split is based on Corners being less than or equal to 2.5.
- Gini of 0.56
- [7,4,16] class distribution

Basic Demo

https://www.loom.com/share/c5dfd69f9b9e4245b8f0387be15ef320?sid=bd4bd6ca-0dc3-4c15-8dc8-68bcc55d9527

Next Steps and Expansion

Next steps and expansion

- Considering other data points such as player index, academic standing, tuition, etc…
- Take into account things that the machine cannot consider because of lack of data.
- Using rolling average data to predict a table of teams based on predicted results
- Out of conference games
- Other sports or schools
- Moving into other types of classifications and predictions using ML knowledge
- Making an ingame outcome model