COURSE PROJECT REPORT: ENGLISH PREMIER LEAGUE PREDICTIONS

Jose Robles

CptS 315

Reet Barik

Washington State University

December 11, 2022

Contents

Introduction 3

Data Mining Task 3

Dataset 4

Methodology 5

Packages 6

Data Partitioning 6

Decision Tree 6

Gini Index 7

Entropy 7

Predictions 7

Evaluation of Methodology 7

Introduction

The English Premier League (EPL) is known as the “topflight” football competition in England. It is the highest level of the football pyramid in England and top talent from around the world come to English clubs to take part in this league. This league has grown to be one of the most popular Football leagues in the world, and with this, there are lots of opportunities for sports betters to make a lot of money. Whether that be predicting a result of a certain fixture, predicting a team’s finish in the league table, or predicting certain stats of players such as goals or assists. With sports betting being more and more popular and advertised within the sport, many programmers have created programs/algorithms to make predictions about certain attributes of the EPL, one of the more popular ones being predicting the number of goals for a player or club. With this paper, the data mining task of predicting goals with a Python program will be explored and explained in further detail.

Data Mining Task

Initially, within my project proposal for CptS 315, I stated that the data mining task for the project was to feed the program a player name, look up the player’s stats in the 2018/19 season, and produce a prediction for the number of goals that player will score in the 2019/20 season. This proposal was most interesting to me, but unfortunately implementing this proved to be more difficult than I had planned initially. Several methods I discovered that could possibly be used to solve this problem were complex and not covered in class. This led to me changing the data mining task.

The new data mining task is as follows. A csv file containing a list of all EPL players and their statistics for the 2018/19 season will be read by the program. Notable attributes are goals scored,

assists, goal involvements, and more. Most importantly though, the attribute of "club_top_rank_in_club_top_scorer" will be used to assign a certain classifier to all players listed in the csv. Depending on their rank (1 to the number of players within the club), they will be assigned a classifier of 2,1,0 where 2 can be thought of as the "best" and 0 as the "worst." For this program, I decided to give players a 2 if they were in the top 5 rank in club top scorers, a 1 if they were in the top 10 but not top 5, and a 0 for anything below that. With these classifiers in place, the program will divide the csv into training and testing data where the program will train to see which attributes are associated with a player with a certain classifier value. Then the program will use the testing data to accurately predict which classifier should be assigned to a player depending on their stats for the 2018/19 season. Essentially, we are predicting where a player is likely to rank amongst their club of top scorers depending on their stats for the season.

Dataset

The dataset to be used for this project can be found at this link: https://footystats.org/download-stats-csv. The Premier League 2018/19 CSVs will be used. There are 5 CVSs relating to the Premier League that contain many stats. The initial plan was to use all of them, but 4 of them ended up being dropped from the project. The england-premier-league-players-2018-to-2019-stats.csv is the sole dataset for this project. This dataset contains columns of redundant information such as a player's birth date, club name, etc. These have been dropped. There are other stats that have been dropped as well such as goals conceded since this is a program that is using attacking stats to make predictions.

After cleaning up the data, a new CSV was produced known as newPlayers.csv. Within this CSV, there are only 19 attributes to consider as opposed to the original CSV's 43. Each attribute

is also related to a player's attacking stats only. This is the dataset that will be used by the Python program.

Within the project there is a file called ModifyTargetAttribute.py. This file is used for modifying the target attribute column within the newPlayers.csv file. Currently, the target attribute column contains classifiers of 2,1,0 that refer to where a player lies within the top goal scorers for their club. The code can be modified to assign different classifiers depending on where a player lies in the rank of club scorers. For example, one can modify the code to assign a classifier of 3 if the clubScorerRank variable is between 3 and 1, a 2 if between 6 and 4, a 3 if between 9 and 7, a 1 if between 12 and 10, and lastly, a 0 for all other ranks. This will change how the program will predict classifiers for the testing data since there are now 4 classifiers to choose from in this example instead of just 3.

It can also be taken a step further if the user wishes to change which column we utilize to determine which classifier to assign. Currently, the rank_in_club_top_scorer column is used. A user can instead change this column to be something like rank_in_league_top_attackers and assign classifiers based on their rank in that column. This can be done by first debugging the code file and expanding the attributes of the player_data Pandas Dataframe variable to see which attributes it contains so a user can know which attribute to choose from.

Methodology

A decision tree was the decided technical approach for this project to compute different accuracies for the testing data by utilizing things learned from the training data. This is a decision support tool that utilizes a tree model that checks for decisions to be made, and their associated consequences. This approach was decided because decision trees were covered in the

class lecture PowerPoints, and I found them particularly interesting in the way they can be used to solve problems/make predictions.

Packages

The Python packages used are pandas for its CSV capabilities, as well as the sklearn package which provides many different machine learning algorithms such as the DecisionTreeClassifier object, as well as other functions that allow us to run our program. See the README associated with the code files for installation steps.

Data Partitioning

In the dataset subsection, it was highlighted how there was an original, more bloated CSV file that was cleaned up with more relevant attributes that is used for this project. This new CSV known as newPlayers.csv is first read into a pandas DataFrame variable using a CSV read function from the pandas library. With this variable, we can split it up into training and testing data, but first, a selection of a target attribute must occur. This will be one column from the CSV that will be stored in a variable, the rest of the columns will be stored in another variable, only then can we split up the data. There is a 70-30 split for data in this program. 70% will be used to train the decision tree, and the remaining 30% will be used as testing data.

Decision Tree

A DecisionTreeClassifier object is utilized from the sklearn library in order to create our tree object. The attributes of the tree are the following:

random_state=100 – This is a pseudo random number generator that is used for random sampling

max_depth = 5 – Maximum depth of the tree, as depth increases, accuracy will increase due to the tree being able to have as many levels as needed to be accurate.

Two objects will be created using the DecisionTreeClassifier() function, one named gini, another named entropy.

Gini Index

The first tree will be created using Gini Index. This is capable of selecting from the n attributes passed in and selecting which attributes should be root or internal nodes. Specifically, the Gini Index can measure how frequent a randomly chosen element would be incorrectly identified, which means that an attribute with lower gini index is preferred for this Gini Index tree.
The tree created using Gini Index will be trained using the training data, and then attempt to make predictions using the testing data.

Entropy

The second tree will be created using entropy. This is the measure of uncertainty of a random variable. We see entropy change when a node within the tree is used to partition training instances into smaller subsets. The concept of information gain is the measure of this said change in entropy. The sklearn library supports creating decision trees using both Gini Index and entropy. By creating two trees with different methods, we hypothesize that we will get different accuracy results for the testing data since both trees will train with the same data and then predict the same testing data.

Predictions

After both trees are built, both will be trained with the same 70-30 split of training and testing data respectively. The trees were trained on the same data to see how they differ from one another when making predictions on the same testing data. To make predictions, internal functions from the sklearn library are called such as train_test_split(), accuracy_score(), and more.

Evaluation of Methodology

To reiterate, we are taking in a list of players which each have a number of different attributes.

One of these attributes is a classifier assigned (2, 1, or 0) based on their club top scorer ranking. The test data will be split into training and testing, and the decision trees created will take in this training data and attempt to correctly assign classifiers to the testing data.

The results of this program are as follows:

```
[569 rows x 20 columns]
Decision tree results using gini index:
Predicted values:
[2 0 0 0 0 0 0 1 2 2 0 2 0 0 0 0 2 0 1 2 0 0 0 0 0 0 0 0 0 2 2 0 0 0 2 2
 0 2 0 0 2 0 0 0 0 0 0 0 2 0 0 0 2 0 0 0 2 1 0 1 0 0 1 0 2 0 2 0 0 0 2 0 0
 0 2 0 0 0 0 0 1 1 0 1 2 1 1 2 2 2 0 2 0 1 2 0 0 0 2 0 0 0 0 0 0 2 0 1 0 1
 1 0 0 0 0 1 0 0 0 2 0 0 0 1 0 0 0 2 0 0 0 2 0 0 2 0 0 2 0 2 2 0 1 0 0 0 2
 1 0 0 0 0 0 0 1 0 0 0 2 0 0 2 0 1 1 1 2 1 2 1]
Confusion Matrix:
[[107   0   0]
 [  1  24   0]
 [  0   0  39]]
Accuracy:  99.41520467836257
Report:               precision    recall  f1-score   support

           0       0.99      1.00      1.00       107
           1       1.00      0.96      0.98        25
           2       1.00      1.00      1.00        39

    accuracy                           0.99       171
   macro avg       1.00      0.99      0.99       171
weighted avg       0.99      0.99      0.99       171

Decision tree results using entropy:
Predicted values:
[2 0 0 0 0 0 0 1 2 2 0 2 0 0 0 0 2 0 1 2 0 0 0 0 0 0 0 0 0 2 2 0 0 0 2 2
 0 2 0 0 2 0 0 0 0 0 0 0 2 0 0 0 2 0 0 0 2 1 0 1 0 0 1 0 2 0 2 0 0 0 2 0 0
 0 2 0 0 0 0 0 1 1 0 1 2 1 1 2 2 2 0 2 0 1 2 0 0 0 2 0 0 0 0 0 0 2 0 1 0 1
 1 0 0 0 0 1 0 0 0 2 0 0 0 1 0 0 0 2 0 0 0 2 0 0 2 0 0 2 0 2 2 0 1 0 0 0 2
 1 0 0 0 0 0 0 1 0 0 0 2 0 0 2 0 1 1 1 2 1 2 1]
Confusion Matrix:
[[107   0   0]
 [  1  24   0]
 [  0   0  39]]
Accuracy:  99.41520467836257
Report:               precision    recall  f1-score   support

           0       0.99      1.00      1.00       107
           1       1.00      0.96      0.98        25
           2       1.00      1.00      1.00        39

    accuracy                           0.99       171
   macro avg       1.00      0.99      0.99       171
weighted avg       0.99      0.99      0.99       171
```

First, the program prints out the predicted values. These are the classifiers the decision tree computed and will assign to the testing data. Next a confusion matrix is printed, this can be used

to understand the trained classifier behavior over the testing data. Essentially, it allows for the visualization of the performance of an algorithm. After that, the tree's accuracy in correctly assigning classifiers to the testing data is shown. For both trees, the accuracy is 99.41520… percent.

Max Depth 5:

- Accuracy 99.41520…

Max Depth 4:

- Accuracy 99.41520…

Max Depth 3:

- Accuracy 99.41520…

Max Depth 2:

- Accuracy 85.38011 …

Max Depth 1:

- Error…

These accuracy results are for both trees. As depth increases, the accuracy increases. This is because the tree is allowed to create more internal nodes that will allow it to more accurately predict the data.

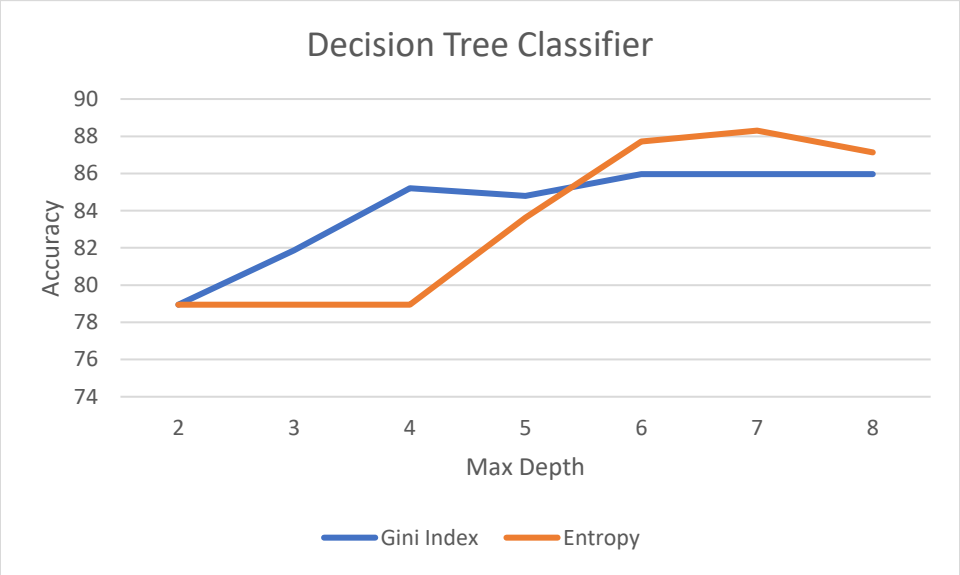
Next, the deletion of the "rank_in_club_top_scorers" column within the csv was necessary. This was to see how accurate the trees would still be if we removed the column it based most of it's classifier assignments on. The attributes of the trees remained unchanged. With this new change in the CSV, the results are as follows:

```
Decision tree results using gini index:
Predicted values:
[2 0 0 0 0 0 1 2 2 2 0 1 1 0 0 0 2 0 1 2 0 1 0 0 0 0 0 0 0 0 2 2 0 0 0 2 2
 0 2 0 0 2 0 0 0 0 0 0 1 2 0 0 1 2 0 0 0 2 1 0 0 0 0 0 0 2 0 2 0 0 0 2 0 0
 0 2 0 0 0 0 0 0 0 1 0 2 2 1 2 1 2 2 0 2 0 2 0 0 0 0 2 0 0 0 0 0 0 2 0 2 0 1
 2 0 0 0 0 0 0 0 0 0 2 0 0 0 1 0 0 0 2 0 0 0 2 0 0 2 0 0 0 0 2 2 0 0 0 0 0 2
 1 0 1 0 0 0 0 0 0 0 0 2 0 0 2 0 1 1 2 2 2 1 1]
Confusion Matrix:
[[101   6   0]
 [  7  10   8]
 [  2   3  34]]
Accuracy:  84.7953216374269
Report:               precision    recall  f1-score   support

           0       0.92      0.94      0.93       107
           1       0.53      0.40      0.45        25
           2       0.81      0.87      0.84        39

    accuracy                           0.85       171
   macro avg       0.75      0.74      0.74       171
weighted avg       0.84      0.85      0.84       171

Decision tree results using entropy:
Predicted values:
[2 0 0 0 0 0 0 2 2 2 0 1 1 0 0 0 2 0 1 2 0 1 0 0 0 0 0 0 0 0 2 2 0 0 0 2 2
 0 2 0 0 2 0 0 0 0 1 0 1 2 0 1 0 2 0 0 0 2 1 1 0 0 0 1 0 2 0 2 0 0 0 2 0 0
 0 2 0 0 0 0 0 0 1 0 1 2 1 1 1 2 2 0 2 1 2 1 0 0 1 2 0 0 0 0 0 0 2 0 1 0 1
 1 0 0 0 0 0 0 0 0 2 0 0 0 1 0 0 0 2 0 0 0 2 0 0 2 1 0 1 0 2 2 0 1 0 0 0 1
 0 0 1 0 0 0 1 1 0 0 0 2 1 0 2 0 1 1 2 1 2 1 1]
Confusion Matrix:
[[95 12  0]
 [ 5 16  4]
 [ 0  7 32]]
Accuracy:  83.62573099415205
Report:               precision    recall  f1-score   support

           0       0.95      0.89      0.92       107
           1       0.46      0.64      0.53        25
           2       0.89      0.82      0.85        39

    accuracy                           0.84       171
   macro avg       0.77      0.78      0.77       171
weighted avg       0.86      0.84      0.85       171
```

With the removal of this column, we see the accuracy fall from ~98% to ~84% and ~83% for gini and entropy respectively. This is still fairly accurate for our decision trees considering that the column that determined which classifier to assign to a player has been removed. Now the tree is fully utilizing the attributes of a player such as goals, assists, etc. in order to determine which classifier to assign that player. When modifying the max depth of the trees we see the following:

Max Depth	2	3	4	5	6	7	8
Gini Index	78.9473	81.8713	85.2105	84.7953	85.9649	85.9649	85.9649
Entropy	78.9473	78.9473	78.9473	83.6257	87.7192	88.304	87.1345

Max depth 1 is excluded because of error. When visualizing the data:



We see that when the max tree depth is lower, less than 6 specifically, the gini index tree is more accurate in assigning classifiers than the entropy tree. But once max depth is greater than or equal to 6, the entropy tree becomes the more accurate tree. Lastly, the highest max depth used was 8 because when raising the depth of the tree higher than that, the accuracies were capped at about 88 percent and would not go any higher.

The initial challenges in this project came from not being familiar with the sklearn machine learning library. Once familiar, more challenges came from the dataset not being compatible with the sklearn library functions due to columns containing strings.

Conclusion

In conclusion, a machine learning algorithm utilizing a decision tree was delivered in order to

assign classifiers for players depending on their stats for the EPL 2018/19 season. The trees were approximately 85% accurate in assigning classifiers to players that had varying statistics. With this project, I was able to connect my enjoyment of soccer/football with my passion for programming as well as the topics learned in this class.