Moneyball and Machine Learning

Members: Matthew Duque and Tavasya Agarwal
Contact: tavasyaagarwal2020@u.northwestern.edu
Course: EECS349, Northwestern University

Abstract:

The Machine Learning task our team chose to undertake is to find out what exactly makes a soccer player's value go up in the future. We aim to create a model that successfully predicts, based on given soccer statistics, whether a player's value is likely to increase in the future or not. This model is particularly relevant in today's sporting climate – today, because of the massive amounts of money being pumped into sports (and specifically the sport of soccer), it is extremely hard for teams with less resources to compete with their richer counterparts. Having a way of identifying players that are currently undervalued by the market, therefore, would be an extremely useful tool with which teams could spend less, but buy more quality.

The first approach we took was to implement decision trees. Given a multitude of statistics, we attempted to see whether we could accurately predict whether a player's value would increase or decrease based on their performance in a given season. We also attempted to use a linear regressor and neural networks in order to improve the accuracy of our predictions, using 10-fold cross validation for the former and leave-one-out cross validation for the latter.

Unsurprisingly (given that a number of teams and people have attempted to tackle this problem), it proved very difficult to achieve a high success rate in predicting which players' prices would rise and whose wouldn't. However, we did manage to achieve an accuracy greater than ZeroR, and found that using decision tree implementations and a linear regressor worked best on our data. We made a lot of progress in changing which features of the data we included, however, and we suspect that it is in this idea that the most success can be found in the future.

Final Report:

Our data was composed of attributes signalling the performance of 234 players in the top tier soccer league in England (the Barclays Premier League) in the 2016/17 season. The attributes were: minutes played, minimum points (from Fantasy leagues), goals, assists, net passes, take-ons, tackles, interceptions, goals conceded, and clean sheets. We later added and removed a number of attributes, and concluded that adding two attributes in particular positively affected our model. These were – 'age', and a binary attribute called 'top6' which was 1 if the player in question played for a 'big 6' club (the traditional best six clubs in England), and 0 if not.

The attribute we trained against and predicted was 'Price Increase?' We scraped the Fantasy league site (with Python and Beautiful Soup) to obtain the prices of players before and after the 16/17 season and determined whether their price increased or didn't as a result of their performance in their season. This value was 1 for players that experienced a price increase, and 0 for those that didn't.

Our preliminary approach was to use Weka's implementation of decision trees. The data we acquired in the beginning did not have the 'age' and 'top6' attributes. We used three approaches: DecisionStump, RandomForest, and Linear Regression.

The ZeroR results are as follows:

```
=== Classifier model (full training set) ===

ZeroR predicts class value: 0.46153846153846156

Time taken to build model: 0 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                 -0.2221
Mean absolute error                      0.4998
Root mean squared error                  0.5014
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances              234
```

We used DecisionStump (a one-level decision tree) to find out which attribute was most important in determining the future valuation of a player. The results are as follows:

```
=== Classifier model (full training set) ===

Decision Stump

Classifications

Mins Played <= 2528.5 : 0.35333333333333333
Mins Played > 2528.5 : 0.6547619047619048
Mins Played is missing : 0.46153846153846156


Time taken to build model: 0 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.1527
Mean absolute error                      0.4743
Root mean squared error                  0.4979
Relative absolute error                 94.9062 %
Root relative squared error             99.2894 %
Total Number of Instances              234
```

We found that the number of minutes played by a player is the best indicator of whether his value goes up or down by the end of the season – this makes intuitive sense, the time a player is given on field is both a signal of his quality and an opportunity to show off his skills. We can also see that while a one-level decision tree does not achieve a 'fantastic' objective accuracy, it does beat ZeroR comfortably.

We also ran the RandomForest and Linear Regressor implementation as follows:

```
=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.14 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.1293
Mean absolute error                      0.469
Root mean squared error                  0.5163
Relative absolute error                 93.8387 %
Root relative squared error            102.9604 %
Total Number of Instances              234
```

Random Forest (above), and Linear Regression (below)

```
=== Classifier model (full training set) ===


Linear Regression Model

Price Increase? =

      0.3948 * Position=Defender,Midfielder,Forward +
      0.0045 * Min Pts +
     -0.0002 * Net Passes +
      0.0025 * Take-Ons +
     -0.004  * Interceptions +
      0.0232 * Clean Sheets +
     -0.1788

Time taken to build model: 0 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.076
Mean absolute error                      0.4845
Root mean squared error                  0.514
Relative absolute error                 96.944  %
Root relative squared error            102.5148 %
Total Number of Instances              234
```

This seemed to imply that our attributes beyond 'minutes played' were being more noisy than useful in prediction, as our correlation coefficient dropped and mean absolute error and root mean squared error increased. Our linear regression also implies that being a goalkeeper seldom leads to increases in value – perhaps this implies that a separate model for goalkeepers is required to determine under- or over-valuation.

We then brainstormed about what features could be most important in determining the future value of a player, and decided on 'age' and 'top6'. Since the career of a player is short, age is often taken as an indicator of whether a player has his 'best years' ahead of or behind him, and hence could be a good predictor of future value. 'Top6', on the other hand, controls for whether a player plays for a big club – it could be true that these players over-perform due to having higher quality players around them, or that they under-perform due to inflated initial valuations just for being on big name teams.

We obtained the values of these 2 new features via more scraping, of Wikipedia. After adding them, we ran RandomForest and Linear Regression again:

```
=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.51 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.1819
Mean absolute error                      0.4609
Root mean squared error                  0.5018
Relative absolute error                 92.2134 %
Root relative squared error            100.0721 %
Total Number of Instances              234
```

Random Forest (above) and Linear Regression (below) after adding two attributes to our model

```
=== Classifier model (full training set) ===


Linear Regression Model

Price Increase? =

      0.3605 * Position=Defender,Midfielder,Forward +
      0.0055 * Min Pts +
     -0.0057 * Interceptions +
      0.0193 * Clean Sheets +
     -0.0231 * Age +
     -0.1561 * Top6 +
      0.5114

Time taken to build model: 0.14 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.157
Mean absolute error                      0.4674
Root mean squared error                  0.5054
Relative absolute error                 93.5266 %
Root relative squared error            100.7979 %
Total Number of Instances              234
```

We found that this increased our correlation and decreased our error values for both implementations, showing that the two attributes were valuable.

Scikit-Learn

After implementing the above models, we attempted to use scikit-learn neural nets to achieve a higher accuracy. We achieved an accuracy of 57% using sklearn's provided multi-layer perceptrons API and a simple implementation of leave-one-out cross validation that we coded up.

```python
import pandas as pd
df = pd.read_csv('ML-csv3.csv')
from sklearn.neural_network import MLPClassifier

attribute_vectors, classifications = [], []
for i, row in df.iterrows():
    attribute_vectors.append(list(row)[:-1])
    classifications.append(list(row)[-1])

corrects = 0
for i, row in enumerate(attribute_vectors):
    test_vector = attribute_vectors[i]
    true_result = classifications[i]
    LOO_vectors = attribute_vectors[:i] + attribute_vectors[(i+1):]
    LOO_classifications = classifications[:i] + classifications[(i+1):]
    clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(20, 2), random_state=1)
    clf.fit(LOO_vectors, LOO_classifications)
    if clf.predict([test_vector]) == true_result:
        corrects = corrects + 1

accuracy = corrects / len(attribute_vectors)
print("accuracy is")
print(accuracy)
```

However, it should be noted that undertaking this with just 234 examples is not ideal, and this approach should in generally be pursued when it is possible to acquire data for many more examples.

Conclusions

While our group certainly has a long way to go in deciding what it is exactly that makes a player's value rise in the future, we have identified certain features that could play a role. Our thinking for the future is that what matters in a model such like this is feature selection, which we tinkered with a large amount and did see significant results. In the movie Moneyball, Brad Pitt finds 'on-base percentage' to be intuitively unimportant but a very good identifier of a baseball player's future potential. Larger datasets, more features, and different implementations of decision trees seem like the most promising methods of finding the equivalent of that attribute for soccer.

Member Roles

Both Matt and Tav worked interchangeably looking over each others' shoulders throughout the project. Tav focussed on Weka implementations and Matt on scikit-learn.