Prediction of value and overall ability of football players

Siwei Wang sw3551

Shantong Xu sx559

1.Introduction

Soccer is one of the most popular sports around the world. Soccer players are paid more money than most of the players in other sport events. Accord to the data in 2017, the average wage of the 500 most paid soccer players is 109K pound per week, and the number of the 100 most paid soccer players is 208K pound per week. Moreover, the transfer fee of soccer players is going much higher in recent years. In 2000, the transfer fee of Luis Figo was 37 million pounds. In 2009, the transfer fee of Cristiano Ronaldo was 80 million pounds, which is more than the double than the maximum in 2000. Moreover, this year, the transfer fee of Neymar reached 198 million pounds. Except for the fact that the maximum transfer fee of soccer players is at an ascending tendency, the number of soccer players with a higher transfer fee is also increasing. Between 2000 and 2010, there were only 3 players worth more than 50 million pounds, while between 2011 and 2015, that number rise to 8. In this single year, there are 8 players with their transfer fee more than 50 million pounds and 3 of them are even more than 90 million pounds. There are several reasons to the increasing of the transfer fee such as the depreciation of currency and the investments of more financial groups.

Another interesting phenomenon on the field is that players are sometimes in the position which is not their registered one. Such position alternation can disorganize the tactics of the opponents and can sometimes be a coup. But the switch of the position can never be arbitrary. It needs evaluation of the players' overall statistics and such switch cannot apply to all the players.

2 Targets of this report

This report focuses on 2 aspects. A dataset of the overall statistics of many players is used and it is downloaded from 'https://www.kaggle.com/thec03u5/fifa-18-demo-player-dataset/data'.

The first target of this report is to make the prediction of the player's actual value to give the managers a criterion of whether a transfer is worthy or not. The prediction is divided into 2 minor parts. The first step is making the prediction of the overall ability based on some vital indicator such as 'acceleration', 'ball control', 'heading accuracy' and so on. The second step is making the prediction of the value based on the overall ability.

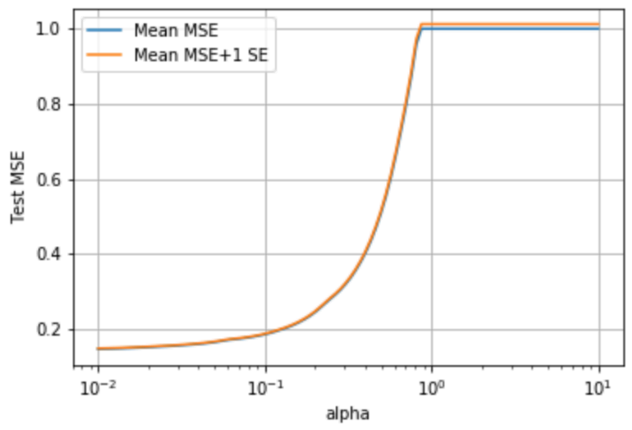
The second target of this report is to make the prediction of the player's favorite position. The dataset has recorded the performance of each player in each different positions of the field. Usually one soccer player should be put in the position which he has the highest performance score, while this is not always the case, since there is correlation between several positions and one can perform better in another position.

3 The prediction of the actual value

3.1 The prediction of the overall ability based on statistics of each player.

When using LASSO, we consider a set of models with different levels of regularization alpha. Higher values of alpha imply greater regularization. We use k-fold cross validation to determine the appropriate alpha. That is, for each alpha value, we evaluate the test error on different training / tests spilt.

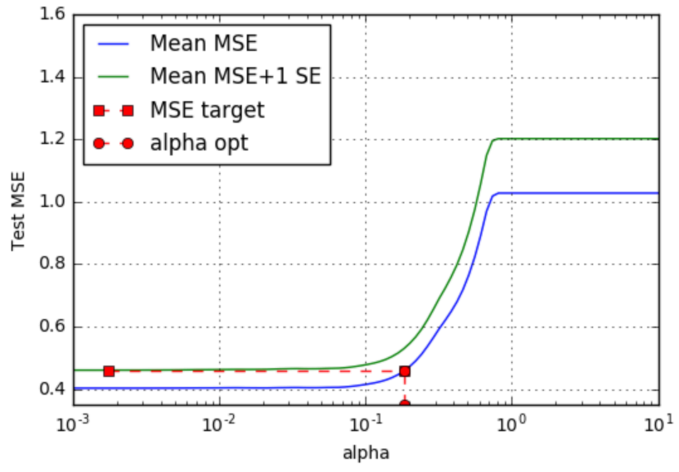
```
nfold = 10
kf = sklearn.model_selection.KFold(n_splits=nfold,shuffle=True)

# Create the LASSO model.  We use the `warm start` parameter so that the fit will start at the pre
vious value.
# This speeds up the fitting.
model = linear_model.Lasso(warm_start=True)

# Regularization values to test
nalpha = 100
alphas = np.logspace(-2,1,nalpha)

# MSE for each alpha and fold value
mse = np.zeros((nalpha,nfold))
for ifold, ind in enumerate(kf.split(X)):


    # Get the training data in the split
    Itr,Its = ind
    X_tr = X[Itr,:]
    y_tr = Y[Itr]
    X_ts = X[Its,:]
    y_ts = Y[Its]

    # Compute the lasso path for the split
    for ia, a in enumerate(alphas):

        # Fit the model on the training data
        model.alpha = a
        model.fit(X_tr,y_tr)

        # Compute the prediction error on the test data
        y_ts_pred = model.predict(X_ts)
        mse[ia,ifold] = np.mean((y_ts_pred-y_ts)**2)
mse_mean = np.mean(mse,axis=1)
mse_std = np.std(mse,axis=1) / np.sqrt(nfold-1)

# Plot the mean MSE and the mean MSE + 1 std dev
plt.semilogx(alphas, mse_mean)
plt.semilogx(alphas, mse_mean+mse_std)
plt.legend(['Mean MSE', 'Mean MSE+1 SE'],loc='upper left')
plt.xlabel('alpha')
plt.ylabel('Test MSE')
plt.grid()
plt.show()
```



Finding the optimal alpha

```
# Find the minimum MSE and MSE target
imin = np.argmin(mse_mean)
mse_tgt = mse_mean[imin] + mse_std[imin]
alpha_min = alphas[imin]

# Find the least complex model with mse_mean < mse_tgt
I = np.where(mse_mean < mse_tgt)[0]
iopt = I[-1]
alpha_opt = alphas[iopt]
print("Optimal alpha = %f" % alpha_opt)

# Plot the mean MSE and the mean MSE + 1 std dev
plt.semilogx(alphas, mse_mean)
plt.semilogx(alphas, mse_mean+mse_std)

# Plot the MSE target
plt.semilogx([alpha_min,alpha_opt], [mse_tgt,mse_tgt], 'rs--')

# Plot the optimal alpha line
plt.semilogx([alpha_opt,alpha_opt], [0.35,mse_mean[iopt]], 'ro--')

plt.legend(['Mean MSE', 'Mean MSE+1 SE', 'MSE target','alpha opt'],loc='upper left')
plt.xlabel('alpha')
plt.ylabel('Test MSE')
plt.ylim([0.35,1.6])
plt.grid()
plt.show()
```

```
Optimal alpha = 0.183074
```



In this section, LASSO is used to select the most weighted factors and eliminate those unimportant indicators. From the result, there are several factors that have a significant influence on the overall ability. Such factors are ‘age’, ‘ball control’, ‘composure’, ‘reactions’ and so on.

```
Age      0.024862
Special  0.052864
Acceleration     0.043361
Ball control     0.230498
Composure        0.193394
Crossing         0.000439
Heading accuracy         0.166758
Interceptions    0.007984
Marking  0.000840
Reactions        0.336106
Short passing    0.094473
Sprint speed     0.062678
Stamina  0.002652
Standing tackle  0.051042
Strength         0.084753
```

Then, a linear LASSO model is created and fit, the accuracy of the prediction is 85.1% which is acceptable.

```
y_pred=model_new.predict(X_new)
RSS=np.mean((y_pred-Y_new)**2)/(np.std(Y_new)**2)
Rsq=1-RSS
print("Rss per sample={0:f}".format(RSS))
print("R^2=          {0:f}".format(Rsq))
```

```
Rss per sample=0.149215
R^2=           0.850785
```

3.2 The prediction of the actual value based on the overall ability

Read the two columns in the dataset and preprocess the data, making the average equals to zero and the square error equals to one. Since there are only one independent variable, we use model selection method to fit.

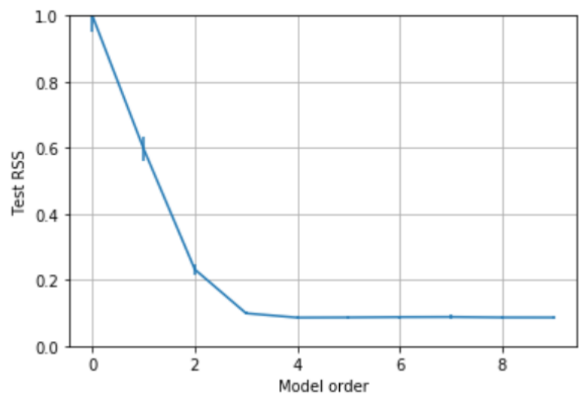
```
y = df['Value'].as_matrix()
x = df['Overall'].as_matrix()
```

```
X = preprocessing.scale(x)
Y = preprocessing.scale(y)
```

Then, a 10-fold cross validation is implemented to find the best order of the model and the best order number is selected according to the RSS.

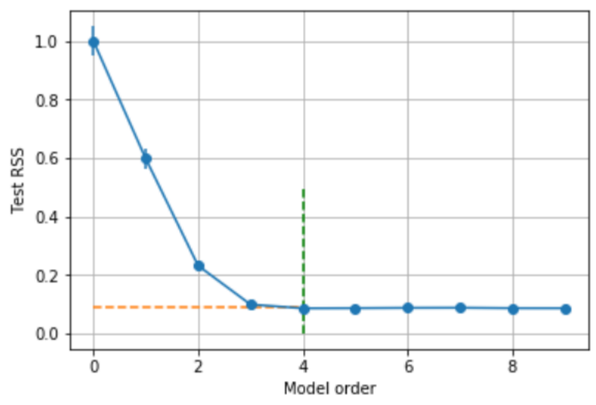
```
nfold = 10
kf = sklearn.model_selection.KFold(n_splits=nfold,shuffle=True)

# Model orders to be tested
dtest = np.arange(0,10)
nd = len(dtest)

# Loop over the folds
RSSts = np.zeros((nd,nfold))
for isplit, Ind in enumerate(kf.split(X)):

    # Get the training data in the split
    Itr, Its = Ind
    xtr = X[Itr]
    ytr = Y[Itr]
    xts = X[Its]
    yts = Y[Its]

    for it, d in enumerate(dtest):

        # Fit data on training data
        beta_hat = poly.polyfit(xtr,ytr,d)

        # Measure RSS on test data
        yhat = poly.polyval(xts,beta_hat)
        RSSts[it,isplit] = np.mean((yhat-yts)**2)
```

```
RSS_mean = np.mean(RSSts,axis=1)
RSS_std  = np.std(RSSts,axis=1) / np.sqrt(nfold-1)
plt.errorbar(dtest, RSS_mean, yerr=RSS_std, fmt='-')
plt.ylim(0,1)
plt.xlabel('Model order')
plt.ylabel('Test RSS')
plt.grid()
```



From the figure above, the selected model order is 4.

```
imin = np.argmin(RSS_mean)
RSS_tgt = RSS_mean[imin] + RSS_std[imin]

# Find the lowest model order below the target
I = np.where(RSS_mean <= RSS_tgt)[0]
iopt = I[0]
dopt = dtest[iopt]

plt.errorbar(dtest, RSS_mean, yerr=RSS_std, fmt='o-')

# Plot the line at the RSS target
plt.plot([dtest[0],dtest[imin]], [RSS_tgt, RSS_tgt], '--')

# Plot the line at the optimal model order
plt.plot([dopt,dopt], [0,0.5], 'g--')

#plt.ylim(0,1)
plt.xlabel('Model order')
plt.ylabel('Test RSS')
plt.grid()

# Print results
print("The estimated model order is %d" % dopt)
```



And the estimated model order is 4 as well.

Then a linear model is trained in fourth order.

```
beta_hat = poly.polyfit(X,Y,dopt)
yhat = poly.polyval(X,beta_hat)

RSS_tr=np.mean((yhat-Y)**2)/(np.std(Y)**2)
Rsq_tr=1-RSS_tr
print("Rss per sample={0:f}".format(RSS_tr))
print("R^2=           {0:f}".format(Rsq_tr))
```

```
Rss per sample=0.084495
R^2=           0.915505
```

The accuracy of this fourth order model is 91.6% which is quite accurate.

4 The prediction of players' favorite position

After reading the dataset into a dataframe, get all the different name of the positions. Goalkeeper is excluded because it's a special position and there is zero possibility for a goalkeeper to switch to any other position.

```
positionvals, positions = np.unique(position, return_inverse = True)
print(positionvals)
```

Get the value of ability of each position, preprocess the data and use a decision tree to make the prediction. The accuracy of the decision tree model is 99.8%.

```
xnames = df.columns[47:73]
```

```
X = np.array(df[xnames].values)
Xs = preprocessing.scale(X)
```

```
dtree = tree.DecisionTreeClassifier()
dtree = dtree.fit(Xs, y)
```

```
yhat2 = dtree.predict(Xs)
```

```
acc2 = np.mean(yhat2 == y)
print('Accurary = {0:f}'.format(acc2))
```

```
Accurary = 0.998151
```

5 Conclusion

In this project, players' overall abilities are predicted based on many other statistics and the accuracy of this prediction is 85.1%. secondly, players' personal values are predicted based on their own overall ability, and the accuracy is 91.6%. Moreover, players' favorite positions are predicted according to their performance on each different position and the accuracy is 99.8%. We use the machine leaning method to solve daily problems, like knowing that there is a fourth order polynomial relationship between player's personal overall ability and his value, which is very interesting. In doing this project, a deeper understanding of several machine learning approaches is added to our mind, including but not limiting to the LASSO, regression and decision tree, but many other as well.