Predicting Player Features of the English Premier League with scikit-Learn

2018-07-09 BY GENE

In a previous installment, I harvested football stats for every player in the English Premier League and saved them as CSV files (separated by tabs actually).

In this installment, I use the same data but predict the player rating (a floating point number) and field position (a string) with the techniques of linear regression and k nearest neighbors, respectively.

First, we read-in the data:

```
import pandas as pd

summary   = pd.read_csv('~/sandbox/dev/Games/Football/Summary-2015-16-proce
offensive = pd.read_csv('~/sandbox/dev/Games/Football/Offensive-2015-16-pro
passing   = pd.read_csv('~/sandbox/dev/Games/Football/Passing-2015-16-proce
defensive = pd.read_csv('~/sandbox/dev/Games/Football/Defensive-2015-16-pro
```

Fortunately, by inspecting, we can see that each of these files is in the same player order (by Rating):

```
>>> print summary.shape
(298, 13)
>>> summary.head()
              Name                  Player   Apps  Mins  Goals  Assists  Yel
R
1    Riyad Mahrez  Leicester, 25, AM(CLR)  36(1)  3058   17.0     11.0  1.0
2   Dimitri Payet    West Ham, 29, M(CLR)  29(1)  2573    9.0     12.0  3.0
3  Alexis Sánchez   Arsenal, 27, M(LR),FW  28(2)  2446   13.0      4.0  1.0
4   Mousa Dembélé    Tottenham, 29, M(CR)  27(2)  2273    3.0      1.0  3.0
5      Mesut Özil     Arsenal, 27, M(CLR)     35  3049    6.0     19.0  4.0

   Red  SpG   PS%  AerialsWon  MotM  Rating
R
1  NaN  2.3  73.6         0.9  10.0    7.84
2  NaN  2.3  80.2         0.1   5.0    7.74
3  NaN  3.6  79.6         0.7   6.0    7.72
4  NaN  0.8  90.0         1.1   4.0    7.69
5  NaN  1.4  86.3         0.1   6.0    7.66

>>> offensive.shape
(298, 14)
>>> offensive.head()
              Name                  Player   Apps  Mins  Goals  Assists  SpG
R
1    Riyad Mahrez  Leicester, 25, AM(CLR)  36(1)  3058   17.0     11.0  2.3
2   Dimitri Payet    West Ham, 29, M(CLR)  29(1)  2573    9.0     12.0  2.3
3  Alexis Sánchez   Arsenal, 27, M(LR),FW  28(2)  2446   13.0      4.0  3.6
4   Mousa Dembélé    Tottenham, 29, M(CR)  27(2)  2273    3.0      1.0  0.8
5      Mesut Özil     Arsenal, 27, M(CLR)     35  3049    6.0     19.0  1.4

   KeyP  Drb  Fouled  Off  Disp  UnsTch  Rating
R
1   1.8  3.5     2.2  0.1   2.4     2.1    7.84
2   4.0  2.2     1.3  0.1   1.7     1.8    7.74
3   2.1  3.4     2.2  0.4   3.3     2.4    7.72
4   1.0  2.9     1.1  0.1   2.2     0.8    7.69
5   4.2  1.3     1.2  0.4   1.7     2.0    7.66
```

Next, after reading-in the data, is to put it in one single dataframe with all the columns and zeros for empty (“na”) values:

```
data = summary

feature_cols = list(offensive)
data[feature_cols] = offensive[feature_cols]

feature_cols = list(passing)
data[feature_cols] = passing[feature_cols]

feature_cols = list(defensive)
data[feature_cols] = defensive[feature_cols]

data = data.fillna(0)
```

There are a couple columns embedded in the “Player” value for each record. This includes the team name, the player age and the field position. This code extracts those for use in our modelling:

```
i = 0
for player in data.Player:
    i = i + 1
    [team, age, position] = player.split(", ")
    data.set_value(i, 'team', team)
    data.set_value(i, 'age', int(age))
    data.set_value(i, 'position', position)

import re

i = 0
for position in data.position:
    i = i + 1
    posn = re.sub('\(.+\)', '', position)
    data.set_value(i, 'posn', posn)
```

Now the data features are all these:

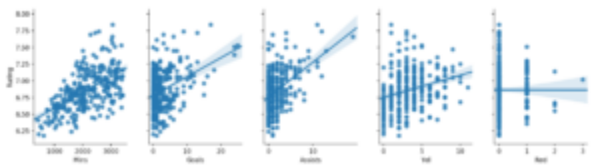
```
>>> list(data)
['Name', 'Player', 'Apps', 'Mins', 'Goals', 'Assists', 'Yel', 'Red', 'SpG',
```

Inspecting a single row:

```
>>> data.loc[[1]]
           Name                   Player   Apps  Mins  Goals  Assists  Yel
R
1  Riyad Mahrez  Leicester, 25, AM(CLR)  36(1)  3058   17.0     11.0  1.0

   Red  SpG   PS%  ...   Inter  Fouls  Offsides  Clear  Blocks  OwnG  \
R                  ...
1  0.0  2.3  73.6  ...     1.0    0.5       0.0    0.4     0.1   0.0

        team   age  position  posn
R
1  Leicester  25.0   AM(CLR)    AM

[1 rows x 34 columns]
>>> data.loc[[1]]['Rating']
R
1    7.84
Name: Rating, dtype: float64
```

Ok. So far so good. How about we visualize the data to get a feel for what relationships might hold?

```
import matplotlib.pyplot as plt
import seaborn as sns

response_col = ['Rating']

def pair_features():
    sns.pairplot(data, x_vars=feature_cols, y_vars=response_col, kind='reg'
    plt.show()

feature_cols = ['Mins', 'Goals', 'Assists', 'Yel', 'Red']
pair_features()
feature_cols = ['SpG', 'PS%', 'AerialsWon', 'MotM', 'KeyP']
pair_features()
feature_cols = ['Drb', 'Fouled', 'Off', 'Disp', 'UnsTch']
pair_features()
feature_cols = ['AvgP', 'Crosses', 'LongB', 'ThrB', 'Tackles']
pair_features()
feature_cols = ['Inter', 'Fouls', 'Offsides', 'Clear', 'Blocks']
pair_features()
feature_cols = ['OwnG', 'age']
pair_features()
```



This shows us that most of the numeric features will contribute to our ML training, but in addition to the non-numeric features, we can remove a couple that will probably not contribute to meaningful prediction. Here is the code to remove those column names and set the **X** and **y** elements to the proper values:

```
feature_cols = list(data)
for col in ['Name','Player','Apps','Rating','team','position','Red','Off','
    feature_cols.remove(col)

X = data[feature_cols]
y = data[response_col]
```

Now we can perform the Rating prediction with linear regression:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

print X_train.shape, y_train.shape
print X_test.shape, y_test.shape

from sklearn.linear_model import LinearRegression

estimator = LinearRegression()
estimator.fit(X_train, y_train)

X_test.iloc[[1]]            # R=235
datum = data.iloc[[234]]    # Bakary Sako - M
X = datum[feature_cols]
y = datum[response_col]
estimator.predict(X)        # array([[6.63099864]])
```

This is a pretty close prediction! The true response value is:

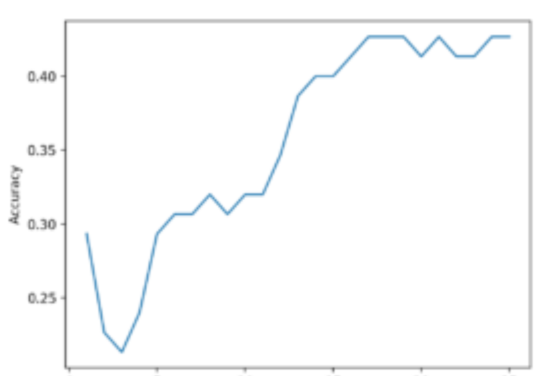
```
>>> X_test.iloc[[1]]  # R=235, Name=Bakary Sako, posn=M
>>> data.iloc[[234]].Rating
R
235    6.61
```

Ok! What about predicting a non-numerical response? For this, we will use the k-nearest-neighbors measure:

```
from sklearn.neighbors import KNeighborsClassifier

feature_cols = list(data) for col in ['Name','Player','Apps','team','positi
    feature_cols.remove(col)

response_col = ['posn']

estimator = KNeighborsClassifier(n_neighbors=18)
estimator.fit(X_train, y_train)

datum = data.iloc[[163]]    # Gary Cahill - D
X = datum[feature_cols]
estimator.predict(X)        # array(['D'], dtype=object)

datum = data.iloc[[51]]     # Erik Lamela - AM
X = datum[feature_cols]
estimator.predict(X)        # array(['AM,FW'], dtype=object)
```

Ha! Decent results!

Here is the accuracy of this model plotted for increasing values of k:



FILED UNDER: DATA, SOFTWARE
TAGGED WITH: FOOTBALL, PYTHON, SCIKIT-LEARN, SOCCER

Epistemologist-at-large

^ Top