assignment_28.1

January 13, 2019

0.1 Predicting score using K-nearest neigbor

In this assignment, I will be using the K-nearest neighbors algorithm to predict how many points NBA players scored in the 2013-2014 season

```
In [1]: import os
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn import model_selection
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import classification_report
        from sklearn.metrics import accuracy_score
        from sklearn.neighbors import KNeighborsRegressor
```

```
In [2]: pd.set_option('display.height', 1000)
        pd.set_option('display.max_rows', 1000)
        pd.set_option('display.max_columns', 500)
        pd.set_option('display.width', 1500)
```

0.2 Load NBA dataset

```
In [3]: nba_data = pd.read_csv('nba_2013.csv')
        nba_data.head()
```

```
Out[3]:           player pos  age bref_team_id   g  gs    mp   fg   fga    fg.  x3p  x3pa
        0     Quincy Acy  SF   23          TOT  63   0   847   66   141  0.468    4    15  0.26
        1   Steven Adams   C   20          OKC  81  20  1197   93   185  0.503    0     0
        2    Jeff Adrien  PF   27          TOT  53  12   961  143   275  0.520    0     0
        3  Arron Afflalo  SG   28          ORL  73  73  2552  464  1011  0.459  128   300  0.42
        4  Alexis Ajinca   C   25          NOP  56  30   951  136   249  0.546    0     1  0.00
```

0.3 Perform analysis of dataset

```
In [4]: nba_data.describe()
```

```
Out[4]:               age           g          gs           mp          fg          fga
        count  481.000000  481.000000  481.000000   481.000000  481.000000   481.000000  479.00
```

```
mean    26.509356   53.253638   25.571726  1237.386694  192.881497   424.463617     0.43
std      4.198265   25.322711   29.658465   897.258840  171.832793   368.850833     0.09
min     19.000000    1.000000    0.000000     1.000000    0.000000     0.000000     0.00
25%     23.000000   32.000000    0.000000   388.000000   47.000000   110.000000     0.40
50%     26.000000   61.000000   10.000000  1141.000000  146.000000   332.000000     0.43
75%     29.000000   76.000000   54.000000  2016.000000  307.000000   672.000000     0.47
max     39.000000   83.000000   82.000000  3122.000000  849.000000  1688.000000     1.00
```

```
In [5]: nba_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 481 entries, 0 to 480
Data columns (total 31 columns):
player          481 non-null object
pos             481 non-null object
age             481 non-null int64
bref_team_id    481 non-null object
g               481 non-null int64
gs              481 non-null int64
mp              481 non-null int64
fg              481 non-null int64
fga             481 non-null int64
fg.             479 non-null float64
x3p             481 non-null int64
x3pa            481 non-null int64
x3p.            414 non-null float64
x2p             481 non-null int64
x2pa            481 non-null int64
x2p.            478 non-null float64
efg.            479 non-null float64
ft              481 non-null int64
fta             481 non-null int64
ft.             461 non-null float64
orb             481 non-null int64
drb             481 non-null int64
trb             481 non-null int64
ast             481 non-null int64
stl             481 non-null int64
blk             481 non-null int64
tov             481 non-null int64
pf              481 non-null int64
pts             481 non-null int64
season          481 non-null object
season_end      481 non-null int64
dtypes: float64(5), int64(22), object(4)
memory usage: 116.6+ KB
```

0.4 Find missing values in columns and fill the missing values

In [6]:
```
nba_data.isnull().any()
```

Out[6]:
```
player          False
pos             False
age             False
bref_team_id    False
g               False
gs              False
mp              False
fg              False
fga             False
fg.              True
x3p             False
x3pa            False
x3p.             True
x2p             False
x2pa            False
x2p.             True
efg.             True
ft              False
fta             False
ft.              True
orb             False
drb             False
trb             False
ast             False
stl             False
blk             False
tov             False
pf              False
pts             False
season          False
season_end      False
dtype: bool
```

In [7]:
```
#fill missing values with mean
nba_data.fillna(nba_data.mean(), inplace = True)
```

0.5 Perform One Hot Encoding on categorical fields pos, bref_team_id

In [8]:
```
nba_data = pd.get_dummies(nba_data, columns=['pos',  'bref_team_id'])
```

In [9]:
```
nba_data.head()
```

Out[9]:

	player	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	x2p	x2pa
0	Quincy Acy	23	63	0	847	66	141	0.468	4	15	0.266667	62	126
1	Steven Adams	20	81	20	1197	93	185	0.503	0	0	0.285111	93	185

```
2    Jeff Adrien   27  53  12   961  143   275  0.520    0     0  0.285111  143   275
3  Arron Afflalo   28  73  73  2552  464  1011  0.459  128   300  0.426667  336   711
4  Alexis Ajinca   25  56  30   951  136   249  0.546    0     1  0.000000  136   248
```

0.6 Prepare data for feature X and target Y

1. Get feature X from nba_data by removing unwanted field 'player', 'season', 'season_end', 'pts' from nba_data
2. Get target Y by taking only field 'pts' from nba_data

In [10]:
```
X = nba_data.drop(['player', 'season', 'season_end', 'pts' ], axis=1)
Y = nba_data[[ 'pts']]
```

In [11]:
```
X.head()
```

Out[11]:
```
   age   g  gs    mp   fg   fga    fg.  x3p  x3pa      x3p.  x2p  x2pa      x2p.  ef
0   23  63   0   847   66   141  0.468    4    15  0.266667   62   126  0.492063  0.4
1   20  81  20  1197   93   185  0.503    0     0  0.285111   93   185  0.502703  0.5
2   27  53  12   961  143   275  0.520    0     0  0.285111  143   275  0.520000  0.5
3   28  73  73  2552  464  1011  0.459  128   300  0.426667  336   711  0.472574  0.5
4   25  56  30   951  136   249  0.546    0     1  0.000000  136   248  0.548387  0.5
```

In [12]:
```
Y.head()
```

Out[12]:
```
    pts
0   171
1   265
2   362
3  1330
4   328
```

0.7 Divide feature and target into train and test dataset using train_test_split

In [13]:
```
# Import train_test_split
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_sta
```

```
E:\anaconda\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: This module w
  "This module will be removed in 0.20.", DeprecationWarning)
```

0.8 Apply KNeighborsRegressor on train data and then calculate accurracy on test data using different values of K

In [14]:
```
#fitting the model
for K in range(25):
    K_value = K+1
    neighbor = KNeighborsRegressor(n_neighbors = K_value, weights='uniform', algorithm
```

```
        neighbor.fit(X_train, y_train)
        y_pred = neighbor.predict(X_test)
        print("Accuracy is ",neighbor.score(X_test,y_test) ," for K-Value:",K_value)
```

```
Accuracy is  0.9550293841767484  for K-Value: 1
Accuracy is  0.9667533821036304  for K-Value: 2
Accuracy is  0.9722822534883708  for K-Value: 3
Accuracy is  0.9726339258124398  for K-Value: 4
Accuracy is  0.9733831983576475  for K-Value: 5
Accuracy is  0.9756387408444478  for K-Value: 6
Accuracy is  0.9746027472541899  for K-Value: 7
Accuracy is  0.9770228646408327  for K-Value: 8
Accuracy is  0.9763740703273166  for K-Value: 9
Accuracy is  0.9748523596389462  for K-Value: 10
Accuracy is  0.9751299739952604  for K-Value: 11
Accuracy is  0.9745794319825458  for K-Value: 12
Accuracy is  0.9739574285892518  for K-Value: 13
Accuracy is  0.9729335547824521  for K-Value: 14
Accuracy is  0.9712182744081604  for K-Value: 15
Accuracy is  0.9700118313803022  for K-Value: 16
Accuracy is  0.9691800087657443  for K-Value: 17
Accuracy is  0.9676174344271444  for K-Value: 18
Accuracy is  0.9651109011024993  for K-Value: 19
Accuracy is  0.9637795637548277  for K-Value: 20
Accuracy is  0.9622185118638689  for K-Value: 21
Accuracy is  0.9621200499613172  for K-Value: 22
Accuracy is  0.9616378565744167  for K-Value: 23
Accuracy is  0.9597992719248764  for K-Value: 24
Accuracy is  0.9589699778682376  for K-Value: 25
```

0.9 Conclusion:

From the result it is evident that accuracy is 97% for K value between 4 and 16, I can take K value as 8 as as gives maximum accuracy is 97.70%