Chapter 5 - Ex2: NBA Players

Cho dữ liệu nba_2013.csv

Sử dụng thuật toán KNN để dự đoán số điểm (points) mà các cầu thủ NBA ghi được trong mùa giải 2013-2014.

Mỗi hàng trong dữ liệu chứa thông tin về player thực hiện trong mùa giải 2013-2014 NBA. (với player -- tên player/ pos -- vị trí của player/ g -- số trận mà player đã tham gia/ gs -- số trận mà player đã bắt đầu/ pts -- tổng số point mà player đã ghi được)

1. Đọc dữ liệu và gán cho biến data. Xem thông tin data: shape, type, head(), tail(), info. Tiền xử lý dữ liệu (nếu cần)
2. Tạo **inputs** data với các cột không có giá trị null trừ cột 'player', 'bref_team_id', 'season', 'season_end', 'pts', và **outputs** data với 1 cột là 'pts' => Vẽ biểu đồ quan sát mối liên hệ giữa inputs và outputs data
3. Từ inputs data và outputs data => Tạo X_train, X_test, y_train, y_test với tỷ lệ 80:20
4. Thực hiện KNN với X_train, y_train
5. Dự đoán y từ X_test => so sánh với y_test
6. Xem kết quả => Nhận xét model
7. Ghi model nếu model phù hợp

In [0]:
```
from google.colab import drive
drive.mount("/content/gdrive", force_remount=True)

path = '/content/gdrive/My Drive/LDS6_MachineLearning/'
```

```
Mounted at /content/gdrive
```

In [0]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

In [0]:
```
data = pd.read_csv(path +"practice/Chapter5_KNN/nba_2013.csv", sep=",")
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
```

In [0]:
```
data.shape
```

Out[85]:
```
(481, 31)
```

In [0]:
```
# HV tự tìm cách fill dữ liệu thiếu/drop dựa trên các kiến thức đã học
data = data.dropna()
```

In [0]:
```
data.shape
```

Out[87]:
```
(403, 31)
```

In [0]:
```
data.head()
```

Out[88]:

mp	fg	fga	fg.	x3p	x3pa	x3p.	x2p	x2pa	x2p.	efg.	ft	fta	ft.	orb	drb
847	66	141	0.468	4	15	0.266667	62	126	0.492063	0.482	35	53	0.660	72	144
552	464	1011	0.459	128	300	0.426667	336	711	0.472574	0.522	274	336	0.815	32	230
951	136	249	0.546	0	1	0.000000	136	248	0.548387	0.546	56	67	0.836	94	183
498	652	1423	0.458	3	15	0.200000	649	1408	0.460938	0.459	296	360	0.822	166	599
072	134	300	0.447	2	13	0.153846	132	287	0.459930	0.450	33	50	0.660	119	192

In [0]:
```
data.tail()
```

Out[89]:

	player	pos	age	bref_team_id	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	x2p
476	Tony Wroten	SG	20	PHI	72	16	1765	345	808	0.427	40	188	0.212766	305
477	Nick Young	SG	28	LAL	64	9	1810	387	889	0.435	135	350	0.385714	252
478	Thaddeus Young	PF	25	PHI	79	78	2718	582	1283	0.454	90	292	0.308219	492
479	Cody Zeller	C	21	CHA	82	3	1416	172	404	0.426	0	1	0.000000	172
480	Tyler Zeller	C	24	CLE	70	9	1049	156	290	0.538	0	1	0.000000	156

In [0]:
```
# The columns that we will be making predictions with.
inputs = data.drop(["player","bref_team_id","season","season_end","pts"],axis=1)
inputs.shape
```

Out[90]:
```
(403, 26)
```

In [0]:
```
inputs.head()
```

Out[91]:

	pos	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	x2p	x2pa	x2p.	efg.
0	SF	23	63	0	847	66	141	0.468	4	15	0.266667	62	126	0.492063	0.482
3	SG	28	73	73	2552	464	1011	0.459	128	300	0.426667	336	711	0.472574	0.522
4	C	25	56	30	951	136	249	0.546	0	1	0.000000	136	248	0.548387	0.546
6	PF	28	69	69	2498	652	1423	0.458	3	15	0.200000	649	1408	0.460938	0.459
7	PF	24	65	2	1072	134	300	0.447	2	13	0.153846	132	287	0.459930	0.450

In [0]:
```
# import seaborn as sns
# sns.pairplot(inputs)
# plt.show()
```

In [0]:
```
inputs = pd.get_dummies(inputs)
inputs.head()
```

Out[93]:

	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	x2p	x2pa	x2p.	efg.	ft	
0	23	63	0	847	66	141	0.468	4	15	0.266667	62	126	0.492063	0.482	35	
3	28	73	73	2552	464	1011	0.459	128	300	0.426667	336	711	0.472574	0.522	274	3
4	25	56	30	951	136	249	0.546	0	1	0.000000	136	248	0.548387	0.546	56	
6	28	69	69	2498	652	1423	0.458	3	15	0.200000	649	1408	0.460938	0.459	296	3
7	24	65	2	1072	134	300	0.447	2	13	0.153846	132	287	0.459930	0.450	33	

In [0]:
```
#inputs.info()
```

In [0]:
```
# The column that we want to predict.
outputs = data["pts"]
outputs = np.array(outputs)
outputs.shape
```

Out[95]:
```
(403,)
```

In [0]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs, outputs,
                                                    test_size=0.20,
                                                    random_state = 42)
```

In [0]:
```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import accuracy_score
```

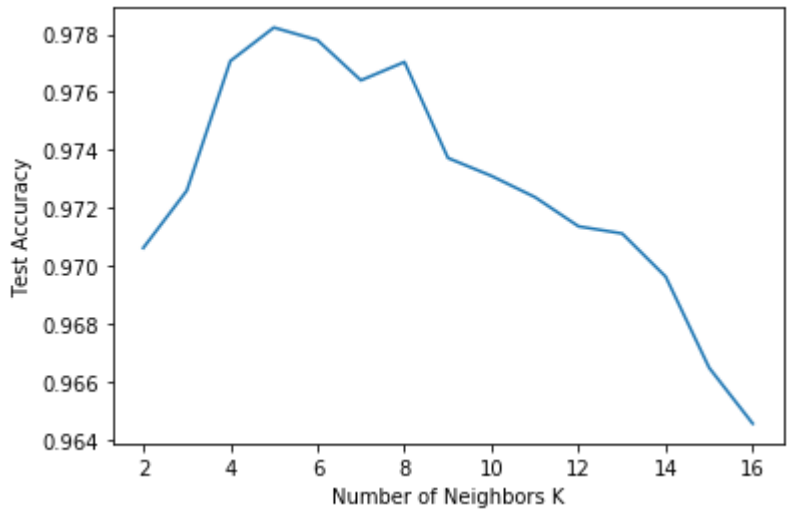
In [0]:
```
list_k = []
list_score = []
for K_value in range(2,int(y_train.shape[0]**0.5)):
    list_k.append(K_value)
    neigh = KNeighborsRegressor(n_neighbors = K_value)
    neigh.fit(X_train, y_train)
    y_pred = neigh.predict(X_test)
    score = neigh.score(X_test, y_test)
    list_score.append(score)
    print("Accuracy is ", score,"% for K-Value:",K_value)
```

```
Accuracy is  0.9706106398874882 % for K-Value: 2
Accuracy is  0.9725933456177593 % for K-Value: 3
Accuracy is  0.9770730768900523 % for K-Value: 4
Accuracy is  0.978230348461507 % for K-Value: 5
Accuracy is  0.977798100546243 % for K-Value: 6
Accuracy is  0.9764061450426895 % for K-Value: 7
Accuracy is  0.9770400934699496 % for K-Value: 8
Accuracy is  0.9737204054920425 % for K-Value: 9
Accuracy is  0.9731007587419361 % for K-Value: 10
Accuracy is  0.9723657389625773 % for K-Value: 11
Accuracy is  0.9713562563279552 % for K-Value: 12
Accuracy is  0.9711109581151784 % for K-Value: 13
Accuracy is  0.9696185629376203 % for K-Value: 14
Accuracy is  0.9664662550549205 % for K-Value: 15
Accuracy is  0.9645375047786899 % for K-Value: 16
```

In [0]:
```
vi_tri = list_score.index(max(list_score))
k = list_k[vi_tri]
print("\nThe optimal number of neighbors is", k, "with", list_score[vi_tri])
```

```
The optimal number of neighbors is 5 with 0.978230348461507
```

In [0]:
```
plt.plot(list_k, list_score)
plt.xlabel('Number of Neighbors K')
plt.ylabel('Test Accuracy')
plt.show()
```



In [0]:
```
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train)
```

Out[101]:
```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                    weights='uniform')
```

In [0]:
```
# Kiểm tra độ chính xác
print("The Train/ Score is: ", knn.score(X_train,y_train)*100,"%")
print("The Test/ Score accuracy is: ", knn.score(X_test,y_test)*100,"%")
```

```
The Train/ Score is:  98.01127110964177 %
The Test/ Score accuracy is:  97.8230348461507 %
```

In [0]:
```
# Tính MSE
from sklearn import metrics
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
```

```
Mean Squared Error: 7451.879581404321
```

Nhận xét:

- Training và Testing cùng có R^2 cao và gần bằng nhau
- Mô hình trên cho R^2 cao ~ 0.98, cho thấy nó fit 98% dữ liệu => mô hình phù hợp

In [0]:
```
y_pred = knn.predict(X_test)
#y_pred
```

In [0]:
```
df = pd.DataFrame({'Actual': pd.DataFrame(y_test)[0].values,
                   'Prediction': pd.DataFrame(y_pred)[0].values})
df.head()
```

Out[105]:

	Actual	Prediction
0	490	450.8
1	548	467.6
2	820	796.4
3	217	198.6
4	491	516.4

In [0]:
```
# Xuất model
import pickle
# Save to file in the current working directory
pkl_filename = "NBA_model.pkl"
with open(pkl_filename, 'wb') as file:
    pickle.dump(knn, file)
```

In [0]:
```
with open(pkl_filename, 'rb') as file:
    nba_model = pickle.load(file)
```

In [0]:
```
nba_model
```

Out[108]:
```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                    weights='uniform')
```