

Chapter 6 - Exercise 3: NBA Players

Cho dữ liệu nba_2013.csv

Sử dụng thuật toán Decision Tree 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 Decision Tree 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 [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
In [2]: # import some data to play with
data = pd.read_csv("nba_2013.csv", sep=",")
#data.info()
```

```
In [3]: data.shape
```

Out[3]: (481, 31)

```
In [4]: # HV tự tìm cách fill dữ liệu thiếu/drop dựa trên các kiến thức đã học
data = data.dropna()
```

```
In [5]: data.shape
```

Out[5]: (403, 31)

```
In [6]: data.head()
```

Out[6]:

	player	pos	age	bref_team_id	g	gs	mp	fg	fga	fg.	...	drb	trb	ast	stl	blk	tov	pf	pts	season	season_end
0	Quincy Acy	SF	23	TOT	63	0	847	66	141	0.468	...	144	216	28	23	26	30	122	171	2013-2014	2013
3	Arron Afflalo	SG	28	ORL	73	73	2552	464	1011	0.459	...	230	262	248	35	3	146	136	1330	2013-2014	2013
4	Alexis Ajinca	C	25	NOP	56	30	951	136	249	0.546	...	183	277	40	23	46	63	187	328	2013-2014	2013
6	LaMarcus Aldridge	PF	28	POR	69	69	2498	652	1423	0.458	...	599	765	178	63	68	123	147	1603	2013-2014	2013
7	Lavoy Allen	PF	24	TOT	65	2	1072	134	300	0.447	...	192	311	71	24	33	44	126	303	2013-2014	2013

5 rows × 31 columns

```
In [7]: data.tail()
```


Out[7]:

	player	pos	age	bref_team_id	g	gs	mp	fg	fga	fg.	...	drb	trb	ast	stl	blk	tov	pf	pts	season	season_end
476	Tony Wroten	SG	20	PHI	72	16	1765	345	808	0.427	...	159	228	217	78	16	204	151	939	2013-2014	2013
477	Nick Young	SG	28	LAL	64	9	1810	387	889	0.435	...	137	166	95	46	12	95	156	1144	2013-2014	2013
478	Thaddeus Young	PF	25	PHI	79	78	2718	582	1283	0.454	...	310	476	182	167	36	165	213	1417	2013-2014	2013
479	Cody Zeller	C	21	CHA	82	3	1416	172	404	0.426	...	235	353	92	40	41	87	170	490	2013-2014	2013
480	Tyler Zeller	C	24	CLE	70	9	1049	156	290	0.538	...	179	282	36	18	38	60	137	399	2013-2014	2013

5 rows × 31 columns

```
In [8]: # The columns that we will be making predictions with.
inputs = data.drop(["player", "bref_team_id", "season", "season_end"], axis=1)
inputs.shape
```

Out[8]: (403, 27)

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

Out[9]:

	pos	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	...	ft.	orb	drb	trb	ast	stl	blk	tov	pf	pts
0	SF	23	63	0	847	66	141	0.468	4	15	...	0.660	72	144	216	28	23	26	30	122	171
3	SG	28	73	73	2552	464	1011	0.459	128	300	...	0.815	32	230	262	248	35	3	146	136	1330
4	C	25	56	30	951	136	249	0.546	0	1	...	0.836	94	183	277	40	23	46	63	187	328
6	PF	28	69	69	2498	652	1423	0.458	3	15	...	0.822	166	599	765	178	63	68	123	147	1603
7	PF	24	65	2	1072	134	300	0.447	2	13	...	0.660	119	192	311	71	24	33	44	126	303

5 rows × 27 columns

```
In [10]: inputs = pd.get_dummies(inputs)
inputs.head()
```

Out[10]:

	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	...	blk	tov	pf	pts	pos_C	pos_G	pos_PF	pos_PG	pos_SF	pos_SG
0	23	63	0	847	66	141	0.468	4	15	0.266667	...	26	30	122	171	0	0	0	0	1	0
3	28	73	73	2552	464	1011	0.459	128	300	0.426667	...	3	146	136	1330	0	0	0	0	0	1
4	25	56	30	951	136	249	0.546	0	1	0.000000	...	46	63	187	328	1	0	0	0	0	0
6	28	69	69	2498	652	1423	0.458	3	15	0.200000	...	68	123	147	1603	0	0	1	0	0	0
7	24	65	2	1072	134	300	0.447	2	13	0.153846	...	33	44	126	303	0	0	1	0	0	0

5 rows × 32 columns

```
In [11]: #inputs.info()
```

```
In [12]: # The column that we want to predict.
outputs = data["pts"]
outputs = np.array(outputs)
outputs.shape
```

Out[12]: (403,)

```
In [13]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs,
                                                    outputs,
                                                    test_size=0.30,
                                                    random_state=42)
```

```
In [14]: from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score
```

```
In [15]: # Create decision tree regressor object
model = DecisionTreeRegressor()
# Train model
model.fit(X_train, y_train)
```

Out[15]: DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=None, splitter='best')


```
In [16]: # Kiểm tra độ chính xác
print("The Train/ Score is: ", model.score(X_train,y_train)*100,"%")
print("The Test/ Score accuracy is: ", model.score(X_test,y_test)*100,"%")
```

The Train/ Score is: 100.0 %
The Test/ Score accuracy is: 99.74514106739079 %

```
In [17]: # Tính MSE
from sklearn import metrics
y_pred = model.predict(X_test)
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
```

Mean Squared Error: 570.0

Nhận xét:

- Training và Testing cùng có R^2 cao và gần bằng nhau
- Mô hình trên cho R^2 cao ~ 0.99 , cho thấy nó fit 99% dữ liệu
- MSE vừa phải

=> mô hình phù hợp

```
In [18]: df = pd.DataFrame({'Actual': pd.DataFrame(y_test)[0].values,
                           'Prediction': pd.DataFrame(y_pred)[0].values})
df.head(10)
```

Out[18]:

	Actual	Prediction
0	490	490.0
1	548	544.0
2	820	821.0
3	217	224.0
4	491	490.0
5	47	54.0
6	1737	1851.0
7	202	201.0
8	520	530.0
9	18	22.0

```
In [19]: # 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(model, file)
```

```
In [20]: with open(pkl_filename, 'rb') as file:
nba_model = pickle.load(file)
```

```
In [21]: nba_model
```

Out[21]: DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=None, splitter='best')