Chapter 8 - Exercise 2: NBA Players

Cho dữ liệu nba_2013.csv

Sử dụng thuật toán Linear 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'.
3. Từ inputs data và outputs data => Tạo X_train, X_test, y_train, y_test với tỷ lệ 80:20
4. Thực hiện Linear 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. Lưu model nếu model có kết quả tốt.
8. Áp dụng Pipeline cho bài toán trên.

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

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

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	p
476	Tony Wroten	SG	20	PHI	72	16	1765	345	808	0.427	...	159	228	217	78	16	204	151	9
477	Nick Young	SG	28	LAL	64	9	1810	387	889	0.435	...	137	166	95	46	12	95	156	11
478	Thaddeus Young	PF	25	PHI	79	78	2718	582	1283	0.454	...	310	476	182	167	36	165	213	14
479	Cody Zeller	C	21	CHA	82	3	1416	172	404	0.426	...	235	353	92	40	41	87	170	4
480	Tyler Zeller	C	24	CLE	70	9	1049	156	290	0.538	...	179	282	36	18	38	60	137	3

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

	pos	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	...	ft.	orb	drb	trb	ast	stl	blk	tov	pf	pts
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]:
```
# Xem xét mối tương quan giữa pts với các features khác
# Lọc ra các features có corr >=0.6
corr_pts = inputs.corr().tail(1)
```

In [11]:
```
corr_pts[corr_pts >=0.6].T.dropna()
```

Out[11]:

	pts
g	0.708630
gs	0.797006
mp	0.920194
fg	0.991289
fga	0.988128
x3p	0.624143
x3pa	0.640738
x2p	0.925905
x2pa	0.929844
ft	0.923201
fta	0.915259
drb	0.783448
trb	0.722322
ast	0.710765
stl	0.767984
tov	0.900949
pf	0.761402
pts	1.000000

In [12]:
```
inputs = pd.get_dummies(inputs)
inputs.head()
```

Out[12]:

	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	...	blk	tov	pf	pts	pos_C	pos_G	pos_PF	pos_
0	23	63	0	847	66	141	0.468	4	15	0.266667	...	26	30	122	171	0	0	0	
3	28	73	73	2552	464	1011	0.459	128	300	0.426667	...	3	146	136	1330	0	0	0	
4	25	56	30	951	136	249	0.546	0	1	0.000000	...	46	63	187	328	1	0	0	
6	28	69	69	2498	652	1423	0.458	3	15	0.200000	...	68	123	147	1603	0	0	1	

	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	...	blk	tov	pf	pts	pos_C	pos_G	pos_PF	pos_
7	24	65	2	1072	134	300	0.447	2	13	0.153846	...	33	44	126	303	0	0	1	

5 rows × 32 columns

In [13]:
```
#inputs.info()
```

In [14]:
```
# The column that we want to predict.
outputs = data["pts"]
outputs = np.array(outputs)
outputs.shape
```

Out[14]: (403,)

In [15]:
```
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 [16]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
```

In [17]:
```
# Train model
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[17]: LinearRegression()

In [18]:
```
# 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:  100.0 %
```

In [19]:
```
# 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: 1.1580296316038713e-25
```

Nhận xét:

- Training và Testing cùng có R^2 cao và gần bằng nhau
- Mô hình trên cho R^2 cao
- MSE vừa phải => mô hình phù hợp

In [20]:
```
df = pd.DataFrame({'Actual': pd.DataFrame(y_test)[0].values,
```

```
                    'Prediction': pd.DataFrame(y_pred)[0].values})
df.head(10)
```

Out[20]:

	Actual	Prediction
0	490	490.0
1	548	548.0
2	820	820.0
3	217	217.0
4	491	491.0
5	47	47.0
6	1737	1737.0
7	202	202.0
8	520	520.0
9	18	18.0

In [21]:
```
# 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 [22]:
```
# with open(pkl_filename, 'rb') as file:
#     nba_model = pickle.load(file)
```

PipeLine

In [23]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
```

In [24]:
```
# The columns that we will be making predictions with.
inputs_now = data.drop(["player", "bref_team_id", "season", "season_end"], axis=1)
output_now = data["pts"]
```

In [25]:
```
inputs_now.shape
```

Out[25]: (403, 27)

In [26]:
```
Xp_train, Xp_test, yp_train, yp_test = train_test_split(inputs_now,
                                                        output_now,
                                                        test_size=0.30,
                                                        random_state = 42)
```

In [27]:
```
Input=[('column_tr', make_column_transformer((OneHotEncoder(), ['pos']),
                                             remainder='passthrough')),
       ('model', LinearRegression())]
```

In [28]:
```
pipe = Pipeline(Input)
pipe
```

Out[28]:
```
Pipeline(steps=[('column_tr',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',
                                                  OneHotEncoder(), ['pos'])])),
                ('model', LinearRegression())])
```

In [29]:
```
pipe.fit(Xp_train, yp_train)
```

Out[29]:
```
Pipeline(steps=[('column_tr',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',
                                                  OneHotEncoder(), ['pos'])])),
                ('model', LinearRegression())])
```

In [30]:
```
ypipe= pipe.predict(Xp_test)
```

In [31]:
```
# Kiểm tra độ chính xác
print("The Train/ Score is: ", pipe.score(Xp_train,yp_train)*100,"%")
print("The Test/ Score accuracy is: ", pipe.score(Xp_test,yp_test)*100,"%")
```

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

In [32]:
```
# Tính MSE
print('Mean Squared Error:', metrics.mean_squared_error(yp_test, ypipe))
```

```
Mean Squared Error: 1.070187065899796e-24
```