```
%pip install gurobipy
import pandas as pd                                                    #importing pandas
import numpy as np                                                     #importing numpy
import matplotlib.pyplot as plt                                        #importing matplotlib
import seaborn as sns                                                  #importing seaborn
from sklearn.model_selection import train_test_split                   #importing scikit-learn's function for data splitting
from sklearn.linear_model import LinearRegression                      #importing scikit-learn's linear regressor function
from sklearn.metrics import mean_squared_error, r2_score               #importing scikit-learn's root mean squared error functio
from sklearn.model_selection import cross_validate, train_test_split   #improting scikit-learn's cross validation function
from sklearn.preprocessing import StandardScaler
from scipy.optimize import linprog
import statsmodels.api as sm
```

```
Requirement already satisfied: gurobipy in /usr/local/lib/python3.10/dist-packages (11.0.2)
```

```
df_main = pd.read_csv('/content/player_data_modify_240613.csv')
df_salary = pd.read_csv('/content/target_games.csv')
df_main['playerName'] = df_main['playerName'].str.strip().str.lower()
df_salary['playerName'] = df_salary['Player'].str.strip().str.lower()
df = pd.merge(df_main, df_salary, on='playerName', how='inner')
print(len(df))
# 檢查數據
print(df.head())
```

```
149
      playerName  teamRslt teamRslt1  playStat playStat1  playMin playPos  \
0  kyrie irving         0      Loss         1     Start     36.0      PG
1  kyrie irving         1       Win         1     Start     40.0      PG
2  kyrie irving         0      Loss         1     Start     42.0      PG
3  kyrie irving         1       Win         1     Start     42.0      PG
4  kyrie irving         1       Win         1     Start     39.0      PG

   playHeight  playWeight  playPTS  ...  playORB  playDRB  playTRB  opptRslt  \
0        1.88         195       12  ...        0        3        3       Win
1        1.88         195       36  ...        1        3        4      Loss
2        1.88         195       16  ...        2        0        2       Win
3        1.88         195       33  ...        0        3        3      Loss
4        1.88         195       20  ...        2        2        4      Loss

   FantasyPoints        Player  Pos  Team     Opp  Salary
0          19.25  Kyrie Irving   PG   BOS  vs WAS    8300
1          50.50  Kyrie Irving   PG   BOS  vs WAS    8300
2          24.00  Kyrie Irving   PG   BOS  vs WAS    8300
3          43.25  Kyrie Irving   PG   BOS  vs WAS    8300
4          36.00  Kyrie Irving   PG   BOS  vs WAS    8300

[5 rows x 37 columns]
```

Q2

```
# 創建新特徵
df['points_per_minute'] = df['playPTS'] / df['playMin']
df['assists_per_minute'] = df['playAST'] / df['playMin']
df['rebounds_per_minute'] = df['playTRB'] / df['playMin']

# 定義特徵和目標變量
features = ['playPTS', 'playAST', 'playTRB', 'playSTL', 'playBLK', 'playTO', 'playMin',
            'points_per_minute', 'assists_per_minute', 'rebounds_per_minute']
X = df[features]
y = df['FantasyPoints']

# 添加常數項
X = sm.add_constant(X)

# 分割數據集為訓練集和測試集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 構建線性回歸模型
model = sm.OLS(y_train, X_train).fit()

# 預測
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

# 評估模型性能
mse_train = mean_squared_error(y_train, y_pred_train)
mse_test = mean_squared_error(y_test, y_pred_test)
r2_train = r2_score(y_train, y_pred_train)
r2_test = r2_score(y_test, y_pred_test)

print('訓練集MSE:', mse_train)
print('測試集MSE:', mse_test)
print('訓練集R2:', r2_train)
print('測試集R2:', r2_test)

# 顯示模型摘要
print(model.summary())

# 討論特徵的重要性
importance = model.params
print('特徵重要性:')
print(importance)
# 可視化模型的預測結果
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_train, y_pred_train, alpha=0.5)
plt.plot([y_train.min(), y_train.max()], [y_train.min(), y_train.max()], 'r--', lw=2)
plt.xlabel('Real_Value')
plt.ylabel('Predict_Value')
plt.title('train: Real_Value vs Predict_Value')

plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_test, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Real_Value')
plt.ylabel('Predict_Value')
plt.title('test: Real_Value vs Predict_Value')

plt.tight_layout()
plt.show()

# 可視化特徵重要性
importance = importance.drop('const')  # 去掉常數項
plt.figure(figsize=(10, 5))
importance.sort_values().plot(kind='barh')
plt.title('importance')
plt.xlabel('coefficient')
plt.show()
```

訓練集MSE: 0.43937041060788123
測試集MSE: 0.305809363777326
訓練集R2: 0.9978603727028389
測試集R2: 0.9980615574879119

OLS Regression Results

Dep. Variable:	FantasyPoints	R-squared:	0.998
Model:	OLS	Adj. R-squared:	0.998
Method:	Least Squares	F-statistic:	5037.
Date:	Thu, 20 Jun 2024	Prob (F-statistic):	2.90e-139
Time:	12:29:04	Log-Likelihood:	-119.92
No. Observations:	119	AIC:	261.8
Df Residuals:	108	BIC:	292.4
Df Model:	10		
Covariance Type:	nonrobust		

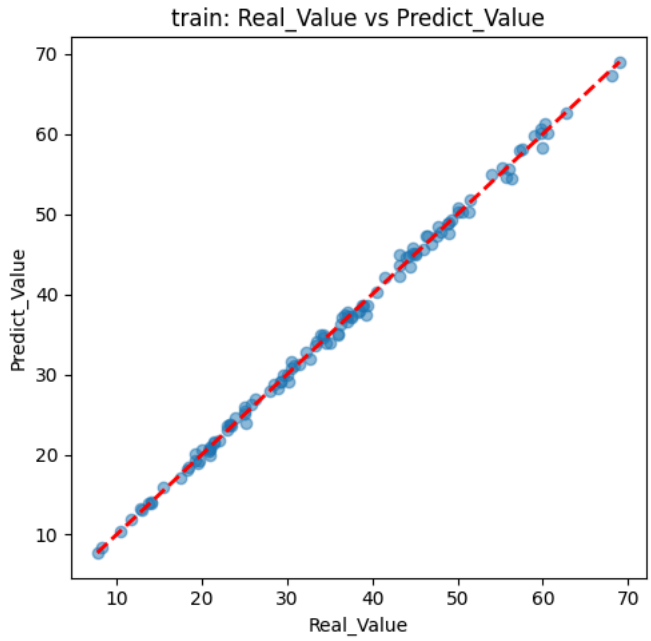
	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.1819	1.033	-1.145	0.255	-3.229	0.865
playPTS	1.0556	0.040	26.468	0.000	0.977	1.135
playAST	1.4104	0.086	16.443	0.000	1.240	1.580
playTRB	1.0929	0.079	13.885	0.000	0.937	1.249
playSTL	1.0431	0.061	17.066	0.000	0.922	1.164
playBLK	2.1154	0.080	26.563	0.000	1.958	2.273
playTO	-0.5140	0.043	-11.823	0.000	-0.600	-0.428
playMin	0.0413	0.030	1.397	0.165	-0.017	0.100
points_per_minute	-0.2985	1.472	-0.203	0.840	-3.217	2.620
assists_per_minute	1.8351	2.725	0.673	0.502	-3.566	7.236
rebounds_per_minute	4.7532	2.773	1.714	0.089	-0.743	10.249

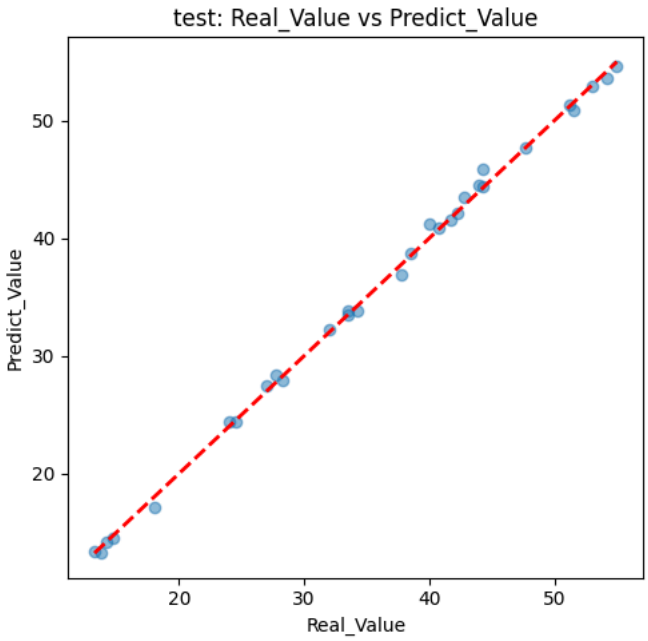
Omnibus:	6.359	Durbin-Watson:	1.948
Prob(Omnibus):	0.042	Jarque-Bera (JB):	5.956
Skew:	0.534	Prob(JB):	0.0509
Kurtosis:	3.250	Cond. No.	2.07e+03

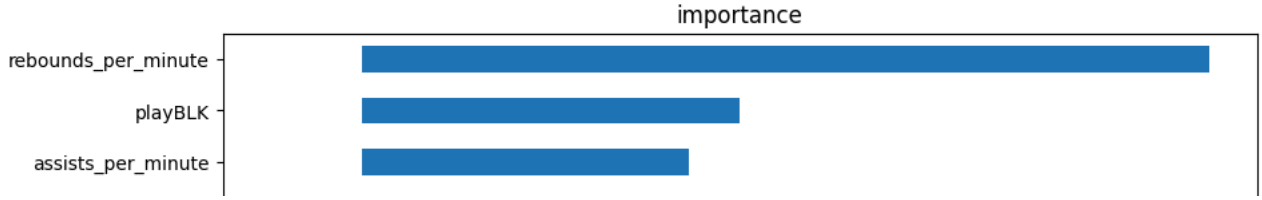
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.07e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

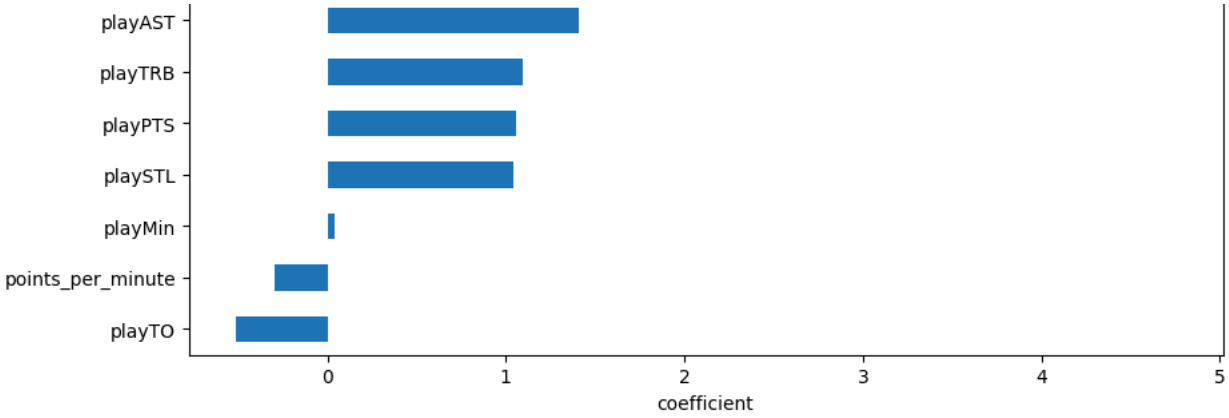
特徵重要性:

```
const                -1.181943
playPTS               1.055634
playAST               1.410390
playTRB               1.092898
playSTL               1.043051
playBLK               2.115359
playTO               -0.514031
playMin               0.041292
points_per_minute    -0.298538
assists_per_minute    1.835087
rebounds_per_minute   4.753220
dtype: float64
```









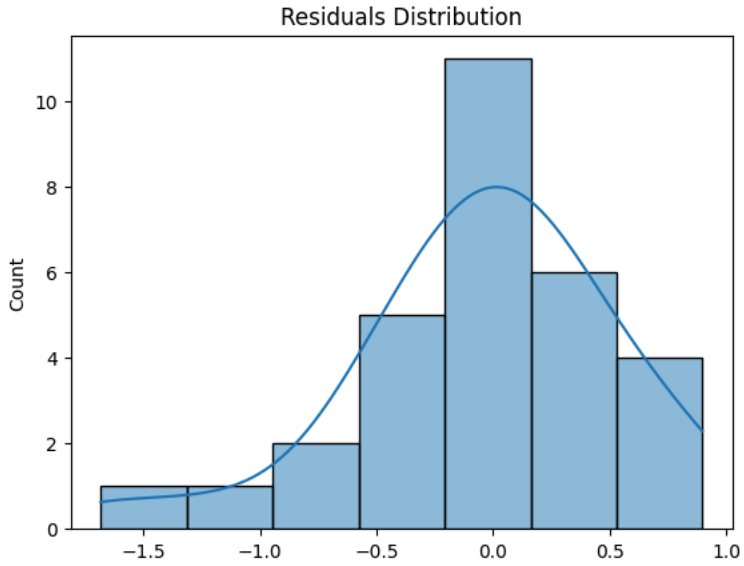
Q4

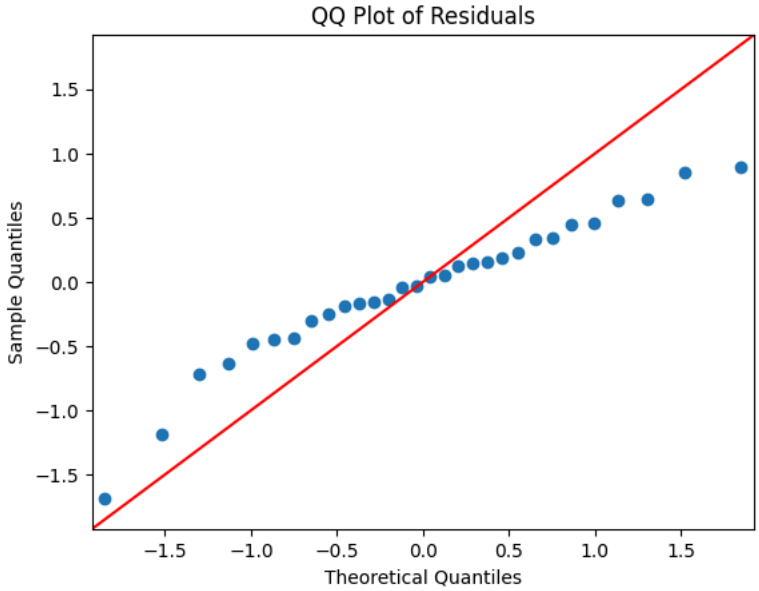
+ 程式碼 + 文字

```
#  殘差
residuals  =  y_test  -  y_pred_test

#  殘差常態分佈檢查
sns.histplot(residuals,  kde=True)
plt.title('Residuals  Distribution')
plt.show()

#  QQ圖檢查殘差
sm.qqplot(residuals,  line  ='45')
plt.title('QQ  Plot  of  Residuals')
plt.show()
```





按兩下 (或按 Enter 鍵) 即可編輯

```
#  收集殘差的平方
residuals_squared  =  residuals  **  2
LSig  =  np.log(residuals_squared)

#  擬合殘差模型
residual_model  =  sm.OLS(LSig,  X_test).fit()
LSig_pred  =  residual_model.predict(X_test)

print(residual_model.summary())
#  模擬誤差
S  =  1000    #  模擬次數
LSig_pred_np  =  np.array(LSig_pred)
simulated_errors  =  np.random.normal(0,  np.exp(LSig_pred_np)[:,  np.newaxis]**0.5,  (len(X_test),  S))
#  Convert  y_pred_test  to  a  NumPy  array  before  adding  simulated  errors
y_pred_test_np  =  np.array(y_pred_test)
simulated_scores  =  y_pred_test_np[:,  np.newaxis]  +  simulated_errors

#  構建分佈
simulated_scores_df  =  pd.DataFrame(simulated_scores)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.279
Model:                            OLS   Adj. R-squared:                 -0.101
Method:                 Least Squares   F-statistic:                    0.7338
Date:                Thu, 20 Jun 2024   Prob (F-statistic):              0.685
Time:                        13:15:51   Log-Likelihood:                -58.488
No. Observations:                  30   AIC:                             139.0
Df Residuals:                      19   BIC:                             154.4
Df Model:                          10
Covariance Type:            nonrobust
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          -0.6840      7.781     -0.088      0.931     -16.969      15.601
playPTS         0.0727      0.261      0.279      0.783      -0.473       0.618
playAST         0.0015      0.830      0.002      0.999      -1.736       1.739
playTRB        -0.3162      0.803     -0.394      0.698      -1.997       1.364
playSTL         0.2904      0.647      0.449      0.659      -1.065       1.645
playBLK         0.1997      0.587      0.340      0.737      -1.029       1.428
playTO          0.5005      0.253      1.978      0.063      -0.029       1.030
playMin        -0.1058      0.250     -0.423      0.677      -0.629       0.417
points_per_minute  -0.7073      8.803     -0.080      0.937     -19.131      17.717
assists_per_minute  -3.8447     22.930     -0.168      0.869     -51.838      44.148
rebounds_per_minute  9.9931     27.265      0.367      0.718     -47.073      67.060
==============================================================================
Omnibus:                        5.198   Durbin-Watson:                   2.402
Prob(Omnibus):                  0.074   Jarque-Bera (JB):                3.788
Skew:                          -0.846   Prob(JB):                        0.150
Kurtosis:                       3.408   Cond. No.                     3.08e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.08e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
#限制
budget = 50000    # 總預算
positions = {'PG': 1, 'SG': 1, 'SF': 1, 'PF': 1, 'C': 1}    # 每個位置至少一個球員

# 計算每個球員的預測得分
df['predicted_score'] = model.predict(sm.add_constant(df[features]))

# 定義優化問題
c = -df['predicted_score'].values    # 最大化總得分，因此使用負值來最小化
A = []
b = []
```

Q3

Formulate a linear integer programming model to optimize your lineup for fantasy games. The objective is to form a team with the highest fantasy scores. Feel free to add or change constraints (total salary limit, position requirement such as at least one PG in the team, etc.). Show and explain your lineup of fantasy NBA players.

```
import pandas as pd

file_path = '/content/player_data_modify_240613.xlsx'
df = pd.read_excel(file_path)

print(df.head())

print(df.columns)
```

```
        playerName  teamRslt  playStat    playMin  playHeight  playWeight  \
0      Luka Doncic  0.666667       1.0  41.333333        2.01         230
1  Anthony Edwards  0.562500       1.0  40.500000        1.93         225
2    Jalen Brunson  0.538462       1.0  39.846154        1.88         190
3  Anthony Edwards  0.562500       1.0  40.500000        1.93         225
4    Jalen Brunson  0.538462       1.0  39.846154        1.88         190

     playPTS   playAST    playTO   playSTL  ...   play3PM   play3P%   playFTA  \
0  28.833333  8.388889  3.944444  1.666667  ...  3.388889  0.341330  7.166667
1  27.562500  6.500000  3.250000  1.500000  ...  2.875000  0.366491  6.375000
2  37.000000  7.384615  2.692308  0.769231  ...  2.000000  0.298119  9.230769
3  27.562500  6.500000  3.250000  1.500000  ...  2.875000  0.366491  6.375000
4  37.000000  7.384615  2.692308  0.769231  ...  2.000000  0.298119  9.230769

    playFTM   playFT%   playORB   playDRB   playTRB  FantasyPoints  playPos
0  5.666667  0.790627  0.888889  8.777778  9.666667      55.777778       PG
1  5.187500  0.758110  1.125000  5.875000  7.000000      48.500000       SG
2  7.153846  0.762143  0.615385  2.692308  3.307692      52.942308       PG
3  5.187500  0.758110  1.125000  5.875000  7.000000      48.500000       SG
4  7.153846  0.762143  0.615385  2.692308  3.307692      52.942308       PG

[5 rows x 29 columns]
Index(['playerName', 'teamRslt', 'playStat', 'playMin', 'playHeight',
       'playWeight', 'playPTS', 'playAST', 'playTO', 'playSTL', 'playBLK',
       'playPF', 'playFGA', 'playFGM', 'playFG%', 'play2PA', 'play2PM',
       'play2P%', 'play3PA', 'play3PM', 'play3P%', 'playFTA', 'playFTM',
       'playFT%', 'playORB', 'playDRB', 'playTRB', 'FantasyPoints', 'playPos'],
      dtype='object')
```

```
print(df['playerName'])
```

```
0        Luka Doncic
1    Anthony Edwards
2      Jalen Brunson
3    Anthony Edwards
4      Jalen Brunson
          ...
65       Alec Burks
66       Josh Green
67     Ben Sheppard
68      Josh Giddey
69    Austin Reaves
Name: playerName, Length: 70, dtype: object
```

決策變數

xi： 二進制變數，表示是否選擇第i個球員。

xi = 1 表示選擇該球員，xi = 0 表示不選擇。

目標是最大化選定球員的總幻想積分。具體的目標函數如下:

Maximize∑(FantasyPoints $i \cdot xi$) 其中 xi是二進制變數，當球員 i 被選中時 xi=1，否則 xi=0。

```
!pip install gurobipy
```

```
Collecting gurobipy
  Downloading gurobipy-11.0.2-cp310-cp310-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (13.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.4/13.4 MB 41.5 MB/s eta 0:00:00
Installing collected packages: gurobipy
Successfully installed gurobipy-11.0.2
```

限制條件

1) 位置約束：每個位置（PG, SG, SF, PF, C）必須正好選擇一名球員。

$\sum xi$(for players in position PG)=1

$\sum xi$(for players in position SG)=1

$\sum xi$(for players in position SF)=1

$\sum xi$(for players in position PF)=1

$\sum xi$(for players in position C)=1

2) 總球員數約束：選擇5名球員。

$\sum xi$=5

3) 助攻失誤比率約束：確保選中球員的助攻失誤比率不低於1.3。

playAST i * xi /(playTO i + 0.000000001)≥ 1.3 * xi − M *(1 − xi)

4) 平均犯規次數約束：確保選中球員的平均犯規次數不超過3.3。

playPF i * xi ≤ 3.3 + M * (1 − xi)

5) 命中率約束：確保選中球員的命中率總和（包括2P%, 3P%, FT%）不低於1.5。

play2P% i * xi + play3P% i * xi + playFT% i * xi ≥ 1.5 * xi − M * (1 − xi)

```
from gurobipy import Model, GRB
import pandas as pd

players_data = df.to_dict('index')

m = Model("Fantasy_Basketball_Model")

# 定義決策變數
x = m.addVars(len(players_data), vtype=GRB.BINARY, name="x")

# 定義目標函數：Max. FantasyPoints
m.setObjective(sum(players_data[i]["FantasyPoints"] * x[i] for i in range(len(players_data))), GRB.MAXIM

# 添加位置constraint：確保每個位置正好有一名球員
positions = ['PG', 'SG', 'SF', 'PF', 'C']
for pos in positions:
        m.addConstr(sum(x[i] for i in range(len(players_data)) if players_data[i]["playPos"] == pos) =

# 添加總球員數constraint：選擇5名球員
m.addConstr(sum(x[i] for i in range(len(players_data))) == 5, name="Total_Players")

M = 1e6
# 助攻失誤比率：確保助攻失誤比率 >= 1.3
for i in range(len(players_data)):
        m.addConstr((players_data[i]["playAST"] * x[i]) / (players_data[i]["playTO"] + 0.000000001) >=

# 平均犯規次數：確保犯規次數 <= 3.3
for i in range(len(players_data)):
        m.addConstr(players_data[i]["playPF"] * x[i] <= 3.3 + M * (1 - x[i]), name=f"PF_Limit_{i}")

# 命中率：確保命中率總和 >= 1.5
```

```
# 命中率：確保命中率總和 >= 1.5
for i in range(len(players_data)):
    m.addConstr(players_data[i]["play2P%"] * x[i] + players_data[i]["play3P%"] * x[i] + players_dat

m.optimize()

# 檢查是否找到解
if m.status == GRB.OPTIMAL:
    selected_players = [players_data[i] for i in range(len(players_data)) if x[i].X == 1]
    print("selected_players info: ", selected_players)
    total_fantasy_point = sum(player["FantasyPoints"] for player in selected_players)
    print("total_point: ", total_fantasy_point)
    for player in selected_players:
        print(f"Selected Player: {player['playerName']}, Position: {player['playPos']}, Fantasy P
else:
    # 這裏可以幫助我們找到到底是哪個限制太嚴苛導致找不到最佳組合
    print("No feasible solution.")
    if m.status == GRB.INFEASIBLE:
        m.computeIIS()
        print("\nThose constraints cannot be satisfied:")
        for c in m.getConstrs():
            if c.IISConstr:
                print(f"{c.ConstrName}")
```

```
Restricted license - for non-production use only - expires 2025-11-24
Gurobi Optimizer version 11.0.2 build v11.0.2rc0 (linux64 - "Ubuntu 22.04.3 LTS")

CPU model: AMD EPYC 7B12, instruction set [SSE2|AVX|AVX2]
Thread count: 1 physical cores, 2 logical processors, using up to 2 threads

Optimize a model with 216 rows, 70 columns and 350 nonzeros
Model fingerprint: 0x1b805b75
Variable types: 0 continuous, 70 integer (70 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+09]
  Objective range  [1e+01, 6e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+06]
Warning: Model contains large matrix coefficients
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Presolve removed 216 rows and 70 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.02 seconds (0.00 work units)
Thread count was 1 (of 2 available processors)

Solution count 1: 260.007

Optimal solution found (tolerance 1.00e-04)
Best objective 2.600069444444e+02, best bound 2.600069444444e+02, gap 0.0000%
selected_players info:  [{'playerName': 'Luka Doncic', 'teamRslt': 0.6666666666666666, 'playStat': 1.0, 'playMin': 41.333333333333336, 'playHeigh
total_point:  260.00694444444446
Selected Player: Luka Doncic, Position: PG, Fantasy Points: 55.77777777777778
Selected Player: Anthony Edwards, Position: SG, Fantasy Points: 48.5
Selected Player: Nikola Jokic, Position: C, Fantasy Points: 60.104166666666664
Selected Player: LeBron James, Position: SF, Fantasy Points: 53.0
Selected Player: Kevin Durant, Position: PF, Fantasy Points: 42.625
```

```
print("finish")
```

```
finish
```

Q6

```
!pip install gurobipy
import pandas as pd
from gurobipy import Model, GRB
import io
import pandas as pd
from google.colab import files
# 上傳csv檔案
uploaded = files.upload()
df = pd.read_excel(io.BytesIO(uploaded['player_data_240619.xlsx']))
```

```
Collecting gurobipy
  Downloading gurobipy-11.0.2-cp310-cp310-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (13.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.4/13.4 MB 15.4 MB/s eta 0:00:00
Installing collected packages: gurobipy
Successfully installed gurobipy-11.0.2
選擇檔案 player_data_240619.xlsx
• player_data_240619.xlsx(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 141101 bytes, last modified: 2024/6/20 - 100% done
Saving player_data_240619.xlsx to player_data_240619.xlsx
```

```
players_data = df.to_dict('index')

# 創建模型
m = Model("Fantasy_Basketball_Model")

# 定義決策變數
x = m.addVars(len(players_data), vtype=GRB.BINARY, name="x")


# 添加位置constraint：確保每個位置正好有一名球員
positions = ['PG', 'SG', 'SF', 'PF', 'C']
for pos in positions:
        m.addConstr(sum(x[i] for i in range(len(players_data)) if players_data[i]["playPos"] == pos) =

# 添加總球員數constraint：選擇5名球員
m.addConstr(sum(x[i] for i in range(len(players_data))) == 5, name="Total_Players")

# 助攻失誤比率：確保助攻失誤比率 >= 1.3
M = 1e6
for i in range(len(players_data)):
        m.addConstr((players_data[i]["playAST"] * x[i]) / (players_data[i]["playTO"] + 1e-9) >= 1.3 *

# 平均犯規次數：確保犯規次數 <= 3.3
for i in range(len(players_data)):
        m.addConstr(players_data[i]["playPF"] * x[i] <= 3.3 + M * (1 - x[i]), name=f"PF_Limit_{i}")

# 命中率：確保命中率總和 >= 1.5
for i in range(len(players_data)):
        m.addConstr(players_data[i]["play2P%"] * x[i] + players_data[i]["play3P%"] * x[i] + players_dat

# 添加L0正則化約束：限制選擇的非零變量數量
max_selected_players = 5
m.addConstr(sum(x[i] for i in range(len(players_data))) <= max_selected_players, name="L0_Constraint")

# 定義目標函數：Max. FantasyPoints + L1正則化
lambda1 =1000 # L1正則化參數
m.setObjective(sum(players_data[i]["FantasyPoints"] * x[i] for i in range(len(players_data))) - lambda1


# 優化模型
m.optimize()
```

```
# 檢查是否找到解
if m.status == GRB.OPTIMAL:
        selected_players = [players_data[i] for i in range(len(players_data)) if x[i].X == 1]
        print("selected_players info: ", selected_players)
        total_fantasy_point = sum(player["FantasyPoints"] for player in selected_players)
        print("total_point: ", total_fantasy_point)
        for player in selected_players:
                print(f"Selected Player: {player['playerName']}, Position: {player['playPos']}, Fantasy P

else:
        print("No feasible solution.")
        if m.status == GRB.INFEASIBLE:
                m.computeIIS()
                print("\nThose constraints cannot be satisfied:")
                for c in m.getConstrs():
                        if c.IISConstr:
                                print(f"{c.ConstrName}")
```

```
Restricted license - for non-production use only - expires 2025-11-24
Gurobi Optimizer version 11.0.2 build v11.0.2rc0 (linux64 - "Ubuntu 22.04.3 LTS")

CPU model: Intel(R) Xeon(R) CPU @ 2.20GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 1 physical cores, 2 logical processors, using up to 2 threads

Optimize a model with 217 rows, 70 columns and 420 nonzeros
Model fingerprint: 0x29d50edc
Variable types: 0 continuous, 70 integer (70 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+09]
  Objective range  [9e+02, 1e+03]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+06]
Warning: Model contains large matrix coefficients
         Consider reformulating model or setting NumericFocus parameter
         to avoid numerical issues.
Presolve removed 217 rows and 70 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.02 seconds (0.00 work units)
Thread count was 1 (of 2 available processors)

Solution count 1: -4739.99
No other solutions better than -4739.99

Optimal solution found (tolerance 1.00e-04)
Best objective -4.739993055556e+03, best bound -4.739993055556e+03, gap 0.0000%
selected_players info:  [{'playerName': 'Luka Doncic', 'teamRslt': 0.6666666666666666, 'playStat': 1.0, 'playMin': 41.333333333333336, 'playHeigl
total_point:  260.00694444444446
Selected Player: Luka Doncic, Position: PG, Fantasy Points: 55.77777777777778
Selected Player: Anthony Edwards, Position: SG, Fantasy Points: 48.5
Selected Player: Nikola Jokic, Position: C, Fantasy Points: 60.104166666666664
Selected Player: LeBron James, Position: SF, Fantasy Points: 53.0
Selected Player: Kevin Durant, Position: PF, Fantasy Points: 42.625
```

```
print("finish")
```

```
finish
```