

How to beat your friends in fantasy football

Alex Molas - Senior Data Scientist @ Stuart

@MolasAlex

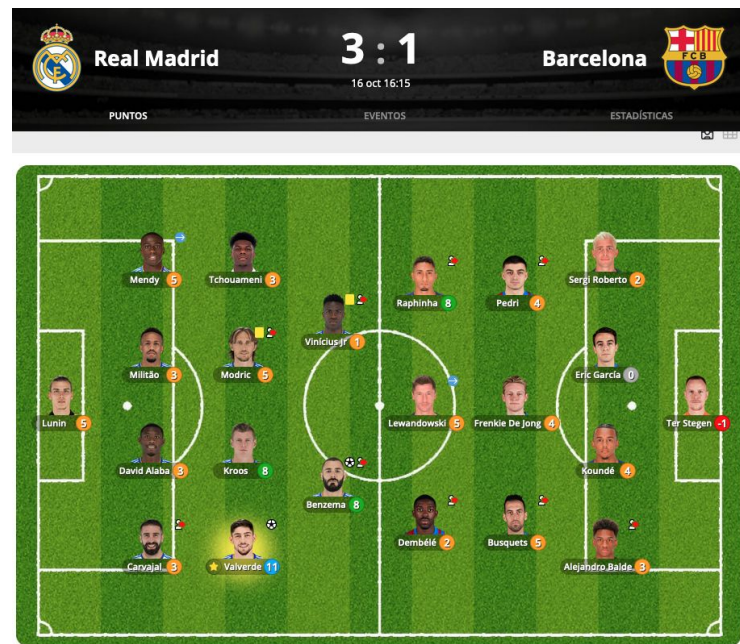
www.alexmolass.com

What's fantasy football?



Participants, using a fixed budget, assemble a team of real life players and score points based on those players' actual performance.

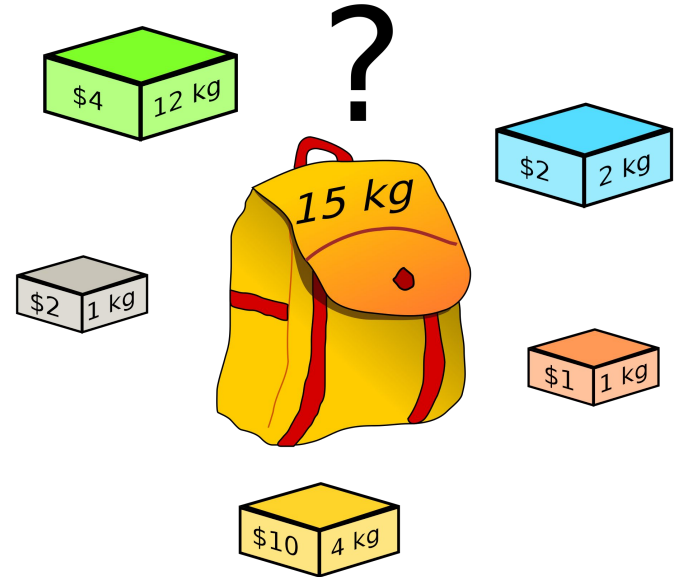
Pos.	Eq.	Jugador	Pts. ▼	VM
DL		Lewandowski	140	65.000.000 €
MC		Valverde	122	30.000.000 €
DL		Griezmann	113	32.000.000 €
DL		Iago Aspas	110	40.000.000 €
MC		Brais Méndez	110	21.000.000 €
MC		Aleix García	108	11.000.000 €
DL		Vinícius Jr	107	54.000.000 €
MC		Pedri	103	43.000.000 €
DL		Dembélé	99	30.000.000 €



Knapsack problem

The problem can be modeled as a **knapsack problem**.

Which boxes should be chosen to maximize the amount of money while still keeping the overall weight under or equal to 15 kg?



Knapsack problem

The problem can be modeled as a **knapsack problem**.

Which boxes should be chosen to maximize the amount of money while still keeping the overall weight under or equal to 15 kg?



Solving knapsack problem

We need

- Set of all players with their characteristics.
- A way to get the value of a player.
- A way to get the price of a player.
- Constraints: max salary we can pay.

```
1 def solve(players: Set[Player],  
2           player_value: Callable[[Player], float],  
3           player_cost: Callable[[Player], float],  
4           max_salary: int) → Set[Player]:  
5
```

Players data

Biwenger doesn't have an open API. Reverse engineering using the **Inspect Tool**.

Pos.	Eq.	Jugador	Pts. ▼	VM	Est.	PJ	Med.	🏠	✂️	Forma
DL		Lewandowski	140	65.000.000 €		14	10,0	76	64	
MC		Valverde	122	30.000.000 €		14	8,7	65	57	
DL		Griezmann	113	32.000.000 €		14	8,1	43	70	
DL		Lago Aspas	110	40.000.000 €		14	7,9	73	37	



```
1 import requests
2 def get_data(competition: str = "la-liga"):
3     data = requests.get(
4         f"{base_path}/competitions/{competition}/data"
5     ).json()
6     return data
7
```

The screenshot shows the Chrome DevTools Network tab with a list of requests. The selected request is a GET request to `cf.biwenger.com/api/v2/competitions/la-liga/data?lang=es&score=2&callback=sonp_1465365483` with a status of 200 OK and a response size of 32.96 kB. The response headers are visible, including `age: 247` and `alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400`.

Players value & cost



```
1 @dataclass(frozen=True)
2 class Player:
3     ...
4     price: int
5     status: str
6     points: int
7     games: int
8     ...
9
```



```
1 def player_value(p: Player) → float:
2     if p.status == 'ok':
3         return p.points / p.games
4     return 0.
5
```

Here you can add some ML!



```
1 def player_cost(p: Player) → int:
2     return p.price
3
```

Solving the problem: ortools




```
1 def solve(players: Set[Player],
2           player_value: Callable[[Player], float],
3           player_cost: Callable[[Player], float],
4           max_salary: int) → Set[Player]:
5     # define the solver
6     solver = pywraplp.Solver('CoinsGridCLP', pywraplp.Solver.CBC_MIXED_INTEGER_PROGRAMMING)
7     ...
```


Solving the problem: ortools



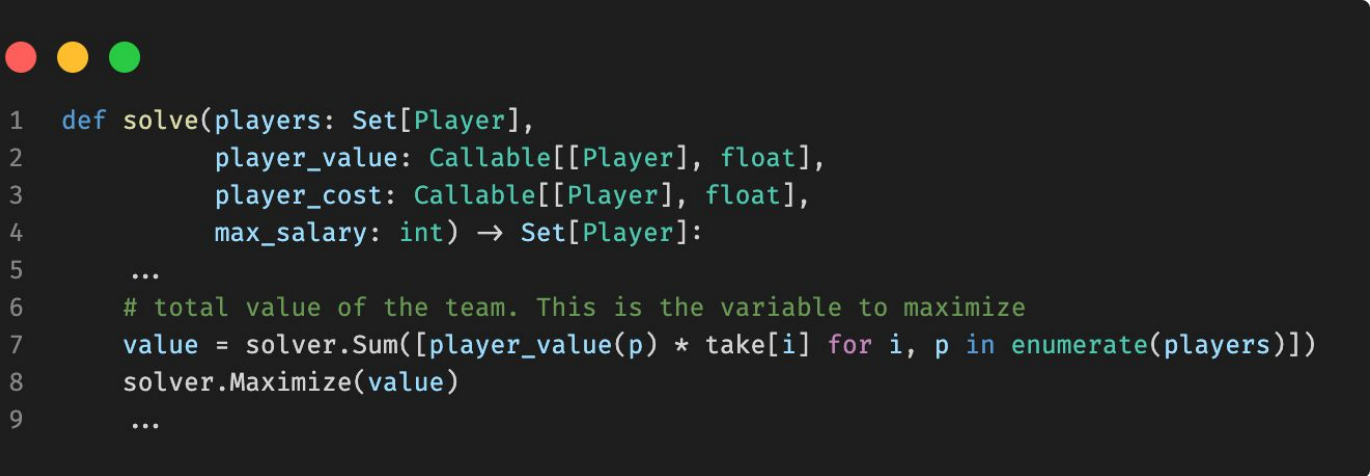
```
1 def solve(players: Set[Player],
2           player_value: Callable[[Player], float],
3           player_cost: Callable[[Player], float],
4           max_salary: int) → Set[Player]:
5     ...
6     # 0 if player is not selected, 1 if player is selected
7     take = [solver.IntVar(0, 1, f'take_{p}') for p in players]
8     ...
```

Solving the problem: ortools



```
1 def solve(players: Set[Player],
2           player_value: Callable[[Player], float],
3           player_cost: Callable[[Player], float],
4           max_salary: int) → Set[Player]:
5     ...
6     # total cost of the team. This variable defines a constraint
7     salary = solver.Sum([player_cost(p) * take[i] for i, p in enumerate(players)])
8     solver.Add(salary ≤ max_salary)
9     ...
```

Solving the problem: ortools



```
1 def solve(players: Set[Player],
2           player_value: Callable[[Player], float],
3           player_cost: Callable[[Player], float],
4           max_salary: int) → Set[Player]:
5     ...
6     # total value of the team. This is the variable to maximize
7     value = solver.Sum([player_value(p) * take[i] for i, p in enumerate(players)])
8     solver.Maximize(value)
9     ...
```

Solving the problem: ortools

```
1 def solve(players: Set[Player],
2           player_value: Callable[[Player], float],
3           player_cost: Callable[[Player], float],
4           max_salary: int) → Set[Player]:
5     ...
6     # solve the problem
7     solver.Solve()
8
9     # build the team
10    team = set()
11    for i, p in enumerate(players):
12        if take[i].SolutionValue():
13            team.add(p)
14    return team
15
```

Solving the problem: ortools

```
1 from ortools.linear_solver import pywraplp
2
3 def solve(players: Set[Player],
4           player_value: Callable[[Player], float],
5           player_cost: Callable[[Player], float],
6           max_salary: int) → Set[Player]:
7     # define the solver
8     solver = pywraplp.Solver('CoinsGridCLP', pywraplp.Solver.CBC_MIXED_INTEGER_PROGRAMMING)
9
10    # 0 if player is not selected, 1 if player is selected
11    take = [solver.IntVar(0, 1, f'take_{p}') for p in players]
12
13    # total cost of the team. This variable defines a constraint
14    salary = solver.Sum([player_cost(p) * take[i] for i, p in enumerate(players)])
15    solver.Add(salary ≤ max_salary)
16
17    # total value of the team. This is the variable to maximize
18    value = solver.Sum([player_value(p) * take[i] for i, p in enumerate(players)])
19    solver.Maximize(value)
20
21    # solve the problem
22    solver.Solve()
23
24    # build the team
25    team = set()
26    for i, p in enumerate(players):
27        if take[i].SolutionValue():
28            team.add(p)
29    return team
```

Best team

- Constraints
 - Salary: 200M
 - 343 / 352 / 433 / 442 / 451 / 532 / 541
- Solution
 - Cost: 198M
 - 343
 - Expected value: 100 points



Thanks for your attention!

@MolasAlex

www.alexmolass.com