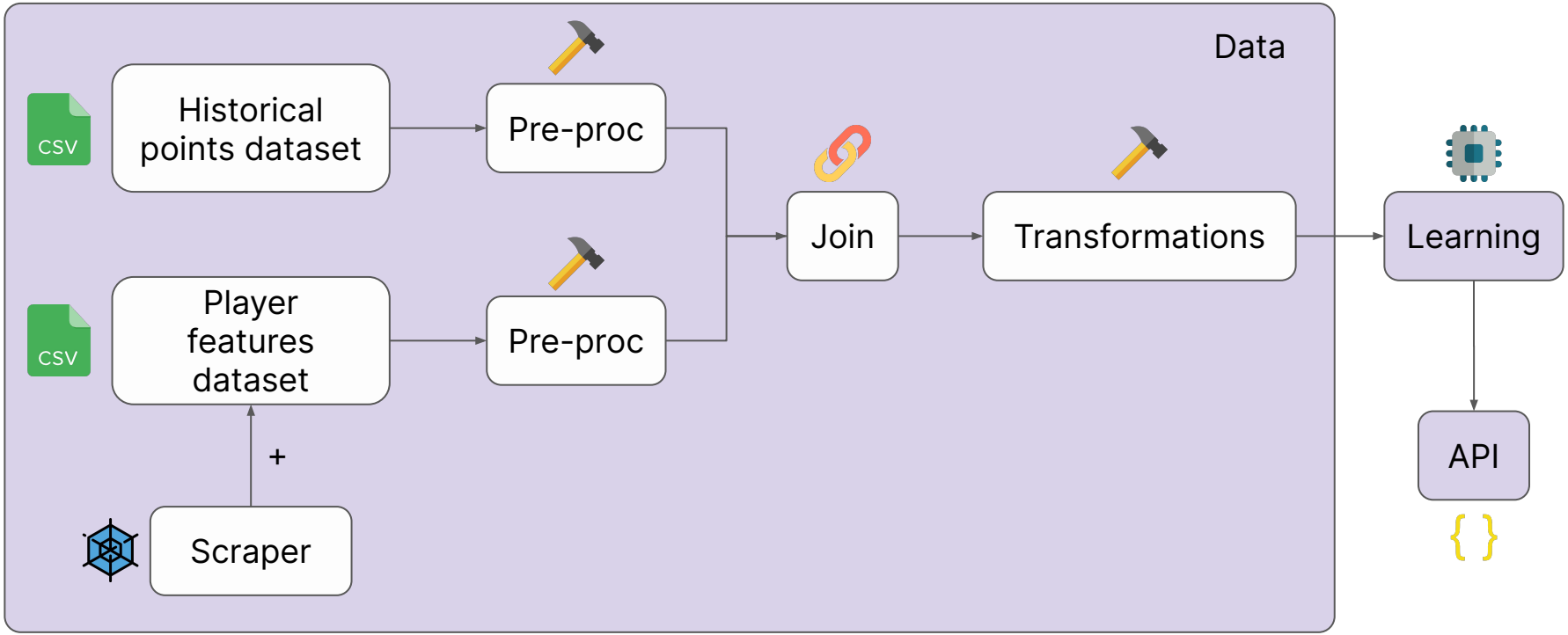
Big Data FC

Predict how many points
European football teams will reach
at the end of the season

Daniele Solombrino, Davide Quaranta

Big Data FC


Data
CSV
Historical points dataset
Pre-proc
CSV
Player features dataset
Pre-proc
+
Scraper
Join
Transformations
Learning
API

Data

What and why:

- **Player features** → players **characterize** teams
- **Player-team associations** → **compose** players into teams
- **League tables** → final team **scores**

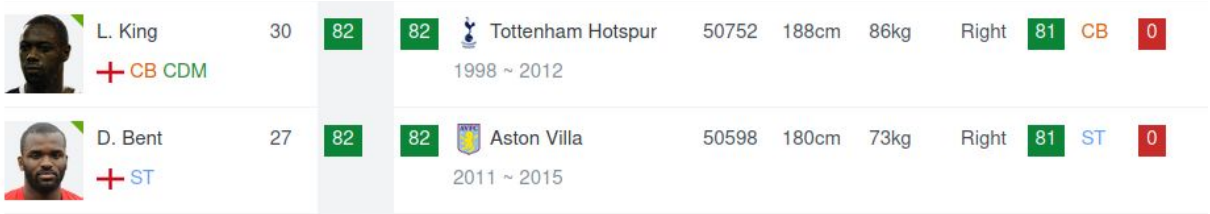
Where:

- FIFA Complete Players Dataset → Kaggle
- European Football dataset → Kaggle

Data

Already available FIFA data (*modern*) only for **limited seasons** (2013 to 2021)

- Scraped seasons (*legacy*): 2007 to 2012
- sofifa.com
- Custom-made scraper:
  - Cookies
  - Parallel requests
  - Caching
  - Auto-throttle



Data

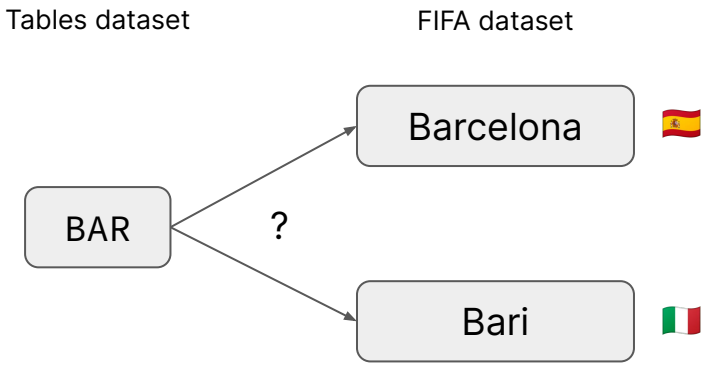
Custom **pre-processing**:

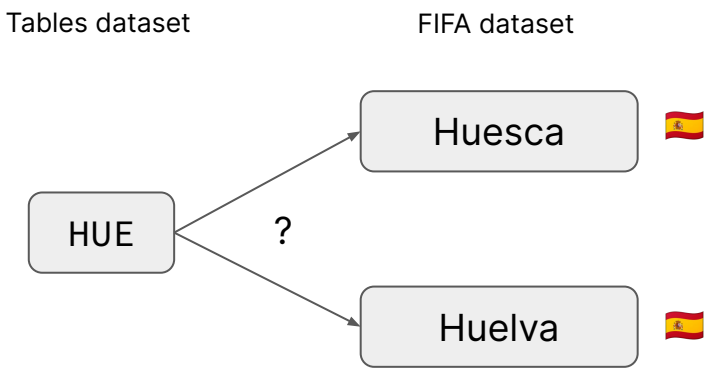
- Different data sources **compatibility**
  - Different **fields naming** (e.g. `pas`/`passing`)
  - **Missing fields** between legacy and modern data
- Heavy **SQL manipulations**:
  - Nested queries
  - Dynamic query generation

Data

Custom **pre-processing**:

- **Team name inconsistencies**/conflicts in score dataset
- Ad-hoc dataset creation





Players DataFrame

```
1 pre_processed_df.select(*columns[0:10]).show()
```

```
+------------+--------------------+--------------------+------+----------------+----------+-------+--------+----+--------+
|  short_name|           club_name|         league_name|season|player_positions|macro_role|overall|   value|pace|shooting|
+------------+--------------------+--------------------+------+----------------+----------+-------+--------+----+--------+
|   S. Padoin|            Juventus|     Italian Serie A|    14|          RM, CM|       2.0|     73| 2100000|  73|      64|
|   S. Padoin|            Juventus|     Italian Serie A|    15|      LB, RM, CM|       1.0|     75| 2600000|  73|      63|
| A. Aquilani|       UD Las Palmas|Spain Primera Div...|    17|         CM, CDM|       2.0|     75| 5000000|  54|      70|
|     S. Pepe|            Juventus|     Italian Serie A|    14|          RM, LM|       2.0|     75| 2900000|  80|      72|
|  F. Modesto|Sporting Club de ...|      French Ligue 1|    14|          CB, RB|       1.0|     72|  400000|  55|      51|
|    A. Boruc|         Bournemouth|English Premier L...|    17|              GK|       0.0|     77|  600000|   0|       0|
|      Aduriz|Athletic Club de ...|Spain Primera Div...|    15|              ST|       3.0|     82|10500000|  71|      83|
|      Aduriz|Athletic Club de ...|Spain Primera Div...|    19|              ST|       3.0|     82| 6500000|  61|      82|
|   G. Pegolo|            Sassuolo|     Italian Serie A|    16|              GK|       0.0|     72| 1200000|   0|       0|
|M. Tacalfred|      Stade de Reims|      French Ligue 1|    14|              CB|       1.0|     71|  700000|  67|      38|
|    J. Audel|          FC Nantes|      French Ligue 1|    15|      LM, RM, ST|       2.0|     71| 1300000|  73|      67|
|    C. Maury|         GFC Ajaccio|      French Ligue 1|    15|              GK|       0.0|     68|  725000|   0|       0|
|   T. Starke|   FC Bayern München|German 1. Bundesliga|    16|              GK|       0.0|     74| 2000000|   0|       0|
|      Dorado|      Rayo Vallecano|Spain Primera Div...|    15|          CB, LB|       1.0|     75| 2400000|  54|      37|
|        Duda|           Málaga CF|Spain Primera Div...|    16|         CAM, RM|       2.0|     72|  250000|  33|      74|
|   Corominas|            Elche CF|Spain Primera Div...|    14| CAM, RM, LM, ST|       2.0|     75| 3100000|  77|      71|
|       Moyá|       Real Sociedad|Spain Primera Div...|    18|              GK|       0.0|     80| 6000000|   0|       0|
|     Garrido|       UD Las Palmas|Spain Primera Div...|    15|         LB, LWB|       1.0|     73| 2000000|  68|      58|
|      Reyes|          Sevilla FC|Spain Primera Div...|    14|          RM, LM|       2.0|     77| 4500000|  72|      71|
|  S. Riether|       FC Schalke 04|German 1. Bundesliga|    15|              RB|       1.0|     72| 1200000|  64|      54|
+------------+--------------------+--------------------+------+----------------+----------+-------+--------+----+--------+
only showing top 20 rows
```

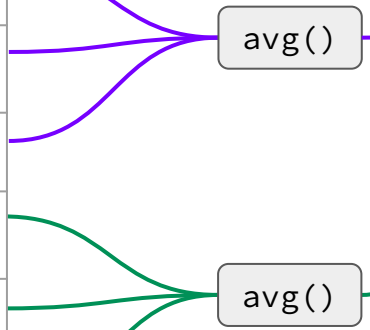
Scores DataFrame

```
1 pre_processed_seasonal_scores_df.show()
```

```
+------+--------------------+--------------------+------+-----+
|season|              league|           club_name|points|place|
+------+--------------------+--------------------+------+-----+
|    18|English Premier L...|   Tottenham Hotspur|  71.0|    4|
|    18|English Premier L...|   Manchester United|  66.0|    6|
|    08|Spain Primera Div...|RC Recreativo de ...|  33.0|   20|
|    16|Spain Primera Div...|            RC Celta|  45.0|   13|
|    16|Spain Primera Div...|       UD Las Palmas|  39.0|   14|
|    15|     Italian Serie A|               Carpi|  38.0|   18|
|    12|  Holland Eredivisie|        FC Groningen|  43.0|    7|
|    13|  Holland Eredivisie|           FC Twente|  63.0|    3|
|    12|      French Ligue 1|      Stade de Reims|  43.0|   14|
|    16|      French Ligue 1|Sporting Club de ...|  34.0|   20|
|    16|English Premier L...|   Tottenham Hotspur|  86.0|    2|
|    07|Spain Primera Div...|          Sevilla FC|  64.0|    5|
|    18|Spain Primera Div...|      Rayo Vallecano|  32.0|   20|
|    17|German 1. Bundesliga|         Hannover 96|  39.0|   13|
|    19|German 1. Bundesliga|   Borussia Dortmund|  69.0|    2|
|    19|  Holland Eredivisie|                Ajax|  56.0|    1|
|    09|      French Ligue 1|  Olympique Lyonnais|  72.0|    2|
|    10|      French Ligue 1|            OGC Nice|  46.0|   17|
|    12|      French Ligue 1|Olympique de Mars...|  71.0|    2|
|    20|      French Ligue 1|           FC Nantes|  40.0|   18|
+------+--------------------+--------------------+------+-----+
only showing top 20 rows
```

Main DataFrame: from players to teams

Season	Club	Name	Passing	Shooting
20		Fofana	70	51
20		Tielemans	82	78
20		Vardy	68	81
18		van Dijk	71	60
18		Wijnaldum	78	75
18		Mané	80	83
...	...	...	..	..



Season	Club	Passing	Shooting
20		73.3	70
18		76.3	72.6
...	...	...	...

Main DataFrame: from players to teams

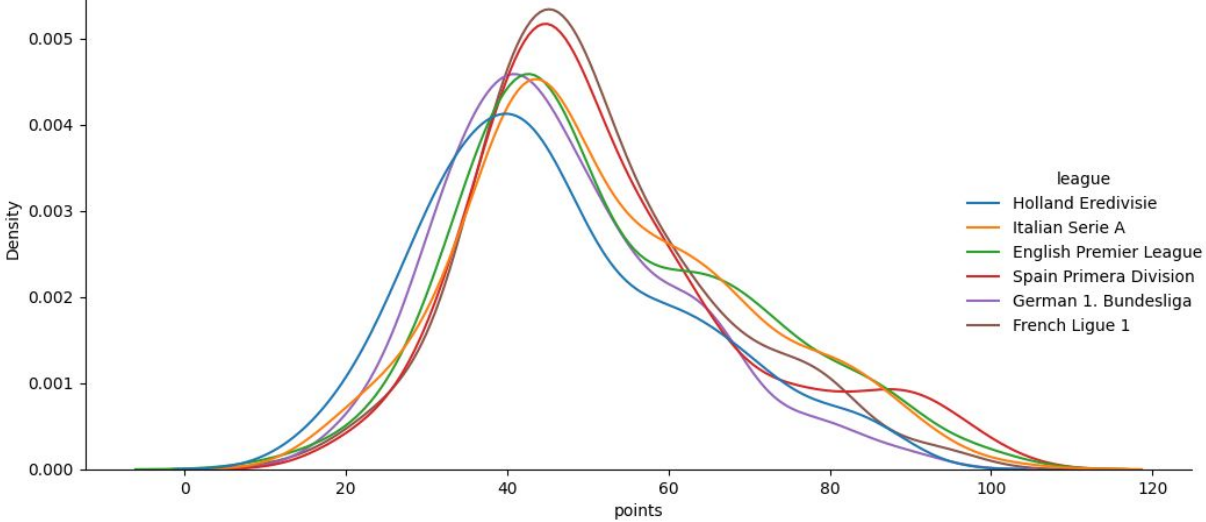
```
football_teams_df = football_teams_df.select(
    "season", "club_name", *PLAYER_FEATURES
).groupBy(
    ["season", "club_name"]
).agg(
    { player_feature: "avg" for player_feature in PLAYER_FEATURES }
)
```

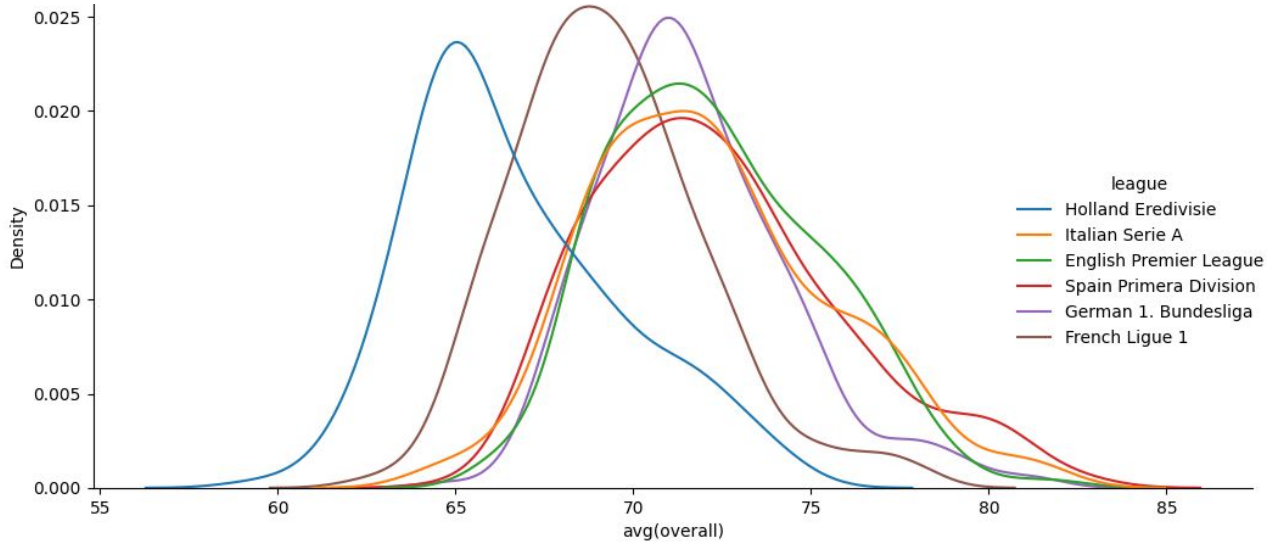
Main DataFrame

```
df.select(
    "season", "league", "club_name", "avg(overall)", "avg(pace)", "points", "place"
).show()
```

```
+------+--------------------+--------------------+------------------+------------------+------+-----+
|season|              league|           club_name|     avg(overall)|         avg(pace)|points|place|
+------+--------------------+--------------------+------------------+------------------+------+-----+
|    18|  Holland Eredivisie|        ADO Den Haag|66.33333333333333|            56.625|  45.0|    9|
|    16|     Italian Serie A|       Chievo Verona|              71.3|              55.9|  43.0|   14|
|    16|English Premier L...|         Southampton|71.93939393939394| 64.66666666666667|  46.0|    8|
|    17|Spain Primera Div...|Athletic Club de ...|             75.75|59.214285714285715|  43.0|   16|
|    15|English Premier L...|   Manchester United|76.25925925925925| 59.96296296296296|  66.0|    5|
|    19|English Premier L...|   Manchester United|76.84848484848484| 67.03030303030303|  66.0|    3|
|    19|German 1. Bundesliga|          Hertha BSC|71.75757575757575| 63.84848484848485|  41.0|   10|
|    15|English Premier L...|         Bournemouth|69.89655172413794| 65.10344827586206|  42.0|   16|
|    20|      French Ligue 1|FC Girondins de B...| 70.6896551724138| 55.44827586206897|  45.0|   12|
|    15|English Premier L...|             Watford|71.32142857142857|62.785714285714285|  45.0|   13|
|    18|      French Ligue 1|    AS Saint-Étienne|70.11111111111111| 63.51851851851852|  66.0|    4|
|    16|English Premier L...|     West Ham United|72.54545454545455| 66.93939393939394|  45.0|   11|
|    15|German 1. Bundesliga|          1. FC Köln|71.79166666666667|62.666666666666664|  43.0|    9|
|    16|  Holland Eredivisie|       SC Heerenveen|66.76666666666667|              59.4|  43.0|    9|
|    14|      French Ligue 1|Sporting Club de ...|66.21212121212122| 60.93939393939394|  47.0|   12|
|    15|Spain Primera Div...|     Atlético Madrid|             76.92|             64.84|  88.0|    3|
|    17|      French Ligue 1|    Stade Rennais FC|69.15151515151516| 61.36363636363637|  58.0|    5|
|    17|English Premier L...|        Swansea City|71.24242424242425| 58.06060606060606|  33.0|   18|
|    17|German 1. Bundesliga| Eintracht Frankfurt|72.33333333333333| 64.27272727272727|  49.0|    8|
|    15|German 1. Bundesliga|       VfB Stuttgart|70.79310344827586|62.206896551724135|  33.0|   17|
+------+--------------------+--------------------+------------------+------------------+------+-----+
only showing top 20 rows
```

Data Distributions





Learning

Our practical debut in ML + Experimental approach

Regression on `points`
Classification on `macro_place`💡

- Linear and Logistic Regression
- Prediction Trees
- Random Forest
- Gradient Boosted Trees
- Support Vector Machines
- Multi Layer Perceptron

Multiple assumptions

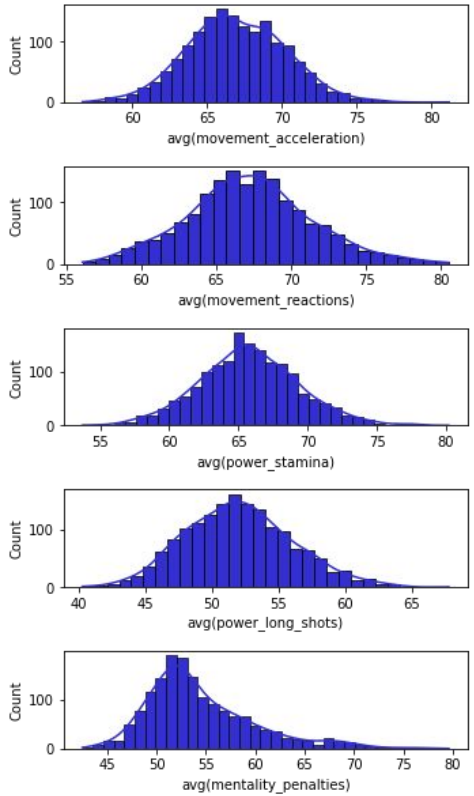
- Attempt 1: naive
- Attempt 2: “less is more”
- Attempt 3: clustering
- Attempt 4: RP coefficient

Learning: set-up

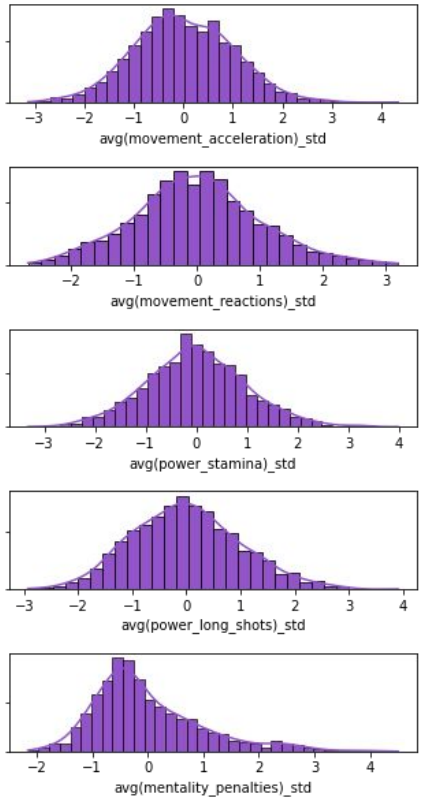
- ~35k players, ~1.7k teams
- **PySpark**
- 70/20/10 train/validation/test split
- 5-fold **cross-validation**
- **Hyper-parameter** tuning via **search grid** method
- **Regression metrics**: R^2, Adjusted R^2 and MSE
- **Classification metrics**: Accuracy, F1, precision, recall, etc.

Attempt 1: data distributions

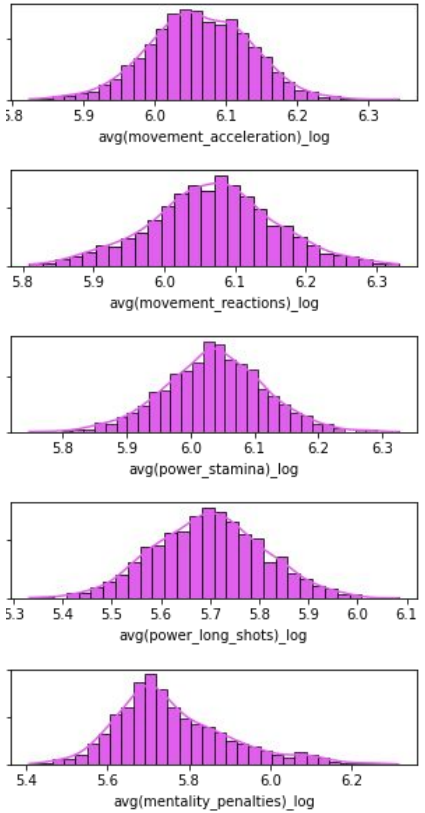
Raw data



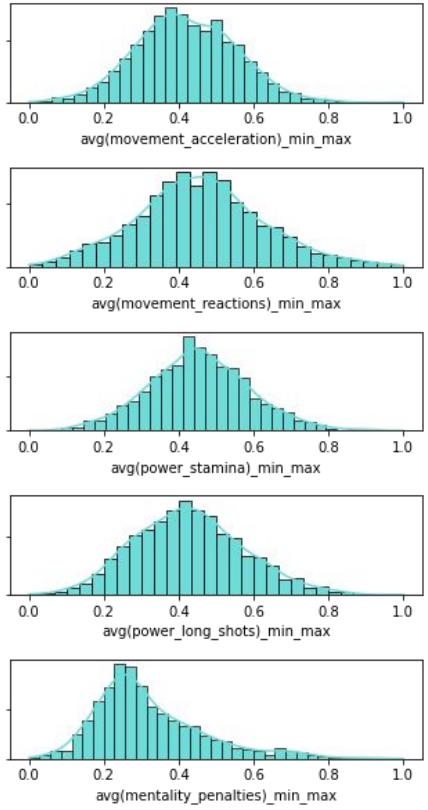
Standardization



Log transformation



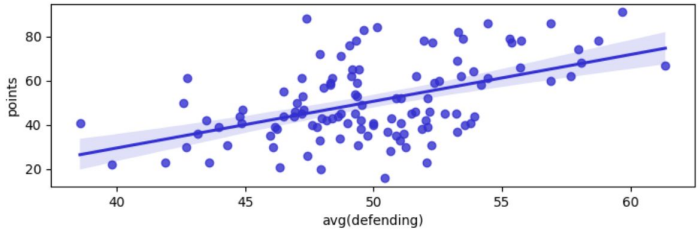
Min-max scaling

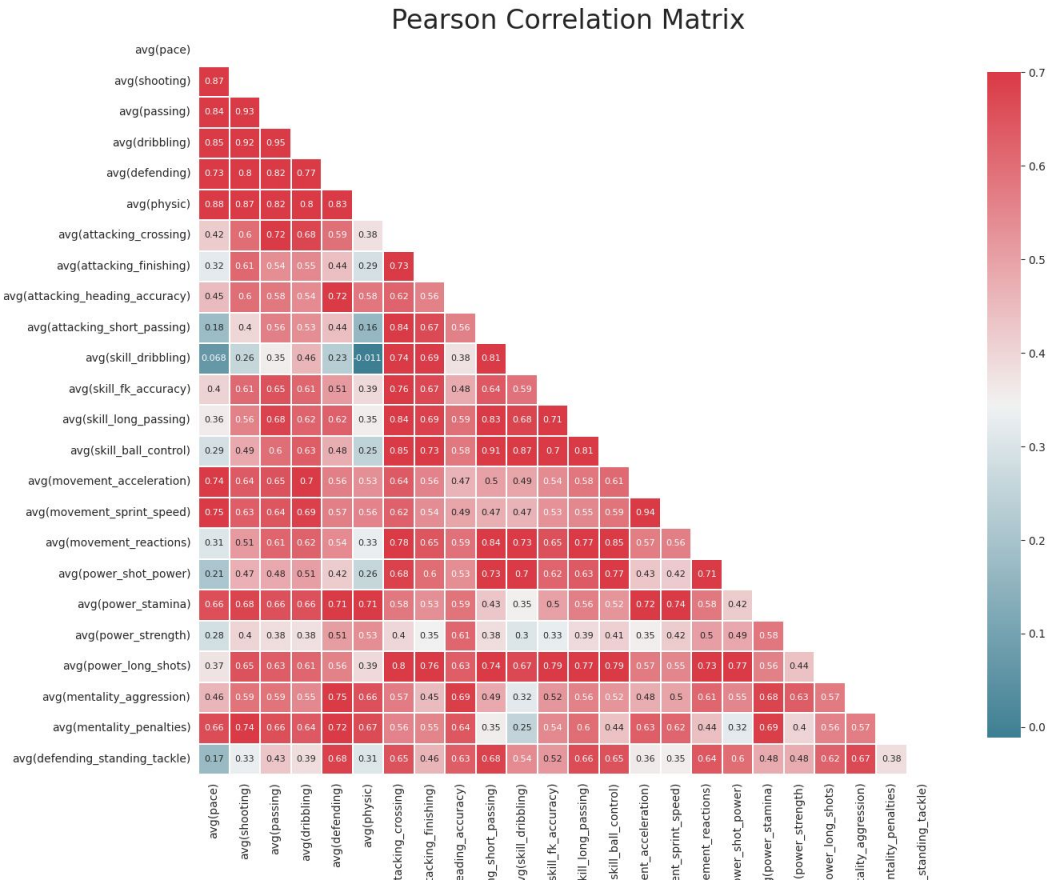


Attempt 1: “naive”

Considers all **24** player **features**:

- High **variance**
- High **correlations**





Attempt 1: learning results

- **Low** test scores
  - Regression → r^2: 35%
  - Classification → accuracy: 30%
- **Tree-based** models perform (slightly) better
- No improvement with **hyper-parameter tuning**

Attempt 2: “less is more”

Dimensionality reduction

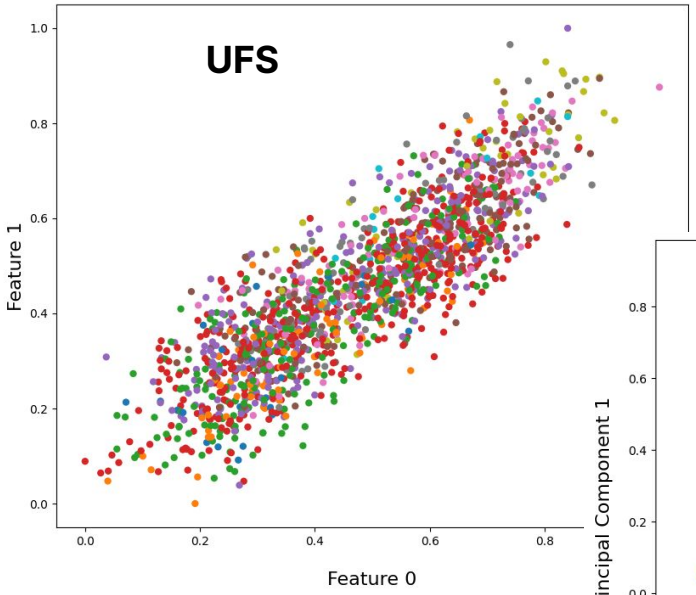
Why:

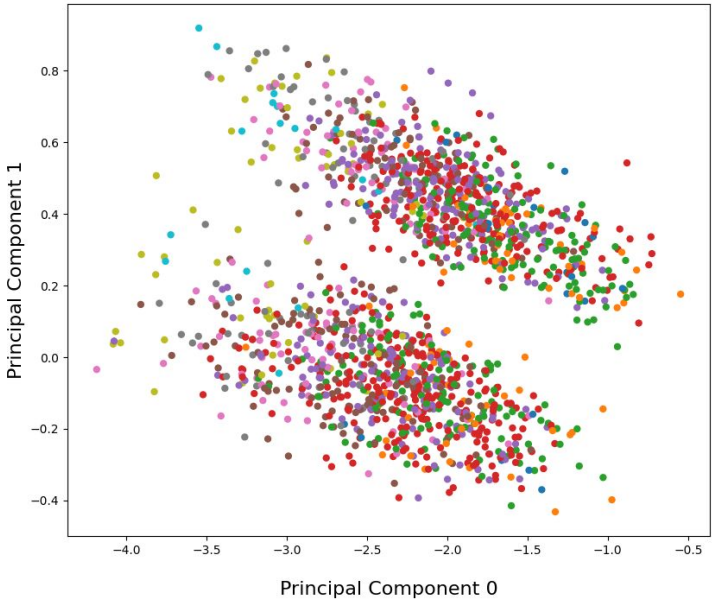
- Address feature **correlation**
- Less **curse of dimensionality** risk

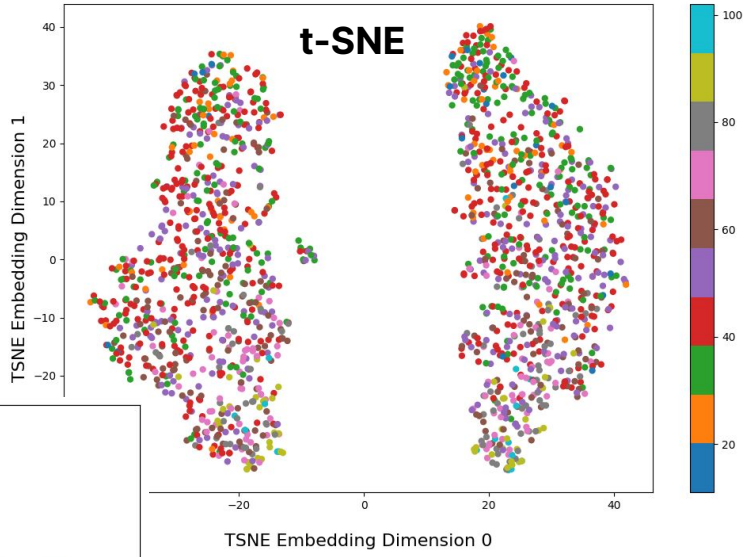
How:

- **PCA**
- **t-SNE**
- **U**nivariate **F**eature **S**election
- **`Overall`** player ability as feature
- Economic player **`value`** as feature

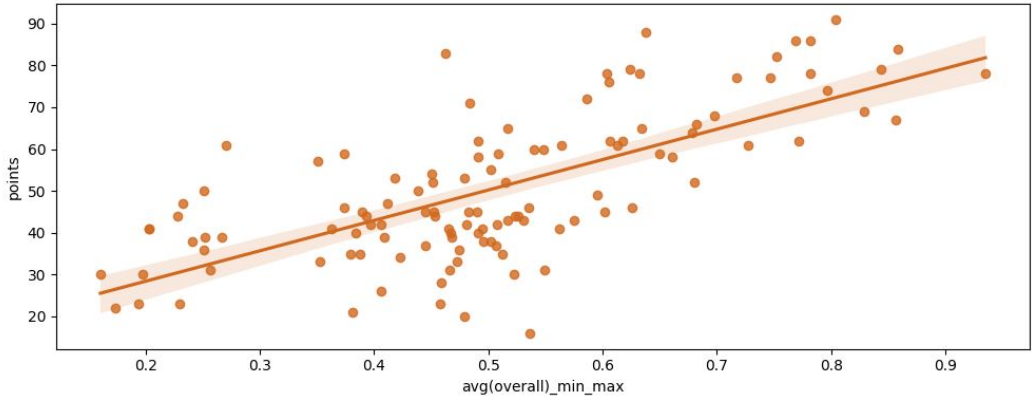
Attempt 2: feature spaces

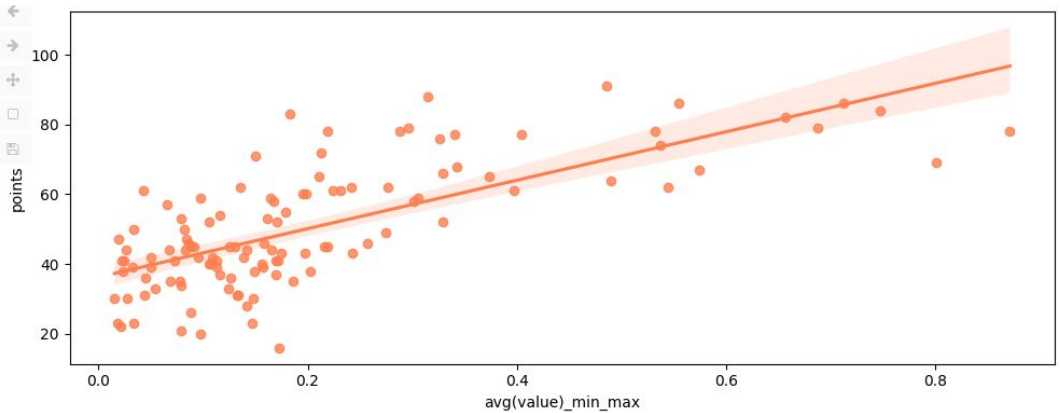






Attempt 2: focus on `overall` and `value`

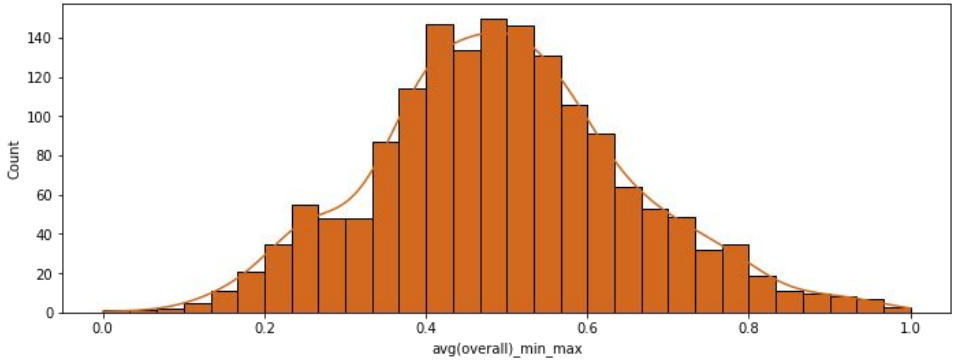


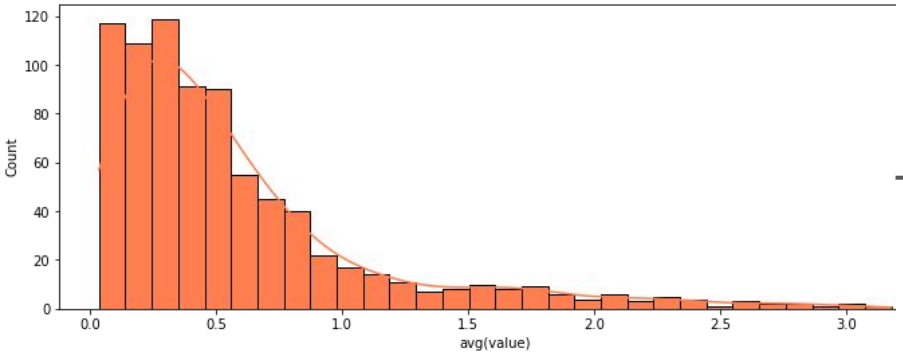


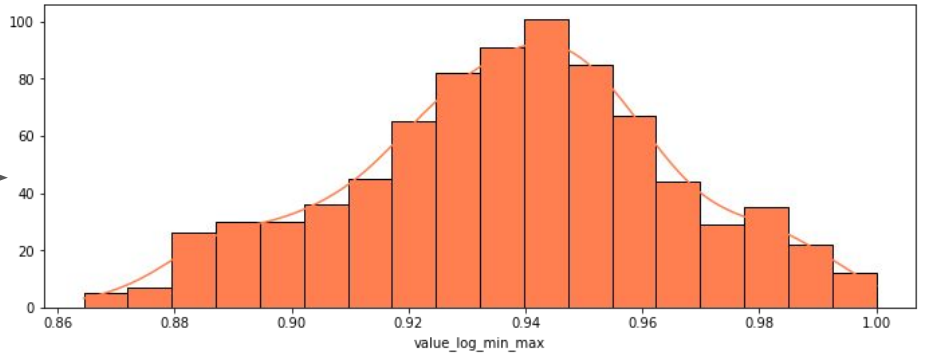
💡 `overall` or `value` capture additional hidden structure

- **Reduced variance**
  - **Confuted** by plots similar to attempt 1
- High overall-value **correlation**

Attempt 2: `overall` and `value` distributions







Attempt 2: learning results

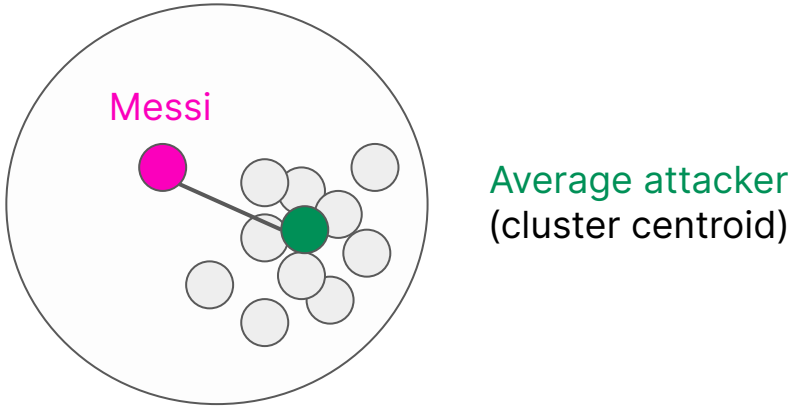
As expected:

- Very **close results** to attempt 1
- Hypothesis **confuted** → overall (or value) do not reduce variance
- Alignment with **papers**:
  - “Causes of Success in the La Liga and How to Predict Them”
  - “Who wins the championship?”

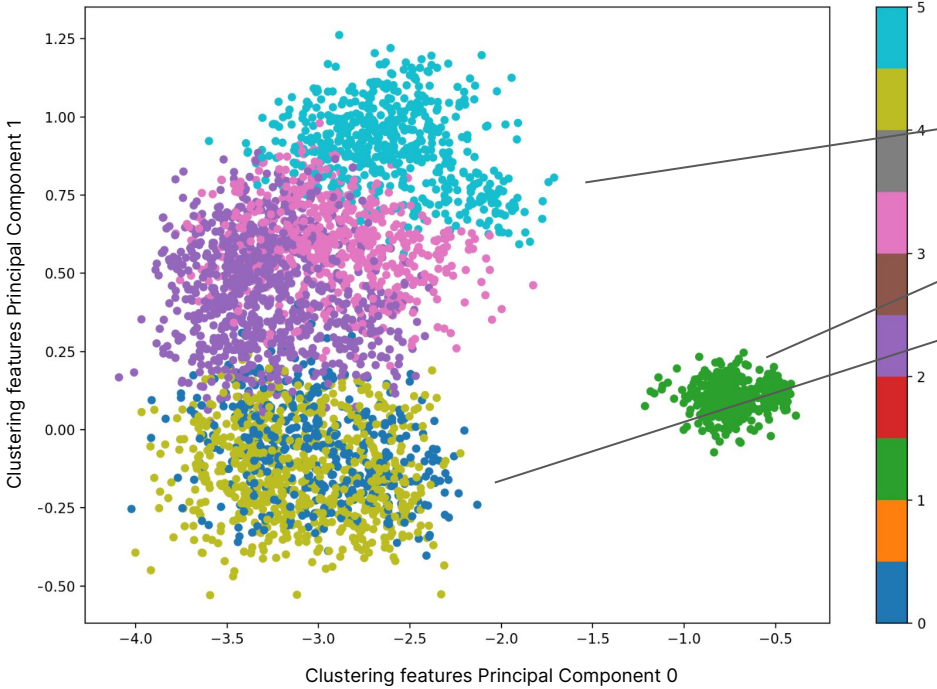
Attempt 3: clustering

From “plain” features to **learning-produced features**:

- Perform **clustering** on players
- Player = **distance** from its **cluster centroid**
  - 💡Cluster centroid ≈ average player
  - 💡Distance from cluster centroid ≈ player uniqueness



Attempt 3: visualizations



Player types	Separation
Defensive	Acceptable
Goalkeeper	Good
Offensive	Bad

- **Elbow method** to elect best number of clusters (6)
- Low **Silhouette coefficient**

Attempt 3: from players to teams

Season	Club	Name	Macro role	Centroid distance
20	Arsenal	White	1	(0.4, 0.5)
20	Arsenal	Holding	1	(0.6, 0.1)
20	Arsenal	Saka	3	(0.1, 0.1)
20	Arsenal	Nketiah	3	(0.2, 0.3)
19	Barcelona	...	...	...
...	...	...	...	...

avg()

avg()

Season	Club	Avg dist. macro role 1	...	Avg dist. macro role 3
20	Arsenal	(0.5, 0.3)	...	(0.15, 0.2)
19	Barcelona	...	...	...
...	...	...	...	...

∀ (club, season, #clusters)

Attempt 3: from players to teams

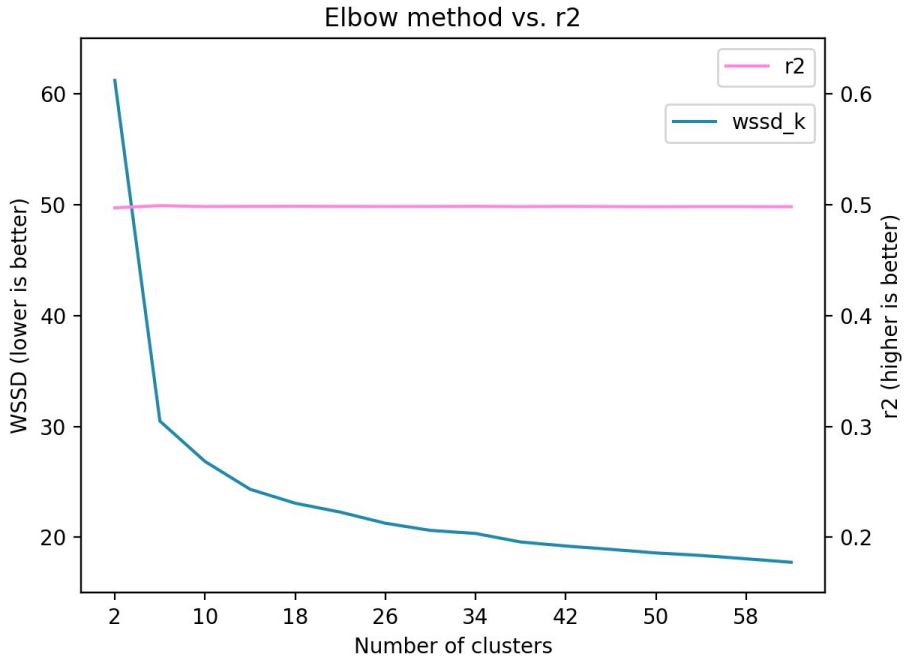
```python
# factory method to get the specific subquery, needed by every number of clusters
def generate_subquery(macro_role, k):
    return f"""(
        case
            when macro_role='{macro_role}' then avg_distance_from_centroid_k_{k}
        else NULL
        end
    ) as avg_dist_macro_role_{int(macro_role)}_k_{k}
    """
```

```python
teams_df.createOrReplaceTempView("t")

temp = dict()

for k in K_RANGE:
    temp[k] = (
        spark.sql(
            f"""
                select season, club_name, {generate_subquery(0.0, k)}, {generate_subquery(1.0, k)},
{generate_subquery(2.0, k)}, {generate_subquery(3.0, k)}, {generate_subquery(4.0, k)}, {generate_subquery(5.0, k)},
{generate_subquery(6.0, k)}, {generate_subquery(7.0, k)}
                from t
            """
        )
        .groupBy("season", "club_name")
        .agg(
            avg(f"avg_dist_macro_role_0_k_{k}").alias(f"avg_dist_macro_role_0_k_{k}"),
            avg(f"avg_dist_macro_role_1_k_{k}").alias(f"avg_dist_macro_role_1_k_{k}"),
            avg(f"avg_dist_macro_role_2_k_{k}").alias(f"avg_dist_macro_role_2_k_{k}"),
            avg(f"avg_dist_macro_role_3_k_{k}").alias(f"avg_dist_macro_role_3_k_{k}"),
            avg(f"avg_dist_macro_role_4_k_{k}").alias(f"avg_dist_macro_role_4_k_{k}"),
            avg(f"avg_dist_macro_role_5_k_{k}").alias(f"avg_dist_macro_role_5_k_{k}"),
            avg(f"avg_dist_macro_role_6_k_{k}").alias(f"avg_dist_macro_role_6_k_{k}"),
            avg(f"avg_dist_macro_role_7_k_{k}").alias(f"avg_dist_macro_role_7_k_{k}"),
        )
    )
```

```python
teams_df = temp[K_RANGE[0]]

for i in range(1, len(K_RANGE)):
    teams_df = teams_df.join(
        temp[K_RANGE[i]],
        on=["season", "club_name"]
    )
```

Attempt 3: learning results

- Same performances as before…
- … 💡what if best k of elbow does not yield to best learning?
  - **Confuted** by plot



Attempt 4: RP coefficient

🔍 Start of the season **data insufficient** to predict final scores

- Additional, **non-easily measurable** factors influence results
  - **Fans** support (on and off the pitch)
  - **Coach** and **staff** competence
  - Players' **psychological state**
  - Societary **solidity**
- Such factors may be captured by a statistical prior
  - 💡 **reward**/**penalize** teams according to **past performances**

Attempt 4: RP coefficient

```
value_rp_normalized = value + rp_tradeoff * rp_coefficient
```

```
add_normalize_by_rp_UDF = udf(
    lambda points, rp, tradeoff: points + tradeoff * rp, DoubleType()
)
```

Attempt 4: RP coefficient

`rp_coefficient`:

- In function of the **average ranking of the team** in previous seasons
- (Tries to) **counter variance**

```
rp_df = spark.sql(
    f"""
    select t_rp_coeff.season, t_rp_coeff.club_name,
        avg(
            (
                select sub.place
                where sub.season < t_rp_coeff.season and
                      sub.club_name == t_rp_coeff.club_name
            )
        ) as rp_coeff
    from t_rp_coeff, t_rp_coeff as sub
    group by t_rp_coeff.season, t_rp_coeff.club_name
    order by t_rp_coeff.season desc
    """
).fillna(MAX_PLACE)

rp_df = rp_df.join(
    df, on=["club_name", "season"]
)

rp_df = rp_df.withColumn("rp_coeff", MAX_PLACE - col("rp_coeff"))

# ...
```

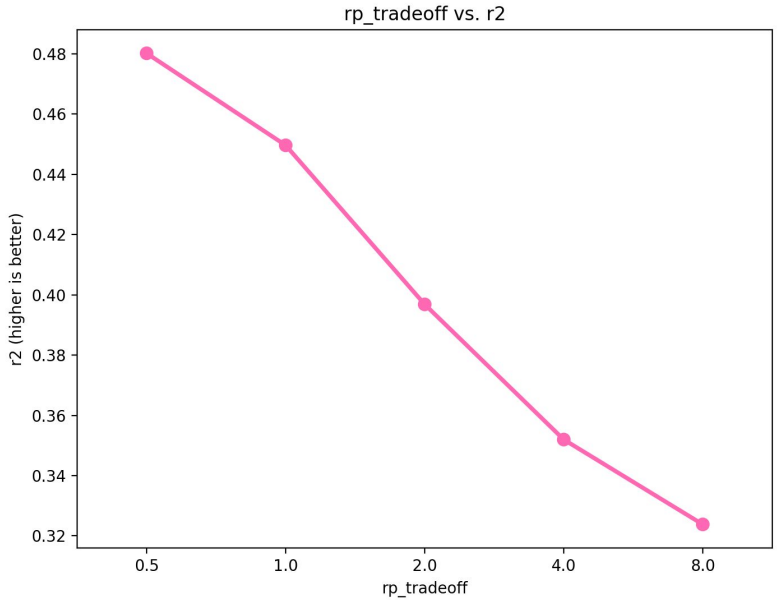
Attempt 4: RP coefficient

`rp_tradeoff`:

- Coefficient weight → Data bias/variance tradeoff
- Usage inspired by **L_p Regularization** formula

```
# ...

for tradeoff, col_name in zip(ADD_RP_TRADEOFF, RP_NORMALIZED_COLS):
    rp_df = rp_df.withColumn(
        col_name, add_normalize_by_rp_UDF(
            col("avg(overall)"), col("rp_coeff"), lit(tradeoff)
        )
    )

    # ...
```

Attempt 4: learning results



- `rp_tradeoff = 0.5` → very small improvement
- `rp_tradeoff = 1` → similar results to other attempts
- `rp_tradeoff > 1` → worse performances
- 🔍 Attempt 4 **hypothesis confuted**

The REST API

Python3 Flask app.

Endpoints:

- GET **`/player`**: (player name, season) → player features
- GET **`/team`**: (team name, season) → team + player features
- GET **`/table`**: (league name, season) → table
- GET **`/predict`**: (league name, season) → predicted table

Available on GitHub

```python
#!python3

import json
from flask import Flask, request, jsonify

import api
from db import Database
from spark import Spark

app = Flask(__name__)
spark = Spark()
db = Database(spark)

@app.route("/predict", methods=["GET"])
def predict_table():
    data = json.loads(request.data)
    prediction = api.get_prediction(
      db,
      data["league"],
      data["season"]

    return jsonify(prediction)

# ...
```

Honorable mentions

Performed tests not submitted/presented due to **no different results**:

Multiple combinations of the following **scenarios**:

1. Using `max` and `min` instead of `avg` to aggregate player features into teams
2. Excluding of bench-players
3. Learning on single seasons
4. Learning on single leagues
5. Excluding a subset of roles
6. More/less granular role encoding
7. More/less granular `macro_place` encoding
8. Different RP coefficient formulas
9. Excluding goalkeepers
10. RP coefficient on `points` rather than `value`

Conclusions

- **Predict** football teams' end-of-the-season **points** from its players
- Cross-linking **multiple** datasets
- Custom-made **scraper**
- **Data inconsistency** handling
- 4 **assumptions** x 9 **learning models** = 36 learnings:
  - Regression (4 models)
  - Classification (5 models)
  - Cumulative 150+ hours spent in training
- Researched **literature** confirms our observations
- **API**