BUSINESS INTELLIGENCE INCLUDES FOOTBALL ANALYSIS

Boussoukaya Islam

Apr 11, 2022

CONTENTS

CHAPTER

ONE

KLOPPY FEATURING INSTAT

1.1 Introduction

Each football data provider uses its own unique format to describe the sequence of a game. Therefore, the software written to analyze this data must be suitable for a specific provider and cannot be used without modifications to analyze the data of other providers.

Kloppy is a Python package that addresses the challenges posed by the variety of data formats and aims to be the fundamental building block for processing soccer tracking and event data.

- How can the kloppy package deal with all these providers and generate the same result for each one? and which providers can support?

1.2 Problematic

Before delving into the details we will identify the problematic . The Company is delivered with football data from different providers, In our case, we will point to the Instat provider. We simply want to adapt the Instat provider to the Kloppy package to be able to use its many features.

1.3 The Kloppy Package

1.3.1 What is Kloppy ?

As previously mentioned kloppy is a great package , It provides (de)serializers, standardized data models, filters, and transformers to facilitate work with different tracking and event data from various providers,So its main features are to load data and transform it into standardized models and vice versa. In the following sections we will try to break-down the package to better understand the process and it works.

1.3.2 Kloppy providers

Kloppy support numerous provider with different format file (JSON & LXML) as the table below shows. To simplify the process, these providers display a different data structure within each format, “Kloppy” will proceed with each one to provide the same final result structure for each game.

Provider	Format
Datafactory	JSON
Metrica	JSON
Opta	LXML
Sportec	LXML
Statsbomb	JSON
Wyscout	JSON

1.3.3 What is events & line-up data in football?

- Events DataEvent data can be considered as a record of the entire game, it records every move on the pitch during the match. It includes the positions of each player and data pertaining to every action during the match which include pass, shot, interception, card, goal etc.
- Line-up Data The line-up data contain the lineup of players, starting & ending tactic . It includes player information full name , jersey number , starting & ending position etc.

1.3.4 Example

To better understand what we are looking for as a result here an code example and its output.

```
from kloppy import opta

dataset = opta.load(
    f7_data="../../kloppy/tests/files/opta_f7.xml",
    f24_data="../../kloppy/tests/files/opta_f24.xml",

    # Optional arguments
    coordinates="opta",
    event_types=["pass", "shot"]
)

dataset.to_pandas().head()
```

	event_id	event_type	result	success	period_id	timestamp	end_timestamp	ball_state	ball_owning_team	team_id	player_id	coordinates_x	coo
0	2370983921	PASS	COMPLETE	True	1	0.160	None	alive	43	43	165809	50.1	
1	2370983925	PASS	COMPLETE	True	1	2.842	None	alive	43	43	146941	29.2	
2	2370983937	PASS	COMPLETE	True	1	5.019	None	alive	43	43	171314	34.7	
3	2370983945	PASS	COMPLETE	True	1	7.444	None	alive	43	43	121145	39.2	
4	2370983957	PASS	COMPLETE	True	1	9.322	None	alive	43	43	171314	24.0	

Fig. 1.1: Code exemple of Opta provider. Using filter for event types as pass and shot.

1.3.5 Kloppy Event Types

"Kloppy" classifies events depending on the type of event. For each event type kloppy defines a class that we will detail.

Events Types
Pass
Shot
Take on
Carry
Substitution
PlayerOn/Off
Card
Recovery
Ball Out
Foul Committed
Generic

Pass Event

A class entitled "Pass Event" contains the following attributes:

- recieve timestamp
- receiver player
- receiver coordinates **Point**
- result **Pass Result**

Attribute	Definition
COMPLETE	Complete pass
INCOMPLETE	Incomplete pass (intercepted)
OUT	Ball went out
OFFSIDE	Offside

- Pass type, a enumeration class named "Pass Type"

PassType
Cross
Hand pass
Head pass
High pass
Launch
Simple pass
Smart pass
Long pass
Through ball
Chipped pass
Flick on
Assit
Assist 2nd
Swithc of play

Shot Event

A class entitled “Shot Event” contains the following attributes:

- result of the shot as **shot Result**

Attribute	Definition
Goal	Shot resulted in a goal
Ooff targer	Shot was off target
Post	Shot hit the post
Blocked	Shot was blocked by another player
Saved	Shot was saved by the keeper

- result coordinates **Point**

Take on Event

A class entitled “Take-on Event” contains the following attributes:

- result of the take on **Take on Result**

Attribute	Definition
Complete	Complete take-on
Incomplete	Incomplete take-on
Out	Ball went out

Carry Event

A class entitled “Carry Event” contains the following attributes:

- end timestamp
- end coordinates **Point**
- Result **Carry Result**

Attribute	Definition
Complete	Complete carry
Incomplete	Incomplete carry

Substitution Event

A class entitled “Substitution Event” contains as attribute:

- replacement player **Player**

Player On/Off Event

Two classes “PlayerOn Event” & “PlayerOff Event” that contains as attribute:

- Event type that the player is concerned on/off

Card Event

A class named “Card Event”, it contains as attribute:

- card type **Card Type**

Attribute	Definition
FIRST_YELLOW	First yellow card
SECOND_YELLOW	Second yellow card
RED	Red card

Recovery Event

A class named “Recovery Event”, it contains as attribute:

- Event type **Recovery**

Ball out Event

A class named “Ball out Event”, it contains as attribute:

- Event type **Ball out**

Foul Commit Event

A class named “Foul Commit Event”, it contains as attribute:

- Event type **Foul Commit**

Generic Event

Kloppy classifies an event which does not match the above mentioned events (Unrecognised event type) as a generic event.

1.3.6 Kloppy qualifiers

Each type of event will be qualified with one or more of the following qualifiers.

Qualifiers	
Set Piece	Corner / FreeKick / Penalty /Throw in /KickOff
Body Part	Chest / Right Foot / Left Foot / Head
Pass Type	Cross / LongBall /ThroughBall / Launch / ChippedBall / Assist / 2nd Assist /SwitchOfPlay

1.3.7 Dependencies

Python libraries that Kloppy depend on

Black : Black is the uncompromising Python code formatter. By using it, you agree to cede control over minutiae of hand-formatting. In return, Black gives you speed, determinism, and freedom from pycodestyle nagging about formatting. Black makes code review faster by producing the smallest diffs possible.

Lxml : The lxml XML toolkit is a Pythonic binding for the C libraries libxml2 and libxslt. It is unique in that it combines the speed and XML feature completeness of these libraries with the simplicity of a native Python API, mostly compatible but superior to the well-known ElementTree API.

Requests : Python module that you can use to send all kinds of HTTP requests. It is an easy-to-use library with a lot of features ranging from passing parameters in URLs to sending custom headers and SSL Verification.

NetworkX : This python package is used for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. It is used to study large complex networks represented in form of graphs with nodes and edges. Using networkx we can load and store complex networks.

PyTest : PyTest is a testing framework that allows users to write test codes using Python programming language. It helps you to write simple and scalable test cases for databases, APIs, or UI. PyTest is mainly used for writing tests for APIs. It helps to write tests from simple unit tests to complex functional tests.

Pandas : pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.

Pre-commit : A framework for managing and maintaining multi-language pre-commit Git hooks.

1.4 The InStat Provider

1.4.1 What is InStat ?

InStat is a worldwide sports performance analysis company, it prepares statistics for pre-game analysis, collects data during games and breaks down the game after it is completed.

1.4.2 InStat Files

As we mentionned before we will analyze the line-up & events file.

Line-up

Events

Each row is considered an event that occurred within the match, every action has the following characteristics: time, type of action , place on the pitch, the player who committed it, the opponent (in the case of challenges).

```
<data>
  <match id="1725171"/>
  <first_team id="247" name="Liverpool FC" color="0xFF7788" color_num="0x000000">
    <lineup>
      <main starting_tactic="4-3-3 down" ending_tactic="4-3-3 down">
        <player id="3668" firstname="James Philip" lastname="Milner" num="7" starting_position_id="100"
        starting_position_name="Substitute player" ending_position_id="12" ending_position_name="Defender - Left"
        birthday="1986-01-04" country1_id="252" country1_name="England" starting_lineup="0" ending_lineup="1"/>
        <player id="3749" firstname="Xherdan" lastname="Shaqiri" num="23" starting_position_id="100"
        starting_position_name="Substitute player" ending_position_id="100" ending_position_name="Substitute player"
        birthday="1991-10-10" country1_id="233" country1_name="Switzerland" country2_id="96" country2_name="Kosovo"
        starting_lineup="0" ending_lineup="0"/>
        <player id="4322" firstname="Georginio Gregion Emile" lastname="Wijnaldum" num="5" starting_position_id="24"
        starting_position_name="Midfielder - Left central" ending_position_id="24" ending_position_name="Midfielder -
        Left central" birthday="1990-11-11" country1_id="138" country1_name="Netherlands" country2_id="198"
        country2_name="Suriname" starting_lineup="1" ending_lineup="0"/>
        <player id="5242" firstname="Thiago" lastname="Alcantara do Nascimento" num="6" starting_position_id="44"
        starting_position_name="Midfielder - Right central" ending_position_id="44" ending_position_name="Midfielder -
        Right central" birthday="1991-04-11" country1_id="77" country1_name="Spain" country2_id="29"
        country2_name="Brazil" starting_lineup="1" ending_lineup="1"/>
        <player id="5886" firstname="Jordan Brian" lastname="Henderson" num="14" starting_position_id="100"
        starting_position_name="Substitute player" ending_position_id="100" ending_position_name="Substitute player"
        birthday="1990-06-17" country1_id="252" country1_name="England" starting_lineup="0" ending_lineup="0"/>
        <player id="9403" firstname="Roberto Firmino" lastname="Firmino Barbosa De Oliveira" num="9"
        starting_position_id="36" starting_position_name="Forward - Central" ending_position_id="36"
        ending_position_name="Forward - Central" birthday="1991-10-02" country1_id="29" country1_name="Brazil"
        starting_lineup="1" ending_lineup="0"/>
        <player id="16425" firstname="Mohamed" lastname="Salah" num="11" starting_position_id="55"
        starting_position_name="Attacking midfielder - Right" ending_position_id="55" ending_position_name="Attacking
        midfielder - Right" birthday="1992-06-15" country1_id="65" country1_name="Egypt" starting_lineup="1"
        ending_lineup="1"/>
```


Fig. 1.2: This a sample of the lineup XML file


```
<match id="1725171"/>
<row id="5137" number="1" action_id="16310" action_name="GK" player_id="137016" player_name="Alisson Ramses Alisson"
team_id="247" team_name="Liverpool FC" position_id="31" position_name="Goalkeeper" half="1" second="0"
standart_id="1" standart_name="Open play" ts="2021-05-24T00:41:57.365" dl="0"/>
<row id="5139" number="66" action_id="16520" action_name="RD" player_id="272150" player_name="Trent Alexander-Arnold"
team_id="247" team_name="Liverpool FC" position_id="52" position_name="Defender - Right" half="1" second="0"
standart_id="1" standart_name="Open play" ts="2021-05-24T00:41:57.365" dl="0"/>
<row id="5141" number="47" action_id="16420" action_name="RCD" player_id="424786" player_name="Nathaniel Phillips"
team_id="247" team_name="Liverpool FC" position_id="42" position_name="Defender - Right central" half="1" second="0"
standart_id="1" standart_name="Open play" ts="2021-05-24T00:41:57.365" dl="0"/>
<row id="5143" number="46" action_id="16220" action_name="LCD" player_id="395318" player_name="Rhys Williams"
team_id="247" team_name="Liverpool FC" position_id="22" position_name="Defender - Left central" half="1" second="0"
standart_id="1" standart_name="Open play" ts="2021-05-24T00:41:57.365" dl="0"/>
<row id="5145" number="26" action_id="16120" action_name="LD" player_id="157248" player_name="Andrew Robertson"
team_id="247" team_name="Liverpool FC" position_id="12" position_name="Defender - Left" half="1" second="0"
standart_id="1" standart_name="Open play" ts="2021-05-24T00:41:57.365" dl="0"/>
<row id="5147" number="6" action_id="16440" action_name="RCM" player_id="5242" player_name="Thiago Alcantara do
Nascimento" team_id="247" team_name="Liverpool FC" position_id="44" position_name="Midfielder - Right central"
half="1" second="0" standart_id="1" standart_name="Open play" ts="2021-05-24T00:41:57.365" dl="0"/>
<row id="5149" number="5" action_id="16240" action_name="LCM" player_id="4322" player_name="Georginio Gregion Emile
Wijnaldum" team_id="247" team_name="Liverpool FC" position_id="24" position_name="Midfielder - Left central" half="1"
second="0" standart_id="1" standart_name="Open play" ts="2021-05-24T00:41:57.365" dl="0"/>
<row id="5151" number="3" action_id="16330" action_name="CDM" player_id="120900" player_name="Fabio Henrique Tavares"
team_id="247" team_name="Liverpool FC" position_id="33" position_name="Defensive midfielder - Central" half="1"
second="0" standart_id="1" standart_name="Open play" ts="2021-05-24T00:41:57.365" dl="0"/>
<row id="5153" number="10" action_id="16150" action_name="LAM" player_id="88763" player_name="Sadio Mane"
team_id="247" team_name="Liverpool FC" position_id="15" position_name="Attacking midfielder - Left" half="1"
second="0" standart_id="1" standart_name="Open play" ts="2021-05-24T00:41:57.365" dl="0"/>
<row id="5155" number="9" action_id="16360" action_name="CF" player_id="9403" player_name="Roberto Firmino Firmino
Barbosa De Oliveira" team_id="247" team_name="Liverpool FC" position_id="36" position_name="Forward - Central"
half="1" second="0" standart_id="1" standart_name="Open play" ts="2021-05-24T00:41:57.365" dl="0"/>
<row id="5157" number="11" action_id="16550" action_name="RAM" player_id="16425" player_name="Mohamed Salah"
team_id="247" team_name="Liverpool FC" position_id="55" position_name="Attacking midfielder - Right" half="1"
second="0" standart_id="1" standart_name="Open play" ts="2021-05-24T00:41:57.365" dl="0"/>
```


Fig. 1.3: This a sample of the events XML file

1.4.3 InStat Actions

InStat has a specific way of structuring events as well as its type of action, each action has a specefic identifier .

GENERIC ACTION	ACTION
Pass	Attacking pass - Key pass - Assist - Key Assit
Challenge	Challenge - Air challenge - Tackle - Dribble
Disipline	Yellow card - Red card - RC for 2 YC
Shot	Shot on target - Shot into the bar - Shot blocked
Set Piece	Open play - Throw-in - Free-Kick - Corner
Goal	Goal - Own goal
Cross	Cross accurate - cross accurate

1.4.4 InStat Coordinates System

The Coordinates system is X,Y coordinates of the player on the pitch , it refers to his position. The origin of the coordinates is in the bottom left and the vertical orientation is from bottom to top of the pitch. The dimension of the picth is 105 * 68 .

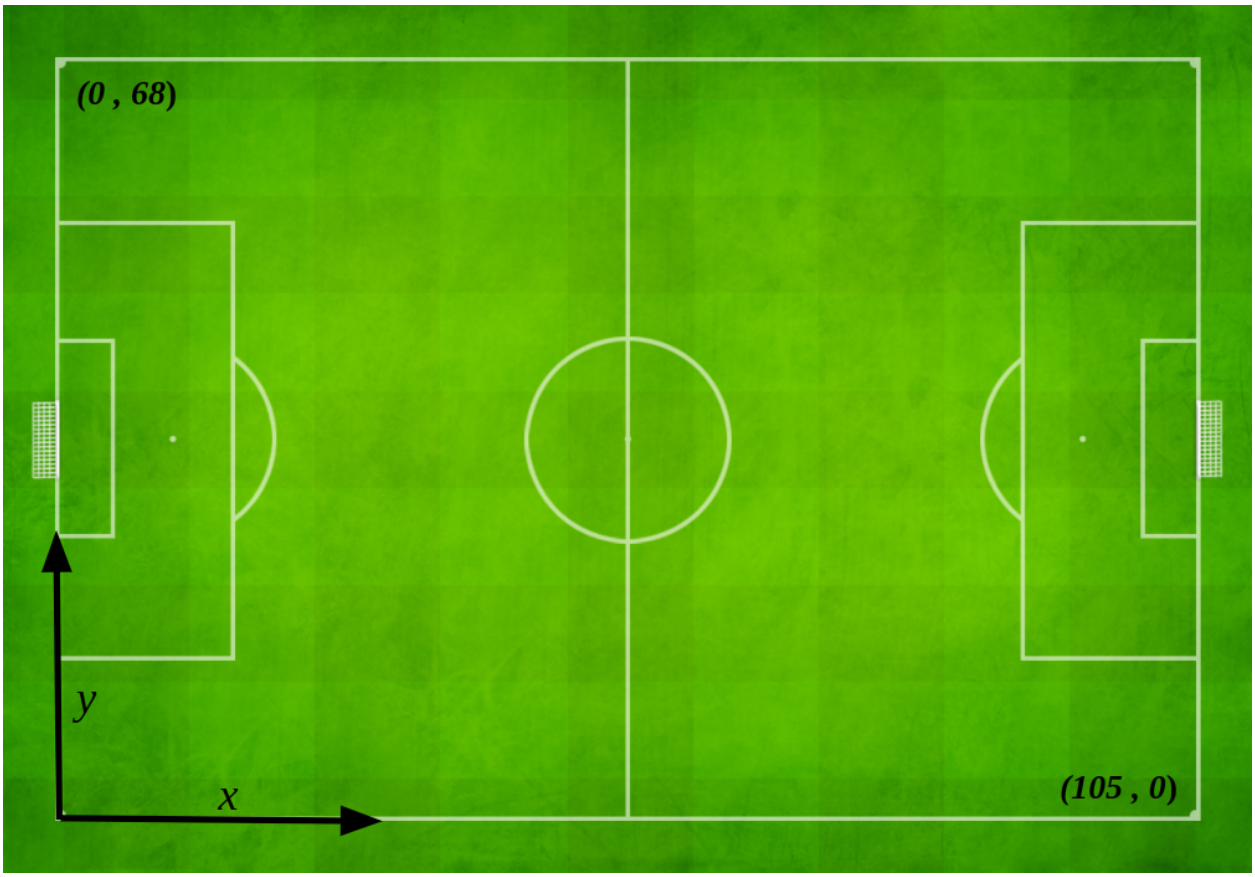

Fig. 1.4: InStat pitch representation.

1.4.5 InStat Standart

Every event is qualified with a standart name as the following table shows.

Standart name
Open Play
Throw-in
Indirect Freekick
Direct free kick
Corner

1.4.6 InStat Body Part

Every event is qualified with a Body name that reffers to the body part as the following table shows.

Body name
Right foot
Left foot
Header
Hand
Other body part

1.5 Workflow

In this section, we will describe in detail the approach adopted.

1.5.1 Instat Inputs

Every provider in Kloppy has an input class that contains the files that will be used. We created the InStatInputs class which has as attributes the file line-up and events as the following code states.

```
class InStatInputs(NamedTuple):
    lineup_data: IO[bytes]
    events_data: IO[bytes]
```

1.5.2 Coordinates System

Each provider has its own coordinate system, so we defined a data class that that specifies the pitch parameters.

```
@dataclass
class InstatCoordinateSystem(CoordinateSystem):
    @property
    def provider(self) -> Provider:
        return Provider.INSTAT

    @property
    def origin(self) -> Origin:
```

(continues on next page)

(continued from previous page)

```
        return Origin.BOTTOM_LEFT

    @property
    def vertical_orientation(self) -> VerticalOrientation:
        return VerticalOrientation.BOTTOM_TO_TOP

    @property
    def pitch_dimensions(self) -> PitchDimensions:
        return PitchDimensions(
            x_dim=Dimension(0, 105),
            y_dim=Dimension(0, 68),
        )
```

1.5.3 Action Id

In this section, we will try to match the types of InStat events and actions to Kloppy classes in order to proceed with the deserialization process. We have created a file called "InStat Events" that contains all types of events with their identifiers. This is a sample of the code.

```
EVENT_TYPE_BALL_OUT = "27000"
EVENT_TYPE_CORNER_AWARDED ="5060"
BALL_OUT_EVENTS = [EVENT_TYPE_BALL_OUT, EVENT_TYPE_CORNER_AWARDED]

EVENT_BODYPART_RIGHT_FOOT = 1
EVENT_BODYPART_LEFT_FOOT = 2
EVENT_BODYPART_HEADER = 3
EVENT_BODYPART_BODY = 4
EVENT_BODYPART_HAND = 5

SET_PIECE_THROW_IN = 2
SET_PIECE_INDIRECT_FREE_KICK = 3
SET_PIECE_DIRECT_FREE_KICK = 4
SET_PIECE_CORNER = 5
```

1.5.4 Qualifiers Functions

Qualifiers describe each single event with one or more of these following qualifiers (Body part, Card, Set piece and Pass).

Body Part Qualifier

This qualifier function returns which body part is used with each event.

```
def _get_event_bodypart(body_id : int) -> List[Qualifier]:
    qualifiers =  []
    if body_id == instat_events.EVENT_BODYPART_RIGHT_FOOT:
        qualifiers.append(BodyPartQualifier(value=BodyPart.RIGHT_FOOT))
    elif body_id == instat_events.EVENT_BODYPART_LEFT_FOOT:
        qualifiers.append(BodyPartQualifier(value=BodyPart.LEFT_FOOT))
    elif body_id == instat_events.EVENT_BODYPART_HEADER:
        qualifiers.append(BodyPartQualifier(value=BodyPart.HEAD))
```

(continues on next page)

(continued from previous page)

```
    elif body_id == instat_events.EVENT_BODYPART_BODY :
        qualifiers.append(BodyPartQualifier(value=BodyPart.OTHER))
    elif body_id == instat_events.EVENT_BODYPART_HAND :
        qualifiers.append(BodyPartQualifier(value=BodyPart.OTHER))

    return qualifiers
```

Card Qualifier

This qualifier function returns which card type is used with each card event.

```
def _get_event_card(action_id : str) -> List[Qualifier]:
    qualifiers =  []
    if action_id == instat_events.EVENT_TYPE_RED_CARD:
        qualifiers.append(CardQualifier(value=CardType.RED))
    elif action_id == instat_events.EVENT_TYPE_SECOND_YELLOW_CARD:
        qualifiers.append(CardQualifier(value=CardType.SECOND_YELLOW))
    elif action_id == instat_events.EVENT_TYPE_FIRST_YELLOW_CARD:
        qualifiers.append(CardQualifier(value=CardType.FIRST_YELLOW))
    return qualifiers
```

Pass Qualifier

This qualifier function returns the type of the pass concerned with each pass event.

```
def _get_event_pass(action_id : str) -> List[Qualifier]:
    qualifiers =  []
    if action_id in instat_events.EVENT_TYPE_CROSS:
        qualifiers.append(PassQualifier(value=PassType.CROSS))
    elif action_id in instat_events.EVENT_TYPE_ASSIST:
        qualifiers.append(PassQualifier(value=PassType.ASSIST))
    elif action_id in instat_events.EVENT_TYPE_ASSISIT_2ND:
        qualifiers.append(PassQualifier(value=PassType.ASSIST_2ND))
    return qualifiers
```

Set Piece Qualifier

As we mentioned previously, the equivalent of set piece in Instat is **standart**. This qualifier function returns the type of the set piece concerned with each event.

```
def _get_event_setpiece(standart_id) -> List[Qualifier]:
    qualifiers = []
    if standart_id == instat_events.SET_PIECE_CORNER:
        qualifiers.append(SetPieceQualifier(value=SetPieceType.CORNER_KICK))
    elif standart_id == instat_events.SET_PIECE_DIRECT_FREE_KICK:
        qualifiers.append(SetPieceQualifier(value=SetPieceType.FREE_KICK))
    elif standart_id == instat_events.SET_PIECE_INDIRECT_FREE_KICK:
        qualifiers.append(SetPieceQualifier(value=SetPieceType.FREE_KICK))
    elif standart_id == instat_events.SET_PIECE_THROW_IN:
        qualifiers.append(SetPieceQualifier(value=SetPieceType.THROW_IN))
    return qualifiers
```

1.5.5 Parse Functions

The parse functions are used to retrieve data from the file to specific classes set by kloppy. This is where de-serialization takes its parts and transforms the data extracted from the XML file into the classes identified by KLoppy. As stated in the preceding chapter, Qualifier describes each event type with the correspondents qualifiers.

Parse Team

This function identifies the team (id, name, ground, team side and starting formation) & players of the team (player id, team, jersey number, first name, last name, starting, position) as the following code shows.

```
def _parse_team(team_root , team_side
                )-> Team:
    team_id = team_root.attrib["id"]
    formation = "-".join(re.findall(r'\d+', team_root.lineup.main.attrib["starting_
↪tactic"]))
    team = Team(
        team_id=str(team_id),
        name=team_root.attrib["name"],
        ground=Ground.HOME
        if str(team_side) == "first_team"
        else Ground.AWAY,
        starting_formation=FormationType(formation),
    )
    team.players = [
        Player(
            player_id=player_elm.attrib["id"],
            team=team,
            jersey_no=int(player_elm.attrib["num"]),
            first_name=player_elm.attrib["firstname"],
            last_name=player_elm.attrib["lastname"],
            starting=True if player_elm.attrib["starting_lineup"] == 1 else False,
            position=Position(
                position_id=player_elm.attrib["starting_position_id"],
                name=player_elm.attrib["starting_position_name"],
                coordinates=None,
            ),
        )
        for player_elm in team_root.lineup.main.iterchildren("player")
    ]
    return team , team_id
```

Parse Score

This function returns the score of both teams, the home and away team.

```
def _parse_score (events_root,home_team_id,away_team_id):

    home_score = 0
    away_score = 0
    try:
        for event in events_root.iterchildren("row"):
            if event.attrib["action_id"]=="8010" and event.attrib["team_id
↪"]==str(home_team_id):
```

(continues on next page)

(continued from previous page)

```
                home_score +=1
            elif event.attrib["action_id"]=="8010" and event.attrib["team_id
↪"]==str(away_team_id):
                away_score +=1

    except KeyError:
        pass
    return home_score,away_score
```

Parse Card

This function returns the type of the card.

```
def _get_event_card(action_id : str) -> List[Qualifier]:
    qualifiers =  []
    if action_id == instat_events.EVENT_TYPE_RED_CARD:
        qualifiers.append(CardQualifier(value=CardType.RED))
    elif action_id == instat_events.EVENT_TYPE_SECOND_YELLOW_CARD:
        qualifiers.append(CardQualifier(value=CardType.SECOND_YELLOW))
    elif action_id == instat_events.EVENT_TYPE_FIRST_YELLOW_CARD:
        qualifiers.append(CardQualifier(value=CardType.FIRST_YELLOW))
    return qualifiers
```

Parse pass

This function takes as an input a pass event and the action id, it returns the attributes used in the Pass Event class.

```
def _parse_pass(action_id: str, row_elm) -> Dict:
    if action_id in instat_events.EVENT_TYPE_CROSS:
        if action_id in instat_events.EVENT_TYPE_CROSS_INCOMPLETE:
            result = PassResult.INCOMPLETE
        elif action_id in instat_events.EVENT_TYPE_CROSS_COMPLETE:
            result = PassResult.COMPLETE
    elif action_id in instat_events.EVENT_TYPE_ASSIST:
        result = PassResult.COMPLETE
    elif action_id in instat_events.EVENT_TYPE_ASSISIT_2ND:
        result = PassResult.COMPLETE

    receiver_coordinates = Point(
            x=float(row_elm.attrib["pos_dest_x"]), y=float(row_elm.attrib["pos_dest_y
↪"])
        )
    qualifiers = _get_event_qualifiers(row_elm)

    return dict(
        result=result,
        receiver_coordinates=receiver_coordinates,
        receiver_player=None,
        receive_timestamp=None,
        qualifiers=qualifiers,
    )
```

Parse shot

This function takes as an input a shot event, the action id, and the coordinates. It returns the attributes used in the Shot Event class.

```
def _parse_shot(
            action_id: str,
            coordinates: Point,
            row_elm
               ) -> Dict:
    if action_id == instat_events.EVENT_TYPE_SHOT_GOAL:
        result = ShotResult.GOAL
    elif action_id == instat_events.EVENT_TYPE_SHOT_OWN_GOAL:
        result = ShotResult.OWN_GOAL
    elif action_id == instat_events.EVENT_TYPE_SHOT_BLOCKED:
        result = ShotResult.BLOCKED
    elif action_id == instat_events.EVENT_TYPE_SHOT_POST:
        result = ShotResult.POST
    elif action_id == instat_events.EVENT_TYPE_SHOT_SAVED:
        result = ShotResult.SAVED
    else:
        result = None
    qualifiers = _get_event_qualifiers(row_elm)
    return dict(
        coordinates=coordinates,
        result=result,
        qualifiers=qualifiers)
```

Parse take-on

This function takes as an input the action id and it returns the result of the take on.

```
def _parse_take_on(action_id: str) -> Dict:
    if action_id in instat_events.EVENT_TYPE_TAKE_ON_COMPLETE:
        result = TakeOnResult.COMPLETE
    elif action_id == instat_events.EVENT_TYPE_TAKE_ON_INSUCC_DRIBBLING:
        result = TakeOnResult.INCOMPLETE
    return dict(result=result)
```

1.5.6 InStat Deserializer

It is the final class that includes all previous sections and will return the final Dataset after the de-serialization process took part.

```
return EventDataset(
            metadata=metadata,
            records=events,
        )
```

EventDataset is a data class that contains the metadata & records of the game.

Metadata

The Metadata contains both team (home & away team), Period (timestamp of the first & second half), Score of the game, coordiantes system of Instat .

```
metadata = Metadata(
             teams=teams,
             periods=periods,
             pitch_dimensions=transformer.get_to_coordinate_system().pitch_dimensions,
             score=score,
             frame_rate=None,
             orientation=Orientation.ACTION_EXECUTING_TEAM,
             flags=DatasetFlag.BALL_OWNING_TEAM,
             provider=Provider.INSTAT,
             coordinate_system=transformer.get_to_coordinate_system(),
         )
```

Records

The records refer to the list of events after it is generated (de-serialized) into Kloppy classes.

1.6 Result

```
dataset = instat.load(
    lineup_data="/home/islam/Downloads/data/lineup_1725171.xml",
    events_data="/home/islam/Downloads/data/events_1725171.xml",

    # Optional arguments
    coordinates="INSTAT",

)
dataset.to_pandas().head()
```

	event_id	event_type	result	success	period_id	timestamp	end_timestamp	ball_state	ball_owning_team	team_id	player_id	coordinates_x	coor
0	10074	GENERIC:unknown	None	None	1	0.00	None	alive	1014	1014	26667	52.0	
1	10075	GENERIC:unknown	None	None	1	1.67	None	alive	1014	1014	6000	37.0	
2	10076	GENERIC:unknown	None	None	1	4.79	None	alive	247	247	424786	25.5	
3	12733	GENERIC:unknown	None	None	1	4.79	None	alive	247	247	424786	25.5	
4	25581	RECOVERY	None	None	1	4.79	None	alive	247	247	424786	25.5	

Fig. 1.5: Code exemple

CHAPTER

TWO

PHYSICAL REPORT

2.1 Introduction

2.2 Tracking Data

2.3 Workflow

2.3.1 Minutes played

2.3.2 Distance covered

2.3.3 Report

2.4 Result

```
x = "1 Mathew Ryan (GK)"
(?=_)
```

```
  Input In [1]
    (?=_)
     ^
SyntaxError: invalid syntax
```

CHAPTER

THREE

DATA VISUALIZATION

3.1 Introduction

In this chapter, we will visualize the data we were working on, so that we can present them with the best layout and the best way.

3.2 FrameWork

Away from the basic softwares used for data visualisation, We will stick to Python and create the dashboard from scratch using dash-plotly as a framework. We're going to build an analytical web application.

3.2.1 What is Dash ?

Dash is a python framework created by plotly for creating interactive web applications. Dash is written on the top of Flask, Plotly.js and React.js.Dash is open source and the application build using this framework are viewed on the web browser.

The Dash application is made of two building blocks

Layout : It defines the structure of the application . Elements used such as dropdowns , buttons , graphs etc and the placement , size , color etc. Dash contains Dash HTML components which we can create and style HTML content such as heading, paragraph, images etc using python . Elements such as dropdowns, graphs, cards are created using Dash Core components. Dash also includes Dash Bootstrap components that makes it easier to build consistently styled apps with complex, responsive layouts.

Callbacks : Are used to bring interactivity to the dash application. These are the functions using which, for example, we can define the activity that would happen on clicking a button or a dropdown.

3.2.2 Dash Bootstrap Components

For improved layout design, we will use Bootstrap. Dash-bootstrap-components is a library of Bootstrap components for Plotly Dash, that makes it easier to build consistently styled apps with complex, responsive layouts.

3.3 Dashboard Content

3.3.1 Match page

The landing page which contains game information such as home team and away team, name and score also two buttons to pass to display the line-up for each team and comparison of players.

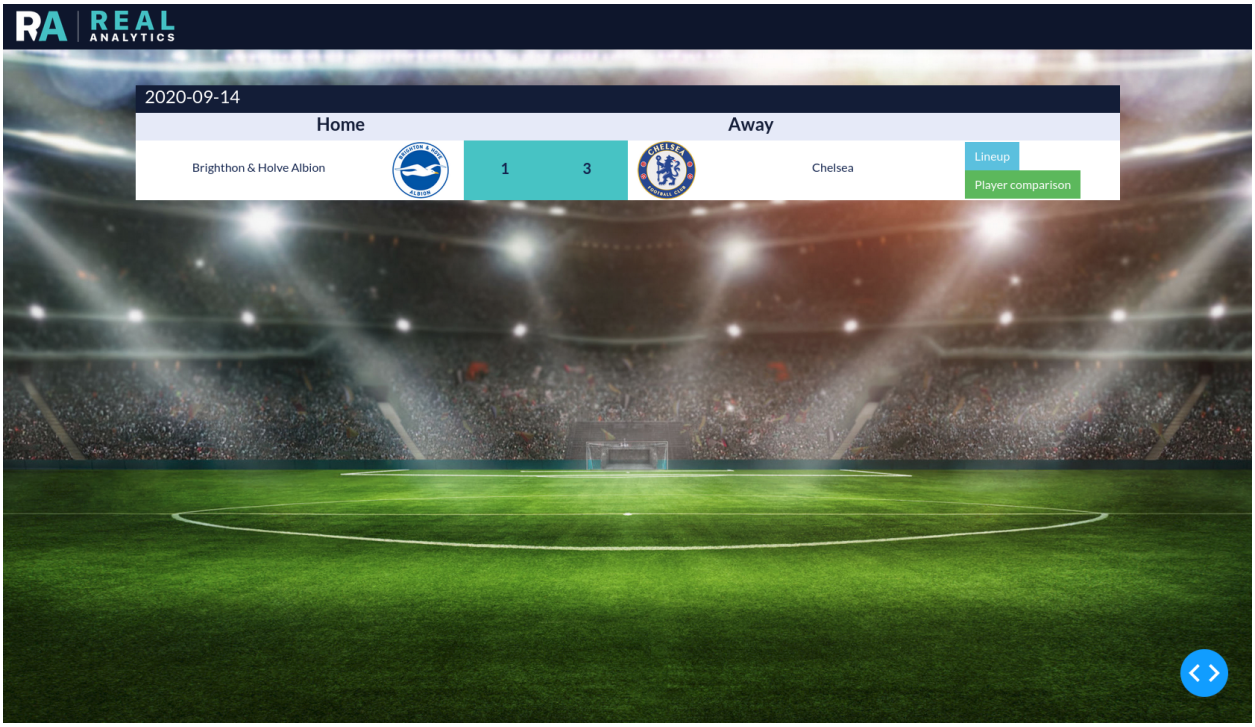

Fig. 3.1: The landing page.

Line-up : The line-up is ordered as player positions in the field.

Position	name
GK	Goal Keeper
Defenders	CF center forward / RF right forward
Midfielders	DM defensive Midfield / RM right midfield / LM left midfield
Forwards	CB center back / RCB right center back / LCB left center back

3.3.2 Player page

It is the main page on which the player performance information will be displayed.

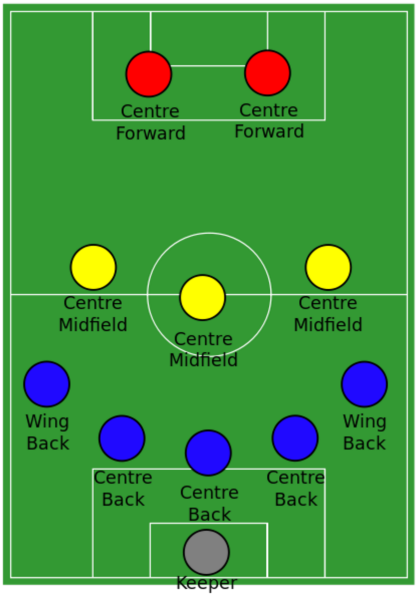

Fig. 3.2: This images illustrate the players position on the field.

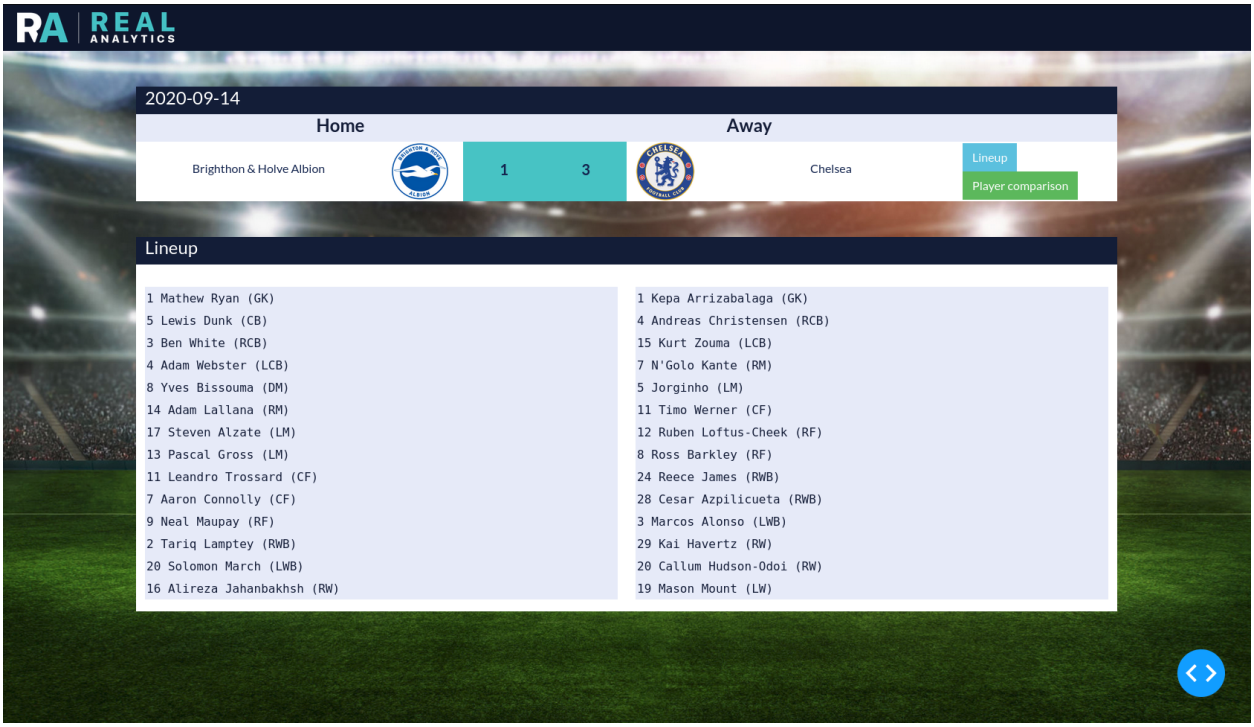

Fig. 3.3: Displayed content after clicking on line-up button.

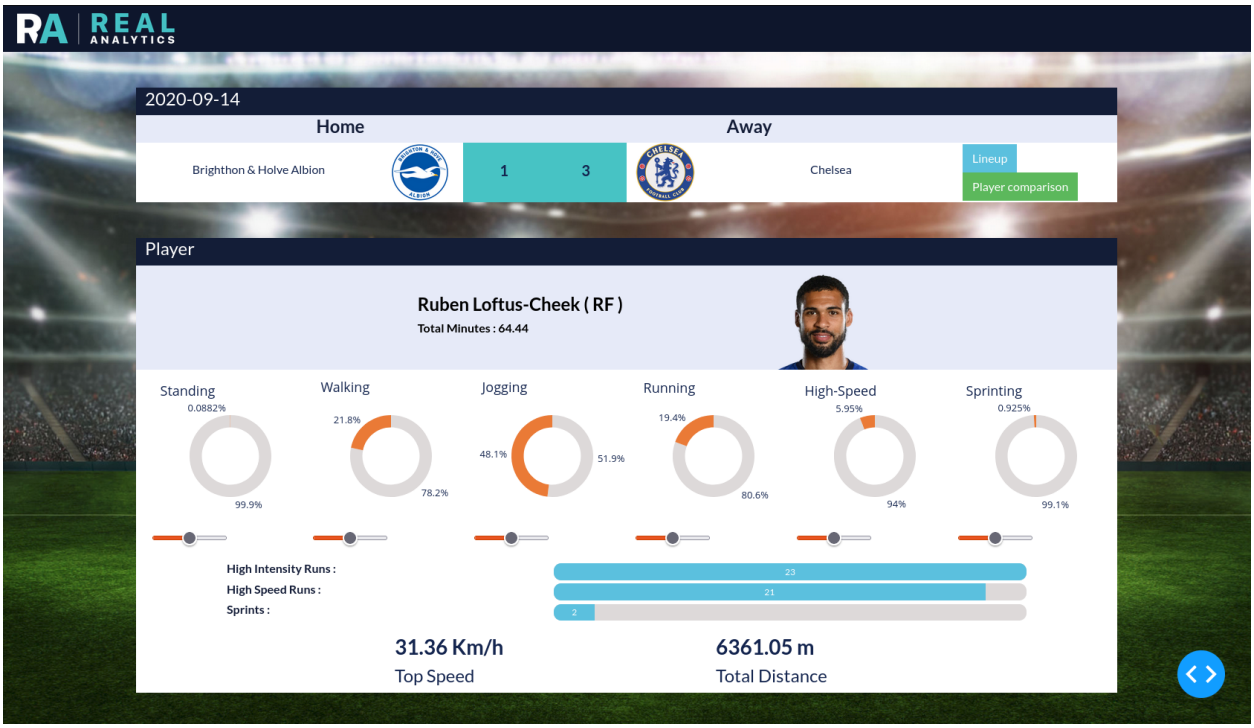

Fig. 3.4: Displayed content after clicking on a player in the line-up.

3.3.3 Player Comparison

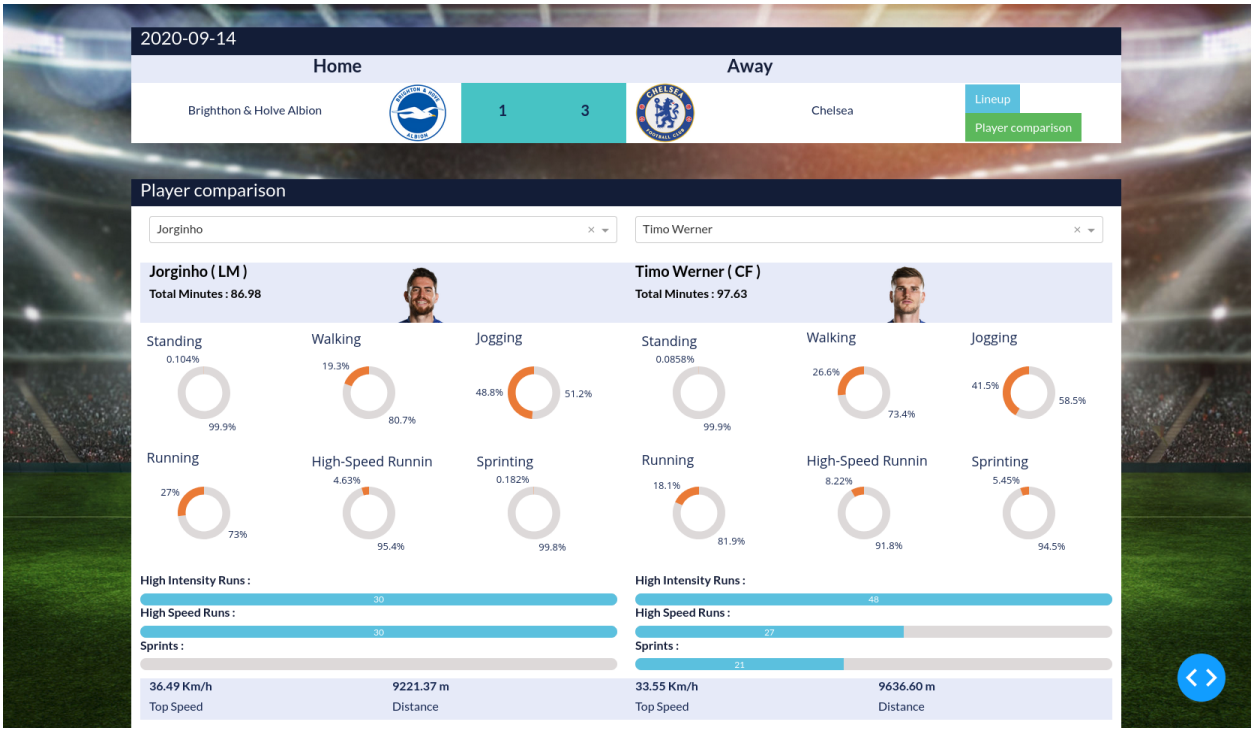

Fig. 3.5: Displayed content after clicking on the player comparison button.