«به نام پروردگار یکتا»

گزارش پروژه‌ی فوتبال فانتزی لیگ برتر / درس برنامه‌نویسی پیشرفته / استاد درس: دکتر حسین‌پور دهکردی

ارائه دهندگان : فرهاد روحبخش ۶۱۰۳۹۸۱۲۷ / پارسا بهرامی ۶۱۰۳۹۸۱۰۷ / رشته : ریاضیات و کاربردها

موضوع پروژه:

فوتبال فانتزی لیگ برتر یک بازی آنلاین سبک پیش‌بینی ورزشی است که بیش از ۷ میلیون نفر در سراسر دنیا در آن شرکت می‌کنند. معرفی مختصر بازی : شما با بودجه ۱۰۰ میلیون پوند باید ۱۵ نفر از بین بازیکنان لیگ برتر انگلیس را انتخاب کرده که شامل ۲ دروازه‌بان، ۵ مدافع، ۵ هافبک و ۳ مهاجم است. هر هفته می‌توانید ۱۱ نفر از این بازیکنان را در ترکیب اصلی خود قرار دهید و این ۱۱ نفر بر اساس عملکرد آنها در بازی آن هفته‌ی لیگ برتر انگلیس امتیاز می‌گیرند. مهم‌ترین عوامل موثر در امتیازگیری بازیکنان عبارتند از گل، پاس گل، کلین‌شیت و دقایق بازی. هدف نهایی این است که ۱۵ نفر را طوری انتخاب کنید که بیشترین امتیاز را در هفته بگیرند. از آنجایی که بازیکنان با کیفیت بیشتر قیمت بالاتری دارند بنابراین شما باید یک تعادل بین بازیکنان ارزان و گران‌قیمت در ترکیب خود ایجاد کنید و این موضوع جذابیت بازی را دوچندان می‌کند. برای مطالعه‌ی کامل قوانین بازی می‌توانید به لینک زیر مراجعه کنید:

https://fantasy.premierleague.com/help/rules

هدف اصلی ما در این پروژه بررسی اطلاعات فوتبالی بازیکنان لیگ برتر انگلیس از طریق ۲ منبع آماری است. سرور خود بازی که اطلاعات اصلی مورد نیاز را در اختیار ما قرار می‌دهد و دیگری وبسایت آمار فوتبالی fbref.com است که برای داده‌های فرعی‌تر به آن مراجعه می‌کنیم.

با ادغام اطلاعات و بررسی آماری این دو وبسایت می‌خواهیم پارامترهای اثرگذار در امتیازگیری بازیکنان را کشف کنیم و همچنین با رسم نمودارهای مختلف، بتوانیم یک درک شهودی از تاثیر پارامترها بر یکدیگر و اینکه کدام بازیکنان در کدام پارامترها بهتر عمل کرده‌اند داشته باشیم؛ و در نهایت نمودارهای رسم شده را در یک نرم‌افزار تعاملی (interactive) نمایش خواهیم داد تا بدین ترتیب بتوانیم خیل عظیم اطلاعات را منظم تر درک کنیم.

چکیده:

ابتدا از دو منبع سرور بازی و سایت fbref.com جداول اطلاعات بازیکنان را به صورت دیتافریم‌های پانداس دریافت می‌کنیم. سپس نام‌های بازیکنان در دیتافریم‌های هر دو منبع را بررسی و یکسان سازی می‌کنیم و بعد از پاک‌سازی داده‌های پرت یا بی‌ربط کل اطلاعات مورد نیاز برای یک بازیکن را با توجه به پست بازی، در ۴ دیتافریم مختلف ذخیره می‌کنیم. سپس با توجه به این ۴ دیتافریم به رسم نمودارهای مختلف از پارامترهای مختلف مانند نمودارهای دایره‌ای، ستونی، نقطه‌ای و ... می‌پردازیم. با توجه به این نمودارها روی همبستگی و تاثیر مستقیم یا غیر مستقیم یک سری پارامترهای مهم روی یکدیگر بحث می‌کنیم. در نهایت در یک محیط تعاملی نمودارهای مختلف را نمایش می‌دهیم تا قابلیت رسم نمودار بر حسب پارامترها، بازیکن‌ها، یا تیم‌هایی که کاربر انتخاب می‌کند داشته باشیم. در لینک زیر می‌توانید نمونه‌ی اجرا شده‌ی برنامه و محیط تعاملی را مشاهده کنید. (لینک با فیلترشکن باز می‌شود.)

https://share.streamlit.io/farhadrouhbakhsh/projectap/main/ww/APP.py

کتابخانه‌های مورد استفاده در این پروژه:

requests: درخواست دادن برای استفاده‌ی آنلاین و لحظه‌ای از آمار وبسایت‌ها

beautiful soup: برای خواندن اطلاعات از روی وبسایت fbref.com که به زبان html نوشته شده بود

re: برای html parsing و پیدا کردن محل سطر و ستون‌های جداول مورد نیاز در کد html

pandas: برای کار کردن با دیتافریم‌های استخراج شده از ۲ منبع ذکر شده و ادغام آنها

plotly: برای رسم نمودارهای تعاملی، به طور مثال نمودارهایی که صرفا یک عکس ثابت نیستند و کاربر با بردن نشانگر موس یا کلیک روی بخش‌های مختلف نمودار اطلاعات مختلفی را مشاهده می‌کند.

plotly_express: ماژولی که از آن برای رسم نمودارهای plotly با کدهای کوتاه‌تر و قابل‌فهم تر نسبت به خود plotly استفاده کردیم تا بتوانیم نمودارها را در زمان کم با کارایی زیاد رسم کنیم.

streamlit: برای ایجاد کردن یک محیط تعاملی data science تحت وب

مقدمه:

خواندن اطلاعات از سایت fbref.com:

از آنجایی که جداول این سایت به صورت محافظت شده و بین علامت‌های خاص قرار گرفته بود، ما در قدم اول نتوانستیم صرفا با استفاده از request به جداول دسترسی داشته باشیم. بنابراین، از آنجایی که به زبان html و همچنین web-scrapping آشنا نبودیم، با سرچ در اینترنت از کد مقاله‌ای با لینک زیر کمک گرفتیم:

https://chmartin.github.io/2019/02/18/EPL-History-Scraping.html

همچنین بخش‌هایی از کد که مربوط به تبدیل string یک خانه از جدول به int یا float است را تغییر دادیم. ورودی تابع urlی است که می‌خواهیم جدول را از آن بخوانیم و columns لیستی از ستون‌های مورد نیاز در آن جدول است که در واقع اطلاعات این ستون‌ها را از جدول می‌خواهیم. خروجی تابع جدول خواسته شده به صورت دیتافریم است.

ابتدا آن بخش از کد html که کامنت هست را حذف می‌کنیم. سپس کد جدول‌ها را با یافتنtbody در کد اصلی مشخص می‌کنیم. هر سطر جدول با tr شروع می‌شود و هر ستون در آن سطر با th. همچنین از row scope می‌توان برای اطمینان از پیدا کردن درست سطرها استفاده کنیم. اطلاعات یک سلول از جدول را با یافتن data-stat در کد html می‌توان مشخص کرد. حالا می‌توانیم روی هر سطر و ستون for بزنیم.

یک سلول جدول را در متغیر text می‌ریزیم و اگر عدد بود فرمتش را به int یا float تغییر می‌دهیم. بعد از مشخص شدن text آن را برابر value یک key از دیکشنری قرار می‌دهیم که key همان ستون مورد نظر است. اگر از قبل این key وجود نداشت آن را ایجاد می‌کنیم.

```
def read_fbref(url,columns):
    res=requests.get(url)
    comm=re.compile("<!--|-->")
    soup = BeautifulSoup(comm.sub("",res.text),'lxml')#remove comments in htm
l code
    all_tables=soup.findAll("tbody")#find tables
    player_table=all_tables[1] #player data is in second table
    pre_df_player = dict()#dict for saving data
    rows_player = player_table.find_all('tr') #find rows
    for row in rows_player:
        if(row.find('th',{"scope":"row"}) != None):#check if row is found cor
rectly
            for cl in columns:
```

```
                cell = row.find("td",{"data-stat": cl})
                a = cell.text.strip().encode()
                text=a.decode("utf-8")#text in string format
                if text.isdigit():
                    text=int(text)
                elif '.' in text:
                    text=float(text)
                elif text!='' and type(text)==str:
                    if text[1]==',':# if text was 1,234 convert it to 1234
                        text=int(text.replace(',',''))
                if cl in pre_df_player:
                    pre_df_player[cl].append(text)
                else:
                    pre_df_player[cl] = [text]
    df_player = pd.DataFrame.from_dict(pre_df_player)#dict to dataframe
    return df_player[columns]
```

خواندن اطلاعات از سرور فانتزی:

ابتدا یک request به سرور بازی ارسال کرده و سپس اطلاعات را به صورت json ذخیره می‌کنیم. از این json ذخیره شده اطلاعات بازیکن‌ها را در elements_df و اطلاعات تیم‌ها را در teams_df به صورت دیتافریم ذخیره می‌کنیم.

```
import requests
import pandas as pd
url = 'https://fantasy.premierleague.com/api/bootstrap-static/'
r = requests.get(url)
json = r.json()

elements_df = pd.DataFrame(json['elements'])
teams_df = pd.DataFrame(json['teams'])
```

حذف بازیکنانی که اصلا بازی نکرده اند:

```
def del_null_pl (df) #remove players who haven't played yet
    for row in df.index:
        if df['minutes'][row]==0: #find the player
            df.drop(index=row,inplace=True) #remove the player
    df.reset_index(inplace=True,drop=True)
```

توضیح بقیه‌ی بخش‌های کد:

بخش اول: کار با دیتافریم‌ها

دیتافریم players را به صورت players=del_null_pl(elements_df) تعریف می‌کنیم

حذف بازیکنان تکراری:

ابتدا دیتافریم گرفته شده از fbref را بر اساس ستون نام بازیکن مرتب کرده و سپس اگر دو نام یکسان وجود داشت، سطر بازیکنی که دقایق کمتری بازی کرده را حذف می‌کنیم. به این دلیل که دقایق بازی کمتر نشان دهنده‌ی تیم قبلی بازیکن است و ما نیازی به اطلاعات تیم قبلی بازیکن نداریم.

```
def del_id_pl(df):#removing identical players from dataframes
    df.sort_values('player',inplace=True)
    df.reset_index(inplace=True,drop=True)
    for row in df.index:
        if row-1 in df.index and df['player'][row]==df['player'][row-1]:
                if df['minutes_90s'][row]>df['minutes_90s'][row-1]:#checking
minutes per 90
                    df.drop(index=row-1,inplace=True)
                else:
                    df.drop(index=row,inplace=True)
    df.reset_index(inplace=True,drop=True)
```

تصحیح نام تیم‌ها:

به علت تفاوت نام تیم‌ها در fbref با سرور بازی، نام تیم‌ها در teams_df را با نام آنها در fbref منطبق می‌کنیم.

```
def correct_team_names(teams_df):
    for row in teams_df.index: #changing team names in fantasy stats to use
in fbref
        team_name=teams_df['name'][row]
        if team_name=='Leicester':
            teams_df['name'][row]='Leicester City'
        if team_name=='Man City':
            teams_df['name'][row]='Manchester City'
        if team_name=='Man Utd':
            teams_df['name'][row]='Manchester Utd'
```

```
        if team_name=='Newcastle':
            teams_df['name'][row]='Newcastle Utd'
        if team_name=='Spurs':
            teams_df['name'][row]='Tottenham'
        if team_name=='Leeds':
            teams_df['name'][row]='Leeds United'
```

اضافه کردن نام تیم‌ها به players:

کار این تابع اضافه کردن نام تیم به players از teams_df بر اساس id تیم است. دلیل این کار این است که در players ستون نام تیم را نداریم.

```
def add_team_name(df):#adding team name
    df['team_name']=''
    for row in df.index:
        team_id=df['team'][row]
        team_row=teams_df.loc[teams_df['id']==team_id] #find team row by id
        team_row.index=[0]
        team_name=team_row['name'][0]
        df['team_name'][row]=team_name #set team name
```

تصحیح پست بازی:

به علت تفاوت نام‌گذاری پست بازیکنان در fbref با players ، پست آنها را با fbref منطبق می‌کنیم.

```
def update_position(df):#Updating position slot
    for row in df.index:
        pos=df['element_type'][row]
        if pos==1:
            df['element_type'][row]='GK'
        if pos==2:
            df['element_type'][row]='DF'
        if pos==3:
            df['element_type'][row]='MF'
        if pos==4:
            df['element_type'][row]='FW'
```

اضافه کردن نام کامل بازیکن:

اضافه کردن یک ستون به نام name به players زیرا در players صرفا ستون‌های first_name ، second_name و web_name موجود بود که آخری مربوط به نام نمایش داده‌شده‌ی بازیکن در وبسایت

بازی است که معمولا مشابه نام خانوادگی بازیکن است. ستون name را به این صورت تعریف می‌کنیم که نام و نام خانوادگی بازیکن را کنار هم در یک رشته داشته باشیم.

```
def add_full_name(df):#add full name of players
    df['name']=''
    for row in df.index:
        first_name=df['first_name'][row]
        web_name=df['web_name'][row]
        if first_name in web_name: #check if web name is enough
            name=web_name
        else:
            name=first_name+" "+web_name #if not enough, make a new name!
        df['name'][row]=name
```

یکسان سازی اسم‌ها از دو منبع:

ورودی اول دیتافریم fbref و ورودی دوم players. ابتدا روی این دو ورودی بر حسب نام تیم groupby می‌زنیم. سپس با اجرای یک حلقه روی تیم‌ها، به صورت تک تک و تیم به تیم، نام بازیکنان هر تیم در دو دیتافریم را مقایسه می‌کنیم. (با این روش دیگر نیازی نخواهیم داشت که هر بازیکن را با تمام دیگر بازیکنان مقایسه کنیم. بلکه او را فقط با هم‌تیمی‌های خودش مقایسه خواهیم کرد).

مقایسه اسامی از ۲ منبع به این شکل است که اگر نام بازیکن در دیتافریم fbref با نام او در players دقیقا یکی شد، id بازیکن در players را در دیتافریم fbref قرار می‌دهد(این id مشترک باعث می‌شود که جلوتر بتوانیم اطلاعات بازیکنان در هر دو دیتافریم را ادغام کنیم). اگر مقایسه با موفقیت انجام شد بولین checked برابر True خواهد شد. اگر هنوز مقایسه انجام نشده بود و نام بازیکن پیدا نشد، نتیجه می‌گیریم نام بازیکن در players با نام او در fbref دقیقا یکی نیست. بنابراین مقایسه را دقیق‌تر می‌کنیم. حالت اول این است که web_name یا second_name در نام بازیکن در fbref موجود باشد. اگر نبود، first_name را بررسی می‌کنیم به همراه پست بازیکن تا ببینیم نام‌ها یکسان هستند یا خیر. با این روش نهایتا همه‌ی بازیکنان در دو منبع id یکسان خواهند داشت.

```
def find_names(df,players):
    df['id']=0
    df_groups=df.groupby('squad')
    pl_groups=players.groupby('team_name') #group by team name
    for team in teams_list:
        fbref=df_groups.get_group(team)
```

```
        fantasy=pl_groups.get_group(team)
        for row in fantasy.index: #iterate on players of a specific team
            checked=False
            player_id=fantasy['id'][row]
            team_name=fantasy['team_name'][row]
            name=fantasy['name'][row]
            first_name=fantasy['first_name'][row]
            second_name=fantasy['second_name'][row]
            web_name=fantasy['web_name'][row]
            pos=fantasy['element_type'][row]
            for s_row in fbref.index:
                fb_name=fbref['player'][s_row]
                fb_team=fbref['squad'][s_row]
                fb_pos=fbref['position'][s_row]
                if name==fb_name or first_name+' '+second_name==fb_name:
#find names which are totally equal
                    checked=True
                    df.loc[df['player']==fb_name,'id']=player_id # set id
                    fantasy.drop(index=row,inplace=True)

 #remove checked player
                    fbref.drop(index=s_row,inplace=True)
                    break
            if checked==False: #find names which are not totally equal
                for s_row in fbref.index:
                    fb_name=fbref['player'][s_row]
                    fb_team=fbref['squad'][s_row]
                    fb_pos=fbref['position'][s_row]
                    if web_name in fb_name or second_name in fb_name:

#check other conditions
                        df.loc[df['player']==fb_name,'id']=player_id
                        checked=True
                    elif (first_name in fb_name or fb_name in first_name) and
pos in fb_pos:
                        df.loc[df['player']==fb_name,'id']=player_id
                        checked=True
```

حال روی دیتافریم players، groupby می‌زنیم و ۴ تا دیتافریم برای تقسیم‌بندی بازیکنان بر اساس پست دریافت می‌کنیم.(به این دلیل که قوانین امتیازگیری برای هر پست در فانتزی متفاوت است.)

```
posts = players.groupby('element_type') #group by position(element_type)
```

```
GK = posts.get_group('GK')
DF = posts.get_group('DF')
MF = posts.get_group('MF')
FW = posts.get_group('FW')
```

اضافه کردن اطلاعات تیم‌ها به ۴ پست:

برای اضافه کردن ستون‌های دلخواه از دیتافریم teams_df به هر کدام از ۴ پست با هدف اینکه اطلاعات بازیکنان در هر پست را با اطلاعات تیمشان ادغام کنیم.

```
def add_cl_teams(post,df,cl): #add columns from teams_df to each post
    post[cl] = 0
    for row in post.index:
        team_id = post['team'][row]
        team_row = df.loc[df['id'] == team_id] #check team id
        post[cl][row] = team_row[cl] #set data
```

ادغام کردن اطلاعات fbref با ۴ پست:

برای اضافه کردن لیستی از ستون‌های دلخواه (cl_list) از دیتافریم fbref به هر کدام از ۴ پست. در اینجا id داده شده در دیتافریم fbref را با id بازیکن در ۴ دیتافریم پست مطابقت می‌دهیم و سپس اطلاعات ستون مورد نیاز آن بازیکن در fbref را به سطر مربوط به آن بازیکن در دیتافریم مربوط به آن پست اضافه می‌کنیم.

```
def add_cl_fbref(post,fbref,cl_list): #add columns from fbref to each post
    for cl in cl_list:
        post[cl]=0.0
    for row in post.index:
        for cl in cl_list:
            player_id=post['id'][row]
            fb_row=fbref.loc[fbref['id']==player_id] #check player id
            fb_row.index=[0]
            post[cl][row]=fb_row[cl] #set data
```

بخش دوم: رسم نمودارها و بررسی آماری

نمودار دایره‌ای (Piechart):

هدف اصلی : مقایسه متغیرهایی که تاثیر مستقیم بر امتیاز گیری بازیکنان دارند.

(دقت کنید که با توجه به قوانین بازی، امتیاز گیری در هر پست به متغیر های متفاوتی بستگی دارد.)

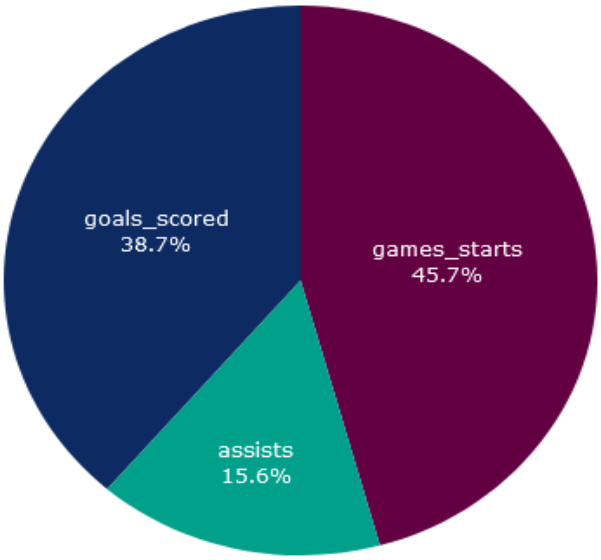
در اینجا به طور مثال برای پست مهاجم piechart را رسم می‌کنیم. ابتدا متغیر value را به عنوان امتیاز کل تقسیم بر قیمت تعریف کرده و بازیکنان را بر حسب value مرتب می‌کنیم. تعدادی از نفرات برتر را بر حسب پست جدا می‌کنیم. سپس لیست پارامترهایی با تاثیر مستقیم بر امتیاز در آن پست را ایجاد و ارزش هر کدام از آنها را برابر ضرب میانگین آن پارامتر بین نفرات برتر در میزان امتیاز آن بر حسب قانون بازی قرار می‌دهیم.

```
import streamlit as st

import pandas as pd
import plotly.express as px

#Define value & sort by it
FW['value'] = FW['total_points'] / FW['now_cost']
FW.sort_values(by='value',ascending=False,inplace=True)
best_fw=FW.head(20)

#Define parameters & values of them
para_fw = ['goals_scored','assists','games_starts']
val_fw = [best_fw['goals_scored'].mean()*4,best_fw['assists'].mean()*3
        ,best_fw['games_starts'].mean()*2]#Parameter.mean()*regression
#Draw piechart with plotly express
fig_fw = px.pie(values=val_fw,names=para_fw,title='Forwards parameters',label
s=para_fw
        ,color=para_fw
       ,color_discrete_map={'goals_scored':'0d2a63',
            'assists':'00a08b','games_starts':'620042'})
fig_fw.update_traces(textposition='inside', textinfo='percent+label')
```



نتیجه می‌گیریم در پست مهاجم گل زدن و حضور داشتن در زمین عوامل بسیار مهمی هستند.

نمودار دایره‌ای چند تکه (sunburst):

هدف اصلی : بررسی متغیر های با تاثیر غیر مستقیم بر امتیاز گیری و ارتباط آن ها با متغیر های اصلی برای هر پست.

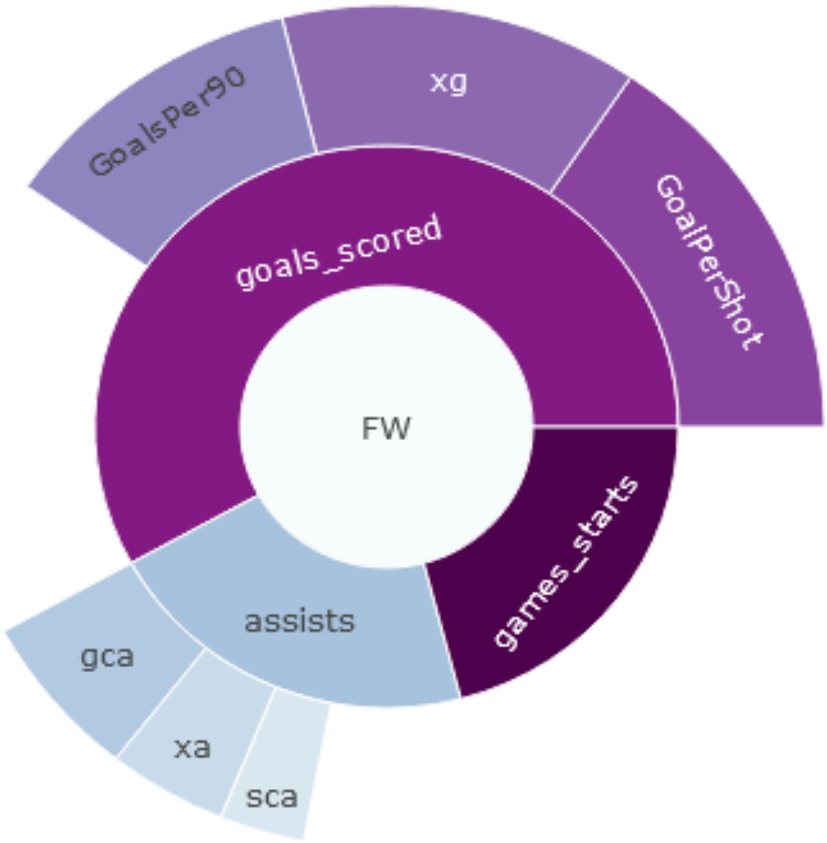
ابتدا یک لیست برای تمام متغیر ها درست کرده سپس دو لیست دیگر یکی برای والد های لیست اول (متغیر های با تاثیر مستقیم) و لیست دوم را برای ارزش های هر کدام از متغیر ها درست می کنیم. مقادیر این لیست تعیین می‌کنند یک بخش از نمودار چه سهمی از یک دایره‌ی کامل دارد. در لیست دوم میانگین نفرات برتر در پارامترهای والد را در ضریب رگرسیون پارامتر والد نسبت به پارامترهای فرعی مربوط به آن ضرب می‌کنیم. (این ضریب رگرسیون را جلوتر در نمودار نقطه‌ای به دست می‌آوریم.) هر چقدر عدد بزرگتری به دست بیاید نشان‌دهنده‌ی همبستگی و اهمیت بیشتر آن پارامتر فرعی است و همچنین بخش بزرگتری از دایره را اشغال می‌کند.

```
#Define parameters
para_adv_lfw=["FW",'goals_scored','assists','games_starts',
                "xg","GoalPer90","GoalPerShot",
                "xa","gca","sca"]

#Define parents (the most influential parameters)
para_adv_pfw=["","FW","FW","FW",'goals_scored','goals_scored','goals_scored',
'assists','assists','assists']

#Define values based on the relation between parent and child
val_adv_fw=[0,best_fw['goals_scored'].mean()*4,best_fw['assists'].mean()*3
        ,best_fw['games_starts'].mean()*2
        ,best_fw['goals_scored'].mean()*4*0.75#ParentInfluence.mean()*regress
ion
        ,best_fw['goals_scored'].mean()*4*0.65
        ,best_fw['goals_scored'].mean()*4*0.88
        ,best_fw['assists'].mean()*3*0.62
        ,best_fw['assists'].mean()*3*0.89
        ,best_fw['assists'].mean()*3*0.44]

#Draw sunburst with px
fig_adv_fw = px.sunburst(names=para_adv_lfw,parents=para_adv_pfw, values=val_
adv_fw
                ,color=val_adv_fw
                ,color_continuous_scale= px.colors.sequential.BuPu
                ,title="Forwards Sunburst chart"
                ,height=600#size chart
                ,width=600)
```


GoalsPer90
xg
GoalPerShot
goals_scored
FW
games_starts
assists
gca
xa
sca


xg
GoalPerShot
goals_scored
GoalsPer90

نمودار چندضلعی (radar chart):

هدف اصلی : مقایسه چند متغیر همزمان برای دو بازیکن مشخص بر حسب پست.

ابتدا پست و ۲ بازیکن مورد نظر را انتخاب می کنیم . سپس متغیر های مناسب بر حسب انتخاب پست در نظر گرفته می‌شوند. در نهایت این متغیر ها را برحسب مقداری که برای ۲ بازیکن انتخاب شده داشتند در یک لیست ذخیره می‌کنیم که در واقع مقادیر نشان داده شده روی نمودار هستند. دقت کنید مقدار خانه‌ی اول این لیست باید با خانه‌ی آخر یکی باشد تا سر و ته خطوط نمودار به هم برسند. نکته دیگری که وجود دارد این است که این نمودار در واقع یک نمودار نقطه‌ای است که با چرخش محور x ها به صورت دایره‌ای در آمده است. در انتها برای زیبا شدن نمودار شعاع دایره برابر حداکثر مقدار پارامتر های ۲ بازیکن قرار داده شده است تا تناسب رعایت شود و مقادیر رسم شده روی نمودار خیلی کوچک به نظر نرسند.

```
import plotly.graph_objects as go

def posts_eval(post):
     var=globals()[post]#convert "post" to post
     return var
GK['save_pct']=GK['save_pct']*100
def app():
    post=st.selectbox(
    'Select Post!',['GK','DF','MF','FW'])
    df=posts_eval(post)
    #Select players by name
    option1 = st.selectbox(
    'Select player 1 for comparison!',
    df['name'].unique())
    option2= st.selectbox(
    'Select player 2 for comparison!',
    df['name'].unique())
    if option1 and option2:
          pl1=df[df['name']==option1]#find the player row in dataframe
          pl2=df[df['name']==option2]
          pl1.set_index('name',inplace=True)
          pl2.set_index('name',inplace=True)
          if post=='FW' or post=='MF':
                elements=['goals_scored','assists','xg','xa','npxg'] # elemen
ts of comparison
          elif post=='GK':
                elements=['clean_sheets_pct','saves','goals_conceded','save_p
ct','total_points']
          elif post=='DF':
                elements=['assists','clean_sheets','bonus','shots_on_target',
'gca']
```

```
            pl1=pl1[elements]
            pl2=pl2[elements]
            row1=list(pl1.loc[option1]) #create a list of parameters in compari
son for 2 players
            row2=list(pl2.loc[option2])
            row1.append(row1[0]) # last value = first value
            row2.append(row2[0])
            elements.append(elements[0])
            fig=go.Figure()
             #radar chart player1
            fig.add_trace(go.Scatterpolar(
                 r=row1,
               theta=elements,
               name=option1
            ))
           #radar chart player2
            fig.add_trace(go.Scatterpolar(
                r=row2,
                theta=elements,
                name=option2
            ))
            #update range from 0 to max
            fig.update_layout(polar=dict( radialaxis=dict( visible=True, range=
[0,max(row1+row2)] )),showlegend=True )
            st.write(fig)
```

Select Post!

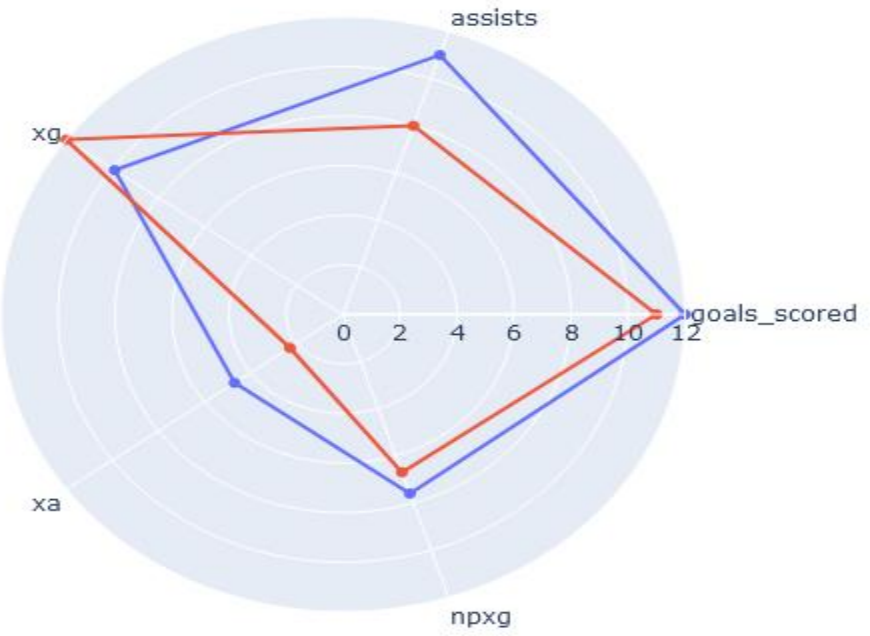
FW

Select player 1 for comparison!

Harry Kane

Select player 2 for comparison!

Jamie Vardy



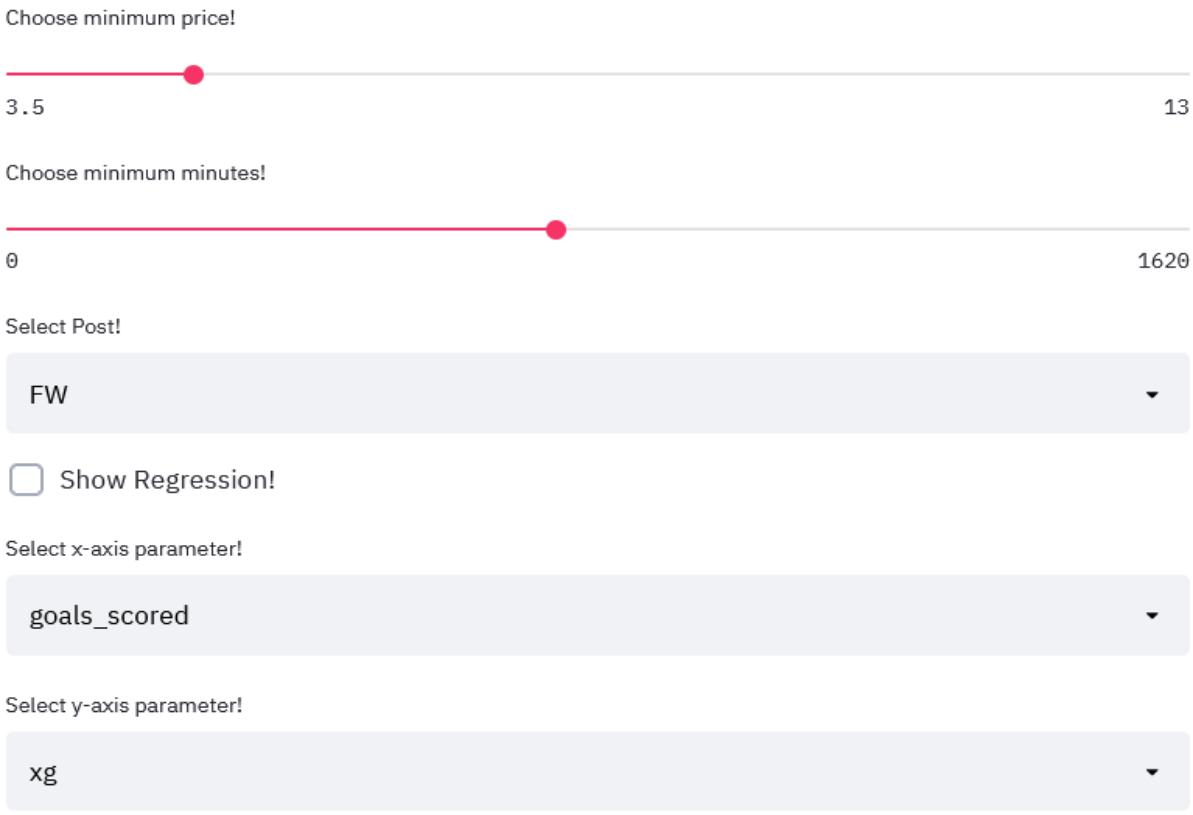
نمودار نقطه‌ای (scatterplot):

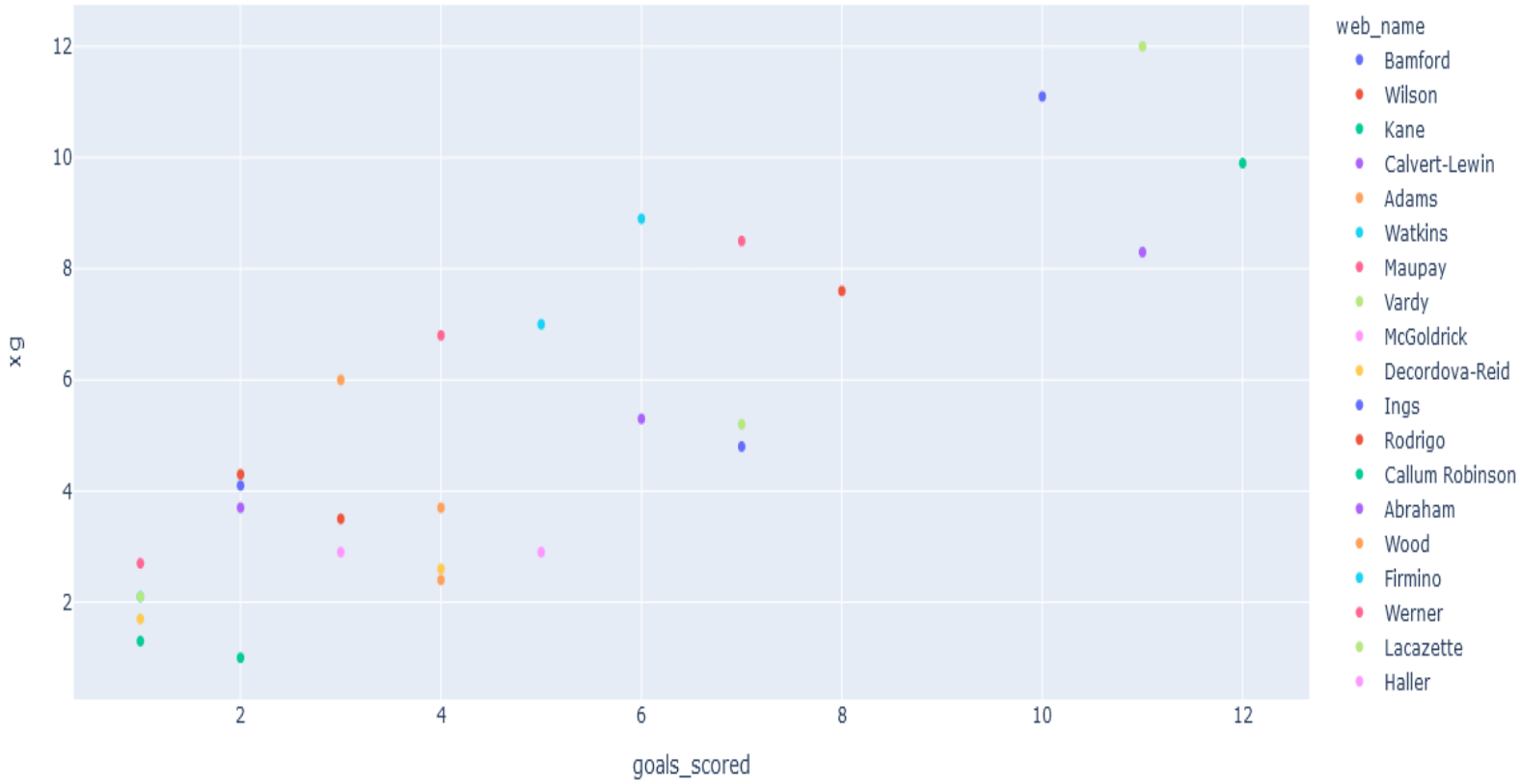
هدف اصلی : بررسی ارتباط بین متغیر ها و نحوه تغییر ارتباط انها با تغییر بازه قیمت و دقایق بازی برای هر پست.

ابتدا دو slider یکی برای حد قیمت و یکی برای دقایق بازی طراحی می کنیم و یک منو برای انتخاب پست و یک چک‌باکس برای انتخاب حالت رسم خط regression می گذاریم. (دقت کنید با رسم خط regression اگر نشانگر موس را روی آن خط ببرید، ضریب رگرسیون نمایش داده می‌شود. چنان که اشاره کردیم از این قابلیت در نمودار sunburst استفاده کرده‌ایم.) همچنین دو منوی دیگر هم برای انتخاب متغیر هایی که میخواهیم مورد بررسی قرار بدهیم وجود دارد. حال بر اساس شرایط تعریف شده و با تغییر هر کدام از آنها، قسمت های بالا جدول ما تغییر و بر حسب تغییرات نمودار های جدید با استفاده از Plotly express رسم شده و توسط stremlit نمایش داده می‌شود.

```
#define price & minutes sliders
price1=st.slider(label='Choose minimum
price!',min_value=3.5,max_value=13.0,step=0.1,format='%s')
```

```
mins=st.slider(label='Choose minimum
minutes!',min_value=0,max_value=2000,step=1)
string_list=['web_name','first_name','second_name','team_code','status','name
','team_name','element_type']
post = st.selectbox('Select Post!',['GK','DF','MF','FW'])
form=st.checkbox('Show Regression!')#checkbox to show regression
df=posts_eval(post)
new_df=df[(df['now_cost']>=price1*10) & (df['minutes']>=mins)]#change
dataframe by sliders
menu=[x for x in df.columns if x not in string_list]#remove columns which are
not numeric
#select parameters to compare
xa=st.selectbox('Select x-axis parameter!',menu)
ya=st.selectbox('Select y-axis parameter!',menu)
if len(new_df)==0:#if there were no players to show
     fig=px.scatter(new_df,x=xa,y=ya)
else:
    if form:# regression mode
        fig=px.scatter(new_df,x=xa,y=ya,trendline='ols')
    else:
        fig=px.scatter(new_df,x=xa,y=ya,color='web_name')
        st.write(fig)#draw scatterplot by conditions
```





نمودار ستونی مستطیلی (bar chart):

هدف اصلی : مقایسه بازیکنان یک تیم بر حسب یک متغیر.(انتخاب بازیکن از هر تیم محدود است)

دو منوی انتخابی یکی برای انتخاب تیم و یکی برای انتخاب متغیر انتخاب می کنیم. بعد جدول بازیکنان تیم انتخاب شده را گرفته و بر حسب متغیر انتخابی رده بندی کرده. سپس با استفاده از plotly express این نمودار را کشیده و با کمک ماژول plotly graph object زنجیر بین هر ستون را اضافه می‌کنیم(که در واقع خودش یک نوع نمودار نقطه‌ای است) و در نهایت توسط streamlit نمایش می‌دهیم.

```
import plotly.express as px

import plotly.graph_objects as go

import streamlit as st

string_list=['web_name','first_name','second_name','team_code','status','name
','team_name','element_type']
menu=[x for x in all_posts.columns if x not in string_list]#remove columns wi
th string type
```

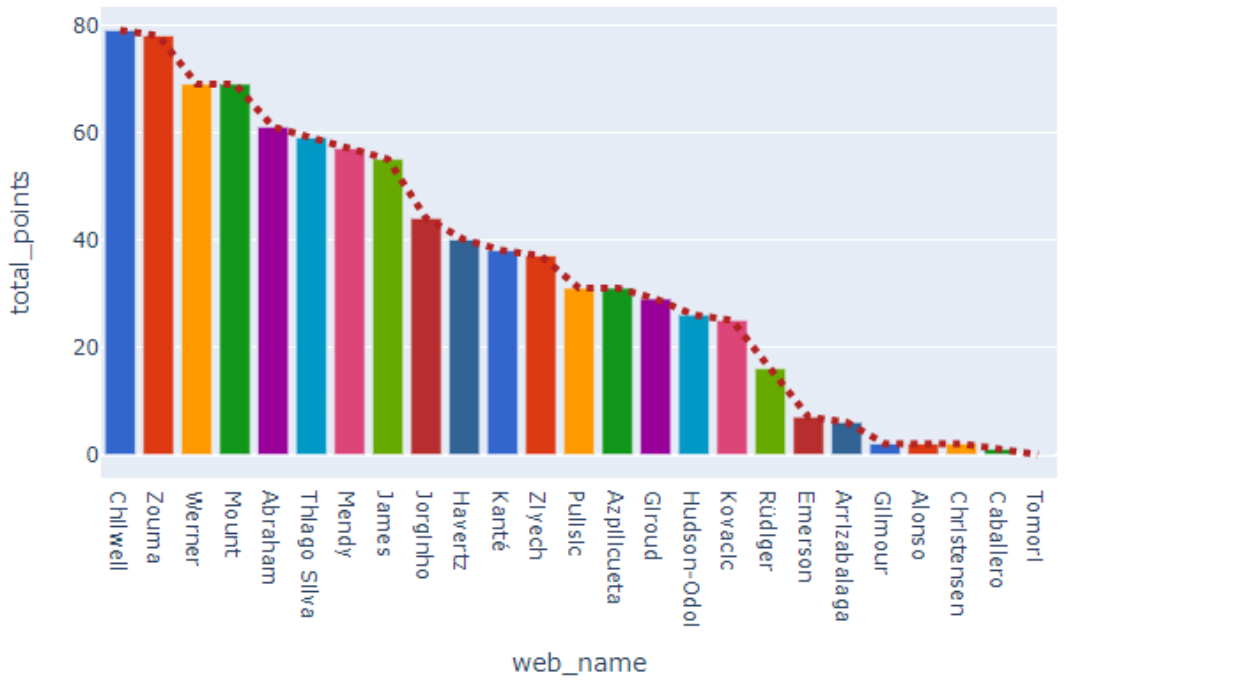
```
#choose team & parameter
team=st.selectbox('select team!',all_posts['team_name'].unique())
param=st.selectbox('select parameter!',menu)
new_df=all_posts[all_posts['team_name']==team]
new_df[param]=pd.to_numeric(new_df[param])#convert string to int,float
new_df.sort_values(param,inplace=True,ascending=False)
#draw Barchart
fig=px.bar(new_df,x='web_name',y=param,color='web_name',color_discrete_sequen
ce=px.colors.qualitative.G10)
fig.add_trace(go.Scatter(x=new_df['web_name'], y=new_df[param],name='',
                         line = dict(color='firebrick', width=4, dash='dot
')))#draw chain of barchart
fig.update_layout(showlegend=False)
st.write(fig)
```

select team!

Chelsea

select parameter!

total_points



صفحه رده‌بندی بازیکنان:

هدف اصلی :رسیدن به لیستی از بهترین بازیکنان برای هر پست.

ابتدا با استفاده از الگوریتمی که بر اساس ضرایب امتیازگیری بازیکنان در هر پست طراحی شده، بهترین بازیکنان را جدا می کنیم و متغیر rank را برای هر پست تعریف می کنیم .

سپس ۱۰ بازیکن برتر هر پست را بر حسب rank مشخص کرده و با استفاده از دستورات plotly figure factory جدول ها را ترسیم می‌کنیم.

```
import plotly.figure_factory as go

gk=GK[GK['minutes']>700]
df=DF[DF['minutes']>700]
fw=FW[FW['minutes']>700]
mf=MF[MF['minutes']>700]
gk['rank']=0.04*gk['clean_sheets_pct']+(1/3)*(gk['saves']/gk['games_starts_gk
'])-(1/2)*(gk['goals_conceded']/gk['games_starts_gk'])
mf['rank']=5*(mf['goals_scored']/mf['games_starts'])+3*(mf['assists']/mf['gam
es_starts'])
df['rank']=4*(df['clean_sheets']/df['games_starts'])+6*(df['goals_scored']/df
['games_starts'])+3*(df['assists']/df['games_starts'])-(1/2)*(df['goals_conce
ded']/df['games_starts'])
fw['rank']=4*(fw['goals_scored']/fw['games_starts'])+3*(fw['assists']/fw['gam
es_starts'])

#Sort posts by rank & give best ten & draw table of the best
bestGK=gk[['web_name','rank']].sort_values('rank',ascending=False).head(10)
del bestGK['rank'] #del rank because we just want names
bestGK=ff.create_table(bestGK)
bestDF=df[['web_name','rank']].sort_values('rank',ascending=False).head(10)
del bestDF['rank']
bestDF=ff.create_table(bestDF)
bestMF=mf[['web_name','rank']].sort_values('rank',ascending=False).head(10)
del bestMF['rank']
bestMF=ff.create_table(bestMF)
bestFW=fw[['web_name','rank']].sort_values('rank',ascending=False).head(10)
del bestFW['rank']
bestFW=ff.create_table(bestFW)
```

Best Defenders

web_name
Stones
Chilwell
Zouma
Vestergaard
Mings
Dias
Cancelo
Walker
Thiago Silva
Aurier

Best Forwards

web_name
Vardy
Kane
Abraham
Calvert-Lewin
Ings
Bamford
Wilson
Watkins
Lacazette
Maupay

Best Goalkeepers

web_name
Martínez
Ederson
Pope
Mendy
McCarthy
Lloris
Fabianski
Meslier
Leno
Schmeichel

Best Midfielders

web_name
Salah
Fernandes
Son
Grealish
Maddison
Zaha
De Bruyne
Rashford
Sterling
Mané