# import sklearn neural network and statistics

import os

import statistics

import scipy as sp

import math

import pandas as pd

import numpy as np

import matplotlib as mpl

import matplotlib.pyplot as plt

import matplotlib.ticker as tick

import seaborn as sns

import statsmodels.api as sm

import statsmodels.formula.api as smf

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

from statsmodels.tools.tools import add\_constant

from statsmodels.regression.linear\_model import OLS

from statsmodels.stats.outliers\_influence import OLSInfluence

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.base import BaseEstimator, RegressorMixin

#importing data

os.chdir('c:\cs project\data')

data = pd.read\_csv('consolidated\_data\_2021.csv',sep=';',engine='python')

data05 =  pd.read\_csv('consolidated\_data\_2019.csv',sep=';',engine='python')

data1 = pd.read\_csv('consolidated\_data\_2020.csv',sep=';',engine='python')

data1=pd.DataFrame.append(data,data1)

data1=pd.DataFrame.append(data1,data05,ignore\_index=True)

data1.sort\_values('player')

#adding dummy variables to dataset and cleaning data names and variables to dataset

data1 = pd.get\_dummies(data1, columns=['league'])

data1 = data1.rename({"league\_Bundesliga":"isBundesliga",

                                "league\_La Liga":"isLaLiga",

                                "league\_Premier League":"isPremierLeague",

                                "league\_Ligue 1":"isLigue1",

                                "league\_Serie A":"isSerieA"},axis='columns')

data1=pd.get\_dummies(data1,columns=['Season'])

data1=pd.get\_dummies(data1,columns=['foot'])

#deleting potential outliers

data1=data1[data1['value']>1000000]

data1=data1[data1['games']>5]

data1=data1[data1['age']>0]

data1=data1[data1['height']>0]

data1

#MIDFIELDERS

dataMID = data1[data1['position2'].str[:10]=='Midfielder']

dataMID1 = data1[data1['position2'].str[:8]=='midfield']

dataMID=pd.DataFrame.append(dataMID,dataMID1)

# log returns

def ln(x):

    return np.log(x)

#Creating a training set on forward attributes by splitting into test & training set of 0.8

trainMID, testMID = train\_test\_split(dataMID, train\_size=0.8)

modelMID1=smf.ols('ln(value)~wins\_gk+clean\_sheets+Pts+W+GDiff+clean\_sheets\_pct+CL+xGDiff+GF+xG+passes\_ground+passes\_completed\_medium+passes\_medium+games+games\_starts+minutes\_90s+minutes+games\_gk+games\_starts\_gk+minutes\_90s\_gk+minutes\_gk+passes\_throws\_gk+passes\_other\_body+passes\_completed+passes\_received+passes\_live+pass\_targets+carries+touches\_live\_ball+passes\_pct\_long+touches\_def\_pen\_area+passes\_completed\_short+passes\_gk+passes\_pressure+passes\_pct+def\_actions\_outside\_pen\_area\_gk+passes\_total\_distance+psxg\_net\_gk+touches\_def\_3rd+passes\_short+passes+touches+ball\_recoveries+through\_balls+dribble\_tackles\_pct+psxg\_net\_per90\_gk+passes\_pct\_launched\_gk+save\_pct+passes\_low+xa\_net+passes\_progressive\_distance+WinCL+carry\_distance+gca\_passes\_dead+errors+passes\_switches+passes\_completed\_long+crosses\_gk+passes\_intercepted+crosses\_stopped\_gk+dribbles\_completed\_pct+passes\_left\_foot+carry\_progressive\_distance+isPremierLeague+MP+avg\_distance\_def\_actions\_gk+saves+draws\_gk+assists+goal\_kicks+gca+foot\_left+isLaLiga+passes\_right\_foot+shots\_on\_target\_against+passes\_pct\_short+aerials\_won\_pct+passes\_dead+assists\_per90+gca\_per90+passes\_completed\_launched\_gk+passes\_long+sca\_passes\_dead+def\_actions\_outside\_pen\_area\_per90\_gk+passes\_pct\_medium+crosses\_stopped\_pct\_gk+passes\_oob+own\_goals\_against\_gk+gca\_passes\_live+pens\_conceded+shots\_on\_target\_pct+throw\_ins+psxg\_gk+pens\_missed\_gk+goals\_assists\_pens\_per90+passes\_received\_pct+height+pens\_allowed+goals\_assists\_per90+passes\_launched\_gk+npxg\_net+pens\_att\_gk+cards\_red+sca+xg\_net+sca\_passes\_live+passes\_high+fouled+free\_kick\_goals\_against\_gk+cards\_yellow+corner\_kicks\_in+xa+passes\_offsides+pens\_saved+dribbles\_completed+dribble\_tackles+assisted\_shots+players\_dribbled\_past+npxg\_per\_shot+xa\_per90+passes\_into\_penalty\_area+pressure\_regain\_pct+tackles\_def\_3rd+passes\_free\_kicks+miscontrols+dribbles+dribbles\_vs+passes\_head+isSerieA+clearances+corner\_kick\_goals\_against\_gk+dribbled\_past+corner\_kicks+shots\_on\_target\_per90+tackles+goals\_against\_gk+pressures\_def\_3rd+tackles\_won+dispossessed+tackles\_mid\_3rd+fouls+shots\_total\_per90+progressive\_passes+offsides+npxg\_xa\_per90+xg\_xa\_per90+goals\_pens\_per90+passes\_blocked+touches\_mid\_3rd+aerials\_won+shots\_on\_target+sca\_dribbles+gca\_shots+pens\_att+pens\_made+pens\_won+nutmegs+goals\_per90+crosses+pressures+blocked\_shots+pressure\_regains+interceptions+goals\_per\_shot+shots\_total+pressures\_mid\_3rd+shots\_free\_kicks+touches\_att\_pen\_area+goals+sca\_fouled+pressures\_att\_3rd+aerials\_lost+touches\_att\_3rd+tackles\_att\_3rd+xg+goals\_per\_shot\_on\_target+own\_goals+npxg+sca\_shots+npxg\_per90+xg\_per90+blocks+blocked\_passes+sca\_per90+crosses\_into\_penalty\_area+passes\_into\_final\_third+D+psnpxg\_per\_shot\_on\_target\_against+goal\_kick\_length\_avg+foot\_right+isBundesliga+isLigue1+passes\_length\_avg\_gk+pct\_goal\_kicks\_launched+losses\_gk+pct\_passes\_launched\_gk+age+goals\_against\_per90\_gk+xGA+GA+L+LgRk+gca\_dribbles+gca\_fouled+gca\_og\_for+corner\_kicks\_out+corner\_kicks\_straight+foot\_both+cards\_yellow\_red+blocked\_shots\_saves',data=dataMID)

#fitting a linear model and minimizing the residual sum of squares between the targets in the dataset, and the targets predicted.

modelMID=smf.ols('ln(value)~age+goals+CL+passes\_completed\_short+passes\_into\_final\_third'

                   '+Pts+xG+xGA'

                   '+xg\_xa\_per90'

                   '+carry\_distance+tackles\_won+'

                   '+isPremierLeague+isLigue1',data=dataMID)

resultsMID=modelMID.fit()

resultsMID\_params=resultsMID.params

resultsMID1=modelMID1.fit()

resultsMID1\_params=resultsMID1.params

#creating a robust regression

modelMIDrobust=sm.RLM(modelMID.endog,modelMID.exog,data=trainMID).fit()

finalMID1 = sm.regression.linear\_model.OLSResults(modelMID,

                                              modelMIDrobust.params,

                                              modelMID.normalized\_cov\_params)

finalMID1.summary()

#getting data for various statistics for seasons and sorting by value\*\*(last 2)

data=dataMID[dataMID['Season\_201920#']==1]

data=data[['player','value','goals','xg\_xa\_per90','passes\_completed\_short','passes\_into\_final\_third','carry\_distance','tackles\_won','minutes']]

data1=data.sort\_values('value',ascending=False)[0:10]

data1

#using log function to plot values

def millions(x, pos):

    'The two args are the value and tick position'

    return '%1.1fM' % (x \* 1e-6)

formatter = mpl.ticker.FuncFormatter(millions)

#removing outliers

dataMID=dataMID[dataMID['xg\_xa\_per90']>0]

dataMID=dataMID[dataMID['passes\_completed\_short']>0]

dataMID=dataMID[dataMID['passes\_into\_final\_third']>0]

dataMID=dataMID[dataMID['carry\_distance']>0]

dataMID=dataMID[dataMID['tackles\_won']>0]

#finding the product moment correlation coefficients

goals=np.corrcoef(dataMID['value'],dataMID['goals'])

xg\_per90=np.corrcoef(dataMID['value'],dataMID['xg\_xa\_per90'])

passes\_completed\_short=np.corrcoef(dataMID['value'],dataMID['passes\_completed\_short'])

passes\_final\_third=np.corrcoef(dataMID['value'],dataMID['passes\_into\_final\_third'])

carry\_distance=np.corrcoef(dataMID['value'],dataMID['carry\_distance'])

tackles\_won=np.corrcoef(dataMID['value'],dataMID['tackles\_won'])

goals=goals[0,1]

xg\_per90=xg\_per90[0,1]

passes\_completed\_short=passes\_completed\_short[0,1]

passes\_final\_third=passes\_final\_third[0,1]

carry\_distance=carry\_distance[0,1]

tackles\_won=tackles\_won[0,1]

#finding the product moment correlation coefficients

fig, ax = plt.subplots(3, 2, figsize=(12, 12))

sns.regplot(ax=ax[0,0],x=dataMID['goals'],y=dataMID['value'],data=dataMID,color='g')

sns.regplot(ax=ax[1,0],x=dataMID['xg\_xa\_per90'],y=dataMID['value'],data=dataMID,color='blue')

sns.regplot(ax=ax[2,0],x=dataMID['passes\_completed\_short'],y=dataMID['value'],data=dataMID,color='orange')

sns.regplot(ax=ax[0,1],x=dataMID['passes\_into\_final\_third'],y=dataMID['value'],data=dataMID,color='cyan')

sns.regplot(ax=ax[1,1],x=dataMID['carry\_distance'],y=dataMID['value'],data=dataMID,color='magenta')

sns.regplot(ax=ax[2,1],x=dataMID['tackles\_won'],y=dataMID['value'],data=dataMID,color='chocolate')

#plotting linear regression graph for goals vs value

ax[0,0].yaxis.set\_major\_formatter(formatter)

ax[0,0].annotate("r=",xy=(0.8,0.85), xycoords="axes fraction")

ax[0,0].annotate("{:.2f}".format(goals),xy=(0.85,0.85), xycoords="axes fraction")

#plotting linear regression graph for xg\_xa per 90 vs value

ax[1,0].yaxis.set\_major\_formatter(formatter)

ax[1,0].annotate("r=",xy=(0.8,0.85), xycoords="axes fraction")

ax[1,0].annotate("{:.2f}".format(xg\_per90),xy=(0.85,0.85), xycoords="axes fraction")

#plotting linear regression graph for passes into the final third vs value

ax[2,0].yaxis.set\_major\_formatter(formatter)

ax[2,0].annotate("r=",xy=(0.8,0.85), xycoords="axes fraction")

ax[2,0].annotate("{:.2f}".format(passes\_completed\_short),xy=(0.85,0.85), xycoords="axes fraction")

#plotting linear regression graph for touches in opponents penalty area vs value

ax[0,1].yaxis.set\_major\_formatter(formatter)

ax[0,1].annotate("r=",xy=(0.8,0.85), xycoords="axes fraction")

ax[0,1].annotate("{:.2f}".format(carry\_distance),xy=(0.85,0.85), xycoords="axes fraction")

#plotting linear regression graph for goal creating actions vs value

ax[1,1].yaxis.set\_major\_formatter(formatter)

ax[1,1].annotate("r=",xy=(0.8,0.85), xycoords="axes fraction")

ax[1,1].annotate("{:.2f}".format(tackles\_won),xy=(0.85,0.85), xycoords="axes fraction")

#plotting linear regression graph for dribbles completed vs value

ax[2,1].yaxis.set\_major\_formatter(formatter)

ax[2,1].annotate("r=",xy=(0.8,0.85), xycoords="axes fraction")

ax[2,1].annotate("{:.2f}".format(goals),xy=(0.85,0.85), xycoords="axes fraction")