#importing SKlearn neural network

import os

import scipy as sp

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('value', ascending=False)

#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'])

#deleing outliers

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

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

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

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

data1

#GOALKEEPERS

dataGK = data1[data1['position2']=='Goalkeeper']

dataGK

#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

trainGK, testGK = train\_test\_split(dataGK, train\_size=0.8)

modelGK=smf.ols('ln(value)~age+CL+wins\_gk+draws\_gk+passes\_pct\_launched\_gk+psnpxg\_per\_shot\_on\_target\_against+'

                  '+isPremierLeague+isLigue1+clean\_sheetsm',data=dataGK)

modelGK1=smf.ols('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=dataGK)

#fitting linear model

resultsGK=modelGK.fit()

resultsGK1=modelGK1.fit()

resultsGK\_params=resultsGK.params

resultsGK1\_params=resultsGK1.params

#creating a regression model

modelGKrobust=sm.RLM(modelGK.endog,modelGK.exog,data=dataGK).fit()

finalGK1 = sm.regression.linear\_model.OLSResults(modelGK,

                                              modelGKrobust.params,

                                              modelGK.normalized\_cov\_params)

finalGK1.summary()

#Sklearn neural net to predict

predictionsGK=finalGK1.predict(dataGK)

dataGK['predsOLS']=np.exp(predictionsGK)

#getting data for various statistics for seasons and sorting by value

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

data=data[['player','value','clean\_sheetsm','psnpxg\_per\_shot\_on\_target\_against','passes\_pct\_launched\_gk']]

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

pd.DataFrame(data.mean()).transpose()

#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

dataGK=dataGK[dataGK['clean\_sheetsm']>0]

dataGK=dataGK[dataGK['passes\_pct\_launched\_gk']>0]

dataGK=dataGK[dataGK['psnpxg\_per\_shot\_on\_target\_against']>0]

#finding the product moment correlation coefficients

clean\_sheets=np.corrcoef(dataGK['value'],dataGK['clean\_sheetsm'])

passes=np.corrcoef(dataGK['value'],dataGK['passes\_pct\_launched\_gk'])

shots\_against=np.corrcoef(dataGK['value'],dataGK['psnpxg\_per\_shot\_on\_target\_against'])

clean\_sheets=clean\_sheets[0,1]

passes=passes[0,1]

shots\_against=shots\_against[0,1]

# use sns to create plot graph for each of the variables

fig, ax = plt.subplots(3, 1, figsize=(12, 12))

sns.regplot(ax=ax[0],x=dataGK['clean\_sheetsm'],y=dataGK['value'],data=dataGK,color='g')

sns.regplot(ax=ax[1],x=dataGK['passes\_pct\_launched\_gk'],y=dataGK['value'],data=dataGK,color='blue')

sns.regplot(ax=ax[2],x=dataGK['psnpxg\_per\_shot\_on\_target\_against'],y=dataGK['value'],data=dataGK,color='orange')

#plotting linear regression graph for clean sheets vs value

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

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

ax[0].annotate("{:.2f}".format(clean\_sheets),xy=(0.825,0.85), xycoords="axes fraction")

#plotting linear regression graph for passes vs value

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

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

ax[1].annotate("{:.2f}".format(passes),xy=(0.825,0.85), xycoords="axes fraction")

#plotting linear regression graph for shots against vs value

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

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

ax[2].annotate("{:.2f}".format(shots\_against),xy=(0.825,0.85), xycoords="axes fraction")