Supplementary Source Code

Master Thesis

Digital technology on financial performance.

Campus:

BI Oslo

Supervisor name:

Espen Andersen

Programme:

Master of Science in Business - Major in Strategy

A description of the scripts can be seen in section 3.4 Text Mining and Analysis in Python in the thesis.

1. Pyckaxe

```
import numpy as np
from selenium import webdriver
from webdriver_manager.firefox import GeckoDriverManager
import time
from selenium.webdriver.common.by import By
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
import pandas as pd
import csv
import random
from selenium.common.exceptions import NoSuchElementException
import PyPDF2
```

In [ ]:

```
######Pyckaxe - Real deal######
#Set def 1
#U = 20 #CSVs (Should i use CSV-loop?)
W = 400 #Browsers
V = 500 #URLs
ID = 432797 #488055 #526031 #532588 #while working #555235 <-new round with
dummy #Last:338724 #360998 #399625 #416456 #447909 #466819 #472775 #555235

#396101 #417960 #442726 #462833 #492458 #495554 #516042 #543847 #555235 #
#Starting over 24.05.2022

#machine learning!
trends = np.array(['IoS','AI', 'blockchain', 'Blockchain', 'IoT', 'IoP',
'IoD',  'AR', 'automation',  'cybersecurity',  'simulation', 'Automation',
'Cybersecurity',  'Simulation','ML'])
xtrends = len(trends)
trends2 = np.array(['artificial intelligence', 'Artificial intelligence',
'Internet of Things', 'Internet of Services', 'Internet of
People','Internet of Data','industrial robotics','Industrial robotics',
'industrial robotic', 'Industrial robotic','Modeling techniques','Semantic
technologies','Semantic technology','semantic technology' 'Cyber-physical
systems','Additive manufacturing','additive manufacturing','modeling
techniques','semantic technologies','Cloud computing', 'Big data','Big
Data', 'big data', 'Augmented reality', 'cyber-physical systems','cloud
computing', 'big data technologies', 'augmented reality','Machine
learning','machine learning', 'Machine Learning']) #machine learning!
xtrends2 = len(trends2)
#Set def 2
csvFile = open('Bigmine.csv', 'w+')
writer = csv.writer(csvFile)
writer.writerow(('ID' , 'Ticker', 'Company', 'Trends', 'Date'))
```

```
for y in range(W):
    browser =
webdriver.Firefox(executable_path="C:/Users/ngbje/Pyerke/Shippyng/geckodriv
er.exe")
    time.sleep(1)
    for x in range(V):
        browser.get('https://newsweb.oslobors.no/message/{}'.format(ID-
((y*V)+x)))
        time.sleep(0.5)
        try:
            text = browser.find_element(By.XPATH,
"/html/body/div/div/main/div[2]/div[2]/div[2]/div").text
            date = browser.find_element(By.XPATH,
"/html/body/div/div/main/div[2]/div[2]/div[1]/div[1]/span[2]").text
            ticker = browser.find_element(By.XPATH,
"/html/body/div/div/main/div[2]/div[2]/div[1]/div[3]/span[2]").text
            comp = browser.find_element(By.XPATH,
"/html/body/div/div/main/div[2]/div[1]").text
            tktext=word_tokenize(text)
            #Dummytest
            info1 = []
            info2 = []
            t1 = []
            t2 = []
            for t in range(xtrends):
                if trends[t] in tktext:
                    info1 = (ticker, comp, trends[t], date)
                    writer.writerow((ID-((y*V)+x), ticker, comp, trends[t],
date))
                    t1 = trends[t]
                    print(info1)
                    print('')
            #New addition to check space separated trends
            for t in range(xtrends2):
                if trends2[t] in text:
                    info2 = (ticker, comp, trends2[t], date)
                    writer.writerow((ID-((y*V)+x), ticker, comp,
trends2[t], date))
                    t2 = trends2[t]
                    print(info2)
                    print('')
            #Dummytest
            if info1 != (ticker, comp, t1, date) and info2 != (ticker,
comp, t2, date):
                info3 = (ticker, comp, 'dummy', date)
                writer.writerow((ID-((y*V)+x), ticker, comp, 'dummy',
date))
                #print(info3)
                #print('')
        except NoSuchElementException:
            writer.writerow((ID-((y*V)+x), "Failed", "Failed", "Failed"))
            print("No, {} failed to find element".format((ID-((y*V)+x))))
            print('')
            pass
    browser.quit()
csvFile.close()
```

In [ ]:

```
csvFile.close()
```

2. Pryceaxe

```
import pandas as pd
import time
import yfinance as yf
from yahoofinancials import YahooFinancials
import csv
```

In [ ]:

```
data = pd.read_csv('Randomtech.csv') #Need to change "." to "-", no!
data
```

In [ ]:

```
dataose = pd.read_csv('OSEBX GR_quote_chart.csv')
dataose
```

In [ ]:

```
dataose['time'] = pd.to_datetime(dataose.time)
dataose['nr'] = dataose.index
dataose.index = dataose['time'].dt.strftime('%d.%m.%Y')
dataose
```

In [ ]:

```
#Getting eod-price for both companies and OSEBX for all dates in panel with
dummies for windows ###### REAL DEAL ######
W = 8200
V = 135
data = pd.read_csv('Randomtech.csv')
dfo = pd.read_csv('OSEBX GR_quote_chart.csv')
dfo['time'] = pd.to_datetime(dfo.time)
dfo['nr'] = dfo.index
dfo.index = dfo['time'].dt.strftime('%d.%m.%Y')

csvFile = open('Stockmarketreactionlastall.csv', 'w+')
writer = csv.writer(csvFile)
writer.writerow(('ID','Ticker', 'Company', 'Trend', 'Date0',
'Time0','Timex',
'Sprice','Mprice','Sreturn','Mreturn','MSreturn','Dumevent','Dumanti','Duma
dju'))

for y in range(W):
    try:
        df = yf.download(("{}.OL".format(data.iloc[y,1]))) #Locate ticker
in this from csv.
        time.sleep(4)
        price0 = df.loc[(data.iloc[y,4])]
        pricet = df.index.searchsorted(data.iloc[y,4])
        price0ose = dfo.loc[(data.iloc[y,4])]
        pricetose = dfo.loc[(data.iloc[y,4]),['nr']]
#dfo.index.searchsorted(data.iloc[y,4])
        for x in range(V):
            try:
```

```
                #instert new order here
                pricex = df.iloc[pricet+(6-x)]
                pricexose = dfo.iloc[pricetose+(6-x)]
                pricexm1 = df.iloc[pricet+(5-x)]
                pricexosem1 = dfo.iloc[pricetose+(5-x)]
                returnx = (pricex[3]/pricexm1[3])
                returnxose = (pricexose.iloc[0,1]/pricexosem1.iloc[0,1])
                if ((6-x) >= 2):
                    dadju = 1
                else:
                    dadju = 0
                if ((6-x) == 1 or (6-x) == 0):
                    devent = 1
                else:
                    devent = 0
                if ((6-x) < 0 and (6-x) > -6):
                    danti = 1
                else:
                    danti = 0   #'ID',          'Ticker',      'Company',
'Trend',      'Date0',     'Time0',       'Timex', 'Sprice',  'Mprice',
'Sreturn','Mreturn','MSreturn', 'Dumevent','Dumanti','Dumadju')

writer.writerow((data.iloc[y,0],data.iloc[y,1],data.iloc[y,2],data.iloc[y,3
],data.iloc[y,4],data.iloc[y,5],(6-x),pricex[3], pricexose.iloc[0,1],
returnx, returnxose,returnx-returnxose, devent, danti, dadju))
            except: #NoSuchElementException:

writer.writerow((data.iloc[y,0],data.iloc[y,1],data.iloc[y,2],data.iloc[y,3
],data.iloc[y,4],data.iloc[y,5],(6-x),
'fail','Mprice','Sreturn','Mreturn','MSreturn','Dumevent','Dumanti','Dumadj
u'))
                print("No, {} failed to find
element".format(data.iloc[[y]]))
                print('')
                pass

    except: #NoSuchElementException:
        print("No, {} failed to find element".format(data.iloc[[y]]))
        print('')
        pass
csvFile.close()
```

In [ ]:

```
csvFile.close()
```

3. Pydfaxe

```
import numpy as np
import pandas as pd
import time
import nltk
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
import csv
import random
import PyPDF2
```

```
import fitz
import pandas as pd
import time
import yfinance as yf
from yahoofinancials import YahooFinancials as yfs
import csv
import os
```

In [ ]:

```
#make all uppercase
path = 'C:\\Users\\ngbje\\Pyerke\\Shippyng\\Master-supyr-
mining\\Pydf\\Yearly reports\\'

for file in os.listdir(path):
    os.rename(path + file, path + file.upper())

then = os.listdir(path)
print(then)
```

In [ ]:

```
#remove '_'
path = 'C:\\Users\\ngbje\\Pyerke\\Shippyng\\Master-supyr-
mining\\Pydf\\Yearly reports\\'

for file in os.listdir(path):
    if file[3:4] == '_':
        os.rename(path + file, path + file[0:2] + file[4:])
```

In [ ]:

```
######Pydfaxe - Real deal######
#Set def 1
W = 7 #Browsers
V = 500 #URLs

trends = np.array(['IoS','AI', 'blockchain', 'Blockchain', 'IoT', 'IoP',
'IoD', 'automation',  'cybersecurity',  'simulation', 'Automation',
'Cybersecurity', 'Simulation','modeling','Modeling','Robotics','robotics',
'ML'])
xtrends = len(trends)
trends2 = np.array(['artificial intelligence', 'Artificial intelligence',
'Internet of Things', 'Internet of Services', 'Internet of
People','Internet of Data','industrial robotics','Industrial robotics',
'industrial robotic', 'Industrial robotic','Modeling techniques','Semantic
technologies','Semantic technology','semantic technology', 'Cyber-physical
systems','Additive manufacturing','additive manufacturing','modeling
techniques','semantic technologies','Cloud computing','Big data',
'Augmented reality','cyber physical systems' ,'Cyber physical systems',
'cyber-physical systems','cloud computing', 'big data', 'augmented
reality', 'Machine learning', 'machine learning', 'Machine Learning'])
xtrends2 = len(trends2)
#Set def 2

csvFile = open('Pydfmine.csv', 'w+')
writer = csv.writer(csvFile)
writer.writerow(('Year' , 'Company', 'Trends', 'Count','Trendtest'))
eqs = pd.read_excel(r"C:\Users\ngbje\Pyerke\Shippyng\Master-supyr-
mining\OSBCompanies.xlsx") #Pydf\Euronext_Equities.xlsx")
```

```
for x in range(V):
    #time.sleep(1)
    for y in range(W):
        # creating a pdf file object

        try:
            pdfile = open(r'C:\Users\ngbje\Pyerke\Shippyng\Master-supyr-
mining\Pydf\Yearly reports\{}{}{}{}'.format((eqs.iloc[x,1])-y,'.
',eqs.iloc[x,0],'.PDF'), 'rb')
            time.sleep(1)
            pdfreader = PyPDF2.PdfFileReader(pdfile)
            pagenum = pdfreader.numPages
            #Testing
            print((eqs.iloc[x,1])-y,(eqs.iloc[x,1]),y,eqs.iloc[x,0])
            #Textmining
            with fitz.open(r'C:\Users\ngbje\Pyerke\Shippyng\Master-supyr-
mining\Pydf\Yearly reports\{}{}{}{}'.format((eqs.iloc[x,1])-y,'.
',eqs.iloc[x,0],'.PDF')) as doc:
                text = ""
                for page in doc:
                    text += page.get_text()

            tktext=word_tokenize(text)
            #Trendsmining
            for t in range(xtrends):
                if trends[t] in tktext:
                    tcount = (trends[t], tktext.count(trends[t]))
                    #print(tcount)
                    info = ((eqs.iloc[x,1])-y,eqs.iloc[x,0], trends[t],
tcount[1],tcount[0])
                    writer.writerow(((eqs.iloc[x,1])-y,eqs.iloc[x,0],
trends[t], tcount[1],tcount[0]))
                    print(info)
                    print('')

            for t in range(xtrends2):
                if trends2[t] in text:
                    tcount2 = (trends2[t], text.count(trends2[t]))
                    #print(tcount2)
                    info2 = ((eqs.iloc[x,1])-y,eqs.iloc[x,0], trends2[t],
tcount2[1],tcount2[0])
                    writer.writerow(((eqs.iloc[x,1])-y,eqs.iloc[x,0],
trends2[t], tcount2[1],tcount2[0]))
                    print(info2)
                    print('')
        except:# RuntimeError or FileNotFoundError:
            writer.writerow(((eqs.iloc[x,1])-y,eqs.iloc[x,0], "Failed",
"Failed","Failed"))
            print("No, failed to find {} file".format(((eqs.iloc[x,1])-
y,eqs.iloc[x,0])))
            print('')
            pass
csvFile.close()


csvFile.close()
```

```
data = pd.read_csv('Pydfminelastfinal.csv')
data


fin = pd.read_excel('Compustatfinancialfull+.xlsx')#, index_col = ['Global
Company Key', 'Data Year - Fiscal'])
fin


comp = pd.read_excel('OSBCompanies.xlsx')
comp


#Fixed issues new real deal ####
W = 205
#V = 135
data = pd.read_csv('Pydfminelastfinal.csv')
fin = pd.read_excel('Compustatfinancialfullall.xlsx')
comp = pd.read_excel('OSBCompanies.xlsx')

csvFile = open('Pydfin.csv', 'w+')
writer = csv.writer(csvFile) #Make with EBIT, +1,+2,+3
writer.writerow(('CompanyID','ISIN','Ticker', 'Company', 'Year', 'EBIT',
'ROA','ROE','Log(SizeAssets)','Log(SizeSales)','Log(SizeEmployees)','AssetT
urnover','Debtratio','TangibleAssetRatio','Production','Service','Trendsum'
, 'Numdifftrend','Blockchain', 'Ios', 'Ai', 'Iot', 'Iop', 'Iod', 'Ar',
'Automation', 'Cybersecurity', 'Simulation', 'Cps', 'Ml', 'Robotics',
'Modeling', 'Semantic', 'Additive', 'Cloud',
'Bigdata','Notrends','Noblockchain', 'Noios', 'Noai', 'Noiot', 'Noiop',
'Noiod', 'Noar', 'Noautomation', 'Nocybersecurity', 'Nosimulation',
'Nocps', 'Noml', 'Norobotics', 'Nomodeling', 'Nosemantic', 'Noadditive',
'Nocloud',
'Nobigdata','Miningconstruction','Foodtextileapparel','Forestpaperpublishin
g','Chemicalspharma', 'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Otherservices','Administrationandother'))


for y in range(W):
    isin = comp.iloc[y,3]
    ticker = comp.iloc[y,2]
    company = comp.iloc[y,0]
        #price0 = df.loc[(data.iloc[y,4])]
    for x in range(comp.iloc[y,1]):
        #year = fin.iloc[x,2]
        finans = fin.loc[(fin['ISIN'] == isin) & (fin['Year'] == 2020-x)]
        #for lagged?
        finansT1 = fin.loc[(fin['ISIN'] == isin) & (fin['Year'] == 2020-
x+0)] #regulere lagged effekt
        year = finans[['Year']]
        ebit = finansT1[['EBIT/Rev']]
            #Included
        roa = finansT1[['ROA']]
        roe = finansT1[['ROE']]
        sizeAss = finansT1['Size(Assets)']
        sizeSal = finansT1['Size(Sales)']
        sizeEmp = finansT1['Size(Employees)']
```

```
        turnover = finansT1['AssetTurnover']
        debt = finansT1['Debt/Assets'] #prøve med debt/assets? done
        tangible = finansT1['TangibleAssets']
        sic = finansT1['SIC']
        comid = finansT1['Global Company Key']
        sic = sic.to_numpy()
        if 100 < sic < 4000:
#miningconstruction,foodtextileapparel,forestpaperpublishing,chemicalspharm
a, refiningrubberplastic, containerssteelheavy,
computersautosaerospace,transportation,telephoneutilities, wholesaleretail,
bankfinancial,hotelentertainment,hospitalmanagement
            production = 1
        else:
            production = 0
        if 4000 < sic < 9999:
#miningconstruction,foodtextileapparel,forestpaperpublishing,chemicalspharm
a, refiningrubberplastic, containerssteelheavy,
computersautosaerospace,transportation,telephoneutilities, wholesaleretail,
bankfinancial,hotelentertainment,hospitalmanagement
            service = 1
        else:
            service = 0
        if 100 < sic < 2000:
#miningconstruction,foodtextileapparel,forestpaperpublishing,chemicalspharm
a, refiningrubberplastic, containerssteelheavy,
computersautosaerospace,transportation,telephoneutilities, wholesaleretail,
bankfinancial,hotelentertainment,hospitalmanagement
            miningconstruction = 1
        else:
            miningconstruction = 0
        if (sic > 1999) & (sic < 2391):
            foodtextileapparel = 1
        else:
            foodtextileapparel = 0
        if (sic > 2390) & (sic < 2781):
            forestpaperpublishing = 1
        else:
            forestpaperpublishing = 0
        if (sic > 2780) & (sic < 2891):
            chemicalspharma = 1
        else:
            chemicalspharma = 0
        if (sic > 2890) & (sic < 3200):
            refiningrubberplastic = 1
        else:
            refiningrubberplastic = 0
        if (sic > 3199) & (sic < 3570):
            containerssteelheavy = 1
        else:
            containerssteelheavy = 0
        if (sic > 3569) & (sic < 3991):
            computersautosaerospace = 1
        else:
            computersautosaerospace = 0
        if (sic > 3990) & (sic < 4732):
            transportation = 1
        else:
```

```
            transportation = 0
        if (sic > 4731) & (sic < 4992):
            telephoneutilities = 1
        else:
            telephoneutilities = 0
        if (sic > 4991) & (sic < 5991):
            wholesaleretail = 1
        else:
            wholesaleretail = 0
        if (sic > 6149) & (sic < 6701):
            bankfinancial = 1
        else:
            bankfinancial = 0
        if (sic > 6799) & (sic < 9000):
            otherservices = 1
        else:
            otherservices = 0
        if (sic > 8999) & (sic < 9999):
            administrationandother = 1
        else:
            administrationandother = 0

        trends = data.loc[(data['Company'] == company) & (data['Year'] ==
x+1)] #.count((1+x and company)) #trendsum = trends['Count'].sum()
        trendsum = len(trends) #Må fikse denne counten. Teller bare
forskjellige rader, og ikke hvor mange av hvert ord
        blockchain = 0; ios = 0; ai = 0; iot = 0; iop = 0; iod = 0; ar = 0;
automation = 0; cybersecurity = 0; simulation= 0; cps = 0; ml = 0; robotics
= 0; modeling = 0; semantic = 0; additive = 0; cloud = 0; bigdata = 0
        noblockchain = 0; noios = 0; noai = 0; noiot = 0; noiop = 0; noiod
= 0; noar = 0; noautomation = 0; nocybersecurity = 0; nosimulation= 0;
nocps = 0; noml = 0; norobotics = 0; nomodeling = 0; nosemantic = 0;
noadditive = 0; nocloud = 0; nobigdata = 0
        for z in range (len(trends)):
            trend = trends.iloc[z,2]
            trendcount = trends.iloc[z,3]
            if trendcount == 'Failed':
                trendcount = 0
            else:
                trendcount = int(trendcount)
            print('{} shoulde be number of {} trends'.format(trendcount,
trend))
            if trend in ('blockchain Blockchain'):
                blockchain = 1
                noblockchain += trendcount
                #else:
                 #   blockchain = 0
            if trend in ('IoS Internet of Services'):
                ios = 1
                noios += trendcount
                #else:
                 #   ios = 0
            if trend in ('AI artificial intelligence Artificial
intelligence'):
                ai = 1
                noai += trendcount
                #else:
```

```
                #   ai = 0
            if trend in ('IoT Internet of Things'):
                iot = 1
                noiot += trendcount
                #else:
                #   iot = 0
            if trend in ('IoP Internet of People'):
                iop = 1
                noiop += trendcount
                #else:
                #   iop = 0
            if trend in ('IoD Internet of Data'):
                iod = 1
                noiod += trendcount
                #else:
                #   iod = 0
            if trend in ('Augmented reality augmented reality'):
                ar = 1
                noar += trendcount
                #else:
                #   ar = 0
            if trend in ('automation Automation'):
                automation = 1
                noautomation += trendcount
                #else:
                #   automation = 0
            if trend in ('cybersecurity Cybersecurity'):
                cybersecurity = 1
                nocybersecurity += trendcount
                #else:
                #   cybersecurity = 0
            if trend in ('simulation Simulation'):
                simulation = 1
                nosimulation += trendcount
                #else:
                #   simulation = 0
            if trend in ('Cyber-physical systems cyber-physical systems
cyber physical systems Cyber physical systems'):
                cps = 1
                nocps += trendcount
                #else:
                #   cps = 0
            if trend in ('ML Machine learning machine learning Machine
Learning'):
                ml = 1
                noml += trendcount
                #else:
                #   ml = 0
            if trend in ('industrial robotics Industrial robotics
industrial robotic Industrial robotic robotics Robotics'):
                robotics = 1
                norobotics += trendcount
                #else:
                #   robotics = 0
            if trend in ('Modeling techniques modeling techniques modeling
Modeling'):
                modeling = 1
```

```
                nomodeling += trendcount
                #else:
                 #   modeling = 0
            if trend in ('Semantic technologies Semantic technology
semantic technology semantic technologies'):
                semantic = 1
                nosemantic += trendcount
                #else:
                 #   semantic = 0
            if trend in ('Additive manufacturing additive manufacturing'):
                additive = 1
                noadditive += trendcount
                #else:
                 #   additive = 0
            if trend in ('Cloud computing cloud computing'):
                cloud = 1
                nocloud += trendcount
                #else:
                 #   cloud = 0
            if trend in ('Big data Big Data big data'):
                bigdata = 1
                nobigdata += trendcount
                #else:
                 #   bigdata = 0
        numtrends = (blockchain + ios + ai + iot + iop + iod + ar +
automation + cybersecurity + simulation + cps + ml + robotics + modeling +
semantic + additive + cloud + bigdata)
        notrends = (noblockchain + noios + noai + noiot + noiop + noiod +
noar + noautomation + nocybersecurity + nosimulation + nocps + noml +
norobotics + nomodeling + nosemantic + noadditive + nocloud + nobigdata)

#'ID','Ticker','Company','Trend','Date0',Time0','Timex','Sprice','Mprice',S
return','Mreturn','MSreturn', 'Dumevent','Dumanti','Dumadju')

writer.writerow((comid.to_numpy(),isin,ticker,company,year.to_numpy(),ebit.
to_numpy(),
roa.to_numpy(),roe.to_numpy(),sizeAss.to_numpy(),sizeSal.to_numpy(),sizeEmp
.to_numpy(),turnover.to_numpy(),debt.to_numpy(),tangible.to_numpy(),product
ion,service,trendsum,numtrends, blockchain, ios, ai, iot, iop, iod, ar,
automation, cybersecurity, simulation, cps, ml, robotics, modeling,
semantic, additive, cloud, bigdata,notrends,noblockchain, noios, noai,
noiot, noiop, noiod, noar, noautomation, nocybersecurity, nosimulation,
nocps, noml, norobotics, nomodeling, nosemantic, noadditive, nocloud,
nobigdata,miningconstruction,foodtextileapparel,forestpaperpublishing,chemi
calspharma, refiningrubberplastic, containerssteelheavy,
computersautosaerospace,transportation,telephoneutilities, wholesaleretail,
bankfinancial,otherservices,administrationandother))

        #print(comid.to_numpy())
        #print(isin)
        #print(ticker)
        #print(company)
        #print((year.to_numpy()))
        #print((ebit.to_numpy()))
        #print((roa.to_numpy()))
        #print((roe.to_numpy()))
        #print(size.to_numpy())
```

```
        #print((debt.to_numpy()))
```

In [ ]:

```
csvFile.close() #change [] to nan and remove [ and ] in excel
```

4. Casting

```
import pandas as pd
import csv
import numpy as np
```

In [ ]:

```
#Real Deal
W = 200000
data = pd.read_csv('Stockmarketreactionlastx.csv')

csvFile = open('Stockmarketreactiontest.csv', 'w+')
writer = csv.writer(csvFile)
writer.writerow(('ID','Ticker', 'Company', 'Trend', 'Date0',
'Time0','Timex',
'Sprice','Mprice','Sreturn','Mreturn','MSreturn','Dumevent','Dumanti','Duma
dju', 'Blockchain', 'Ios', 'Ai', 'Iot', 'Iop', 'Iod', 'Ar', 'Automation',
'Cybersecurity', 'Simulation', 'Cps', 'Ml', 'Robotics', 'Modeling',
'Semantic', 'Additive', 'Cloud', 'Bigdata'))

for x in range(W):
    try:
        ID = data.iloc[x,0]
        ticker = data.iloc[x,1]
        comp = data.iloc[x,2]
        trend = data.iloc[x,3]
        date0 = data.iloc[x,4]
        time0 = data.iloc[x,5]
        timex = data.iloc[x,6]
        sprice = data.iloc[x,7]
        mprice = data.iloc[x,8]
        sreturn = data.iloc[x,9]
        mreturn = data.iloc[x,10]
        msreturn = data.iloc[x,11]
        #If trends dummy
        if trend in ('blockchain' or 'Blockchain'):
            blockchain = 1
        else:
            blockchain = 0
        if trend in ('IoS' or 'Internet of Services'):
            ios = 1
        else:
            ios = 0
        if trend in ('AI' or 'artificial intelligence' or 'Artificial
intelligence'):
            ai = 1
        else:
            ai = 0
        if trend in ('IoT' or 'Internet of Things'):
            iot = 1
```

```
        else:
            iot = 0
        if trend in ('IoP' or 'Internet of People'):
            iop = 1
        else:
            iop = 0
        if trend in ('IoD' or 'Internet of Data'):
            iod = 1
        else:
            iod = 0
        if trend in ('AR' or 'Augmented reality' or 'augmented reality'):
            ar = 1
        else:
            ar = 0
        if trend in ('automation' or 'Automation'):
            automation = 1
        else:
            automation = 0
        if trend in ('cybersecurity' or 'Cybersecurity'):
            cybersecurity = 1
        else:
            cybersecurity = 0
        if trend in ('simulation' or 'Simulation'):
            simulation = 1
        else:
            simulation = 0
        if trend in ('Cyber-physical systems' or 'cyber-physical systems'):
            cps = 1
        else:
            cps = 0
        if trend in ('ML' or 'Machine learning' or 'machine learning' or
'Machine Learning'):
            ml = 1
        else:
            ml = 0
        if trend in ('industrial robotics' or 'Industrial robotics' or
'industrial robotic' or 'Industrial robotic'):
            robotics = 1
        else:
            robotics = 0
        if trend in ('Modeling techniques' or 'modeling techniques'):
            modeling = 1
        else:
            modeling = 0
        if trend in ('Semantic technologies' or 'Semantic technology' or
'semantic technology' or 'semantic technologies'):
            semantic = 1
        else:
            semantic = 0
        if trend in ('Additive manufacturing' or 'additive manufacturing'):
            additive = 1
        else:
            additive = 0
        if trend in ('Cloud computing' or 'cloud computing'):
            cloud = 1
        else:
            cloud = 0
```

```
        if trend in ('Big data' or 'Big Data' or 'big data'):
            bigdata = 1
        else:
            bigdata = 0

            #if stage dummy
        try:
            if ((timex) >= 2):
                dadju = 1
            else:
                dadju = 0
            if (((timex == 1) or (timex == 0 and int(time0[:2]) < 16))):
                devent = 1
            else:
                devent = 0
            if (((timex) < 0 and timex > -6) or (timex == 0 and
int(time0[:2]) >= 16)):
                danti = 1
            else:
                danti = 0
#'ID','Ticker','Company','Trend','Date0',Time0','Timex','Sprice','Mprice',S
return','Mreturn','MSreturn', 'Dumevent','Dumanti','Dumadju')
            writer.writerow((ID, ticker, comp, trend, date0, time0, timex,
sprice, mprice, sreturn, mreturn, msreturn, devent, danti, dadju,
blockchain, ios, ai, iot, iop, iod, ar, automation, cybersecurity,
simulation, cps, ml, robotics, modeling, semantic, additive, cloud,
bigdata))
        except: #NoSuchElementException:

writer.writerow((data.iloc[y,0],data.iloc[y,1],data.iloc[y,2],data.iloc[y,3
],data.iloc[y,4],data.iloc[y,5],(6-x), 'fail'))
            print("No, {} failed on ifs".format(data.iloc[[x]]))
            print('')
            pass

    except: #NoSuchElementException:
        print("No, {} failed to find element".format(data.iloc[[x]]))
        print('')
        pass
csvFile.close()
```

In [ ]:

```
csvFile.close()
```

5. MeltingPyt

```
import pandas as pd
import glob
import os
import csv
```

In [ ]:

```
f = open('Bigpytlastall.csv', 'w', encoding='UTF8', newline='')
```

```
writer = csv.writer(f)
```

In [ ]:

```
##Read with CSV and ##Append by writerows
rows = []
with open("Bigminedummy-432797-343134.csv", 'r') as file:
    csvreader = csv.reader(file)
    header = next(csvreader)
    for row in csvreader:
        rows.append(row)

writer.writerow(header)
writer.writerows(rows)
#print(header) rate limit Bigminedummy-488055-432797
#print(rows)
```

In [ ]:

```
f.close()
```

In [ ]:

```
#BigpytClean after replacing " with space in excel and adding Time as
header
data= pd.read_csv("Bigpytlastall.csv")
data
```

In [ ]:

```
#Find failed
Fail=[]

V = len(data)
for x in range(V):
    if data.iloc[x,1] == "Failed":
        #print(data.iloc[x,0])
        Fail.append(data.iloc[x,0])
print(len(Fail))
print(Fail)
```

In [ ]:

```
len(Fail)
```

In [ ]:

```
cleandata = pd.read_csv("Bigpytlastall.csv", index_col ="ID" )

# dropping passed values
cleandata.drop(Fail, inplace = True)

# display
cleandata
```

In [ ]:

```
cleandata.to_csv('Biggerpytlastall.csv', sep=',', encoding='utf-8')
```

In [ ]:

```
cleandata = cleandata.reset_index()
```

In [ ]:

```
cleandata
```

In [ ]:

```
#Find tech and randoms REAL DEAL#
randomtech = pd.read_csv("Biggerpytlastall.csv")
Fail=[]
```

```
V = len(randomtech)
for x in range(V):
    ID = randomtech.iloc[x,0].astype(str)
    if randomtech.iloc[x,3] == "dummy" and ID[4:] != '42' and ID[4:] !=
'13':
        Fail.append(randomtech.iloc[x,0])
#dropping passed values
print(len(Fail))
print(Fail)
```

In [ ]:

```
randomtech.set_index(['ID'], inplace=True)
randomtech.drop(Fail, inplace = True)
randomtech
```

In [ ]:

```
randomtech.to_csv('Randomtech.csv', sep=',', encoding='utf-8')
```

In [ ]:

```
#Open the BiggerPyt
f = open('Biggerpytlast2.csv', 'w', encoding='UTF8', newline='')
writer = csv.writer(f)
```

In [ ]:

```
##Read with CSV and ##Append pyts writerows to one
rows = []
with open("Bigpytlaster.csv", 'r') as file:
    csvreader = csv.reader(file)
    header = next(csvreader)
    for row in csvreader:
        rows.append(row)

writer.writerow(header)
writer.writerows(rows)
print(header)
print(rows)
```

In [ ]:

```
f.close()
```

In [ ]:

```
#Removed header in-data after inspecting
```

In [ ]:

```
data = pd.read_csv("Bigpytlaster.csv", index_col ="ID" )
data
```

6. RegressionAnnualReports

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn import datasets, linear_model, metrics
from linearmodels import PooledOLS
import statsmodels.api as sm
```

```
import statsmodels.formula.api as smf
from statsmodels.stats.diagnostic import het_white, het_breuschpagan
from statsmodels.stats.stattools import durbin_watson
from statsmodels.sandbox.regression.predstd import wls_prediction_std
from statsmodels.stats.outliers_influence import variance_inflation_factor
from linearmodels import PanelOLS
from linearmodels import RandomEffects
import numpy.linalg as la
from scipy import stats
import csv
```

In [ ]:

```
data = pd.read_csv('Pydfinlastall2.csv', index_col = ['CompanyID'])
data
```

In [ ]:

```
#REGRESSION OF ANNUAL REPORTS not lagged FINP
```

In [ ]:

```
data = pd.read_csv('Pydfinlastall0.csv', index_col = ['CompanyID'])
data = data.dropna() #Dropping nan's
data = data[data.duplicated(subset=['ISIN'], keep= False)]#drop enties with
only one year/row.
data = data[~data.isin([np.nan, np.inf, -np.inf]).any(1)] #drop inf and
nans
data = data.reset_index()
data.set_index(['CompanyID', 'Year'], inplace=True)
print(data['ISIN'].nunique())
data
```

In [ ]:

```
#Drop companies with less than 3 years
data = data.reset_index()
df = []
for i in range(len(data)):
    obs = (len(data[data['ISIN'] == '{}'.format(data.iloc[i,2])]))
    if obs < 3:
        df.append(i)
        #data.drop([i],axis=0,inplace = False)
        print('for index: {} obs: {} was found'.format(i,obs))
data.drop(df,axis=0,inplace = True)
```

In [ ]:

```
data
```

In [ ]:

```
data.set_index(['CompanyID', 'Year'], inplace=True)
print(data['ISIN'].nunique())
data
```

In [ ]:

```
data.describe()
data[['Noblockchain','Noios','Noai','Noiot','Noiop','Noiod','Noar','Noautom
ation','Nocybersecurity','Nosimulation','Nocps','Noml','Norobotics','Nomode
ling','Nosemantic','Noadditive','Nocloud','Nobigdata']].describe()
```

In [ ]:

```
count =
data[['Blockchain','Ios','Ai','Iot','Iop','Iod','Ar','Automation','Cybersec
urity','Simulation','Cps','Ml','Robotics','Modeling','Semantic','Additive',
'Cloud','Bigdata']].sum()
```

```
nocount =
data[['Noblockchain','Noios','Noai','Noiot','Noiop','Noiod','Noar','Noautom
ation','Nocybersecurity','Nosimulation','Nocps','Noml','Norobotics','Nomode
ling','Nosemantic','Noadditive','Nocloud','Nobigdata']].sum()
print(count)
print(nocount)
plt.hist(x, bins = 10)
plt.show()
```

In [ ]:

```
#[['ROA','EBIT','Log(SizeAssets)','Log(SizeSales)','Log(SizeEmployees)','As
setTurnover','Debtratio','TangibleAssetRatio','Miningconstruction','Foodtex
tileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices','Noblockchain','No
ios','Noai','Noiot','Noiop','Noiod','Noar','Noautomation','Nocybersecurity'
,'Nosimulation','Nocps','Noml','Norobotics','Nomodeling','Nosemantic','Noad
ditive','Nocloud','Nobigdata']]


# the independent variables set
X =
data[['Log(SizeSales)','AssetTurnover','Debtratio','TangibleAssetRatio','Mi
ningconstruction','Foodtextileapparel','Forestpaperpublishing','Chemicalsph
arma', 'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices','Noblockchain','No
ios','Noai','Noiot','Noiop','Noiod','Noar','Noautomation','Nocybersecurity'
,'Nosimulation','Nocps','Noml','Norobotics','Nomodeling','Nosemantic','Noad
ditive','Nocloud','Nobigdata']]

# VIF dataframe
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns

# calculating VIF for each feature
vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]

print(vif_data)
```

In [ ]:

```
#Correlation matrix
corr =
data[['ROA','EBIT','Log(SizeAssets)','Log(SizeSales)','Log(SizeEmployees)',
'AssetTurnover','Debtratio','TangibleAssetRatio','Miningconstruction','Food
textileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices','Noblockchain','No
ios','Noai','Noiot','Noiop','Noiod','Noar','Noautomation','Nocybersecurity'
,'Nosimulation','Nocps','Noml','Norobotics','Nomodeling','Nosemantic','Noad
ditive','Nocloud','Nobigdata']].corr()
corr.style.background_gradient(cmap='coolwarm')
```

In [ ]:

```
data['Notrends2'] = data.Notrends ** 2
```

In [ ]:

```
# Perform PooledOLS
X =
sm.tools.tools.add_constant(data[['Notrends','Notrends2','Log(SizeSales)','
AssetTurnover','Debtratio','TangibleAssetRatio','Foodtextileapparel','Fores
tpaperpublishing','Chemicalspharma', 'Refiningrubberplastic',
'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])

print('For EBIT')
Y = data['EBIT']
mod = PooledOLS(Y, X)
pooledOLS_res = mod.fit(cov_type='clustered', cluster_entity=True)
# Store values for checking homoskedasticity graphically
fittedvals_pooled_OLS = pooledOLS_res.predict().fitted_values
residuals_pooled_OLS = pooledOLS_res.resids
print('3A. Homoskedasticity')
print('3A.1 Residuals-Plot for growing Variance Detection')
fig, ax = plt.subplots()
ax.scatter(fittedvals_pooled_OLS, residuals_pooled_OLS, color = 'blue')
ax.axhline(0, color = 'r', ls = '--')
ax.set_xlabel('Predicted Values', fontsize = 15)
ax.set_ylabel('Residuals', fontsize = 15)
ax.set_title('Homoskedasticity Test', fontsize = 30)
plt.show()
print('3A.2 White-Test')
pooled_OLS_dataset = pd.concat([data, residuals_pooled_OLS], axis=1)
#pooled_OLS_dataset = pooled_OLS_dataset.drop(['Timex'], axis =
1).fillna(0)
X = sm.tools.tools.add_constant(data['Notrends']).fillna(0)
white_test_results = het_white(pooled_OLS_dataset['residual'], X)
labels = ['LM-Stat', 'LM p-val', 'F-Stat', 'F p-val']
print(dict(zip(labels, white_test_results)))
print('3A.3 Breusch-Pagan-Test')
breusch_pagan_test_results =
het_breuschpagan(pooled_OLS_dataset['residual'], X)
labels = ['LM-Stat', 'LM p-val', 'F-Stat', 'F p-val']
print(dict(zip(labels, breusch_pagan_test_results)))
print('3.B Non-Autocorrelation')
print('Durbin-Watson-Test # 0-4 scale where 2 is least biased')
durbin_watson_test_results = durbin_watson(pooled_OLS_dataset['residual'])
print(durbin_watson_test_results)
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Notrends','Notrends2','Log(SizeSales)','
AssetTurnover','Debtratio','TangibleAssetRatio','Foodtextileapparel','Fores
tpaperpublishing','Chemicalspharma', 'Refiningrubberplastic',
'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
```

```
print('For ROA')
Y = data['ROA']
mod = PooledOLS(Y, X)
pooledOLS_res = mod.fit(cov_type='clustered', cluster_entity=True)
# Store values for checking homoskedasticity graphically
fittedvals_pooled_OLS = pooledOLS_res.predict().fitted_values
residuals_pooled_OLS = pooledOLS_res.resids
print('3A. Homoskedasticity')
print('Residuals-Plot for growing Variance Detection')
fig, ax = plt.subplots()
ax.scatter(fittedvals_pooled_OLS, residuals_pooled_OLS, color = 'blue')
ax.axhline(0, color = 'r', ls = '--')
ax.set_xlabel('Predicted Values', fontsize = 15)
ax.set_ylabel('Residuals', fontsize = 15)
ax.set_title('Homoskedasticity Test', fontsize = 30)
plt.show()
print('3A.2 White-Test')
pooled_OLS_dataset = pd.concat([data, residuals_pooled_OLS], axis=1)
#pooled_OLS_dataset = pooled_OLS_dataset.drop(['Timex'], axis =
1).fillna(0)
X = sm.tools.tools.add_constant(data['Notrends']).fillna(0)
white_test_results = het_white(pooled_OLS_dataset['residual'], X)
labels = ['LM-Stat', 'LM p-val', 'F-Stat', 'F p-val']
print(dict(zip(labels, white_test_results)))
print('3A.3 Breusch-Pagan-Test')
breusch_pagan_test_results =
het_breuschpagan(pooled_OLS_dataset['residual'], X)
labels = ['LM-Stat', 'LM p-val', 'F-Stat', 'F p-val']
print(dict(zip(labels, breusch_pagan_test_results)))
print('3.B Non-Autocorrelation')
print('Durbin-Watson-Test # 0-4 scale where 2 is least biased')
durbin_watson_test_results = durbin_watson(pooled_OLS_dataset['residual'])
print(durbin_watson_test_results)
```

In [ ]:

```
# FE und RE model ,'Notrends2'
X =
sm.tools.tools.add_constant(data[['Notrends','Notrends2','Log(SizeSales)','
AssetTurnover','Debtratio','TangibleAssetRatio','Foodtextileapparel','Fores
tpaperpublishing','Chemicalspharma', 'Refiningrubberplastic',
'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])

Y = data['ROA']
# random effects model
model_re = RandomEffects(Y, X)
re_res = model_re.fit()
# fixed effects model
model_fe = PanelOLS(Y, X, time_effects = True)
fe_res = model_fe.fit()
#print results
print(re_res)
print(fe_res)
```

In [ ]:

```
print('Hausman-test')
```

```
def hausman(fe_res, re_res):
    b = fe_res.params
    B = re_res.params
    v_b = fe_res.cov
    v_B = re_res.cov
    df = b[np.abs(b) < 1e8].size
    chi2 = np.dot((b - B).T, la.inv(v_b - v_B).dot(b - B))

    pval = stats.chi2.sf(chi2, df)
    return chi2, df, pval
hausman_results = hausman(fe_res, re_res)
print('chi-Squared: ' + str(hausman_results[0]))
print('degrees of freedom: ' + str(hausman_results[1]))
print('p-Value: ' + str(hausman_results[2]))
```

In [ ]:

```
###TEMPLATE WITH EXAMPLE FOR NOTRENDS##  #Dummies:
,'Miningconstruction','Foodtextileapparel','Forestpaperpublishing','Chemica
lspharma', 'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Hotelentertainment','Hospitalmanagement'
#data['Notrends2'] = data.Notrends ** 2  #'Log(SizeSales)' was omitted due
to high corralation w'Log(SizeAssets)','Log(SizeEmployees)'
X =
sm.tools.tools.add_constant(data[['Notrends','Notrends2','Log(SizeSales)','
AssetTurnover','Debtratio','TangibleAssetRatio','Foodtextileapparel','Fores
tpaperpublishing','Chemicalspharma', 'Refiningrubberplastic',
'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
```

In [ ]:

```
print('Result from Quadratic Multiple OLS with FE for time not lagged on
indirect variables with more than 60 (10*IVs) observations ')

data['Numdifftrend2'] = data.Numdifftrend ** 2 #'Log(SizeSales)' was
omitted due to high corralation w'Log(SizeAssets)','Log(SizeEmployees)'
print('For Numdifftrend')
Y1 = data['EBIT']
Y2 = data['ROA'] #'Numdifftrend2'
X =
sm.tools.tools.add_constant(data[['Numdifftrend','Numdifftrend2','Log(SizeS
ales)','AssetTurnover','Debtratio','TangibleAssetRatio','Foodtextileapparel
','Forestpaperpublishing','Chemicalspharma', 'Refiningrubberplastic',
```

```
'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)

data['Notrends2'] = data.Notrends ** 2 #'Log(SizeSales)' was omitted due to
high corralation w'Log(SizeAssets)','Log(SizeEmployees)'
print('For Notrends')
Y1 = data['EBIT']
Y2 = data['ROA'] #,'Notrends2'
X =
sm.tools.tools.add_constant(data[['Notrends','Notrends2','Log(SizeSales)','
AssetTurnover','Debtratio','TangibleAssetRatio','Foodtextileapparel','Fores
tpaperpublishing','Chemicalspharma', 'Refiningrubberplastic',
'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)

data['Noai2'] = data.Noai ** 2 #'Log(SizeSales)' was omitted due to high
corralation w'Log(SizeAssets)','Log(SizeEmployees)'
print('For Noai')
Y1 = data['EBIT']
Y2 = data['ROA'] #,'Noai2'
X =
sm.tools.tools.add_constant(data[['Noai','Noai2','Log(SizeSales)','AssetTur
nover','Debtratio','TangibleAssetRatio','Foodtextileapparel','Forestpaperpu
blishing','Chemicalspharma', 'Refiningrubberplastic',
'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
```

```
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)

data['Noautomation2'] = data.Noautomation ** 2 #'Log(SizeSales)' was
omitted due to high corralation w'Log(SizeAssets)','Log(SizeEmployees)'
print('For Noautomation')
Y1 = data['EBIT']
Y2 = data['ROA'] #'Noautomation2'
X =
sm.tools.tools.add_constant(data[['Noautomation','Noautomation2','Log(SizeS
ales)','AssetTurnover','Debtratio','TangibleAssetRatio','Foodtextileapparel
','Forestpaperpublishing','Chemicalspharma', 'Refiningrubberplastic',
'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)

data['Nosimulation2'] = data.Nosimulation ** 2 #'Log(SizeSales)' was
omitted due to high corralation w'Log(SizeAssets)','Log(SizeEmployees)'
print('For Nosimulation')
Y1 = data['EBIT']
Y2 = data['ROA'] #'Nosimulation2'
X =
sm.tools.tools.add_constant(data[['Nosimulation','Nosimulation2','Log(SizeS
ales)','AssetTurnover','Debtratio','TangibleAssetRatio','Foodtextileapparel
','Forestpaperpublishing','Chemicalspharma', 'Refiningrubberplastic',
'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)

data['Noml2'] = data.Noml ** 2 #'Log(SizeSales)' was omitted due to high
corralation w'Log(SizeAssets)','Log(SizeEmployees)'
print('For Noml')
Y1 = data['EBIT']
Y2 = data['ROA']
X =
sm.tools.tools.add_constant(data[['Noml','Noml2','Log(SizeSales)','AssetTur
nover','Debtratio','TangibleAssetRatio','Foodtextileapparel','Forestpaperpu
```

```
blishing','Chemicalspharma', 'Refiningrubberplastic',
'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)

data['Nobigdata2'] = data.Nobigdata ** 2 #'Log(SizeSales)' was omitted due
to high corralation w'Log(SizeAssets)','Log(SizeEmployees)'
print('For Nobigdata')
Y1 = data['EBIT']
Y2 = data['ROA']
X =
sm.tools.tools.add_constant(data[['Nobigdata','Nobigdata2','Log(SizeSales)'
,'AssetTurnover','Debtratio','TangibleAssetRatio','Foodtextileapparel','For
estpaperpublishing','Chemicalspharma', 'Refiningrubberplastic',
'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)

#Ai Automation Simulation Ml
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Noblockchain','Log(SizeAssets)','Log(Siz
eSales)','Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRat
io','Foodtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
```

```
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Noai','Log(SizeAssets)','Log(SizeSales)'
,'Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRatio','Foo
dtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Noiot','Log(SizeAssets)','Log(SizeSales)
','Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRatio','Fo
odtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Noar','Log(SizeAssets)','Log(SizeSales)'
,'Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRatio','Foo
dtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Noautomation','Log(SizeAssets)','Log(Siz
eSales)','Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRat
io','Foodtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Nocybersecurity','Log(SizeAssets)','Log(
SizeSales)','Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAsset
Ratio','Foodtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
```

```
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Nosimulation','Log(SizeAssets)','Log(Siz
eSales)','Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRat
io','Foodtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Noml','Log(SizeAssets)','Log(SizeSales)'
,'Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRatio','Foo
dtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
```

```
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Noiod','Log(SizeAssets)','Log(SizeSales)
','Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRatio','Fo
odtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Nomodeling','Log(SizeAssets)','Log(SizeS
ales)','Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRatio
','Foodtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
```

```
#print(ROEres)
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Norobotics','Log(SizeAssets)','Log(SizeS
ales)','Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRatio
','Foodtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

In [ ]:

```
X =
sm.tools.tools.add_constant(data[['Nocloud','Log(SizeAssets)','Log(SizeSale
s)','Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRatio','
Foodtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

In [ ]:

```
# No findings for these: 'Ios', 'Iop', 'CPS','Semantic', 'Bigdata'
X =
sm.tools.tools.add_constant(data[['Noadditive','Log(SizeAssets)','Log(SizeS
ales)','Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRatio
','Foodtextileapparel','Forestpaperpublishing','Chemicalspharma',
```

```
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

In [ ]:

```
# No findings for these: 'Ios', 'Iop', 'CPS','Semantic',
X =
sm.tools.tools.add_constant(data[['Nobigdata','Log(SizeAssets)','Log(SizeSa
les)','Log(SizeEmployees)','AssetTurnover','Debtratio','TangibleAssetRatio'
,'Foodtextileapparel','Forestpaperpublishing','Chemicalspharma',
'Refiningrubberplastic', 'Containerssteelheavy',
'Computersautosaerospace','Transportation','Telephoneutilities',
'Wholesaleretail',
'Bankfinancial','Administrationandother','Otherservices']])
Y1 = data['EBIT']
Y2 = data['ROA']
Y3 = data['ROE']
##Fixed effects model
EBITmodel = PanelOLS(Y1, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
EBITres = EBITmodel.fit() #cov_type='clustered', cluster_entity=True
print(EBITres)
ROAmodel = PanelOLS(Y2, X, time_effects = True)#, entity_effects = True)#,
time_effects = True)
ROAres = ROAmodel.fit() #cov_type='clustered', cluster_entity=True
print(ROAres)
#ROEmodel = PanelOLS(Y3, X)#, entity_effects = True)#, time_effects = True)
#ROEres = ROEmodel.fit() #cov_type='clustered', cluster_entity=True
#print(ROEres)
```

7. RegressionStockMarketReaction

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn import datasets, linear_model, metrics
from linearmodels import PooledOLS
import statsmodels.api as sm
from statsmodels.stats.diagnostic import het_white, het_breuschpagan
```

```
from statsmodels.stats.stattools import durbin_watson
from linearmodels import PanelOLS
from linearmodels import RandomEffects
import numpy.linalg as la
from scipy import stats
import csv

#Tech and randoms in two groups REAL DEAL#
randomtech = pd.read_csv("Stockmarketreactionlastfinalall.csv")
Fail=[]

V = len(randomtech)
for x in range(V):
    if randomtech.iloc[x,3] != "dummy":
        Fail.append(randomtech.iloc[x,0])
#dropping passed values
print(len(Fail))
#print(Fail)

randomtech.set_index(['ID'], inplace=True)
randomtech.drop(Fail, inplace = True)
randomtech
```

In [ ]:

```
randomtech.to_csv('Stockmarketreactioncontrol.csv', sep=',', encoding='utf-
8')
```

In [ ]:

```
datacontrol = pd.read_csv('Stockmarketreactioncontrol.csv', index_col =
['ID', 'Timex'])
data = datacontrol
datacontrol
```

In [ ]:

```
data = pd.read_csv('Stockmarketreactiontech.csv', index_col = ['ID',
'Timex'])
data
```

In [ ]:

```
# Perform PooledOLS
X = sm.tools.tools.add_constant(data['Dumevent'])
Y = data['MSreturn']
mod = PooledOLS(Y, X)
pooledOLS_res = mod.fit(cov_type='clustered', cluster_entity=True)
# Store values for checking homoskedasticity graphically
fittedvals_pooled_OLS = pooledOLS_res.predict().fitted_values
residuals_pooled_OLS = pooledOLS_res.resids
```

In [ ]:

```
# 3A. Homoskedasticity
# 3A.1 Residuals-Plot for growing Variance Detection
fig, ax = plt.subplots()
ax.scatter(fittedvals_pooled_OLS, residuals_pooled_OLS, color = 'blue')
ax.axhline(0, color = 'r', ls = '--')
ax.set_xlabel('Predicted Values', fontsize = 15)
ax.set_ylabel('Residuals', fontsize = 15)
ax.set_title('Homoskedasticity Test', fontsize = 30)
plt.show()
```

In [ ]:

```
print('White-Test')
pooled_OLS_dataset = pd.concat([data, residuals_pooled_OLS], axis=1)
#pooled_OLS_dataset = pooled_OLS_dataset.drop(['Timex'], axis =
1).fillna(0)
X = sm.tools.tools.add_constant(data['Dumevent']).fillna(0)
white_test_results = het_white(pooled_OLS_dataset['residual'], X)
labels = ['LM-Stat', 'LM p-val', 'F-Stat', 'F p-val']
print(dict(zip(labels, white_test_results)))
print('Breusch-Pagan-Test')
breusch_pagan_test_results =
het_breuschpagan(pooled_OLS_dataset['residual'], X)
labels = ['LM-Stat', 'LM p-val', 'F-Stat', 'F p-val']
print(dict(zip(labels, breusch_pagan_test_results)))
```

In [ ]:

```
# 3.B Non-Autocorrelation
print('Durbin-Watson-Test # 0-4 scale where 2 is least biased')
durbin_watson_test_results = durbin_watson(pooled_OLS_dataset['residual'])
print(durbin_watson_test_results)
```

In [ ]:

```
# FE und RE model
X = sm.tools.tools.add_constant(data['Dumevent'])
Y = data['MSreturn']
# random effects model
model_re = RandomEffects(Y, X)
re_res = model_re.fit(cov_type='robust')
# fixed effects model
model_fe = PanelOLS(Y, X, entity_effects = True)
fe_res = model_fe.fit(cov_type='robust')
#print results
print(re_res)
print(fe_res)
```

In [ ]:

```
print('Hausman-test')
def hausman(fe_res, re_res):
    b = fe_res.params
    B = re_res.params
    v_b = fe_res.cov
    v_B = re_res.cov
    df = b[np.abs(b) < 1e8].size
    chi2 = np.dot((b - B).T, la.inv(v_b - v_B).dot(b - B))

    pval = stats.chi2.sf(chi2, df)
    return chi2, df, pval
hausman_results = hausman(fe_res, re_res)
print('chi-Squared: ' + str(hausman_results[0]))
print('degrees of freedom: ' + str(hausman_results[1]))
print('p-Value: ' + str(hausman_results[2]))
```

In [ ]:

```
#For trends in general
X = sm.tools.tools.add_constant(data['Dumevent'])
Y = data['MSreturn']
# random effects model
model_re = PanelOLS(Y, X, entity_effects = True)
re_res = model_re.fit(cov_type='robust')
print(re_res)
```

In [ ]:

```
X = sm.tools.tools.add_constant(datacontrol['Dumevent'])
Y = datacontrol['MSreturn']
# random effects model
#model_re = RandomEffects(Y, X)
model_re = PanelOLS(Y, X, entity_effects = True) #Fixed effects
re_res = model_re.fit(cov_type='robust')
print(re_res)
```

In [ ]:

```
X = sm.tools.tools.add_constant(data['Dumevent']*data['Blockchain'])
Y = data['MSreturn']
# random effects model
#model_re = RandomEffects(Y, X)
model_re = PanelOLS(Y, X, entity_effects = True) #Fixed effects
re_res = model_re.fit(cov_type='robust')
print(re_res)
```

In [ ]:

```
X = sm.tools.tools.add_constant(data['Dumevent']*data['Ai'])
Y = data['MSreturn']
# random effects model
#model_re = RandomEffects(Y, X)
model_re = PanelOLS(Y, X, entity_effects = True) #Fixed effects
re_res = model_re.fit(cov_type='robust')
print(re_res)
```

In [ ]:

```
X = sm.tools.tools.add_constant(data['Dumevent']*data['Iot'])
Y = data['MSreturn']
# random effects model
#model_re = RandomEffects(Y, X)
model_re = PanelOLS(Y, X, entity_effects = True) #Fixed effects
re_res = model_re.fit(cov_type='robust')
print(re_res)
```

In [ ]:

```
X = sm.tools.tools.add_constant(data['Dumevent']*data['Ar'])
Y = data['MSreturn']
# random effects model
#model_re = RandomEffects(Y, X)
model_re = PanelOLS(Y, X, entity_effects = True) #Fixed effects
re_res = model_re.fit(cov_type='robust')
print(re_res)
```

In [ ]:

```
X = sm.tools.tools.add_constant(data['Dumevent']*data['Automation'])
Y = data['MSreturn']
# random effects model
#model_re = RandomEffects(Y, X)
model_re = PanelOLS(Y, X, entity_effects = True) #Fixed effects
re_res = model_re.fit(cov_type='robust')
print(re_res)
```

In [ ]:

```
X = sm.tools.tools.add_constant(data['Dumevent']*data['Cybersecurity'])
Y = data['MSreturn']
# random effects model
#model_re = RandomEffects(Y, X)
model_re = PanelOLS(Y, X, entity_effects = True) #Fixed effects
```

```
re_res = model_re.fit(cov_type='robust')
print(re_res)
```

In [ ]:

```
X = sm.tools.tools.add_constant(data['Dumevent']*data['Simulation'])
Y = data['MSreturn']
# random effects model
#model_re = RandomEffects(Y, X)
model_re = PanelOLS(Y, X, entity_effects = True) #Fixed effects
re_res = model_re.fit(cov_type='robust')
print(re_res)
```

In [ ]:

```
X = sm.tools.tools.add_constant(data['Dumevent']*data['Ml'])
Y = data['MSreturn']
# random effects model
#model_re = RandomEffects(Y, X)
model_re = PanelOLS(Y, X, entity_effects = True) #Fixed effects
re_res = model_re.fit(cov_type='robust')
print(re_res)
```

In [ ]:

```
X = sm.tools.tools.add_constant(data['Dumevent']*data['Additive'])
Y = data['MSreturn']
# random effects model
#model_re = RandomEffects(Y, X)
model_re = PanelOLS(Y, X, entity_effects = True) #Fixed effects
re_res = model_re.fit(cov_type='robust')
print(re_res)
```

In [ ]:

```
X = sm.tools.tools.add_constant(data['Dumevent']*data['Bigdata'])
Y = data['MSreturn']
# random effects model
#model_re = RandomEffects(Y, X)
model_re = PanelOLS(Y, X, entity_effects = True) #Fixed effects
re_res = model_re.fit(cov_type='robust')
print(re_res)
```