```
#%%
#pip install pingouin
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels import regression
import matplotlib.pyplot as plt
import sympy as sym
from pandas import ExcelWriter
from pandas import ExcelFile
from datetime import datetime, timedelta
from datetime import timedelta

# Leser inn excel filer for prisdata, tilbakekjøp og Oslo børs index (dfoseax)
df2000 = pd.read_excel('/Users/sindr/Dropbox/Master Thesis/PythonData/Manipulert_data/stock_returns00-09.xlsx')
df2010 = pd.read_excel('/Users/sindr/Dropbox/Master Thesis/PythonData/Manipulert_data/stock_returns10-18.xlsx')
dfbuyback = pd.read_excel('/Users/sindr/Dropbox/Master Thesis/PythonData/Manipulert_data/buyback10-18-Hoved.xlsx')
dfoseax = pd.read_excel('/Users/sindr/Dropbox/Master Thesis/PythonData/Manipulert_data/OSEAX95.xlsx')
dfRiskfree = pd.read_excel('/Users/sindr/Dropbox/Master Thesis/PythonData/Manipulert_data/RiskFreeRate.xlsx')

# Endrer format på alle datasett for å kunne bruke datoer videre som referanse
df2000['Date'] = pd.to_datetime(df2000['Date']).dt.date
df2010['Date'] = pd.to_datetime(df2010['Date']).dt.date
dfbuyback['payment_date'] = pd.to_datetime(dfbuyback['payment_date']).dt.date
dfoseax['Date'] = pd.to_datetime(dfoseax['Date']).dt.date
dfbuyback.rename(columns={'payment_date':'Date'}, inplace=True)
dfRiskfree['Date'] = pd.to_datetime(dfRiskfree['Date']).dt.date

# Sletter uviktige kolonner fra datasett
del df2000['Company']
del df2000['NoSharesTraded']
del df2000['Dividend']
del df2000['RiskPremium']
del df2000['SharesOutstanding']

del df2010['Company']
del df2010['NoSharesTraded']
del df2010['Dividend']
del df2010['RiskPremium']
del df2010['SharesOutstanding']

#del dfbuyback['no_shares_repurchased']
del dfbuyback['Industry']
del dfbuyback['Program']
del dfbuyback['Oppstart']
del dfbuyback['Avsluttet']
del dfbuyback['Intangibles']
del dfbuyback['Goodwill']
del dfbuyback['Total_assets']
del dfbuyback['Dummy_2']
del dfbuyback['Dummy_1']
del dfbuyback['Dummy_int']
del dfbuyback['Dummy_int_t']


# Tar bort duplicater i dfbuyback grunnet noen selskap gjennomfører 2 tilbakekjøp samme dag
#dfbuyback = dfbuyback.drop_duplicates(subset=['OBI_SEC_ID', 'Date'])
#%%

```

```
# Setter sammen prisdata for alle selskaper på Oslo Børs før og etter 2010
dfconcatall = pd.concat([df2000,df2010], sort=False)

# Legger til alle tilbakekjøp i datasettet, nå inneholder datasettet all informasjon vi trenger på hver selskap
dfmerged = pd.merge(dfconcatall, dfbuyback, on=['OBI_SEC_ID', 'Date'], how='outer')


# Slette alle selskaper som ikke har kjøpt tilbake aksjer
antall = dfbuyback[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_index(drop=True)
dfmergedbuyback = pd.merge(dfmerged, antall, on=['OBI_SEC_ID'])
dffinal = pd.merge(dfmergedbuyback, dfoseax, on = ['Date'])
# OBS: Går fra 578437 rows i dfmergedbuyback til 541295 rows når man merger på dfoseax. Mulig grunnet vi kun har datoer fra 2009 på oseax.

# Reset index for dffinal for bruk i steg 2
dffinal = dffinal.sort_values(by = ['OBI_SEC_ID', 'Date'], ascending = True).reset_index(drop = True)



# Trekker ut rows i dfbuyback med dummy lik 1
dfbuyback1 = dfbuyback.query("Dummy_uten_ansatte == 1").reset_index(drop=True)
antall_int_company = dfbuyback1[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_index(drop=True)

#%% Tilbakekjøp per selskap

test = dfbuyback1['OBI_SEC_ID'].value_counts()


test_1 = test.between(1, 1, inclusive=True).sum()
test_under_20 = test.between(2, 10, inclusive=True).sum()
test_under_50 = test.between(11, 50, inclusive=True).sum()
test_under_100 = test.between(51, 100, inclusive=True).sum()
test_over_100 = test.between(101, 750, inclusive=True).sum()

#Average repurchase before the 30-day filter
test_avg = test.sum()/len(test)

#Reelle etter 30-day filter
test_2 = dato_events['OBI_SEC_ID'].value_counts()

#Average repurchase after 30-day filter
        test_2_avg = test_2.sum()/len(test_2)

        #   Verdi tilbakekjøpt
        dfbuyback1['Verdi_tilbakekjøp'] = dfbuyback1['no_shares_repurchased'] * dfbuyback1['share price']
        totalt_tilbakekjøp = dfbuyback1['Verdi_tilbakekjøp'].sum()
        #%% Totalt tilbakekjøp fordelt på år


        #Year 2010
        start_2010 = '01-01-2010'
        start_date_2010 = datetime.strptime(start_2010, '%m-%d-%Y').date()
        slutt_2010 = '12-31-2010'
        slutt_date_2010 = datetime.strptime(slutt_2010, '%m-%d-%Y').date()
        tilbakekjøp_2010 = dfbuyback1.loc[(dfbuyback1['Date'] >= start_date_2010) & (dfbuyback1['Date'] <= slutt_date_2010)]
        totalt_tilbakekjøp_2010 = tilbakekjøp_2010['Verdi_tilbakekjøp'].sum()

        #Year 2011
        start_2011 = '01-01-2011'
        start_date_2011 = datetime.strptime(start_2011, '%m-%d-%Y').date()
```

```
        slutt_2011 = '12-31-2011'
        slutt_date_2011 = datetime.strptime(slutt_2011, '%m-%d-%Y').date()
        tilbakekjøp_2011 = dfbuyback1.loc[(dfbuyback1['Date'] >= start_date_2011) &
    (dfbuyback1['Date'] <= slutt_date_2011)]
        totalt_tilbakekjøp_2011 = tilbakekjøp_2011['Verdi_tilbakekjøp'].sum()

        #Year 2012
        start_2012 = '01-01-2012'
        start_date_2012 = datetime.strptime(start_2012, '%m-%d-%Y').date()
        slutt_2012 = '12-31-2012'
        slutt_date_2012 = datetime.strptime(slutt_2012, '%m-%d-%Y').date()
        tilbakekjøp_2012 = dfbuyback1.loc[(dfbuyback1['Date'] >= start_date_2012) &
    (dfbuyback1['Date'] <= slutt_date_2012)]
        totalt_tilbakekjøp_2012 = tilbakekjøp_2012['Verdi_tilbakekjøp'].sum()

        #Year 2013
        start_2013 = '01-01-2013'
        start_date_2013 = datetime.strptime(start_2013, '%m-%d-%Y').date()
        slutt_2013 = '12-31-2013'
        slutt_date_2013 = datetime.strptime(slutt_2013, '%m-%d-%Y').date()
        tilbakekjøp_2013 = dfbuyback1.loc[(dfbuyback1['Date'] >= start_date_2013) &
    (dfbuyback1['Date'] <= slutt_date_2013)]
        totalt_tilbakekjøp_2013 = tilbakekjøp_2013['Verdi_tilbakekjøp'].sum()

        #Year 2014
        start_2014 = '01-01-2014'
        start_date_2014 = datetime.strptime(start_2014, '%m-%d-%Y').date()
        slutt_2014 = '12-31-2014'
        slutt_date_2014 = datetime.strptime(slutt_2014, '%m-%d-%Y').date()
        tilbakekjøp_2014 = dfbuyback1.loc[(dfbuyback1['Date'] >= start_date_2014) &
    (dfbuyback1['Date'] <= slutt_date_2014)]
        totalt_tilbakekjøp_2014 = tilbakekjøp_2014['Verdi_tilbakekjøp'].sum()

        #Year 2015
        start_2015 = '01-01-2015'
        start_date_2015 = datetime.strptime(start_2015, '%m-%d-%Y').date()
        slutt_2015 = '12-31-2015'
        slutt_date_2015 = datetime.strptime(slutt_2015, '%m-%d-%Y').date()
        tilbakekjøp_2015 = dfbuyback1.loc[(dfbuyback1['Date'] >= start_date_2015) &
    (dfbuyback1['Date'] <= slutt_date_2015)]
        totalt_tilbakekjøp_2015 = tilbakekjøp_2015['Verdi_tilbakekjøp'].sum()

        #Year 2016
        start_2016 = '01-01-2016'
        start_date_2016 = datetime.strptime(start_2016, '%m-%d-%Y').date()
        slutt_2016 = '12-31-2016'
        slutt_date_2016 = datetime.strptime(slutt_2016, '%m-%d-%Y').date()
        tilbakekjøp_2016 = dfbuyback1.loc[(dfbuyback1['Date'] >= start_date_2016) &
    (dfbuyback1['Date'] <= slutt_date_2016)]
        totalt_tilbakekjøp_2016 = tilbakekjøp_2016['Verdi_tilbakekjøp'].sum()

        #Year 2017
        start_2017 = '01-01-2017'
        start_date_2017 = datetime.strptime(start_2017, '%m-%d-%Y').date()
        slutt_2017 = '12-31-2017'
        slutt_date_2017 = datetime.strptime(slutt_2017, '%m-%d-%Y').date()
        tilbakekjøp_2017 = dfbuyback1.loc[(dfbuyback1['Date'] >= start_date_2017) &
    (dfbuyback1['Date'] <= slutt_date_2017)]
        totalt_tilbakekjøp_2017 = tilbakekjøp_2017['Verdi_tilbakekjøp'].sum()

        #Year 2018
        start_2018 = '01-01-2018'
        start_date_2018 = datetime.strptime(start_2018, '%m-%d-%Y').date()
        slutt_2018 = '12-31-2018'
        slutt_date_2018 = datetime.strptime(slutt_2018, '%m-%d-%Y').date()
```

```
        tilbakekjøp_2018 = dfbuyback1.loc[(dfbuyback1['Date'] >= start_date_2018) &
    (dfbuyback1['Date'] <= slutt_date_2018)]
        totalt_tilbakekjøp_2018 = tilbakekjøp_2018['Verdi_tilbakekjøp'].sum()

        #%% Totalt tilbakekjøp fordelt på sector og selskaper fra hver sector









        #%%

        test = dffinal.loc[dffinal['OBI_SEC_ID'] == 1249672]
        test2 = df2010.loc[df2010['OBI_SEC_ID'] == 1249672]



        #%% Gjennomsnittlig int_ratio for hver sector

        sec1 = dfbuyback1.query('Dummy_sector == 1')
        sec_main1 = dato_events.query('Dummy_sector == 1')
        avg_sec1 = (sec1['Int_ratio'].sum())/len(sec1['Int_ratio'])
        sec1_sum = sec1['Verdi_tilbakekjøp'].sum()
        sec1_antall = sec1[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_index
    (drop=True)


        sec2 = dfbuyback1.query('Dummy_sector == 2')
        avg_sec2 = (sec2['Int_ratio'].sum())/len(sec1['Int_ratio'])
        sec2_sum = sec2['Verdi_tilbakekjøp'].sum()
        sec2_antall = sec2[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_index
    (drop=True)

        sec3 = dfbuyback1.query('Dummy_sector == 3')
        avg_sec3 = (sec3['Int_ratio'].sum())/len(sec1['Int_ratio'])
        sec3_sum = sec3['Verdi_tilbakekjøp'].sum()
        sec3_antall = sec3[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_index
    (drop=True)

        sec4 = dfbuyback1.query('Dummy_sector == 4')
        avg_sec4 = (sec4['Int_ratio'].sum())/len(sec1['Int_ratio'])
        sec4_sum = sec4['Verdi_tilbakekjøp'].sum()
        sec4_antall = sec4[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_index
    (drop=True)

        sec5 = dfbuyback1.query('Dummy_sector == 5')
        avg_sec5 = (sec5['Int_ratio'].sum())/len(sec1['Int_ratio'])
        sec5_sum = sec5['Verdi_tilbakekjøp'].sum()
        sec5_antall = sec5[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_index
    (drop=True)

        sec6 = dfbuyback1.query('Dummy_sector == 6')
        avg_sec6 = (sec6['Int_ratio'].sum())/len(sec1['Int_ratio'])
        sec6_sum = sec6['Verdi_tilbakekjøp'].sum()
        sec6_antall = sec6[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_index
    (drop=True)

        sec7 = dfbuyback1.query('Dummy_sector == 7')
        avg_sec7 = (sec7['Int_ratio'].sum())/len(sec1['Int_ratio'])
        sec7_sum = sec7['Verdi_tilbakekjøp'].sum()
        sec7_antall = sec7[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_index
    (drop=True)

        sec8 = dfbuyback1.query('Dummy_sector == 8')
        avg_sec8 = (sec8['Int_ratio'].sum())/len(sec1['Int_ratio'])
```

```
235.        sec8_sum = sec8['Verdi_tilbakekjøp'].sum()
236.        sec8_antall = sec8[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_index
    (drop=True)
237.
238.        sec9 = dfbuyback1.query('Dummy_sector == 9')
239.        avg_sec9 = (sec9['Int_ratio'].sum())/len(sec1['Int_ratio'])
240.        sec9_sum = sec9['Verdi_tilbakekjøp'].sum()
241.        sec9_antall = sec9[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_index
    (drop=True)
242.
243.        sec10 = dfbuyback1.query('Dummy_sector == 10')
244.        avg_sec10 = (sec10['Int_ratio'].sum())/len(sec1['Int_ratio'])
245.        sec10_sum = sec10['Verdi_tilbakekjøp'].sum()
246.        sec10_antall = sec10[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_ind
    ex(drop=True)
247.
248.        #Høyest int_ratio er sec9, sec4 og sec1
249.        #Lavest int_ratio er sec10, sec6 og sec5
250.        #%%Tilbakekjøp per sector
251.
252.        sec_main1 = dato_events.query('Dummy_sector == 1')
253.        sec_main2 = dato_events.query('Dummy_sector == 2')
254.        sec_main3 = dato_events.query('Dummy_sector == 3')
255.        sec_main4 = dato_events.query('Dummy_sector == 4')
256.        sec_main5 = dato_events.query('Dummy_sector == 5')
257.        sec_main6 = dato_events.query('Dummy_sector == 6')
258.        sec_main7 = dato_events.query('Dummy_sector == 7')
259.        sec_main8 = dato_events.query('Dummy_sector == 8')
260.        sec_main9 = dato_events.query('Dummy_sector == 9')
261.        sec_main10 = dato_events.query('Dummy_sector == 10')
262.
263.
264.
265.        #%% Finne aktuelle tilbakekjøp
266.
267.        # Extract unique values from list
268.        def unique(list1):
269.            """Returns all unique items in a list"""
270.            return list(set(list1))
271.
272.        my_events = dict()
273.        for i in dfbuyback1.OBI_SEC_ID.unique():
274.            my_events[i] = dfbuyback1.loc[dfbuyback1['OBI_SEC_ID'].isin([i])].copy()

275.
276.        unique([1,1,3])
277.        dato_events = pd.DataFrame()
278.        counter = 1
279.        unique_sec_ids = unique(dfbuyback1['OBI_SEC_ID'])
280.
281.        for obi_id in unique_sec_ids:
282.            print("Starting iteration: ", counter, " of ", len(unique_sec_ids))
283.            counter += 1
284.            df = my_events[obi_id].sort_values( by = ['Date'], ascending = True)
285.
286.            # Loop for each company
287.            used_dates = []
288.            for i, rows in df.iterrows():
289.                dato = rows['Date']
290.
291.                if dato not in used_dates:
292.                    array = df.loc[(df['Date'] <= dato + timedelta(days=30)) & (df['
    Date']>= dato)]
293.                    #Gå på index, finne neste obs etter denne datoen
294.                    dato_events = dato_events.append(rows)
295.
```

```
                    for j in range(0,30):
                        used_dates.append(dato + timedelta(days=j))

        #%%
        dato_events = dfbuyback1

        test = pd.merge(df2010,dato_events, on=['OBI_SEC_ID', 'Date'], how='outer')


        dato_events =

        for i in buyback_id1():
            common = pd.merge(dato_events,my_dict,on=['Date'])


        print(common)
        df1[(~df1.col1.isin(common.col1))&(~df1.col2.isin(common.col2))]
           col1  col2
        0     1    10
        1     2    11
        2     3    12

        #%% Antall obervasjoner  og antall observasjoner per selskap

        antall_buyback_events = len(dato_events)

        count = dato_events['OBI_SEC_ID'].value_counts()
        sum_count = count.sum()


        #%% Steg 2:

        dfbuyback_events = pd.DataFrame(columns=dffinal.columns)

        for index, rows in dato_events.iterrows():
            temp = rows.copy()
            secid = temp["OBI_SEC_ID"]
            date = temp["Date"]

            idag = dffinal[(dffinal.Date == date) & (dffinal.OBI_SEC_ID == secid)].i
    ndex
            #igar = dffinal.iloc[idag-1]
            #igar2 = dffinal.iloc[idag-2]
            #igar3 = dffinal.iloc[idag-3]
            #igar4 = dffinal.iloc[idag-4]
            #igar5 = dffinal.iloc[idag-5]
            #igar6 = dffinal.iloc[idag-6]
            #igar7 = dffinal.iloc[idag-7]
            #igar8 = dffinal.iloc[idag-8]
            #igar9 = dffinal.iloc[idag-9]
            #igar10 = dffinal.iloc[idag-10]

            imara = dffinal.iloc[idag+1]
            imara2 = dffinal.iloc[idag+2]
            #imara3 = dffinal.iloc[idag+3]
            #imara4 = dffinal.iloc[idag+4]
            #imara5 = dffinal.iloc[idag+5]
            #imara6 = dffinal.iloc[idag+6]
            #imara7 = dffinal.iloc[idag+7]
            #imara8 = dffinal.iloc[idag+8]
            #imara9 = dffinal.iloc[idag+9]
            #imara10 = dffinal.iloc[idag+10]
            #imara11 = dffinal.iloc[idag+11]
            dagens = dffinal.iloc[idag]


```

```
360.            tmp = pd.DataFrame(columns=dffinal.columns)
361.            #tmp = tmp.append(igar10, ignore_index= True)
362.            #tmp = tmp.append(igar9, ignore_index= True)
363.            #tmp = tmp.append(igar8, ignore_index= True)
364.            #tmp = tmp.append(igar7, ignore_index= True)
365.            #tmp = tmp.append(igar6, ignore_index= True)
366.            #tmp = tmp.append(igar5, ignore_index= True)
367.            #tmp = tmp.append(igar4, ignore_index= True)
368.            #tmp = tmp.append(igar3, ignore_index= True)
369.            #tmp = tmp.append(igar2, ignore_index= True)
370.            #tmp = tmp.append(igar, ignore_index= True)
371.            tmp = tmp.append(dagens, ignore_index= True)
372.            tmp = tmp.append(imara, ignore_index = True)
373.            tmp = tmp.append(imara2, ignore_index = True)
374.            #tmp = tmp.append(imara3, ignore_index = True)
375.            #tmp = tmp.append(imara4, ignore_index = True)
376.            #tmp = tmp.append(imara5, ignore_index = True)
377.            #tmp = tmp.append(imara6, ignore_index = True)
378.            #tmp = tmp.append(imara7, ignore_index = True)
379.            #tmp = tmp.append(imara8, ignore_index = True)
380.            #tmp = tmp.append(imara9, ignore_index = True)
381.            #tmp = tmp.append(imara10, ignore_index = True)
382.            #tmp = tmp.append(imara11, ignore_index = True)
383.            tmp['Event date'] = date
384.            tmp['Buyback ID'] = str(secid) + '_' + str(date)
385.
386.            dfbuyback_events = dfbuyback_events.append(tmp, ignore_index = True)
387.
388.
389.
390.        dfbuyback_events['Event_date'] = 1
391.        dfbuyback_events = dfbuyback_events.sort_values(by = ['Buyback ID'], ascendi
    ng = True)
392.        #dfbuyback_events = dfbuyback_events.drop_duplicates(subset=['OBI_SEC_ID', '
    Date'])
393.
394.
395.        #%% Tester resultater
396.
397.        test = dfbuyback_events['Buyback ID'].value_counts()
398.
399.        #Finner event som ikke er med i dato_events
400.        #test1 = dato_events[~dato_events['Date'].isin(dfbuyback_events['Date'])]
401.
402.        #%% Steg 3, Steg 4 og Steg 5:
403.
404.        # For loop som går gjennom alle unike OBI_SEC_ID fra df_merged_events, gjenn
    omfører regresjon
405.        # og lagrer på en ny dict.Nå med training 365 dager på beta estimering og al
    pha.
406.
407.        my_dict = dict()
408.        my_dict2 = dict()
409.
410.        for i in dato_events.OBI_SEC_ID.unique():
411.            my_dict[i] = dffinal.loc[dffinal['OBI_SEC_ID'].isin([i])].copy()
412.
413.        my_params = dict()
414.        my_params1 = dict()
415.        my_params2 = dict()
416.        my_resids = dict()
417.        my_autocorr = dict()
418.
419.        days_trailing = 365
420.        buyback_id = ''
421.
```

```
        buyback_id1 = []
        Num =[]
        for date, obi_id in zip(dato_events['Date'], dato_events['OBI_SEC_ID']):
            buyback_id = str(obi_id) + '_' + str(date)
            #beta
            y = my_dict[obi_id].loc[ (my_dict[obi_id]['Date']<=date- timedelta(days=10)) & (my_dict[obi_id]['Date']>=date - timedelta(days=365))]['Return']
            x = my_dict[obi_id].loc[ (my_dict[obi_id]['Date']<=date- timedelta(days=10)) & (my_dict[obi_id]['Date']>=date - timedelta(days=365))]['ReturnOSEAX']
            x = sm.add_constant(x)
            model = sm.OLS(y, x).fit()
            my_params[buyback_id] = model.params

            #Lagged beta(-1) on market returns
            y1 = my_dict[obi_id].loc[ (my_dict[obi_id]['Date']<=date- timedelta(days=10)) & (my_dict[obi_id]['Date']>=date - timedelta(days=365))]['Return']
            y1.drop(y1.tail(1).index,inplace=True)
            x1 = x.shift(-1)
            x1.drop(x1.tail(1).index,inplace=True)
            model1 = sm.OLS(y1, x1).fit()
            my_params1[buyback_id] = model1.params

            # Lead beta (1) on market returns
            y2 = my_dict[obi_id].loc[ (my_dict[obi_id]['Date']<=date- timedelta(days=10)) & (my_dict[obi_id]['Date']>=date - timedelta(days=365))]['Return']
            y2.drop(y2.head(1).index,inplace=True)
            x2 = x.shift(1)
            x2.drop(x1.head(1).index,inplace=True)
            model2 = sm.OLS(y2, x2).fit()
            my_params2[buyback_id] = model2.params

            #print("buybackid: " + buyback_id + " , antall obs: " + str(model.nobs))

            my_resids[buyback_id] = pd.DataFrame(model.params[0]+model.resid,columns={'AR'})
            my_dict2[buyback_id] = pd.merge(my_dict[obi_id],my_resids[buyback_id],left_index=True,right_index=True)
            Num1 = (np.count_nonzero(my_dict2[buyback_id]['Return']))
            Num.append(Num1)
            buyback_id1.append(buyback_id)

            t_return_oseax = my_dict2[buyback_id]['ReturnOSEAX']
            my_autocorr[buyback_id] = t_return_oseax.autocorr(1)


        return_oseax = dict()
        for k in my_params:
            return_oseax[k] = my_params[k].ReturnOSEAX


        #Konverterer liste til dataframe og ta bort tilbakekjøp med under 50% trading days

        Num = pd.DataFrame(Num, columns =['Num'])
        buyback_id1 = pd.DataFrame(buyback_id1, columns =['Buyback ID'])
        t_obs_count = pd.merge(buyback_id1, Num, left_index = True, right_index = True)

        #Tar bort observasjoner grunnet for lite data for å predikere beta
        t_obs_count_1 = t_obs_count[t_obs_count['Num'] >= 125]
        #t_obs_count_test = t_obs_count[t_obs_count['Num'] >= 125]

        # Tester hvor mange selskaper som blir tatt bort pga kravet om 125 trading days
        test_3 = pd.merge(dfbuyback_events, t_obs_count_1, on = ['Buyback ID'], how = 'inner' )
```

```
        t_obs_count_1_antall = test_3[['OBI_SEC_ID']].drop_duplicates('OBI_SEC_ID').reset_index(drop=True)

        #%% Calculating Expected Return using the market model and calculating Abnormal Return

        dfbuyback_events['AR'] = 'nan'

        t_exp = []
        t_A = []
        t_B = []
        #dfbuyback_events = dfbuyback_events.reset_index()
        #del dfbuyback_events['index']


        for bb_id, index in zip(dfbuyback_events['Buyback ID'], dfbuyback_events.index):
            t_alpha = my_params[bb_id][0]
            t_beta = (my_params[bb_id][1] + my_params1[bb_id][1] + my_params2[bb_id][1])/(1+2*my_autocorr[bb_id])
            t_mkt_ret = dfbuyback_events.loc[index]['ReturnOSEAX']
            Exp = t_alpha + t_beta * t_mkt_ret
            t_exp.append(Exp)
            #t_A.append(t_alpha)
            #t_B.append(t_beta)

        dfbuyback_events['expected_return'] = t_exp
        dfbuyback_events['AR'] = dfbuyback_events['Return'] - dfbuyback_events['expected_return']
        #dfbuyback_events['Alpha'] = t_A
        #dfbuyback_events['Beta'] = t_B

        #%% Simple beta estimation

        dfbuyback_events['AR'] = 'nan'

        t_exp = []
        t_A = []
        t_B = []
        #dfbuyback_events = dfbuyback_events.reset_index()
        #del dfbuyback_events['index']


        for bb_id, index in zip(dfbuyback_events['Buyback ID'], dfbuyback_events.index):
            t_alpha = my_params[bb_id][0]
            t_beta = my_params[bb_id][1]
            t_mkt_ret = dfbuyback_events.loc[index]['ReturnOSEAX']
            Exp = t_alpha + t_beta * t_mkt_ret
            t_exp.append(Exp)
            #t_A.append(t_alpha)
            #t_B.append(t_beta)

        dfbuyback_events['expected_return'] = t_exp
        dfbuyback_events['AR'] = dfbuyback_events['Return'] - dfbuyback_events['expected_return']
        #dfbuyback_events['Alpha'] = t_A
        #dfbuyback_events['Beta'] = t_B

        #%% Test

        bb_id="16511.0_2018-12-04"
        t_alpha = my_params[bb_id][0]
        t_beta = my_params[bb_id][1]
        t_oseax = dfbuyback_events['ReturnOSEAX'][670]

```

```
536.        t_ExReturn = t_alpha + t_beta * t_oseax
537.        t_AR1 = dfbuyback_events['Return'][670]
538.
539.        t_mkt_ret = dfbuyback_events.loc[670]['ReturnOSEAX']
540.
541.
542.        #%% Calculating averag CAR and test the significance
543.        dfbuyback_events.sort_values(by=['Buyback ID','Date'], inplace=True)
544.
545.        #Lage ny dataframe for å gjøre analyse av CAR
546.        my_CAR = dfbuyback_events[['OBI_SEC_ID','Date','Return','AR','Buyback ID']]
547.        my_CAR.sort_values(by=['Buyback ID','Date'], inplace=True)
548.        my_CAR1 = my_CAR.dropna().reset_index(drop = True)
549.
550.        # Legge sammen 3 og 3 for å finne CAR
551.        my_CAR1['CAR'] = my_CAR1.groupby(my_CAR1.index // 3)['AR'].transform('sum')[
    lambda x: ~(x.duplicated(keep='last'))]
552.        dfbuyback_events['CAR'] = dfbuyback_events.groupby(dfbuyback_events.index //
     3)['AR'].transform('sum')[lambda x: ~(x.duplicated(keep='last'))]
553.        # Extract all rows with numbers in the CAR Column
554.
555.        my_CAR1 = my_CAR1.dropna().reset_index(drop = True)
556.        my_CAR2 = pd.merge(my_CAR1, t_obs_count_1, on = ['Buyback ID'], how ='inner'
    )
557.
558.
559.        #Estimated average CAR across all firms
560.
561.            #Sum of CAR in list/ length of my_events2
562.        AvgCAR = (my_CAR2['CAR'].sum())/len(my_CAR2['CAR'])
563.
564.        #test_sumCAR = dfbuyback_events['AR'].sum()
565.        #test_avgCAR = test_sumCAR / 691
566.
567.
568.        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
569.        import pingouin as pg
570.        ttest_sample = my_CAR2['CAR']
571.        Two_sided_test = pg.ttest(x=ttest_sample,y=0)
572.
573.        #%% Filtrer to grupper med mange og få transasksjoner
574.        my_CAR4 = my_CAR2
575.
576.        #Reelle etter 30-day filter
577.        test4 = my_CAR4['OBI_SEC_ID'].value_counts()
578.        test4 = test4.reset_index()
579.        test4.columns = ['OBI_SEC_ID','Events']
580.        test5 = pd.merge(my_CAR4, test4, on = ['OBI_SEC_ID'])
581.
582.        #filtrer på antall obs
583.        High_events = test5.loc[(test5['Events'] >= 6)]
584.        High_events = High_events['OBI_SEC_ID']
585.        High_events = High_events.drop_duplicates()
586.
587.        Low_events = test5.loc[(test5['Events'] <= 5)]
588.        Low_events = Low_events['OBI_SEC_ID']
589.        Low_events = Low_events.drop_duplicates()
590.
591.        my_CAR_high = pd.merge(High_events, my_CAR4, on = ['OBI_SEC_ID'], how = 'inn
    er')
592.        my_CAR_low = pd.merge(Low_events, my_CAR4, on = ['OBI_SEC_ID'], how = 'inner
    ')
593.
594.
595.
```

```
            #Average CAR for both high and low events
        AvgCAR_high = (my_CAR_high['CAR'].sum())/len(my_CAR_high['CAR'])
        AvgCAR_low = (my_CAR_low['CAR'].sum())/len(my_CAR_low['CAR'])


        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
        import pingouin as pg

        ttest_sample = my_CAR_high['CAR']
        Two_sided_test_high = pg.ttest(x=ttest_sample,y=0)

        ttest_sample = my_CAR_low['CAR']
        Two_sided_test_low = pg.ttest(x=ttest_sample,y=0)


        #%% Ny my_CAR som inneholder både int_ratio og CAR


        my_CAR2 = dfbuyback_events[['OBI_SEC_ID','Date','Return','AR','Buyback ID','Int_ratio']]
        my_CAR2 = my_CAR2.dropna().reset_index(drop = True)
        my_CAR2 = my_CAR2[['Buyback ID', 'Int_ratio']]
        my_CAR2 = pd.merge(my_CAR2, my_CAR1, on = ['Buyback ID'], how = 'inner')
        my_CAR2 = my_CAR2.drop_duplicates(['Buyback ID'])
        my_CAR2 = pd.merge(my_CAR2, t_obs_count_1, on = ['Buyback ID'], how ='inner')


        #%% Univariate analysis using OLS with CAR(-1,1) = alpha + Beta*Int_ratio

         y = my_CAR2['CAR']
         x = my_CAR2['Int_ratio']
         x = sm.add_constant(x)
         model_int = sm.OLS(y, x).fit()
         int_model = model_int.params
         print(model_int.summary())

        Avgint = (my_CAR2['Int_ratio'].sum())/len(my_CAR2['Int_ratio'])

        #%% Beregnet CAR for alle events med både int_ratio og sectorer

        my_CAR3 = dfbuyback_events[['OBI_SEC_ID','Date','Return','AR','Buyback ID','Dummy_sector','Int_ratio']]
        t_sector = my_CAR3[['Buyback ID', 'Dummy_sector']]
        t_sector = t_sector.dropna()
        t_int_ratio = my_CAR3[['Buyback ID','Int_ratio']]
        t_int_ratio = t_int_ratio.dropna()
        t_merge = pd.merge(t_sector, t_int_ratio, on = ['Buyback ID'])
        t_merge = t_merge.drop_duplicates(['Buyback ID'])
        del my_CAR3['Dummy_sector']
        del my_CAR3['Int_ratio']
        my_CAR3['CAR'] = my_CAR3.groupby(my_CAR3.index // 3)['AR'].transform('sum')[lambda x: ~(x.duplicated(keep='last'))]
        my_CAR3 = my_CAR3.dropna()
        my_CAR3 = pd.merge(my_CAR3, t_merge, on= ['Buyback ID'], how = 'inner')
        my_CAR3 = my_CAR3.drop_duplicates(['Buyback ID'])
        my_CAR3 = pd.merge(my_CAR3, t_obs_count_1, on = ['Buyback ID'], how ='inner')


        #%% Hypotesis test for top 3 high int_ratio sectors

        t_high9 = my_CAR3.query('Dummy_sector == 6')
        t_high1 = my_CAR3.query('Dummy_sector == 9')
        t_high4 = my_CAR3.query('Dummy_sector == 3')
```

```
657.        t_high5 = my_CAR3.query('Dummy_sector == 2')
658.
659.
660.
661.        high_int = pd.concat([t_high9,t_high1], sort=False)
662.        high_int = pd.concat([high_int,t_high4], sort=False)
663.        high_int = pd.concat([high_int,t_high5], sort=False)
664.
665.
666.        avg_high = (high_int['CAR'].sum())/len(high_int['CAR'])
667.
668.        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
669.        ttest_sample_high = high_int['CAR']
670.        Two_sided_test_high = pg.ttest(x=ttest_sample_high,y=0)
671.
672.        #%% Hypotesis test for top 3 low int_ratio sectors
673.
674.        t_low6 = my_CAR3.query('Dummy_sector == 10')
675.        t_low10 = my_CAR3.query('Dummy_sector == 4')
676.        t_low5 = my_CAR3.query('Dummy_sector == 8')
677.        t_low8 = my_CAR3.query('Dummy_sector == 7')
678.
679.        low_int = pd.concat([t_low6,t_low10], sort=False)
680.        low_int = pd.concat([low_int,t_low5], sort=False)
681.        low_int = pd.concat([low_int, t_low8], sort=False)
682.
683.        avg_low = (low_int['CAR'].sum())/len(low_int['CAR'])
684.
685.        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
686.        ttest_sample_low = low_int['CAR']
687.        Two_sided_test_low = pg.ttest(x=ttest_sample_low,y=0)
688.
689.        #%% Test
690.
691.        t_high1 = my_CAR3.query('Dummy_sector == 1')
692.        #t_high2 = my_CAR3.query('Dummy_sector == 2')
693.        #t_high3 = my_CAR3.query('Dummy_sector == 3')
694.        #t_high4 = my_CAR3.query('Dummy_sector == 4')
695.        #t_high5 = my_CAR3.query('Dummy_sector == 5')
696.        #t_high6 = my_CAR3.query('Dummy_sector == 6')
697.        #t_high7 = my_CAR3.query('Dummy_sector == 7')
698.        #t_high8 = my_CAR3.query('Dummy_sector == 8')
699.        #t_high9 = my_CAR3.query('Dummy_sector == 9')
700.        #t_high10 = my_CAR3.query('Dummy_sector == 10')
701.
702.        high_int = t_high1
703.        #high_int = pd.concat([t_high1,t_high2], sort=False)
704.        #high_int = pd.concat([high_int,t_high3], sort=False)
705.        #high_int = pd.concat([high_int,t_high4], sort=False)
706.        #high_int = pd.concat([high_int,t_high5], sort=False)
707.        #high_int = pd.concat([high_int,t_high6], sort=False)
708.        #high_int = pd.concat([high_int,t_high7], sort=False)
709.        #high_int = pd.concat([high_int,t_high8], sort=False)
710.        #high_int = pd.concat([high_int,t_high9], sort=False)
711.        #high_int = pd.concat([high_int,t_high10], sort=False)
712.
713.
714.        avg_high = (high_int['CAR'].sum())/len(high_int['CAR'])
715.
716.        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
717.        ttest_sample_high = high_int['CAR']
718.        Two_sided_test_high = pg.ttest(x=ttest_sample_high,y=0)
719.
720.        #%%
721.        my_CAR.rename(columns={'Buyback ID':'bb_id'}, inplace=True)
722.        my_CAR_dict = dict()
```

```
    for i in my_CAR.bb_id.unique():
        my_CAR_dict[i] = my_CAR.loc[my_CAR['bb_id'].isin([i])].copy()

    for k in my_CAR.bb_id.unique():
        del my_CAR_dict[k]['Return']
        del my_CAR_dict[k]['OBI_SEC_ID']
        del my_CAR_dict[k]['Date']
        del my_CAR_dict[k]['bb_id']

    #%% AR for -10
    list_neg10 = []
    for k in my_CAR.bb_id.unique():
        tmp = my_CAR_dict[k].iloc[0]
        list_neg10.append(tmp)

    neg10 = pd.DataFrame(list_neg10)

    avg_neg10 = (neg10['AR'].sum())/len(neg10['AR'])

    # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
    ttest_sample_neg10 = neg10['AR']
    Two_sided_test_neg10 = pg.ttest(x=ttest_sample_neg10,y=0)

    #%% AR for -9
    list_neg9 = []
    for k in my_CAR.bb_id.unique():
        tmp = my_CAR_dict[k].iloc[1]
        list_neg9.append(tmp)

    neg9 = pd.DataFrame(list_neg9)

    avg_neg9 = (neg9['AR'].sum())/len(neg9['AR'])

    # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
    ttest_sample_neg9 = neg9['AR']
    Two_sided_test_neg9 = pg.ttest(x=ttest_sample_neg9,y=0)

    #%% AR for -8
    list_neg8 = []
    for k in my_CAR.bb_id.unique():
        tmp = my_CAR_dict[k].iloc[2]
        list_neg8.append(tmp)

    neg8 = pd.DataFrame(list_neg8)

    avg_neg8 = (neg8['AR'].sum())/len(neg8['AR'])

    # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
    ttest_sample_neg8 = neg8['AR']
    Two_sided_test_neg8 = pg.ttest(x=ttest_sample_neg8,y=0)

    #%% AR for -7
    list_neg7 = []
    for k in my_CAR.bb_id.unique():
        tmp = my_CAR_dict[k].iloc[3]
        list_neg7.append(tmp)

    neg7 = pd.DataFrame(list_neg7)

    avg_neg7 = (neg7['AR'].sum())/len(neg7['AR'])

    # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
    ttest_sample_neg7 = neg7['AR']
    Two_sided_test_neg7 = pg.ttest(x=ttest_sample_neg7,y=0)

    #%% AR for -6
```

```
789.        list_neg6 = []
790.        for k in my_CAR.bb_id.unique():
791.            tmp = my_CAR_dict[k].iloc[4]
792.            list_neg6.append(tmp)
793.
794.        neg6 = pd.DataFrame(list_neg6)
795.
796.        avg_neg6 = (neg6['AR'].sum())/len(neg6['AR'])
797.
798.        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
799.        ttest_sample_neg6 = neg6['AR']
800.        Two_sided_test_neg6 = pg.ttest(x=ttest_sample_neg6,y=0)
801.
802.        #%% AR for -5
803.        list_neg5 = []
804.        for k in my_CAR.bb_id.unique():
805.            tmp = my_CAR_dict[k].iloc[5]
806.            list_neg5.append(tmp)
807.
808.        neg5 = pd.DataFrame(list_neg5)
809.
810.        avg_neg5 = (neg5['AR'].sum())/len(neg5['AR'])
811.
812.        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
813.        ttest_sample_neg5 = neg5['AR']
814.        Two_sided_test_neg5 = pg.ttest(x=ttest_sample_neg5,y=0)
815.
816.        #%% AR for -4
817.        list_neg4 = []
818.        for k in my_CAR.bb_id.unique():
819.            tmp = my_CAR_dict[k].iloc[6]
820.            list_neg4.append(tmp)
821.
822.        neg4 = pd.DataFrame(list_neg4)
823.
824.        avg_neg4 = (neg4['AR'].sum())/len(neg4['AR'])
825.
826.        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
827.        ttest_sample_neg4 = neg4['AR']
828.        Two_sided_test_neg4 = pg.ttest(x=ttest_sample_neg4,y=0)
829.
830.        #%% AR for -3
831.        list_neg3 = []
832.        for k in my_CAR.bb_id.unique():
833.            tmp = my_CAR_dict[k].iloc[7]
834.            list_neg3.append(tmp)
835.
836.        neg3 = pd.DataFrame(list_neg3)
837.
838.        avg_neg3 = (neg3['AR'].sum())/len(neg3['AR'])
839.
840.        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
841.        ttest_sample_neg3 = neg3['AR']
842.        Two_sided_test_neg3 = pg.ttest(x=ttest_sample_neg3,y=0)
843.
844.        #%% AR for -2
845.        list_neg2 = []
846.        for k in my_CAR.bb_id.unique():
847.            tmp = my_CAR_dict[k].iloc[8]
848.            list_neg2.append(tmp)
849.
850.        neg2 = pd.DataFrame(list_neg2)
851.
852.        avg_neg2 = (neg2['AR'].sum())/len(neg2['AR'])
853.
854.        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
```

```
        ttest_sample_neg2 = neg2['AR']
        Two_sided_test_neg2 = pg.ttest(x=ttest_sample_neg2,y=0)

        #%% AR for -1
        list_neg1 = []
        for k in my_CAR.bb_id.unique():
            tmp = my_CAR_dict[k].iloc[9]
            list_neg1.append(tmp)

        neg1 = pd.DataFrame(list_neg1)

        avg_neg1 = (neg1['AR'].sum())/len(neg1['AR'])

        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
        ttest_sample_neg1 = neg1['AR']
        Two_sided_test_neg1 = pg.ttest(x=ttest_sample_neg1,y=0)

        #%% AR for 0
        list_neg0 = []
        for k in my_CAR.bb_id.unique():
            tmp = my_CAR_dict[k].iloc[10]
            list_neg0.append(tmp)

        neg0 = pd.DataFrame(list_neg0)

        avg_neg0 = (neg0['AR'].sum())/len(neg0['AR'])

        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
        ttest_sample_neg0 = neg0['AR']
        Two_sided_test_neg0 = pg.ttest(x=ttest_sample_neg0,y=0)

        #%% AR for 1
        list_pos1 = []
        for k in my_CAR.bb_id.unique():
            tmp = my_CAR_dict[k].iloc[11]
            list_pos1.append(tmp)

        pos1 = pd.DataFrame(list_pos1)

        avg_pos1 = (pos1['AR'].sum())/len(pos1['AR'])

        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
        ttest_sample_pos1 = pos1['AR']
        Two_sided_test_pos1 = pg.ttest(x=ttest_sample_pos1,y=0)

        #%% AR for 2
        list_pos2 = []
        for k in my_CAR.bb_id.unique():
            tmp = my_CAR_dict[k].iloc[12]
            list_pos2.append(tmp)

        pos2 = pd.DataFrame(list_pos2)

        avg_pos2 = (pos2['AR'].sum())/len(pos2['AR'])

        # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
        ttest_sample_pos2 = pos2['AR']
        Two_sided_test_pos2 = pg.ttest(x=ttest_sample_pos2,y=0)

        #%% AR for 3
        list_pos3 = []
        for k in my_CAR.bb_id.unique():
            tmp = my_CAR_dict[k].iloc[13]
            list_pos3.append(tmp)

        pos3 = pd.DataFrame(list_pos3)
```

```
921.
922.       avg_pos3 = (pos3['AR'].sum())/len(pos3['AR'])
923.
924.       # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
925.       ttest_sample_pos3 = pos3['AR']
926.       Two_sided_test_pos3 = pg.ttest(x=ttest_sample_pos3,y=0)
927.
928.       #%% AR for 4
929.       list_pos4 = []
930.       for k in my_CAR.bb_id.unique():
931.           tmp = my_CAR_dict[k].iloc[14]
932.           list_pos4.append(tmp)
933.
934.       pos4 = pd.DataFrame(list_pos4)
935.
936.       avg_pos4 = (pos4['AR'].sum())/len(pos4['AR'])
937.
938.       # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
939.       ttest_sample_pos4 = pos4['AR']
940.       Two_sided_test_pos4 = pg.ttest(x=ttest_sample_pos4,y=0)
941.
942.       #%% AR for 5
943.       list_pos5 = []
944.       for k in my_CAR.bb_id.unique():
945.           tmp = my_CAR_dict[k].iloc[15]
946.           list_pos5.append(tmp)
947.
948.       pos5 = pd.DataFrame(list_pos5)
949.
950.       avg_pos5 = (pos5['AR'].sum())/len(pos5['AR'])
951.
952.       # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
953.       ttest_sample_pos5 = pos5['AR']
954.       Two_sided_test_pos5 = pg.ttest(x=ttest_sample_pos5,y=0)
955.
956.       #%% AR for 6
957.       list_pos6 = []
958.       for k in my_CAR.bb_id.unique():
959.           tmp = my_CAR_dict[k].iloc[16]
960.           list_pos6.append(tmp)
961.
962.       pos6 = pd.DataFrame(list_pos6)
963.
964.       avg_pos6 = (pos6['AR'].sum())/len(pos6['AR'])
965.
966.       # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
967.       ttest_sample_pos6 = pos6['AR']
968.       Two_sided_test_pos6 = pg.ttest(x=ttest_sample_pos6,y=0)
969.
970.       #%% AR for 7
971.       list_pos7 = []
972.       for k in my_CAR.bb_id.unique():
973.           tmp = my_CAR_dict[k].iloc[17]
974.           list_pos7.append(tmp)
975.
976.       pos7 = pd.DataFrame(list_pos7)
977.
978.       avg_pos7 = (pos7['AR'].sum())/len(pos7['AR'])
979.
980.       # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
981.       ttest_sample_pos7 = pos7['AR']
982.       Two_sided_test_pos7 = pg.ttest(x=ttest_sample_pos7,y=0)
983.
984.       #%% AR for 8
985.       list_pos8 = []
986.       for k in my_CAR.bb_id.unique():
```

```
            tmp = my_CAR_dict[k].iloc[18]
            list_pos8.append(tmp)

    pos8 = pd.DataFrame(list_pos8)

    avg_pos8 = (pos8['AR'].sum())/len(pos8['AR'])

    # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
    ttest_sample_pos8 = pos8['AR']
    Two_sided_test_pos8 = pg.ttest(x=ttest_sample_pos8,y=0)

    #%% AR for 9
    list_pos9 = []
    for k in my_CAR.bb_id.unique():
        tmp = my_CAR_dict[k].iloc[19]
        list_pos9.append(tmp)

    pos9 = pd.DataFrame(list_pos9)

    avg_pos9 = (pos9['AR'].sum())/len(pos9['AR'])

    # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
    ttest_sample_pos9 = pos9['AR']
    Two_sided_test_pos9 = pg.ttest(x=ttest_sample_pos9,y=0)

    #%% AR for 10
    list_pos10 = []
    for k in my_CAR.bb_id.unique():
        tmp = my_CAR_dict[k].iloc[20]
        list_pos10.append(tmp)

    pos10 = pd.DataFrame(list_pos10)

    avg_pos10 = (pos10['AR'].sum())/len(pos10['AR'])

    # PerformTwo-sided  t-test with H0: CAR=0 H1: CAR not equal 0
    ttest_sample_pos10 = pos10['AR']
    Two_sided_test_pos10 = pg.ttest(x=ttest_sample_pos10,y=0)
```