In [28]:
```python
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
import re
import yfinance as yf
import pandas_market_calendars as mcal
import random
import sweetviz as sv

from sklearn import linear_model
from sklearn.model_selection import train_test_split

import plotly.express as px
```

In [3]:
```python
df = pd.read_csv('Assets\CONVICTIONSUMMARY_270_opt3_best2(1).csv', sep='|')
```

In [4]:
```python
df2 = pd.DataFrame()
df2 = df.loc[df['TYPE'] == 'MERGED']
```

Extract Time and Date

In [5]:
```python
df2["Time"] = df2.apply(lambda row: re.split(" ",row.DATE)[0], axis=1)

df2["DATE"] = df2.apply(lambda row: re.split(" ",row.DATE)[-1],axis=1)
```

```
<ipython-input-5-0de3e1e305d2>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  df2["Time"] = df2.apply(lambda row: re.split(" ",row.DATE)[0], axis=1)
<ipython-input-5-0de3e1e305d2>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  df2["DATE"] = df2.apply(lambda row: re.split(" ",row.DATE)[-1],axis=1)
```

In [6]:
```python
df2.dtypes
```

Out[6]:
```
DATE              object
ID                object
STOCK             object
SECTOR            object
TYPE              object
CONVICTIONS      float64
Time              object
dtype: object
```

In [7]:
```python
df2['DATE'] = pd.to_datetime(
                          df2['DATE'],
                          format='%Y-%m-%d')
```

```
<ipython-input-7-3660468bee21>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  df2['DATE'] = pd.to_datetime(
```

In [8]:
```python
df2 = df2.sort_values(by="DATE")
```

Cleaning

In [14]:
```python
df2.columns = ['DATE', 'ID', 'STOCK',"SECTOR","TYPE","SCORE","TIME"]
```

In [15]:
```python
df2['SCORE']=df2['SCORE'].replace(0, np.nan)
```

In [79]:
```python
df2 = df2.loc[df2['SCORE'] != 0]
```

Adding returns

gettting info

In [21]:
```python
list_of_companies = df2["STOCK"].unique()
list_of_dates = df2["DATE"].unique()
```

In [22]:
```python
start_date = df2.DATE.min()
end_date = df2.DATE.max()
```

In [82]:
```python
final_date = end_date + pd.DateOffset(months=12)
print(final_date)
```

```
2005-07-14 00:00:00
```

Define functions

In [113…
```python
def get_history_data(ticker_symbol, start_date, end_date):
    tickerData = yf.Ticker(ticker_symbol)
    tickerDf = tickerData.history(period='1d', start=start_date, end=end_date)
    tickerDf["STOCK"] = ticker_symbol

    return tickerDf
```

In [84]:
```python
def get_closest_trading_day(day,trading_days):
    trading_days = trading_days[trading_days>=pd.to_datetime(day, utc=True)]
    return trading_days.sort_values()[0]
```

In [112…
```python
def get_traning_days(start_date, end_date,stock_market_name):
    nyse = mcal.get_calendar(stock_market_name)
    valid_days = nyse.valid_days(start_date=start_date, end_date=final_date)
    return valid_days
```

In [111…
```python
def get_return_on_date(date,df):
  try:
    x = df.loc[date]
    x = float(x['Close'])
  except:
    x = 0.0
  return x
```

In [87]:
```python
def calculate_returns(start_date, df, trading_days):
    returns = pd.DataFrame(columns=['DATE','STOCK','1MReturn','3MReturn','6MRerturn','12MReturn'])

    date1m = get_closest_trading_day(start_date + pd.DateOffset(months=1),trading_days)
    date3m = get_closest_trading_day(start_date + pd.DateOffset(months=3),trading_days)
    date6m = get_closest_trading_day(start_date + pd.DateOffset(months=6),trading_days)
    date12m = get_closest_trading_day(start_date + pd.DateOffset(months=12),trading_days)

    name = df['STOCK']
    try:
        current = df.loc[start_date]
    except:
        return returns
    data1m = get_return_on_date(date1m,df)
    data3m = get_return_on_date(date3m,df)
    data6m = get_return_on_date(date6m,df)
    data12m = get_return_on_date(date12m,df)

    current = float(current['Close'])

    return1m = (data1m - current)/current
    return3m = (data3m - current)/current
    return6m = (data6m - current)/current
    return12m = (data6m - current)/current


    returns = pd.DataFrame([[start_date,name[0],return1m, return3m, return6m,return12m]], columns=['DATE','STOC



    return returns
```

In [88]:
```python
def calculate_returns_for_date(list_of_dates,df,trading_days):
    returns = pd.DataFrame(columns=['DATE','STOCK','1MReturn','3MReturn','6MReturn','12MReturn'])
    for d in list_of_dates:
        temp = calculate_returns(pd.to_datetime(d),df,trading_days)

        returns = returns.append(temp, ignore_index=True)

    return returns
```

In [89]:
```python
def calculate_returns_for_companies(list_of_dates,list_of_companies,start_date, end_date):
    returns = pd.DataFrame(columns=['DATE','STOCK','1MReturn','3MReturn','6MReturn','12MReturn'])
    trading_days = get_traning_days(start_date,end_date, "NYSE")
    for c in list_of_companies:
        c = c.replace(".XX1","")
        c = c.replace(".XX2","")
        data = get_history_data(c,start_date, end_date)
        len_of_data = data.shape[0]
        if len_of_data !=0:
            returns_for_company = calculate_returns_for_date(list_of_dates,data,trading_days)
            returns = returns.append(returns_for_company, ignore_index=True)

    return returns
```

Get Data

In [182…
```python
test = calculate_returns_for_companies(list_of_dates, list_of_companies[1:100],start_date, end_date)
```

```
- PKZ: No data found for this date range, symbol may be delisted
- PALDF: No data found, symbol may be delisted
- CHRS: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- AGU: No data found for this date range, symbol may be delisted
- BJS: No data found for this date range, symbol may be delisted
- AAN: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- NMG.B: No data found for this date range, symbol may be delisted
- HBG: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- SSINQ: No data found, symbol may be delisted
- DSCP: No data found for this date range, symbol may be delisted
- BRK.B: No data found, symbol may be delisted
- EPL: No data found for this date range, symbol may be delisted
- APCC.XX9: No data found, symbol may be delisted
- PP.XX9: No data found, symbol may be delisted
- INET1: No data found, symbol may be delisted
- SYD: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- SKO: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- SWFT: No data found for this date range, symbol may be delisted
- BNI: No data found for this date range, symbol may be delisted
- TBL: No data found for this date range, symbol may be delisted
- USG: No data found, symbol may be delisted
- COL: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- MTLQQ: No data found for this date range, symbol may be delisted
- ASCA: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- SPOK: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- NMG.A: No data found, symbol may be delisted
- OUTR: No data found for this date range, symbol may be delisted
- ANN: No data found for this date range, symbol may be delisted
- BGPIQ: No data found, symbol may be delisted
- OMM: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- CEPH: No data found for this date range, symbol may be delisted
- DLM: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- EFII: No data found, symbol may be delisted
- NEV: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- BRK.A: No data found, symbol may be delisted
- SEM.XX9: No data found, symbol may be delisted
- BRY: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- PETM: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- AXE: No data found, symbol may be delisted
- RSHCQ: No data found for this date range, symbol may be delisted
- MTLG: No data found for this date range, symbol may be delisted
- LABH: No data found, symbol may be delisted
- AOT: No data found for this date range, symbol may be delisted
- BRP: Data doesn't exist for startDate = 1076454000, endDate = 1089756000
- GTM: No data found for this date range, symbol may be delisted
- TRK: No data found, symbol may be delisted
- EDMC: No data found for this date range, symbol may be delisted
```

In [183…
```python
test.head()
```

Out[183…

	DATE	STOCK	1MReturn	3MReturn	6MReturn	12MReturn	6MRerturn
0	2004-02-11	TTEK	-0.071624	-0.238595	-1.0	-1.0	NaN
1	2004-02-18	TTEK	-0.040246	-0.233428	-1.0	-1.0	NaN
2	2004-02-25	TTEK	-0.017266	-0.203357	-1.0	-1.0	NaN
3	2004-03-03	TTEK	0.073337	-0.151044	-1.0	-1.0	NaN
4	2004-03-10	TTEK	-0.006816	-0.154820	-1.0	-1.0	NaN

In [184…
```python
new_df = pd.merge(df2, test,  how='right', left_on=['DATE','STOCK'], right_on = ['DATE','STOCK'])
```

In [185…
```python
new_df.head()
```

Out[185…

	DATE	ID	STOCK	SECTOR	TYPE	SCORE	TIME	1MReturn	3MReturn	6MReturn	12MReturn	6MRerturn
0	2004-02-11	V60F1Z-R	TTEK	Industrial Services	MERGED	0.320429	10:01:54.408	-0.071624	-0.238595	-1.0	-1.0	NaN
1	2004-02-18	V60F1Z-R	TTEK	Industrial Services	MERGED	0.313523	10:01:55.720	-0.040246	-0.233428	-1.0	-1.0	NaN
2	2004-02-25	V60F1Z-R	TTEK	Industrial Services	MERGED	0.321246	10:01:56.309	-0.017266	-0.203357	-1.0	-1.0	NaN
3	2004-03-03	V60F1Z-R	TTEK	Industrial Services	MERGED	0.320851	10:01:57.220	0.073337	-0.151044	-1.0	-1.0	NaN
4	2004-03-10	V60F1Z-R	TTEK	Industrial Services	MERGED	0.322088	10:01:58.763	-0.006816	-0.154820	-1.0	-1.0	NaN

Aditional data

Simply Regression

1 month

In [166…
```python
regression_data = new_df.filter(['SCORE',"1MReturn"],axis=1)
regression_data.dropna(inplace = True)
X = regression_data.filter(['SCORE'], axis=1)
Y = regression_data.filter(['1MReturn'], axis=1)

lm = linear_model.LinearRegression()
```

In [167…
```python
model = lm.fit(X,Y)

print("Intercept: \n", lm.intercept_)
print("Coefficients: \n", lm.coef_)
```

```
Intercept:
 [-0.27868712]
Coefficients:
 [[0.52452624]]
```

In [168…
```python
predictions = lm.predict(X)
```

In [169…
```python
lm.score(X,Y)
```

Out[169…
```
0.019749343191932667
```

3 months

In [186…
```python
regression_data = new_df.filter(['SCORE',"3MReturn","STOCK"],axis=1)
regression_data.dropna(inplace = True)

companies_in_sample = regression_data["STOCK"].unique()

companies_train, companies_test = train_test_split(companies_in_sample, test_size=0.2,random_state =2)
```

In [189…
```python
train_data = regression_data[regression_data['STOCK'].isin(companies_train)]
test_data = regression_data[regression_data['STOCK'].isin(companies_test)]
```

In [190…
```python
X_train = train_data.filter(['SCORE'], axis=1)
Y_train = train_data.filter(['3MReturn'], axis=1)

X_test = test_data.filter(["SCORE"],axis=1)
Y_train - test_data.filter(["3MReturn"],axis=1)


lm3 = linear_model.LinearRegression()
```

In [191…
```python
model = lm3.fit(X_train,Y_train)

print("Intercept: \n", lm3.intercept_)
print("Coefficients: \n", lm3.coef_)
```

```
Intercept:
 [-0.37580189]
Coefficients:
 [[-0.38539599]]
```

In [193…
```python
lm3.score(X_train,Y_train)
```

Out[193…
```
0.007003022434370121
```

In [ ]:

Exploratory analysis

In [60]:
```python
sweet_report = sv.analyze(df2)
```

```
Done! Use 'show' commands to display/save.   |██████████| [100%]   00:00 -> (00:00 left)
```

In [61]:
```python
sweet_report.show_html('sweet_report.html')
```

```
Report sweet_report.html was generated! NOTEBOOK/COLAB USERS: the web browser MAY not pop up, regardless, the r
eport IS saved in your notebook/colab files.
```

23% of all scores are nulls

many companies dont have historical stock prices

In [40]:
```python
stock_with_score_by_date = df2[df2['SCORE'].notna()].groupby('DATE')['STOCK'].nunique()
stock_with_score_by_date = stock_with_score_by_date.to_frame()
```

In [45]:
```python
stock_with_dates_count = df2[df2['SCORE'].notna()].groupby('STOCK')['DATE'].nunique()
stock_with_dates_count = stock_with_dates_count.to_frame()
```

In [38]:
```python
type(stock_with_score_by_date)
```

Out[38]:
```
pandas.core.series.Series
```

In [44]:
```python
fig = px.line(stock_with_score_by_date,title="number of Stocks in dates")
fig.update_xaxes(
    dtick="M1",rangeslider_visible=True)
fig.show()
```

In [56]:
```python
stocks_in_full_timeline = stock_with_dates_count [stock_with_dates_count.values ==  23]
```

In [59]:
```python
print("the percage of companies that appear in full timeline is equal to", len(stocks_in_full_timeline)/len(lis
```

```
the percage of companies that appear in full timeline is equal to 0.20259019426456984
```

In [67]:
```python
sectors_size = df2.groupby(['SECTOR']).size()
sectors_size = sectors_size.to_frame()
```

In [73]:
```python
sectors_size.reset_index(inplace=True)
sectors_size = sectors_size.rename(columns = {'index':'Count'})
```

In [74]:
```python
sectors_size.columns
```

Out[74]:
```
Index(['Count', 'SECTOR', 0], dtype='object')
```

In [76]:
```python
fig = px.pie(sectors_size, values='Count', names='SECTOR', title='Sectors representations')
fig.show()
```

In [ ]: