ANC Project Report on Glassdoor

Author :
M. Zakaria Iraqi

September 29, 2020

Contents

Introduction

In this report, we will be discussing how to use efficiently glassdoor data in order to extract a signal, that will help us do some feature selection and select stocks accordingly.

We will start by describing how to collect and webscrape data, from various databases, then, once we are done extracting the data, we will be using a signal that we will be predefining that will help us in our stock selection process.

Chapter 1

Data collection

We will start by collecting our datas from various sources, abd show how to do so, especially for Glassdoor

1.1 S&P 500 stocks

First of all, we want to collect the stocks that will be used in our study, we chose to work with S&P 500 current stocks division, although It may not make sense to just use current relavant stocks for the past years, finding S&P 500 data yearly appeared to be difficult.

We will be using a public dataset to store the data, this data can be found through https://datahub.io/core/s-and-p-500-companies/datapackage.json.

The Notebooks joined to this file contain a way to quickly import It.

See https://colab.research.google.com/drive/1narp1bD$_ZDI0Ly8cV1qVxHyNod1QVQLm?usp=$ *sharing* for instance. Here is a caption of the stocks :

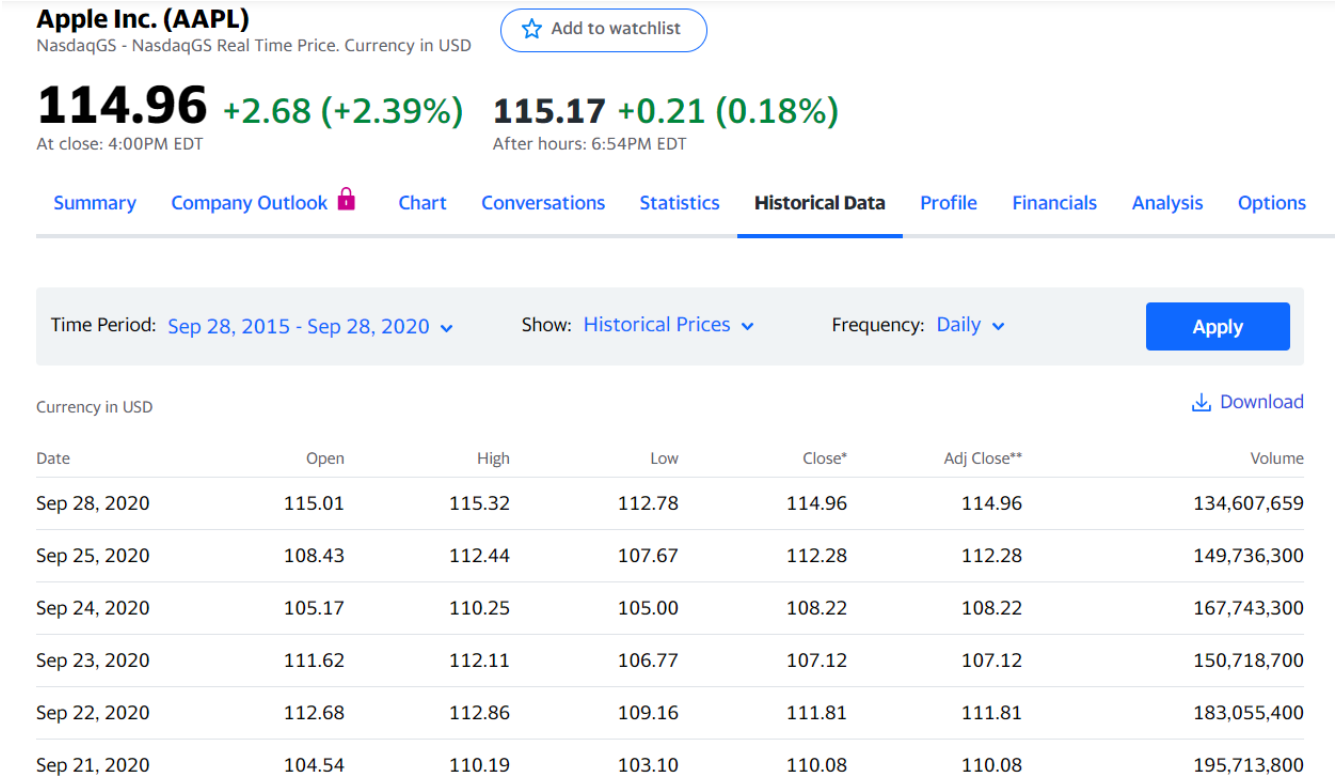
	Symbol	Name	Sector
0	MMM	3M Company	Industrials
1	AOS	A.O. Smith Corp	Industrials
2	ABT	Abbott Laboratories	Health Care
3	ABBV	AbbVie Inc.	Health Care
4	ABMD	ABIOMED Inc	Health Care

Figure 1.1: S&P 500 snapshot

1.2 Yahoo Finance data

Yahoo Finance is one of the most easy-to-use and user friendly data providers for financial markets, this ease to use will be the main incentive behind using It.

Here is a caption of the Yahoo finance data for Apple(AAPL)



Apple Inc. (AAPL)
NasdaqGS - NasdaqGS Real Time Price. Currency in USD
Add to watchlist

114.96 +2.68 (+2.39%) At close: 4:00PM EDT
115.17 +0.21 (0.18%) After hours: 6:54PM EDT

Summary | Company Outlook | Chart | Conversations | Statistics | **Historical Data** | Profile | Financials | Analysis | Options

Time Period: Sep 28, 2015 - Sep 28, 2020 Show: Historical Prices Frequency: Daily Apply

Currency in USD Download

Date	Open	High	Low	Close*	Adj Close**	Volume
Sep 28, 2020	115.01	115.32	112.78	114.96	114.96	134,607,659
Sep 25, 2020	108.43	112.44	107.67	112.28	112.28	149,736,300
Sep 24, 2020	105.17	110.25	105.00	108.22	108.22	167,743,300
Sep 23, 2020	111.62	112.11	106.77	107.12	107.12	150,718,700
Sep 22, 2020	112.68	112.86	109.16	111.81	111.81	183,055,400
Sep 21, 2020	104.54	110.19	103.10	110.08	110.08	195,713,800

Figure 1.2: AAPL data in Yahoo Finance website

We notice a download button, that will give us the data for the alst 5 years as a csv; let us use It efficiently.

The implementation of this, in order to save the data to a folder is done in:

https://colab.research.google.com/drive/1narp1bD$_Z$DI0Ly8cV1qVxHyNod1QVQLm?usp = *sharing*

This colab file will save the stock data to a given google drive folder prselected by the user; It can be run either locally or online.

1.3 Glassdoor data

1.3.1 Glassdoor overview

Glassdoor is a website where current and former employees anonymously review companies. Glassdoor also allows users to anonymously submit and view salaries as well as search and apply for jobs on its platform.

A stock page for a given company have this outlook:

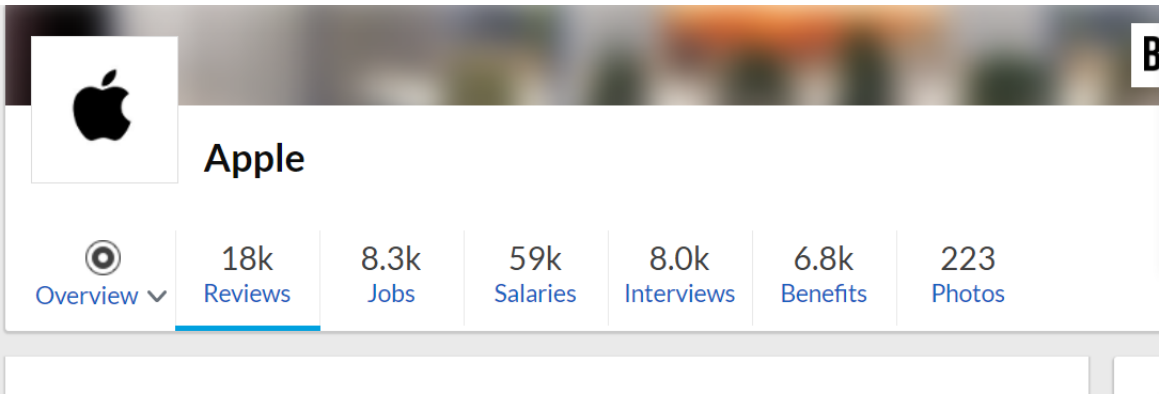


Figure 1.3: Apple review page in glassdoor

In this caption, we find, for a given company, the number of reviews, and some macro details about It.

When we scroll through the webpage, we get he overall rating of the compnay and Its CEO approval

If one wants to go through a stock's review, he may decide to sort them by date, and go through all the webpages to collect this data to use It as a signal.

Here is a caption of a given testimony:

From this testimony, we can extract various factors and comments; but the most important are :

- The overall review
- The number of people who found It helpful
- The date of the review
- The business outlook

Some of these datas are not shown here, but are given through webscrapig the website.

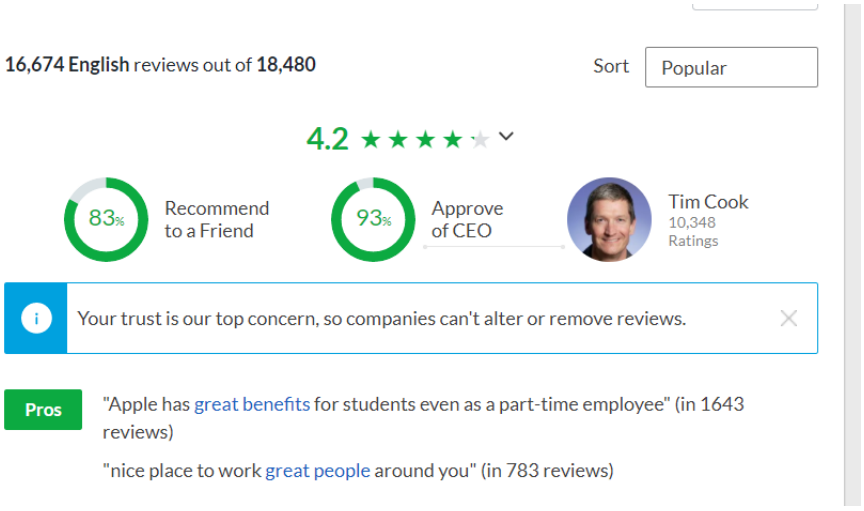


Figure 1.4: Apple metrics on glassdoor

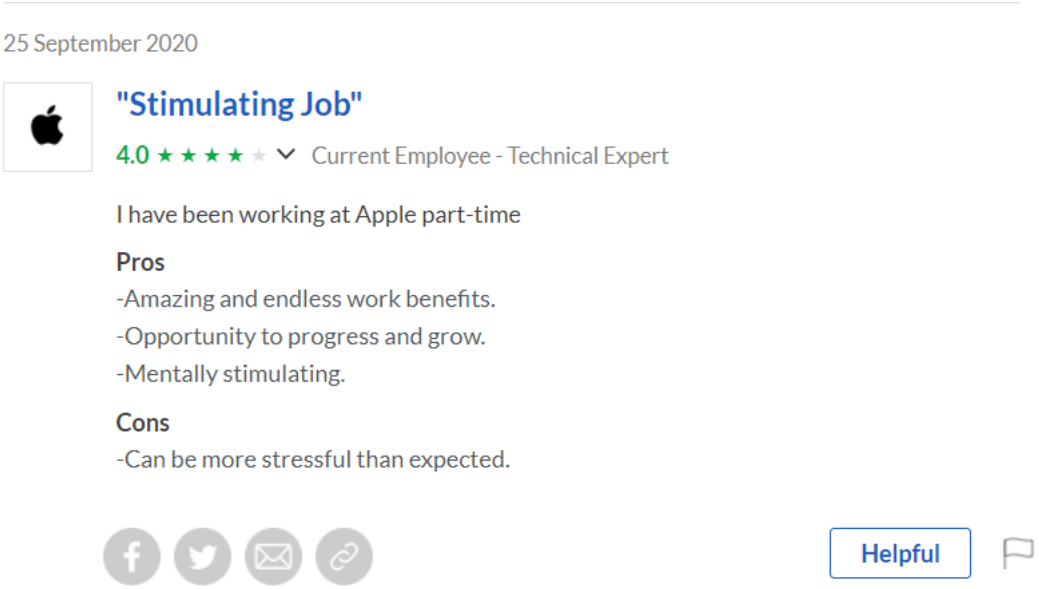


Figure 1.5: Apple example of testimony

1.3.2 Webscraping

Here is the process we will be doing to extract and save the data from glassdoor

- Go to Google.com, and look for "<Stock company's name> glassdoor review".
- Find the relevant URL.
- Process the first page and detect overall statistics.

- Browse through all the webpages, after filtering the reviews descendingly, and stopping when we reach the 5 years limit of our study, for each step, save all the relevant data in folders

This process can be done by using the colab notebooks available and linked to this file; unfortunately, they have to be run locally(even though the save folder can be external), since Colab is using a firewall.

To do so, we will use various Python packages, such as urllib, and also use chromedriver extensively. We can also multithread the process. This is what I have done in my local machine, and It required 48 hours of runtime to save the overall number of testimonies (found in folder /testimonies on the joint zip file), and saved as much as 5 times in speed.

See implementation of both :

Glassdoor overall statistics saver:

https://colab.research.google.com/drive/1ynKaqHV2zdHvJTrJUL3fFc2F$_U qLxJwX?usp = sharing$

Glassdoor detailed testimonies saver:

https://colab.research.google.com/drive/1MFQ2thHHjKz2qn1jsXcBoBTBM3cxNxuM?usp=sharing

Once we are done collecting, and testing out the data, we will be applying a strategy and processing It through a signal coming from these previous datas.

Chapter 2

Data Processings

In this chapter, we will be processing the data obtained from the previous webscraping in order to create a signal that will be discussed later on

2.1 The signal

Let us consider a stock S, for which we saved a set of testimonies.

Let us consider impact(t) the value of the testimony given at time t; This impact can be the overall rating, the business overview from the reviewer's standpoint, or some other factor.

Let us weight our impacts with the parameter importance, giving more importance to a testimony that was deemed interesting by other users.

Let us consider a decay factor λ, which will determine how much we value current testimonies compared to previous one(in default, we set $\lambda = 1$

$importance(testimony) = 3 + \#upvotes(testimony) - \#downvotes(testimony)$

We thus can consider, for a given vote at time t, a signal that will determine the overall voting value of a stock, which will be defined as:

$signal(t) = \frac{\sum_{t \geq t_i} impact(t_i) e^{-\lambda \times (t-t_i)} importance(t_i))}{\sum_{t \geq t_i} e^{-\lambda \times (t-t_i)} importance(t_i)}$.

In order to avoid numerical issues, we define :

$factor(t) = \frac{1}{\sum_{t \geq t_i} e^{-\lambda \times (t-t_i)} importance(t_i)}$

By considering t' the first time preceeding t, we thus have:

$signal(t) = signal(t') e^{-\lambda \times (t-t')} + impact(t) \times factor(t) \times importance(t)$

$factor(t) = \frac{1}{\frac{1}{factor(t')} \times e^{-\lambda \times (t-t')} + importance(t)}$

Both signals defined above are equal, but, numerically speaking, the one in the last part is less affected by high values, and is thus more stable.

For instance, if one considers the impact factor to be the overall rating, we have the following result for Amazon: .

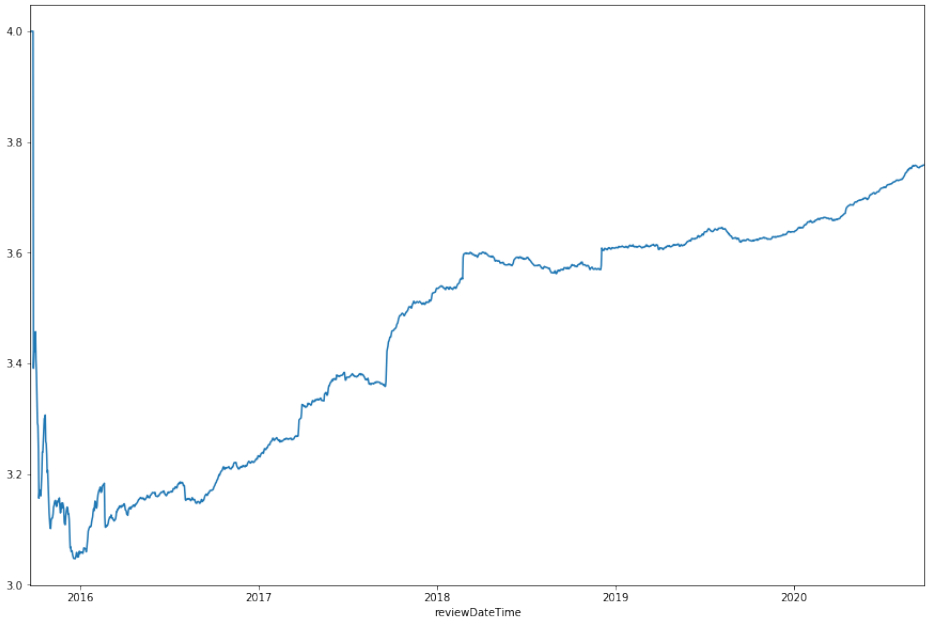


Figure 2.1: Amazon testimony signal

Now that we have processed the datas and transformed It to signals, we will shift them in one time step, to avoid data leakage; this is an important thing to do too.

We can then proceed to merge all the signals into one file, for readability and processing purposes. We also may be feeling the voids with a forward fill: If no new review was made in the last days, then stick to the last signal. Just for completion, we fill the first empty lines using a backward fill.

The process mentioned above is done in the following jupyter notebook joined to this work Here is an overview of the table of signals:

Here is a snap of the code for the signal creation:

reviewDateTime	A	AAL	AAP	AAPL	ABBV	ABC	ABMD	ABT	ACN	ADBE	...	XLNX	XOM	XRAY	XRX	XYL
2015-09-20	3.290203	3.714286	3.540516	3.722222	4.494445	3.21815	3.362337	4.0	1.75	3.446491	...	3.192771	3.5	3.002468	4.0	1.551891
2015-09-21	3.290203	3.714286	3.540516	3.722222	4.494445	3.21815	3.362337	4.0	1.75	3.446491	...	3.192771	3.5	3.002468	4.0	1.551891
2015-09-22	3.290203	3.714286	3.540516	3.722222	4.494445	3.21815	3.362337	4.0	1.75	3.446491	...	3.192771	3.5	3.002468	4.0	1.551891
2015-09-23	3.290203	3.714286	3.540516	3.722222	4.494445	3.21815	3.362337	4.0	1.75	3.446491	...	3.192771	3.5	3.002468	4.0	1.521964
2015-09-24	3.290203	2.964528	3.570991	3.722222	4.494445	3.21815	3.362337	4.0	1.75	3.446491	...	3.192771	3.5	3.002468	4.0	1.521964

Figure 2.2: Overview of merged signals from testimonies

```
testimonies_folder = "./testimonies"
files = os.listdir(testimonies_folder)
```

```
def process_data(file, decay_weight= 1):

    data = pd.read_csv(file)
    data.reviewDateTime = data.apply(lambda x: parser.parse(x["reviewDateTime"]).date(), axis=1)
    data = data.sort_values(by=['reviewDateTime'], ascending=True)
    start_date = min(data['reviewDateTime'])
    #data["deltaTime"] = data.reviewDateTime - start_date
    #data["decay"] = data.apply(lambda x: x["deltaTime"].total_seconds() / (3600* 24* 30), axis=1)
    data["valuation"] = data["ratingOverall"]
    data["importance"] = (3 - data["countNotHelpful"] + data["countHelpful"])
    data["crossVal"] = data["valuation"] *  data["importance"]
    signal = []
    factor = []
    data = data[["reviewDateTime", "importance", "crossVal"]]

    for index, row in data.iterrows():
        date = row["reviewDateTime"]
        importance= row["importance"]
        try :
            delta = (date - previous_date).total_seconds()  / (3600* 24* 30 * 12) * decay_weight
            current_factor =  exp(-delta)
            new_factor = 1/ ((1/factor[-1]) * current_factor + importance)
            new_signal = signal[-1]*current_factor * new_factor / factor[-1]  + row["crossVal"] * new_factor
            factor.append(new_factor)
            signal.append(new_signal)
            previous_date = date
            deltas.append(current_factor)
        except:
            left_parts = [0]
            factor = [1/importance]
            signal = [row["crossVal"] / row['importance']]
            deltas = [1]
            previous_date = date
    data = data.set_index(["reviewDateTime"])
    data["impact"] = signal
    data = data[["impact"]]
    data = data.groupby("reviewDateTime").mean()
    return data
```

```
datas = []
for stock in files:
    #print("processing stock %s"%(stock))
    data = process_data(testimonies_folder + "/"+ stock, decay_weight= 1)
    data.rename(columns={"impact": stock[:-4]}, inplace=True)
    datas.append(data)
df_merged = reduce(lambda  left,right: pd.merge(left,right,on=['reviewDateTime'],
                                            how='outer'), datas)
df_merged = df_merged[df_merged.index>(datetime.today() - timedelta(days=5 * 365 + 10)).date()].reset_index()
df_merged = df_merged.sort_values(by=['reviewDateTime'], ascending=True).set_index(['reviewDateTime'])
df= df_merged.fillna(method="ffill")
df= df.fillna(method="bfill") # for initial values
df.head()
```

Figure 2.3: Snapshot of code for signal generation

2.2 The strategy

The strategy that will be used is the following:

$$\alpha(Company, Category, date) = \begin{cases} 1: & \text{go long} \\ 0: & \text{do nothing} \\ -1: & \text{go short} \end{cases} \tag{2.1}$$

Basically, given a fixed stock sector, we will be longing our N best rated companies, and shorting the lowest N rated companies.

Here is a snapshot of the stock allocation: The stock allocation is given in the jupyter

Top1	Top2	Top3	Top4	Top5	Top6	Top7	Top8	Top9	Top10	date	Bottom1	Bottom2	Bottom3	Bottom4	Bottom5	Bottom6	Bottom7	Bottom8	Bottom9	Bottom10
DIS	FB	T	EA	NFLX	TWTR	ATVI	GOOG	TTWO	FOXA	20/09/2015	LYV	GOOGL	OMC	NWS	NWSA	CTL	DISH	IPG	DISCA	DISCK
DIS	FB	T	EA	NFLX	TWTR	ATVI	GOOG	TTWO	FOXA	21/09/2015	LYV	GOOGL	OMC	NWS	NWSA	CTL	DISH	IPG	DISCA	DISCK
DIS	FB	T	EA	TWTR	NFLX	ATVI	GOOG	TTWO	FOXA	22/09/2015	LYV	GOOGL	OMC	NWS	NWSA	CTL	DISH	IPG	DISCA	DISCK
DIS	FB	T	EA	TWTR	NFLX	ATVI	GOOG	TTWO	FOXA	23/09/2015	LYV	GOOGL	OMC	NWS	NWSA	DISH	CTL	IPG	DISCA	DISCK
DIS	FB	T	EA	TWTR	NFLX	ATVI	GOOG	TTWO	FOXA	24/09/2015	LYV	GOOGL	OMC	NWS	NWSA	DISH	CTL	IPG	DISCA	DISCK
DIS	FB	T	TWTR	EA	NFLX	ATVI	GOOG	TTWO	FOXA	25/09/2015	LYV	GOOGL	OMC	CMCSA	NWS	NWSA	DISH	IPG	DISCA	DISCK
DIS	T	TWTR	EA	FB	NFLX	ATVI	GOOG	TTWO	FOXA	26/09/2015	LYV	GOOGL	OMC	DISH	CMCSA	NWS	NWSA	IPG	DISCA	DISCK
T	TWTR	EA	FB	DIS	NFLX	ATVI	GOOG	TTWO	FOXA	27/09/2015	LYV	GOOGL	OMC	DISH	NWS	NWSA	CMCSA	IPG	DISCA	DISCK
T	TWTR	FB	DIS	NFLX	EA	ATVI	TTWO	GOOG	FOXA	28/09/2015	GOOGL	OMC	LYV	DISH	NWS	NWSA	CHTR	CMCSA	IPG	DISCA
T	TWTR	FB	DIS	NFLX	ATVI	TTWO	GOOG	FOXA	FOX	29/09/2015	GOOGL	DISH	OMC	LYV	NWS	NWSA	CHTR	CMCSA	IPG	DISCA
T	TWTR	FB	NFLX	ATVI	TTWO	GOOG	DIS	FOX	FOXA	30/09/2015	GOOGL	OMC	LYV	DISH	NWS	NWSA	CMCSA	IPG	DISCA	DISCK
T	TWTR	FB	DIS	NFLX	GOOG	ATVI	TTWO	FOXA	FOX	01/10/2015	GOOGL	LYV	OMC	DISH	NWS	NWSA	CMCSA	IPG	CTL	CHTR
T	TWTR	FB	DIS	GOOG	NFLX	ATVI	TTWO	FOXA	FOX	02/10/2015	GOOGL	LYV	DISH	OMC	NWS	NWSA	CMCSA	IPG	CTL	VIAC
T	TWTR	FB	GOOG	DIS	NFLX	ATVI	TTWO	FOXA	FOX	03/10/2015	GOOGL	LYV	DISH	OMC	NWS	NWSA	CMCSA	IPG	CTL	CHTR
T	TWTR	FB	GOOG	NFLX	ATVI	TTWO	DIS	FOXA	FOX	04/10/2015	GOOGL	OMC	LYV	DISH	NWS	NWSA	CMCSA	CTL	IPG	CHTR
T	TWTR	FB	GOOG	TTWO	NFLX	DIS	FOXA	FOX	EA	05/10/2015	GOOGL	OMC	LYV	DISH	NWS	NWSA	CMCSA	CTL	IPG	CHTR
T	TWTR	FB	TTWO	NFLX	GOOG	FOXA	FOX	EA	DIS	06/10/2015	GOOGL	OMC	LYV	NWS	NWSA	CMCSA	DISH	CTL	IPG	CHTR
T	TWTR	FB	TTWO	NFLX	GOOG	FOXA	FOX	EA	DIS	07/10/2015	GOOGL	OMC	LYV	CMCSA	NWSA	NWS	CTL	IPG	DISH	CHTR
T	TWTR	FB	TTWO	NFLX	GOOG	FOXA	FOX	EA	DIS	08/10/2015	GOOGL	OMC	LYV	CMCSA	NWS	NWSA	CTL	IPG	DISH	VZ
T	TWTR	FB	TTWO	NFLX	GOOG	FOX	FOXA	EA	DIS	09/10/2015	GOOGL	OMC	LYV	CMCSA	NWS	NWSA	CTL	DISCA	DISCK	IPG
T	TWTR	FB	TTWO	NFLX	GOOG	FOX	FOXA	EA	DIS	10/10/2015	GOOGL	OMC	LYV	CMCSA	NWS	NWSA	CTL	DISCA	DISCK	IPG
T	TWTR	FB	TTWO	NFLX	GOOG	FOX	FOXA	EA	DIS	11/10/2015	GOOGL	OMC	LYV	NWS	NWSA	CMCSA	IPG	CTL	DISCA	DISCK
T	FB	TWTR	TTWO	NFLX	GOOG	FOX	FOXA	EA	DIS	12/10/2015	GOOGL	OMC	NWS	NWSA	LYV	CMCSA	IPG	DISCA	DISCK	CTL
T	FB	TWTR	NFLX	TTWO	GOOG	FOX	FOXA	EA	TMUS	13/10/2015	GOOGL	OMC	NWS	NWSA	CMCSA	LYV	IPG	CTL	DISCA	DISCK
T	FB	TWTR	NFLX	TTWO	GOOG	FOX	FOXA	EA	TMUS	14/10/2015	GOOGL	OMC	NWS	NWSA	CMCSA	LYV	CTL	IPG	DISCA	DISCK

Figure 2.4: Communication services daily alpha allocator

notebook "ANC Project: using testimonies as a trading signal"

We also show later on a snapshot of the alpha selector:

2.2.1 Strategy explanation

Let us assume that we are on our first day of this strategy, that the stocks can be reduced to two separate stocks : S_1 : the high rating stock, which will be longed, and S_2, the one with the low rating. Let us assume our position starts at time 0, And we can buy as infinitesimal bits of the stock as we want, this is an important assumption.

- At the beginning of the day, we will borrow 1$ by selling 1$'s worth of S_2,let's call It $x(S_2, 0)$(which we don't have), and then buying with that amount \$1$ $'s worth of S_1, ,let's call It $x(S_1, 0)$ - At the end of the day, we close our position: sell back $x(S_1, 0)$ of S_1, and buy back $x(S_2, 0)$ of shares of S_2, so that we can give back the $x(S_2, 0)$ of shares of S_2 that we borrowed and sold.

Saving stocks historic data per category, and detecting most and least performant stocks:

In [42]:
```
stocks_folder = "./stocks/"
sectorwise_stock_folder = "./sectorwise_stock/"
files = os.listdir(stocks_folder)
stocks_stored_set = set([file[:-4] for file in files])
sp500 = pd.read_csv("sp500.csv")
sectors = {}
for _, bit in sp500.groupby("Sector"):
    sectors[bit["Sector"].iloc[0]] = set(bit["Symbol"].unique())

data = pd.read_csv("processed_testimonies.csv")
data.reviewDateTime = data.apply(lambda x: parser.parse(x["reviewDateTime"]).date(), axis=1)

overall_data= {}
allocation_sector = {}
data_columns = set(data.columns)
for sector in sectors.keys():
    columns = sectors[sector].intersection(data_columns)
    overall_data[sector] =  data[columns]
    tops =pd.DataFrame((overall_data[sector]).apply(lambda x:list(overall_data[sector].columns[np.array(x).argsort()\
                                        [::-1][:10]]), axis=1).to_list(),  columns=['Top' + str(k) for k in range(1, 11)])
    tops["date"] = data["reviewDateTime"]
    bottoms =pd.DataFrame( (-overall_data[sector]).apply(lambda x:list(overall_data[sector].columns[np.array(x).argsort()\
                                    [::-1][:10]]), axis=1).to_list(),  columns=['Bottom' + str(k) for k in range(1, 11)])
    bottoms["date"] = data["reviewDateTime"]
    overall = pd.merge(tops, bottoms, how="outer")
    allocation_sector[sector] = overall

folder = "./allocation_sector/"
for sector in allocation_sector.keys():
    allocation_sector[sector].to_csv(folder+ sector+".csv")
allocation_sector[sector].head()
```

Out[42]:

	Top1	Top2	Top3	Top4	Top5	Top6	Top7	Top8	Top9	Top10	...	Bottom1	Bottom2	Bottom3	Bottom4	Bottom5	Bottom6	Bottom7	Bottom8	Botton
0	EVRG	ATO	LNT	EXC	D	ED	NI	PPL	SRE	AEP	...	WEC	AWK	EIX	ES	AEE	SO	PEG	NEE	DU
1	EVRG	ATO	LNT	EXC	D	ED	NI	PPL	SRE	AEP	...	WEC	AWK	EIX	ES	AEE	SO	PEG	NEE	DU
2	EVRG	ATO	LNT	EXC	D	ED	NI	PPL	SRE	AEP	...	WEC	AWK	EIX	ES	AEE	SO	PEG	NEE	DU
3	EVRG	ATO	LNT	EXC	D	ED	NI	PPL	SRE	AEP	...	WEC	AWK	EIX	ES	AEE	SO	PEG	NEE	DU
4	EVRG	ATO	LNT	EXC	D	ED	NI	PPL	SRE	AEP	...	WEC	AWK	EIX	ES	AEE	SO	NEE	DUK	NR

5 rows × 21 columns

Figure 2.5: stocks selector algorithm

Our Pnl in that particular day is thus :

$$Pnl(day, 1\$) = x(S_1, 0) \times S_1(Close) - x(S_2, 0) \times S_2(Close) \tag{2.2}$$

$$Pnl(day, 1\$) = 1\$ \times \frac{S_1(Close)}{S_1(Open)} - 1\$ \times \frac{S_2(Close)}{S_2(Open)} \tag{2.3}$$

$$Pnl(day, 1\$) = 1\$ \times ((\frac{S_1(Close)}{S_1(Open)} - 1) - (\frac{S_2(Close)}{S_2(Open)} - 1)) \tag{2.4}$$

$$Pnl(day, 1\$) = 1\$ \times (\frac{S_1(Close) - S_1(Open)}{S_1(Open)} - \frac{S_2(Close) - S_2(Open)}{S_2(Open)}) \tag{2.5}$$

Our cumulative Pnl will thus be:

$$Pnl(Year, K\$) = K \times \Pi_{day \in Year}(1 + Pnl(day, 1\$)) \tag{2.6}$$

2.2.2 Strategy implementation

Following are a snapshot of the daily and monthly $P\&L$ algorithm, ector wise pnl allocation in a daily basis, sector wise pnl allocation in a monthly basis and in a quarterly basis too, And when we combine all stocks to select an index.

We leave the data and the rest of the analysis in the jupyter notebook "ANC Project, Using testimonies as a trading signal"

We leave the data and the rest of the analysis in the jupyter notebook "ANC Project, Using testimonies as a trading signal"

```
daily_positions = {}
for file in files:
    print("processing sector "+ file[:-4])
    allocation = pd.read_csv(folder_allocation + file)
    stocks = pd.read_csv(folder_stocks + file)
    allocation.date = allocation.apply(lambda x: parser.parse(x["date"]).date(), axis=1)
    stocks.Date = stocks.apply(lambda x: parser.parse(x["Date"]).date(), axis=1)
    stocks.rename(columns={"Date":"date"}, inplace=True)
    stocks.set_index("date", inplace=True)
    allocation.set_index("date", inplace=True)
    positions = []
    for row_stock in stocks.iterrows():
        date = row_stock[0]

        counter = 0
        try:
            current_allocation = allocation.loc[row_stock[0] - timedelta(days=counter)]
            columns_top = ["Top%s"%(s) for s in range(1,11) ]
            columns_bot = ["Bottom%s"%(s) for s in range(1,11) ]
            tops = list(current_allocation[columns_top])
            bots = list(current_allocation[columns_bot])
            position_long = 0
            position_short = 0
            null_top = 0
            null_bot = 0

            for top in tops:
                try:
                    value = (row_stock[1]["Close_%s"%(top)] / row_stock[1]["Open_%s"%(top)]) - 1
                    if pd.isnull(value):
                        null_top +=1
                    else :
                        position_long += value
                except: null_top+=1
            for bot in bots:
                try:

                    value = (row_stock[1]["Close_%s"%(bot)] / row_stock[1]["Open_%s"%(bot)]) - 1
                    if pd.isnull(value):
                        null_bot +=1
                    else :
                        position_short += value
                except:
                    null_bot+=1


            try:
                position_long /= (10 - null_top)
                position_short /= (10 - null_bot)
                positions.append([date, position_long, position_short])
            except:
                continue

        except:
            if counter > 365 * 6:
                break
            counter+= 1
    positions = pd.DataFrame(positions, columns= ["date", "long_position", "short_position"])
    positions["daily_pnl"] = positions["long_position"] - positions["short_position"]
    positions["cumulative_pnl"] =  (1 + positions["daily_pnl"]).cumprod()
    positions.set_index("date", inplace=True)
```

Figure 2.6: Daily $P\&L$ algorithm

Monthly allocation P&L:

Now, instead of taking orders daily, we proceed on a buy and hold, sell and wait for stocks, and we refresh each month, taking again the stocks that have high testimony, by selling those wiht low testimonies

In [10]:
```
def get_allocations(date, stocks_columns):
    counter = 0
    try:
        current_allocation = allocation.loc[row_stock[0] - timedelta(days=counter)]
        columns_top = ["Top%s"%(s) for s in range(1,11) ]
        columns_bot = ["Bottom%s"%(s) for s in range(1,11)]
        tops = list(current_allocation[columns_top])
        tops = [s for s in tops  if s in stocks_columns]
        bots = list(current_allocation[columns_bot])
        bots = [s for s in bots  if s in stocks_columns]
        return tops, bots
    except:
        if counter > 365:
            return None
```

In [23]:
```
monthly_positions = {}
for file in files:
    print("processing sector "+ file[:-4])
    allocation = pd.read_csv(folder_allocation + file)
    stocks = pd.read_csv(folder_stocks + file)
    allocation.date = allocation.apply(lambda x: parser.parse(x["date"]).date(), axis=1)
    stocks.Date = stocks.apply(lambda x: parser.parse(x["Date"]).date(), axis=1)
    stocks.rename(columns={"Date":"date"}, inplace=True)
    stocks.set_index("date", inplace=True)
    allocation.set_index("date", inplace=True)
    positions = []

    previous_date = None

    positions =  []

    previous_long_positions = []
    previous_short_positions = []
    for row_stock in stocks.iterrows():
        date = row_stock[0]

        if not previous_date:
            positions.append([date, date, 0,0])
            date_allocation = get_allocations(date, stocks_columns)
            if date_allocation :
                tops, bots = date_allocation
            else :
                continue
            previous_date = date
            previous_long_position = [row_stock[1]["Open_%s"%(top)] for top in tops]
            previous_short_position = [row_stock[1]["Open_%s"%(bot)] for bot in bots]
            continue
        else :

            if date < previous_date + timedelta(days=30) or get_allocations(date, stocks_columns) is None:
                continue
            current_long_position = [row_stock[1]["Open_%s"%(top)] for top in tops]
            current_short_position = [row_stock[1]["Open_%s"%(bot)] for bot in bots]

            value_long = [current_long_position[k]/previous_long_position[k] - 1 for k in range(len(current_long_position))]
            value_short = [current_short_position[k]/previous_short_position[k] - 1 for k in range(len(current_short_position)
            long_position = 0
            short_position = 0
            nan_long = 0
            nan_short = 0
            for element in value_long:
                if pd.isnull(element):
                    nan_long+=1
                else :
                    long_position+=element
            for element in value_short:
                if pd.isnull(element):
                    nan_short+=1
                else :
                    short_position+=element
            if nan_long ==len(tops) or nan_short == len(bots):
                continue
            long_position = long_position/(len(tops)  - nan_long)
            short_position = short_position/(len(bots) - nan_short)
            positions.append([previous_date, date, long_position, short_position])
            tops, bots = get_allocations(date, stocks_columns)
            previous_long_position = [row_stock[1]["Open_%s"%(top)] for top in tops]
            previous_short_position = [row_stock[1]["Open_%s"%(bot)] for bot in bots]
            previous_date = date

    positions = pd.DataFrame(positions, columns= ["start_date", "end_date", "long_position", "short_position"])
    positions["daily_pnl"] = positions["long_position"] - positions["short_position"]
    positions["cumulative_pnl"] = (1 + positions["daily_pnl"]).cumprod()
```

Figure 2.7: Monthly $P\&L$ algorithm

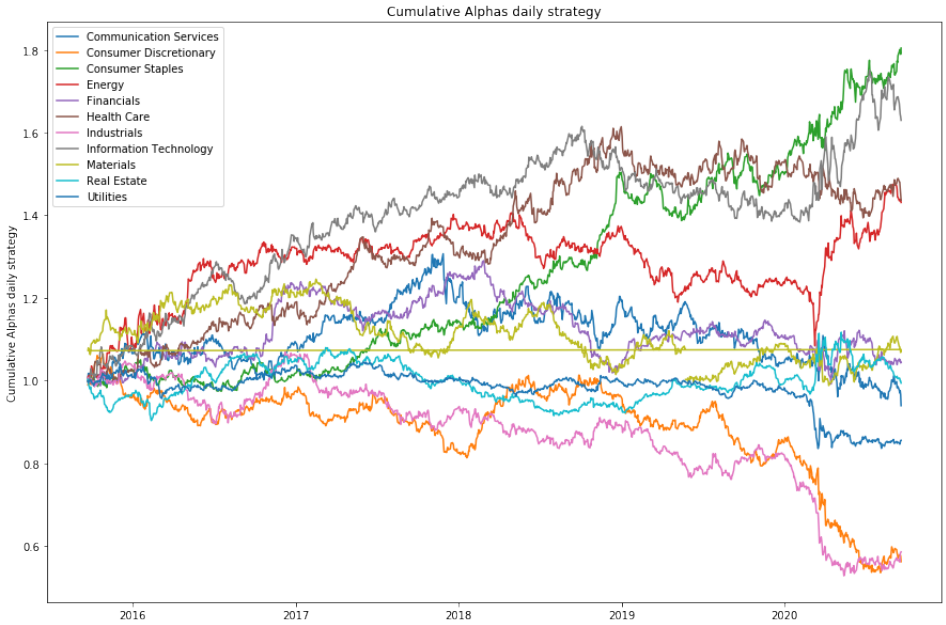


Figure 2.8: Daily strategy cumulative $P\&L$, the $P\&L$ does not start at 1 since there is transactions and revenues starting from the first day

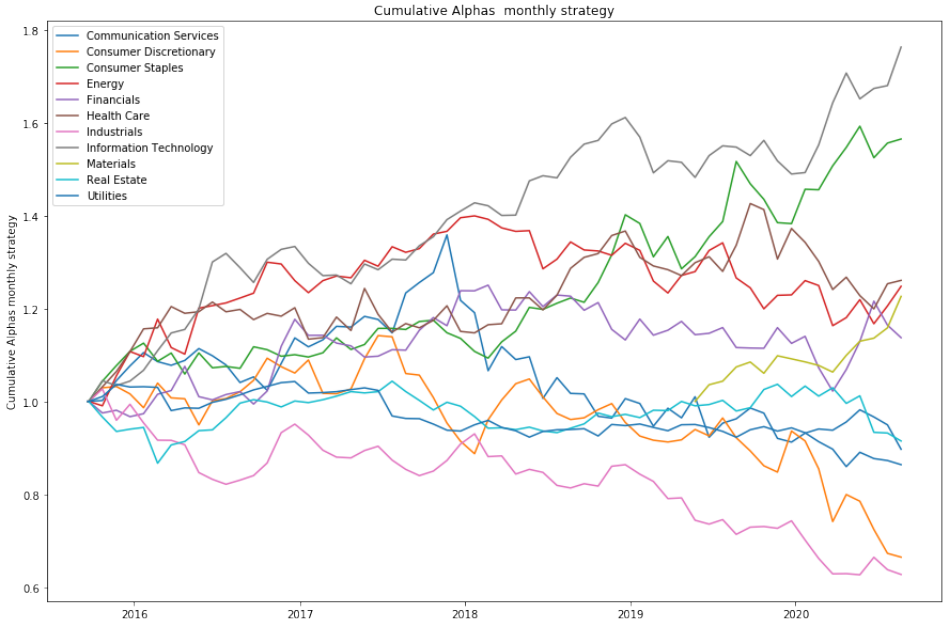


Figure 2.9: Monthly strategy cumulative $P\&L$

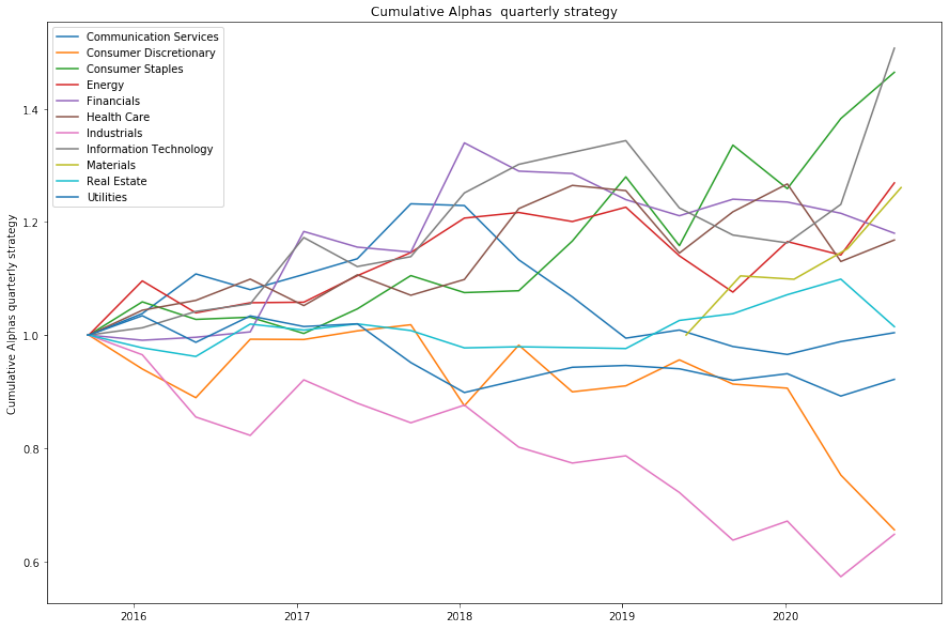


Figure 2.10: Quarterly strategy cumulative $P\&L$

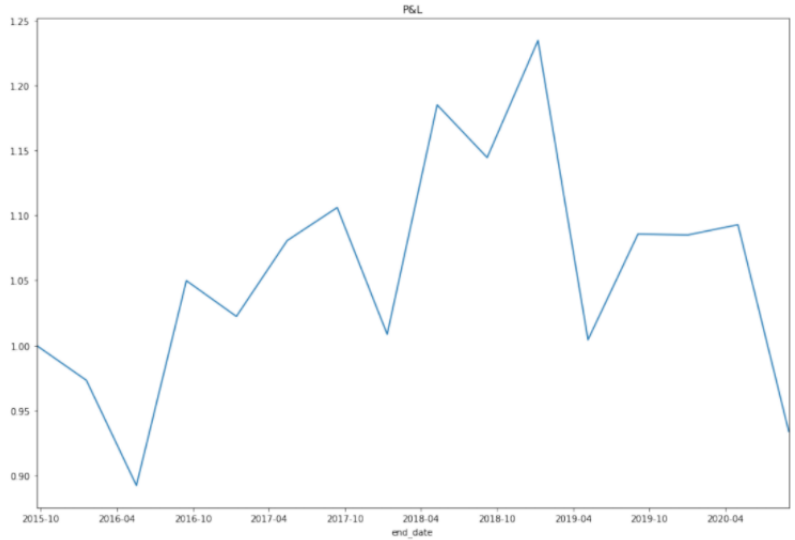


Figure 2.11: Quarterly strategy cumulative $P\&L$ with whole sp500 as input; It follows the same trend as the backtested data from a top tier bank, in the first years, but It stopped working in the last year, may be due to COVID and other trends

Further suggested work

We find, after collecting the data, webscraping It, that glassdoor data can be a relevant estimator for various sectors, while not being as much of a relevant estimator for other ones. Still, in general scenarios, we find that returns of high rated glassdoor companies, during the same period as in the test done by a top tier bank, gives the same trend; there still needs to be work done on the estimators.

One can find an estimator that is using all the input parameters od the stocks, given the sector, and outputs a recommendation through a neural network; one can also feed an LSTM to add more insight about the data : how bad/good the comment judges the company; These methods of feature selection will enhance the model, but may make It less robust and less explainable.