BA870_Individual Project_Ji_Qi

April 28, 2022

1 *Student Name: Ji Qi , Session B1*

2 Import packages

[310]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import statsmodels.api as sm
from statsmodels.sandbox.regression.predstd import wls_prediction_std
from scipy.stats.mstats import winsorize
```

3 Upload CSV file with data

[311]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

3.1 ProjectTickers.csv

- Import ProjectTickers.csv file
- 3 columns: **Ticker** (the stock's ticker symbol), **Name** (the name of each company), and **RetTYD** (the year-to-date stock return of each company from January 1, 2022 to April 14, 2022).

[312]:
```
ticker = pd.read_csv('/content/drive/MyDrive/BA_870/Individual Project/
↪ProjectTickers.csv')
ticker.head()
```

[312]:

	Ticker	Name	RetYTD
0	A	Agilent Technologies	-0.2080
1	AA	Alcoa Corp	0.4731

```
2    AAL     American Airlines Gp   0.0579
3    AAN  Aarons Holdings Company  -0.1327
4   AAON                 Aaon Inc  -0.3456
```

- No missing value for ProjectTickers.csv file
- 1886 unique companies and unique RetYTDs

[313]:
```
ticker.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1886 entries, 0 to 1885
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Ticker  1886 non-null   object
 1   Name    1886 non-null   object
 2   RetYTD  1886 non-null   float64
dtypes: float64(1), object(2)
memory usage: 44.3+ KB
```

3.2 Project-2017-21-Returns.csv

- Download monthly stock return data from January 2017 - December 2021(60 months) for each of the stocks with tickers (1886 stocks) from CRSP on the WRDS database in the file "Project-2017-21-Returns.csv".
- "Project-2017-21-Returns.csv" file contains 114715 data

[314]:
```
Ret17_21 = pd.read_csv('/content/drive/MyDrive/BA_870/Individual Project/
 ↪Project-2017-21-Returns.csv')
Ret17_21.head()
```

[314]:

	PERMNO	date	TICKER	RET
0	10026	20170131	JJSF	-0.043918
1	10026	20170228	JJSF	0.048836
2	10026	20170331	JJSF	0.016293
3	10026	20170428	JJSF	-0.007229
4	10026	20170531	JJSF	-0.033289

[315]:
```
Ret17_21.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114715 entries, 0 to 114714
Data columns (total 4 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   PERMNO  114715 non-null  int64
 1   date    114715 non-null  int64
```

```
 2   TICKER  114653 non-null  object
 3   RET     114700 non-null  object
dtypes: int64(2), object(2)
memory usage: 3.5+ MB
```

- In fact, Although I used 1886 tickers list to query the stock return from 2017 to 2021, the dataset I received includes 1924 tickers. My guess is that companies may change tickers or merge with other companies. This will probably cause the inconsistencies in the numer of tickers.

[316]:
```
Ret17_21.nunique()
```

[316]:
```
PERMNO     1920
date         60
TICKER     1924
RET       94112
dtype: int64
```

- It's worth noting that the data type of RET is `object` which means is a mixture of string and floating points. I will dig into this column later when dealing with missing values.

[317]:
```
Ret17_21.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114715 entries, 0 to 114714
Data columns (total 4 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   PERMNO  114715 non-null  int64
 1   date    114715 non-null  int64
 2   TICKER  114653 non-null  object
 3   RET     114700 non-null  object
dtypes: int64(2), object(2)
memory usage: 3.5+ MB
```

- TICKER column has 62 missing values
- RET column has 15 missing values

[318]:
```
Ret17_21.isnull().sum()
```

[318]:
```
PERMNO     0
date       0
TICKER    62
RET       15
dtype: int64
```

3.3 Project-2021-Financials.csv

- Download Financial Report data for the year 2021 for each of the stocks with tickers (1886 stocks) from Compustat on the WRDS database in the file "Project-2021-Financials.csv".
- "Project-2021-Financials.csv". file contains 1886 data

[319]:
```python
fin_21 = pd.read_csv('/content/drive/MyDrive/BA_870/Individual Project/
↪Project-2021-Financials.csv')
fin_21.head()
```

[319]:
```
   gvkey  datadate  fyear indfmt consol popsrc datafmt  tic curcd      act  \
0   1004  20210531   2020   INDL      C      D     STD  AIR   USD    937.0
1   1045  20211231   2021   INDL      C      D     STD  AAL   USD  17336.0
2   1075  20211231   2021   INDL      C      D     STD  PNW   USD   1551.1
3   1078  20211231   2021   INDL      C      D     STD  ABT   USD  24239.0
4   1161  20211231   2021   INDL      C      D     STD  AMD   USD   8583.0

   …     ebit      invt       lct         lt       ni          re  \
0  …    65.50   591.000   336.800    565.300    35.80     723.400
1  … -5514.00  1795.000  19006.000  73807.000 -1993.00  -14580.000
2  …   805.31   367.167   1756.869  15981.762   618.72    3209.858
3  …  8966.00  5157.000  13105.000  39172.000  7071.00   23154.000
4  …  3678.00  1955.000   4240.000   4922.000  3162.00   -1454.000

        sale      seq costat  prcc_c
0   1651.400    974.4      A   36.22
1  29882.000  -7340.0      A   17.96
2   3803.835   5906.2      A   70.59
3  43075.000  35802.0      A  140.74
4  16434.000   7497.0      A  143.90

[5 rows x 23 columns]
```

[320]:
```python
fin_21.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1886 entries, 0 to 1885
Data columns (total 23 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   gvkey     1886 non-null   int64
 1   datadate  1886 non-null   int64
 2   fyear     1886 non-null   int64
 3   indfmt    1886 non-null   object
 4   consol    1886 non-null   object
 5   popsrc    1886 non-null   object
 6   datafmt   1886 non-null   object
```

```
 7   tic       1886 non-null   object
 8   curcd     1886 non-null   object
 9   act       1431 non-null   float64
 10  at        1886 non-null   float64
 11  ceq       1886 non-null   float64
 12  csho      1886 non-null   float64
 13  ebit      1886 non-null   float64
 14  invt      1852 non-null   float64
 15  lct       1431 non-null   float64
 16  lt        1881 non-null   float64
 17  ni        1886 non-null   float64
 18  re        1883 non-null   float64
 19  sale      1886 non-null   float64
 20  seq       1886 non-null   float64
 21  costat    1886 non-null   object
 22  prcc_c    1886 non-null   float64
dtypes: float64(13), int64(3), object(7)
memory usage: 339.0+ KB
```

- Both `act`(total current assets) and `lct` (total current liabilities) columns have 455 missing values
- `invt`(inventory) column has 34 missing values
- `lt` (total liabilities) column has 5 missing values
- `re` (Retrained Earnings) column has 3 missing values

[321]:
```
fin_21.isnull().sum()
```

[321]:
```
gvkey         0
datadate      0
fyear         0
indfmt        0
consol        0
popsrc        0
datafmt       0
tic           0
curcd         0
act         455
at            0
ceq           0
csho          0
ebit          0
invt         34
lct         455
lt            5
ni            0
re            3
sale          0
seq           0
```

```
costat        0
prcc_c        0
dtype: int64
```

3.4 Project-2021-Sector.csv

- Download Compustat GGROUP for ”Data Date” 2021-01 through 2021-12 from WRDS Compustat and store into ”Project-2021-Sector.csv”
- tProject-2021-Sector CSV file contains 1886 data
- the `ggroup variable` represents `Industry Group GICS code`

[322]:
```
sec_21 = pd.read_csv('/content/drive/MyDrive/BA_870/Individual Project/
↪Project-2021-Sector.csv')
sec_21.head()
```

[322]:
```
   gvkey  datadate  fyear indfmt consol popsrc datafmt  tic curcd costat  \
0   1004  20210531   2020   INDL      C      D     STD  AIR   USD      A
1   1045  20211231   2021   INDL      C      D     STD  AAL   USD      A
2   1075  20211231   2021   INDL      C      D     STD  PNW   USD      A
3   1078  20211231   2021   INDL      C      D     STD  ABT   USD      A
4   1161  20211231   2021   INDL      C      D     STD  AMD   USD      A

   ggroup
0    2010
1    2030
2    5510
3    3510
4    4530
```

- No missing value for Project-2021-Sector.csv

[323]:
```
sec_21.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1886 entries, 0 to 1885
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   gvkey     1886 non-null   int64
 1   datadate  1886 non-null   int64
 2   fyear     1886 non-null   int64
 3   indfmt    1886 non-null   object
 4   consol    1886 non-null   object
 5   popsrc    1886 non-null   object
 6   datafmt   1886 non-null   object
 7   tic       1886 non-null   object
 8   curcd     1886 non-null   object
```

```
 9   costat     1886 non-null   object
 10  ggroup     1886 non-null   int64
dtypes: int64(4), object(7)
memory usage: 162.2+ KB
```

- In total, 24 unique Industry Group GICS code for Project-2021-Sector.csv file

[324]:
```python
sec_21.ggroup.nunique()
```

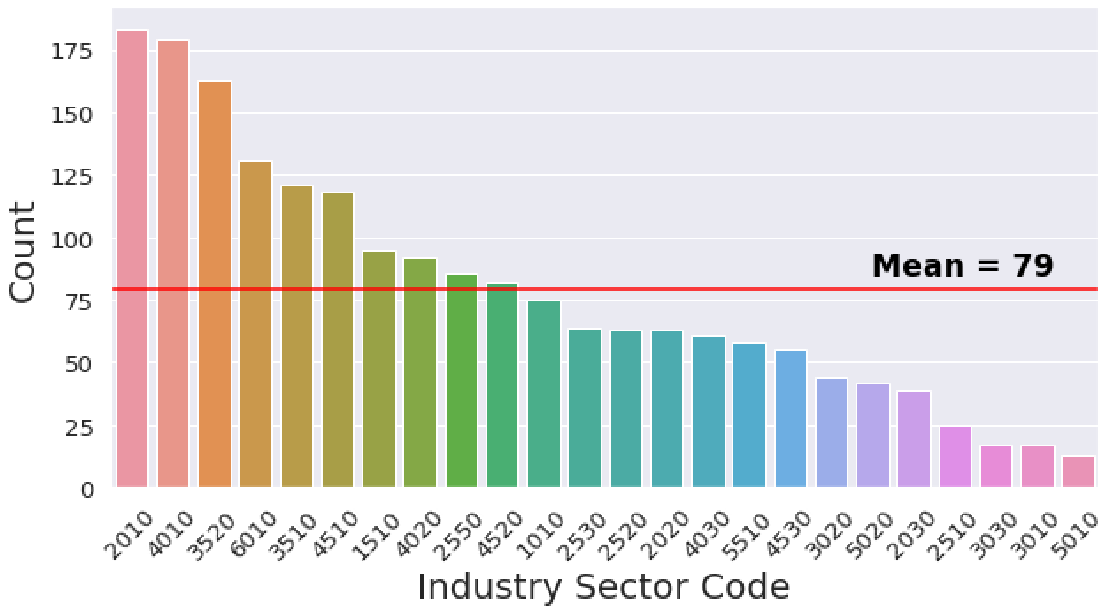
[324]: 24

[325]:
```python
secdf = sec_21.groupby('ggroup').tic.agg('count').to_frame().sort_values('tic',
 ↪ascending = False).reset_index()
secdf['ggroup'] = secdf['ggroup'].astype('string')
secdf.rename(columns={'tic':'count'}, inplace = True)
```

- Count the number of stocks in each industry

[326]:
```python
sns.set(font_scale = 1.2)
plt.figure(figsize = (10, 5))
p = sns.barplot(secdf['ggroup'], secdf['count'])
p.set_xlabel("Industry Sector Code", fontsize = 20)
p.set_ylabel("Count", fontsize = 20)
plt.xticks(rotation=45)
plt.axhline(round(secdf['count'].mean()), color='red')
plt.text(18, 85, "Mean = 79", horizontalalignment='left', size='large',
 ↪color='black', weight='semibold');
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



4 Determine Risk Exposures

4.1 Handel Outliers for Project-2017-21-Returns.csv

- TICKER 'NAN' has 62 missing values including "nan" and 'B'. I will drop this TICKER 'NAN' since it may be mistakenly generated by the WRDS database.
- There also exists 11 TICKERS with one missing value individually, so dropping one value of 60 values won't affect the results of risk exposures

[327]:
```
Ret17_21.loc[Ret17_21.isnull().any(axis = 1)].value_counts('TICKER',␣
↪dropna=False)
```

[327]:
```
TICKER
NaN      62
CASM      1
CHMT      1
CLGX      1
INSY      1
IQNT      1
JNP       1
NAV       1
OHGI      1
PIR       1
PRZM      1
TFCFA     1
```

```
dtype: int64
```

[328]:
```
Ret17_21[Ret17_21.TICKER.isnull()].RET.unique()
```

[328]:
```
array([nan, 'B'], dtype=object)
```

- Drop missing values mentioned above

[329]:
```
Ret17_21 = Ret17_21.loc[~Ret17_21.isnull().any(axis = 1)]
Ret17_21.isnull().sum()
```

[329]:
```
PERMNO    0
date      0
TICKER    0
RET       0
dtype: int64
```

- Convert RET from string to floating points

[330]:
```
Ret17_21.RET = pd.to_numeric(Ret17_21.RET, errors = 'coerce')
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/generic.py:5516:
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#returning-a-view-versus-a-copy
  self[name] = value
```

- Still have 13 nan values

[331]:
```
Ret17_21.RET.isnull().sum()
```

[331]:
```
13
```

- Drop 13 additional missing values

[332]:
```
Ret17_21 = Ret17_21[~Ret17_21.RET.isnull()]
Ret17_21.isnull().sum()
```

[332]:
```
PERMNO    0
date      0
TICKER    0
RET       0
dtype: int64
```

4.2 Create 1886 new dataframes for each stock: monthly data

[333]:
```
# # A 1886-stocks list
# tickerlist = Ret17_21.TICKER.unique().tolist()
```

[334]:
```
# A 1886-stocks list
tickerlist = ticker.Ticker
```

[335]:
```
# Create 1886 new stock dataframes and store into a dictionary
Ret17_21_dict = {}
Ret17_21_count = []
for i in tickerlist:
  Ret17_21_dict[i] = Ret17_21[Ret17_21['TICKER'] == i]
  Ret17_21_count.append(len(Ret17_21[Ret17_21['TICKER'] == i]))
```

[336]:
```
# All 1886 stock tickers
len(Ret17_21_dict.keys())
```

[336]:
```
1886
```

[337]:
```
# Display the first 120 rows (2 DataFrames:'A', 'AA')
list(Ret17_21_dict.values())[:2]
```

[337]:
```
[       PERMNO      date TICKER       RET
 89546   87432  20170131      A  0.074846
 89547   87432  20170228      A  0.047580
 89548   87432  20170331      A  0.033177
 89549   87432  20170428      A  0.041233
 89550   87432  20170531      A  0.096094
 89551   87432  20170630      A -0.014882
 89552   87432  20170731      A  0.008093
 89553   87432  20170831      A  0.082455
 89554   87432  20170929      A -0.008035
 89555   87432  20171031      A  0.061713
 89556   87432  20171130      A  0.017786
 89557   87432  20171229      A -0.030633
 89558   87432  20180131      A  0.096461
 89559   87432  20180228      A -0.065913
 89560   87432  20180329      A -0.024639
 89561   87432  20180430      A -0.015112
 89562   87432  20180531      A -0.058108
 89563   87432  20180629      A -0.001292
 89564   87432  20180731      A  0.070327
 89565   87432  20180831      A  0.022714
 89566   87432  20180928      A  0.044418
 89567   87432  20181031      A -0.079402
 89568   87432  20181130      A  0.116685
```

```
89569   87432  20181231      A -0.065321
89570   87432  20190131      A  0.127335
89571   87432  20190228      A  0.044576
89572   87432  20190329      A  0.011833
89573   87432  20190430      A -0.021349
89574   87432  20190531      A -0.145860
89575   87432  20190628      A  0.113646
89576   87432  20190731      A -0.068247
89577   87432  20190830      A  0.024492
89578   87432  20190930      A  0.079932
89579   87432  20191031      A -0.011484
89580   87432  20191129      A  0.066271
89581   87432  20191231      A  0.058438
89582   87432  20200131      A -0.032235
89583   87432  20200228      A -0.066497
89584   87432  20200331      A -0.068379
89585   87432  20200430      A  0.070371
89586   87432  20200529      A  0.149752
89587   87432  20200630      A  0.004652
89588   87432  20200731      A  0.090076
89589   87432  20200831      A  0.042458
89590   87432  20200930      A  0.005178
89591   87432  20201030      A  0.013176
89592   87432  20201130      A  0.145068
89593   87432  20201231      A  0.013601
89594   87432  20210129      A  0.015816
89595   87432  20210226      A  0.015811
89596   87432  20210331      A  0.041534
89597   87432  20210430      A  0.052651
89598   87432  20210528      A  0.033598
89599   87432  20210630      A  0.070079
89600   87432  20210730      A  0.037981
89601   87432  20210831      A  0.145141
89602   87432  20210930      A -0.102240
89603   87432  20211029      A  0.000978
89604   87432  20211130      A -0.041844
89605   87432  20211231      A  0.057985,
       PERMNO      date TICKER       RET
32595   16347  20170131     AA  0.298077
32596   16347  20170228     AA -0.051029
32597   16347  20170331     AA -0.005493
32598   16347  20170428     AA -0.019477
32599   16347  20170531     AA -0.023421
32600   16347  20170630     AA -0.008804
32601   16347  20170731     AA  0.114855
32602   16347  20170831     AA  0.205494
32603   16347  20170929     AA  0.062443
```

32604	16347	20171031	AA	0.024882
32605	16347	20171130	AA	-0.131226
32606	16347	20171229	AA	0.297760
32607	16347	20180131	AA	-0.034342
32608	16347	20180228	AA	-0.135525
32609	16347	20180329	AA	-0.000222
32610	16347	20180430	AA	0.138790
32611	16347	20180531	AA	-0.061133
32612	16347	20180629	AA	-0.024756
32613	16347	20180731	AA	-0.077005
32614	16347	20180831	AA	0.032355
32615	16347	20180928	AA	-0.095590
32616	16347	20181031	AA	-0.133911
32617	16347	20181130	AA	-0.090883
32618	16347	20181231	AA	-0.164414
32619	16347	20190131	AA	0.116629
32620	16347	20190228	AA	-0.006065
32621	16347	20190329	AA	-0.045424
32622	16347	20190430	AA	-0.052557
32623	16347	20190531	AA	-0.205772
32624	16347	20190628	AA	0.104766
32625	16347	20190731	AA	-0.039299
32626	16347	20190830	AA	-0.202757
32627	16347	20190930	AA	0.119353
32628	16347	20191031	AA	0.035875
32629	16347	20191129	AA	-0.021164
32630	16347	20191231	AA	0.057002
32631	16347	20200131	AA	-0.351464
32632	16347	20200228	AA	-0.005735
32633	16347	20200331	AA	-0.555876
32634	16347	20200430	AA	0.323052
32635	16347	20200529	AA	0.130061
32636	16347	20200630	AA	0.220413
32637	16347	20200731	AA	0.156584
32638	16347	20200831	AA	0.124615
32639	16347	20200930	AA	-0.204514
32640	16347	20201030	AA	0.110920
32641	16347	20201130	AA	0.540248
32642	16347	20201231	AA	0.158291
32643	16347	20210129	AA	-0.219089
32644	16347	20210226	AA	0.363889
32645	16347	20210331	AA	0.323422
32646	16347	20210430	AA	0.127732
32647	16347	20210528	AA	0.082696
32648	16347	20210630	AA	-0.071338
32649	16347	20210730	AA	0.089848
32650	16347	20210831	AA	0.105106

```
 32651   16347  20210930     AA  0.102998
 32652   16347  20211029     AA -0.059052
 32653   16347  20211130     AA  0.012622
 32654   16347  20211231     AA  0.280464]
```

4.3 Upload Fama-French monthly risk factor data

- Fama French Risk Factors for 2017-2021 from the file ”FF-Factors-2017-2021.csv”

[338]:
```
ff_factors = pd.read_csv('/content/drive/MyDrive/BA_870/Individual Project/
↪FF-Factors-2017-2021.csv')
```

4.3.1 List varaibles in FF dataframe

[339]:
```
ff_factors.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   dateff  60 non-null     int64
 1   mktrf   60 non-null     float64
 2   smb     60 non-null     float64
 3   hml     60 non-null     float64
 4   rf      60 non-null     float64
dtypes: float64(4), int64(1)
memory usage: 2.5 KB
```

4.3.2 Look at head and tail of FF dataframe

[340]:
```
ff_factors.head()
```

[340]:

	dateff	mktrf	smb	hml	rf
0	20170131	0.0194	-0.0113	-0.0274	0.0004
1	20170228	0.0357	-0.0204	-0.0167	0.0004
2	20170331	0.0017	0.0113	-0.0333	0.0003
3	20170428	0.0109	0.0072	-0.0213	0.0005
4	20170531	0.0106	-0.0252	-0.0375	0.0006

[341]:
```
ff_factors.tail()
```

[341]:

	dateff	mktrf	smb	hml	rf
55	20210831	0.0290	-0.0048	-0.0013	0.0000

```
56  20210930 -0.0437  0.0080  0.0509  0.0000
57  20211029  0.0665 -0.0228 -0.0044  0.0000
58  20211130 -0.0155 -0.0135 -0.0053  0.0000
59  20211231  0.0310 -0.0157  0.0323  0.0001
```

4.3.3 Rename date column to "date" to match WRDS data "date" column for 1886 monthly stocks

[342]:
```python
ff_factors.rename(columns={'dateff':'date'}, inplace=True)
ff_factors.head()
```

[342]:
```
       date   mktrf     smb     hml      rf
0  20170131  0.0194 -0.0113 -0.0274  0.0004
1  20170228  0.0357 -0.0204 -0.0167  0.0004
2  20170331  0.0017  0.0113 -0.0333  0.0003
3  20170428  0.0109  0.0072 -0.0213  0.0005
4  20170531  0.0106 -0.0252 -0.0375  0.0006
```

4.4 Merge the 1886 monthly stock return data and Fama-French market data based on "date"

- Then list head and tail of dataframe for A

[343]:
```python
stockret_ff = {}
Ret17_21_count = []
for i in Ret17_21_dict.keys():
  stockret_ff[i] = pd.merge(Ret17_21_dict[i], ff_factors, on = 'date', how =␣
↪'inner')
  Ret17_21_count.append(len(stockret_ff[i]))
```

[344]:
```python
stockret_ff['A'].head()
```

[344]:
```
   PERMNO      date TICKER       RET   mktrf     smb     hml      rf
0   87432  20170131      A  0.074846  0.0194 -0.0113 -0.0274  0.0004
1   87432  20170228      A  0.047580  0.0357 -0.0204 -0.0167  0.0004
2   87432  20170331      A  0.033177  0.0017  0.0113 -0.0333  0.0003
3   87432  20170428      A  0.041233  0.0109  0.0072 -0.0213  0.0005
4   87432  20170531      A  0.096094  0.0106 -0.0252 -0.0375  0.0006
```

[345]:
```python
stockret_ff['A'].tail()
```

[345]:
```
    PERMNO      date TICKER       RET   mktrf     smb     hml      rf
55   87432  20210831      A  0.145141  0.0290 -0.0048 -0.0013  0.0000
56   87432  20210930      A -0.102240 -0.0437  0.0080  0.0509  0.0000
57   87432  20211029      A  0.000978  0.0665 -0.0228 -0.0044  0.0000
```

```
58   87432  20211130      A -0.041844 -0.0155 -0.0135 -0.0053  0.0000
59   87432  20211231      A  0.057985  0.0310 -0.0157  0.0323  0.0001
```

- Create a Dataframe for checking the total amount of monthly stock risk exposure factors for each stock from 2017 to 2021

[346]:
```python
expfa = pd.DataFrame({'tic':tickerlist.tolist(), 'count':Ret17_21_count})
```

- All 11 companies has less than 60 montly risk exposures

[347]:
```python
expfa[expfa['count'] < 60]
```

[347]:
```
       tic  count
3      AAN     59
118   ANAB     59
151   ARNC     59
396   CNDT     59
697   FOXA     59
805    HGV     59
944   JELD     59
948   JNCE     59
1368    PK     59
1451  REVG     59
1791  VREX     59
```

[348]:
```python
expfa['count'].value_counts().reset_index().rename(columns={'index': '# of␣
↪monthly risk exposures'})
```

[348]:
```
   # of monthly risk exposures  count
0                           60   1875
1                           59     11
```

4.5 Run OLS regression for 1886 stocks (60 months) using FF 3-factor model:

- [Ret(stock)-Rf] = alpha + B1(RetMkt-Rf) + b2(SMB) + b3(HML) + e

[349]:
```python
# Create a empty output dataframe
output = pd.DataFrame(columns = ['TICKER', 'R-squared', 'Adj. R-squared',␣
↪'const', 'mktrf', 'smb', 'hml'])
output
```

[349]:
```
Empty DataFrame
Columns: [TICKER, R-squared, Adj. R-squared, const, mktrf, smb, hml]
Index: []
```

[350]:
```python
# Define a Linear Regression function for FF model
def ffmodel(data,i):
```

```
  y = data[i]["RET"] - data[i]["rf"]
  X = data[i][['mktrf' , 'smb' , 'hml']]
  # Use statsmodels
  X = sm.add_constant(X) # adding a constant
  model = sm.OLS(y, X).fit()

  #return regression output
  return (i, model.rsquared, model.rsquared_adj,
          model.params[0], model.params[1],
          model.params[2],model.params[3])

for i in list(stockret_ff.keys())[:]:
  output.loc[len(output.index)] = ffmodel(stockret_ff, i)

# Display the output regression statistics
output
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/tsatools.py:117:
FutureWarning: In a future version of pandas all arguments of concat except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

[350]:

	TICKER	R-squared	Adj. R-squared	const	mktrf	smb	hml
0	A	0.498110	0.471223	0.007241	1.014152	-0.253674	-0.143608
1	AA	0.539152	0.514464	0.014194	1.984149	0.527862	1.924844
2	AAL	0.528181	0.502905	-0.017850	1.315550	0.612825	1.248123
3	AAN	0.388654	0.355308	0.009753	1.648965	0.242525	0.815613
4	AAON	0.167367	0.122761	0.008716	0.516779	0.422130	-0.117068
…	…	…	…	…	…	…	…
1881	ZEN	0.489536	0.462189	0.010584	1.026803	1.363839	-0.774104
1882	ZION	0.795943	0.785011	0.004964	1.084739	0.869374	1.151468
1883	ZNGA	0.176268	0.132140	0.012054	0.101034	1.200987	-0.714916
1884	ZTS	0.377264	0.343903	0.014726	0.728418	-0.560814	-0.179065
1885	ZUMZ	0.443239	0.413412	0.009920	1.249317	2.336902	0.499124

[1886 rows x 7 columns]

[351]:

```
output.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1886 entries, 0 to 1885
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   TICKER          1886 non-null   object
 1   R-squared       1886 non-null   float64
 2   Adj. R-squared  1886 non-null   float64
 3   const           1886 non-null   float64
```

```
 4   mktrf          1886 non-null   float64
 5   smb            1886 non-null   float64
 6   hml            1886 non-null   float64
dtypes: float64(6), object(1)
memory usage: 117.9+ KB
```

[352]:
```
output.describe().T
```

[352]:
```
                 count      mean       std       min       25%       50%  \
R-squared       1886.0  0.382156  0.185622  0.002022  0.235846  0.380695
Adj. R-squared  1886.0  0.349054  0.195568 -0.051441  0.194909  0.347518
const           1886.0  0.004178  0.017644 -0.079578 -0.003677  0.002600
mktrf           1886.0  1.065523  0.624864 -6.676723  0.720385  1.007568
smb             1886.0  0.667360  1.232145 -6.286219  0.016011  0.501588
hml             1886.0  0.317198  0.699372 -3.593321 -0.047773  0.360752

                     75%        max
R-squared       0.516755   0.866154
Adj. R-squared  0.490867   0.858983
const           0.009878   0.433986
mktrf           1.337691   6.002016
smb             1.019407  31.005941
hml             0.760726   5.919490
```

[353]:
```
print("{0}% of smb beta is more than 0".format(round(len(output[output.smb >␣
↪0]) / len(output),2) * 100))
```

```
76.0% of smb beta is more than 0
```

[354]:
```
print("{0}% of hml beta is more than 0".format(round(len(output[output.hml >␣
↪0]) / len(output),2) * 100))
```

```
73.0% of hml beta is more than 0
```

- Calculate outlier percentages for all 3 factor betas using IQR

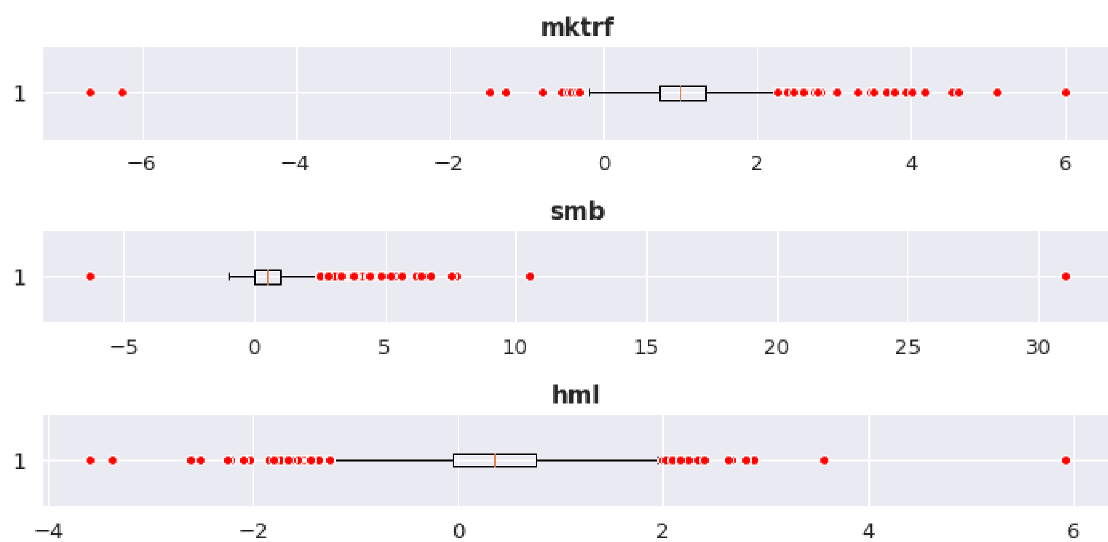
[355]:
```
ffiqr = []
for i in list(output.columns)[-3:]:
  Q1 = output[i].quantile(0.25)
  Q3 = output[i].quantile(0.75)
  IQR = Q3 - Q1
  high = Q3 + 1.5 * IQR
  low = Q1 - 1.5 * IQR
  ffiqr.append(round(len(output[(output[i] > high) |(output[i] < low)]) /␣
↪len(output),2))

pd.DataFrame({'Risk Factor':list(output.columns)[-3:], 'Outlier Ratio':ffiqr})
```

[355]:

	Risk Factor	Outlier Ratio
0	mktrf	0.04
1	smb	0.04
2	hml	0.03

[356]:

```
#Creating boxplot of each column with its own scale
red_circle = dict(markerfacecolor='red', marker='o', markeredgecolor='white')
mean_shape = dict(markerfacecolor='green', marker='D', markeredgecolor='green')
output1 = output.iloc[:,4:]

fig, axs = plt.subplots(3,1, figsize=(10,5))

for i, ax in enumerate(axs.flat):
    ax.boxplot(output1.iloc[:,i], flierprops=red_circle, meanprops=mean_shape,␣
 ↪vert = False)
    ax.set_title(output1.columns[i], fontsize=15, fontweight='bold')
    ax.tick_params(axis='y', labelsize=14)

plt.tight_layout()
```



4.6 Data Merging (ProjectTickers.csv & FF-Exposures dataframe)

[357]:

```
ffexp = pd.merge(ticker, output, left_on = 'Ticker', right_on = 'TICKER', how =␣
 ↪'left').iloc[:,1:]
```

4.7 Export FF-Exposures.csv file

[358]:
```
# Store the output into a csv file
output_file = ffexp.to_csv('FF-Exposures.csv', index = False)
```

5 Calculate Financial Ratios

5.1 Data Merging (ProjectTickers.csv & Project-2021-Financials.csv)

- No missing value for this merged dataset

[359]:
```
fin = pd.merge(ticker, fin_21, how = 'outer', left_on= 'Ticker', right_on=␣
↪'tic', indicator= True)
fin._merge.value_counts()
```

[359]:
```
both          1886
left_only        0
right_only       0
Name: _merge, dtype: int64
```

- Both `act`(total current assets) and `lct` (total current liabilities) columns have 455 missing values
- `invt` (inventory) column has 34 missing values
- `lt` (total liabilities) column has 5 missing values
- `re` (Retrained Earnings) column has 3 missing values

[360]:
```
fin.isnull().sum().to_frame().reset_index().rename(columns={'index':'fin␣
↪variable', 0:'count'}).sort_values('count', ascending = False).iloc[:5]
```

[360]:

	fin variable	count
18	lct	455
12	act	455
17	invt	34
19	lt	5
21	re	3

- Merge Project-2021-Financials with Project-2021-Sector

[361]:
```
fin_sec = pd.merge(fin_21, sec_21, how = 'outer', on= 'tic', indicator= True)
fin_sec._merge.value_counts()
```

[361]:
```
both          1886
left_only        0
right_only       0
Name: _merge, dtype: int64
```

- Bank and Real Estate industries have more missing values in total assets and total liabilities

[362]:
```
fin_sec[fin_sec.lct.isnull()].groupby('ggroup').agg({'ggroup':'count'}).
↪rename(columns={'ggroup': 'count'}).reset_index().sort_values('count',␣
↪ascending = False)
```

[362]:

	ggroup	count
4	4010	179
10	6010	114
5	4020	74
6	4030	54
1	2520	15
0	2010	10
3	2550	3
7	4510	3
2	2530	1
8	5020	1
9	5510	1

5.2 Fill the NaN values using the Median and Predictive Model

5.2.1 Data Imputation for `Retrained Earnings` Using Median Value

[363]:
```
fin['re'] = fin['re'].fillna(fin['re'].median(skipna = True))
fin['re'].isnull().sum()
```

[363]: 0

5.2.2 Data Imputation for `Total Liabilities` Using Median Value

[364]:
```
fin['lt'] = fin['lt'].fillna(fin['lt'].median(skipna = True))
fin['lt'].isnull().sum()
```

[364]: 0

5.2.3 Data Imputation for `Inventory` Using Median Value

[365]:
```
fin['invt'] = fin['invt'].fillna(fin['invt'].median(skipna = True))
fin['invt'].isnull().sum()
```

[365]: 0

5.2.4 Data Imputation for Current Assets (Random Forest)

[366]:
```
# DataFrame without any missing value
fin_nan_ca = fin[~fin.isna().any(axis=1)]
fin_nan_ca
```

[366]:
```
     Ticker                     Name   RetYTD   gvkey  datadate  fyear indfmt  \
0         A     Agilent Technologies  -0.2080  126554  20211031   2021   INDL
1        AA               Alcoa Corp   0.4731   27638  20211231   2021   INDL
2       AAL     American Airlines Gp   0.0579    1045  20211231   2021   INDL
4      AAON                 Aaon Inc  -0.3456   21542  20211231   2021   INDL
5       AAP   Advance Auto Parts Inc  -0.0884  145977  20211231   2021   INDL
…       …                       …       …       …         …      …      …
1880   ZBRA     Zebra Technologies  -0.3386   24405  20211231   2021   INDL
1881    ZEN              Zendesk Inc   0.2002   20229  20211231   2021   INDL
1883   ZNGA         Zynga Inc Cl A    0.3969  187576  20211231   2021   INDL
1884    ZTS          Zoetis Inc Cl A  -0.2325   13721  20211231   2021   INDL
1885   ZUMZ               Zumiez Inc  -0.1792  162988  20210131   2020   INDL

     consol popsrc datafmt  …      invt        lct         lt        ni  \
0         C      D     STD  …   830.000   1708.000   5316.000  1210.000
1         C      D     STD  …  1956.000   3223.000   8741.000   429.000
2         C      D     STD  …  1795.000  19006.000  73807.000 -1993.000
4         C      D     STD  …   136.019     86.768    184.010    58.758
5         C      D     STD  …  4659.018   5180.307   9065.918   616.108
…       …      …       …   …  …         …          …          …
1880      C      D     STD  …   491.000   1800.000   3231.000   837.000
1881      C      D     STD  …     0.000    911.339   1962.061  -223.644
1883      C      D     STD  …     0.000   1563.400   3247.000  -104.200
1884      C      D     STD  …  1923.000   1797.000   9356.000  2037.000
1885      C      D     STD  …   134.354    195.452    445.768    76.227

             re       sale        seq  costat  prcc_c  _merge
0        66.000   6319.000   5389.000       A  159.65    both
1     -4907.000  12152.000   4672.000       A   59.58    both
2    -14580.000  29882.000  -7340.000       A   17.96    both
4       384.306    534.517    466.170       A   79.43    both
5      4583.164  10997.989   3128.291       A  239.88    both
…           …          …          …       …      …
1880   3544.000   5627.000   2984.000       A  595.20    both
1881  -1149.154   1338.603    489.218       A  104.29    both
1883  -2513.100   2800.500   3111.900       A    6.40    both
1884   6422.000   7786.000   4543.000       A  244.03    both
1885    380.968    990.652    552.596       A   36.78    both

[1431 rows x 27 columns]
```

[367]:
```
# Select useful columns
fin_nan_ca = fin_nan_ca[['act', 'at', 'ceq', 'csho', 'lct', 'lt', 're', 'sale',
       'seq', 'prcc_c']]
```

[368]:
```
# Fitting Random Forest Regression to the dataset
# import the regressor
from sklearn.ensemble import RandomForestRegressor

 # create regressor object
regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)

Xpca = fin_nan_ca.drop(columns={'act', 'lct'})
ypca = fin_nan_ca['act']

# fit the regressor with x and y data
regressor.fit(Xpca, ypca)
```

[368]:
```
RandomForestRegressor(random_state=0)
```

[369]:
```
print('R squared for the Random Forest method is around {:.4f}.'.
 ↪format(regressor.score(Xpca, ypca)))
```

```
R squared for the Random Forest method is around 0.9552.
```

[370]:
```
CA = fin.iloc[fin[fin.isnull().any(axis =1)].index,:][['at', 'ceq', 'csho',␣
 ↪'lt', 're', 'sale',
       'seq', 'prcc_c']]
```

- Predict `Current Assets` for the missing rows

[371]:
```
fin.loc[fin[fin.isnull().any(axis =1)].index,['act']] = regressor.predict(CA)
```

5.2.5 Data Imputation for `CURRENT LIABIL` (Linear Regression)

[372]:
```
# DataFrame without any missing value
fin_nan_cl = fin[~fin.isna().any(axis=1)]
fin_nan_cl
```

[372]:
```
     Ticker                   Name  RetYTD   gvkey  datadate  fyear indfmt  \
0         A     Agilent Technologies -0.2080  126554  20211031   2021   INDL
1        AA              Alcoa Corp  0.4731   27638  20211231   2021   INDL
2       AAL     American Airlines Gp  0.0579    1045  20211231   2021   INDL
4      AAON                Aaon Inc -0.3456   21542  20211231   2021   INDL
5       AAP   Advance Auto Parts Inc -0.0884  145977  20211231   2021   INDL
…      …                     …      …       …      …      …
1880   ZBRA      Zebra Technologies -0.3386   24405  20211231   2021   INDL
```

```
1881    ZEN             Zendesk Inc   0.2002   20229  20211231   2021   INDL
1883   ZNGA          Zynga Inc Cl A   0.3969  187576  20211231   2021   INDL
1884    ZTS         Zoetis Inc Cl A  -0.2325   13721  20211231   2021   INDL
1885   ZUMZ              Zumiez Inc  -0.1792  162988  20210131   2020   INDL

     consol popsrc datafmt  …      invt       lct        lt        ni  \
0         C      D     STD  …   830.000  1708.000  5316.000  1210.000
1         C      D     STD  …  1956.000  3223.000  8741.000   429.000
2         C      D     STD  …  1795.000 19006.000 73807.000 -1993.000
4         C      D     STD  …   136.019    86.768   184.010    58.758
5         C      D     STD  …  4659.018  5180.307  9065.918   616.108
…       …    …       …   …        …         …        …        …
1880      C      D     STD  …   491.000  1800.000  3231.000   837.000
1881      C      D     STD  …     0.000   911.339  1962.061  -223.644
1883      C      D     STD  …     0.000  1563.400  3247.000  -104.200
1884      C      D     STD  …  1923.000  1797.000  9356.000  2037.000
1885      C      D     STD  …   134.354   195.452   445.768    76.227

             re       sale       seq  costat  prcc_c  _merge
0        66.000   6319.000  5389.000       A  159.65    both
1     -4907.000  12152.000  4672.000       A   59.58    both
2    -14580.000  29882.000 -7340.000       A   17.96    both
4       384.306    534.517   466.170       A   79.43    both
5      4583.164  10997.989  3128.291       A  239.88    both
…           …         …        …       …      …       …
1880   3544.000   5627.000  2984.000       A  595.20    both
1881  -1149.154   1338.603   489.218       A  104.29    both
1883  -2513.100   2800.500  3111.900       A    6.40    both
1884   6422.000   7786.000  4543.000       A  244.03    both
1885    380.968    990.652   552.596       A   36.78    both

[1431 rows x 27 columns]
```

- Select useful columns for the independent varibales
- add a intercept column

[373]:
```
fin_nan_cl = fin_nan_cl[['act', 'at', 'ceq', 'csho', 'lct', 'lt', 're', 'sale',
       'seq', 'prcc_c']]
fin_nan_cl['constant'] = 1 # add a intercept

Xcl = fin_nan_cl.drop(columns={'lct'})
ycl = fin_nan_cl[['lct']]
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py: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#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

- Run the linear regression model

[374]:
```python
modelcl = sm.OLS(ycl, Xcl)
resultscl = modelcl.fit()
print(resultscl.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    lct   R-squared:                       0.947
Model:                            OLS   Adj. R-squared:                  0.947
Method:                 Least Squares   F-statistic:                     2824.
Date:                Thu, 28 Apr 2022   Prob (F-statistic):               0.00
Time:                        02:02:25   Log-Likelihood:                -13255.
No. Observations:                1431   AIC:                         2.653e+04
Df Residuals:                    1421   BIC:                         2.658e+04
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
act            0.6258      0.010     62.595      0.000       0.606       0.645
at             0.0768      0.104      0.736      0.462      -0.128       0.281
ceq            0.2847      0.209      1.364      0.173      -0.125       0.694
csho          -1.2428      0.149     -8.335      0.000      -1.535      -0.950
lt             0.1007      0.106      0.950      0.342      -0.107       0.309
re            -0.0102      0.009     -1.163      0.245      -0.028       0.007
sale           0.0362      0.003     10.348      0.000       0.029       0.043
seq           -0.5554      0.228     -2.436      0.015      -1.003      -0.108
prcc_c        -1.6381      0.356     -4.601      0.000      -2.337      -0.940
constant     -71.2162     80.203     -0.888      0.375    -228.546      86.114
==============================================================================
Omnibus:                      666.866   Durbin-Watson:                   1.936
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           381952.815
Skew:                           0.743   Prob(JB):                         0.00
Kurtosis:                      83.023   Cond. No.                     7.05e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 7.05e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[375]:
```
print('R squared for the Linear Regression method is around {:.4f}.'.
 ↪format(resultscl.rsquared))
```

```
R squared for the Linear Regression method is around 0.9470.
```

- Predict `Current Liabilities` for the missing rows

[376]:
```
CL = fin[fin.isna().any(axis=1)][['act', 'at', 'ceq', 'csho', 'lt', 're',␣
 ↪'sale',
       'seq', 'prcc_c']]
CL['constant'] = 1
resultscl.predict(CL)
```

[376]:
```
3          340.987495
7          346.592838
11        4282.208122
15        2364.170964
17        1397.479158
              …
1857     10389.769104
1859     -1066.976318
1861      3221.686199
1869       252.535794
1882     16590.685451
Length: 455, dtype: float64
```

- No missing values for Financial Report Data

[377]:
```
fin.loc[CL.index,['lct']] = resultscl.predict(CL)
fin.isnull().sum()
```

[377]:
```
Ticker      0
Name        0
RetYTD      0
gvkey       0
datadate    0
fyear       0
indfmt      0
consol      0
popsrc      0
datafmt     0
tic         0
curcd       0
act         0
at          0
ceq         0
csho        0
ebit        0
```

```
invt        0
lct         0
lt          0
ni          0
re          0
sale        0
seq         0
costat      0
prcc_c      0
_merge      0
dtype: int64
```

5.3 Feature Engineering (Creates 4 Market Ratios)

- **Market Value Ratios**
- Price/Book = (PRCC_C * CSHO) / CEQ
- P/E = (PRCC_C * CSHO) / NI
- P/EBIT = (PRCC_C * CSHO) / EBIT
- P/SALES = (PRCC_C * CSHO) / SALES
- **Efficiency Ratios**
- Retention Ratio = Retained Earnings / Net Income
- Quick Ratio = CURRENT ASSETS - Inventory / CURRENT LIABIL
- **Debt Management Ratios**
- Debt ratio = Total liabilities / Total Assets
- **Profitability Ratios**
- Return on Assets = NI / TOTAL ASSETS
- Return on Equity = NI / Shareholder Equity
-

5.4 Asset Turnover Ratio = Total Sales / Total Assets

- Invert each of those ratios to make them ”better behaved”.
- Book/Price = 1/[Price/Book]
- E/P = 1/[P/E]
- EBIT/P = 1/[P/EBIT]
- SALE/P = 1/[P/SALES]

[378]:
```python
# Market ratios
fin['B/P'] = fin['ceq'] / (fin['prcc_c'] * fin['csho'])
fin['E/P'] = fin['ni'] / (fin['prcc_c'] * fin['csho'])
fin['ebit/P'] = fin['ebit'] / (fin['prcc_c'] * fin['csho'])
fin['sale/P'] = fin['sale'] / (fin['prcc_c'] * fin['csho'])

# Efficiency Ratios
# fin['curr_ratio'] = fin['act'] / fin['lct']
fin['rent_ratio'] = fin['re'] / fin['ni']
fin['quick_ratio'] = (fin['act'] - fin['invt']) / fin['lct']

# Debt Management Ratios
fin['debt_ratio'] = fin['lt'] / fin['at']

# Profitability Ratios
# fin['net_profit_margin'] = fin['ni'] / fin['sale']
fin['ROA'] = fin['ni'] / fin['at']
fin['ROE'] = fin['ni'] / fin['seq']
fin['ATR'] = fin['sale'] / fin['at']
```

5.5 Check and Handle outliers

- Create a new dataframe by selecting the 10 finance ratios as the independent variables and RetYTD as a dependent variable.

[379]:
```python
fin.columns
```

[379]:
```
Index(['Ticker', 'Name ', 'RetYTD', 'gvkey', 'datadate', 'fyear', 'indfmt',
       'consol', 'popsrc', 'datafmt', 'tic', 'curcd', 'act', 'at', 'ceq',
       'csho', 'ebit', 'invt', 'lct', 'lt', 'ni', 're', 'sale', 'seq',
       'costat', 'prcc_c', '_merge', 'B/P', 'E/P', 'ebit/P', 'sale/P',
       'rent_ratio', 'quick_ratio', 'debt_ratio', 'ROA', 'ROE', 'ATR'],
      dtype='object')
```

[380]:
```python
fin = fin[['Ticker', 'Name ', 'RetYTD', 'B/P', 'E/P', 'ebit/P', 'sale/P',
       'rent_ratio','quick_ratio', 'debt_ratio', 'ROA', 'ROE', 'ATR']]
fin.head()
```

[380]:
```
  Ticker                     Name   RetYTD       B/P       E/P    ebit/P  \
0      A     Agilent Technologies  -0.2080  0.111695  0.025079  0.029204
1     AA               Alcoa Corp   0.4731  0.425940  0.039111  0.189357
2    AAL     American Airlines Gp   0.0579 -0.630953 -0.171320 -0.473988
3    AAN  Aarons Holdings Company  -0.1327  0.940491  0.143967  0.206699
4   AAON                Aaon Inc  -0.3456  0.111730  0.014083  0.017648

     sale/P  rent_ratio  quick_ratio  debt_ratio       ROA       ROE       ATR
```

```
0  0.130970    0.054545     1.738290    0.496590  0.113031  0.224531  0.590285
1  1.107882  -11.438228     0.952529    0.581764  0.028552  0.091824  0.808785
2  2.568684    7.315605     0.817689    1.110431 -0.029985  0.271526  0.449576
3  2.416822    0.889688     2.044556    0.501711  0.076276  0.153076  1.280475
4  0.128111    6.540488     0.945752    0.283014  0.090372  0.126044  0.822106
```

- No missing value for each created financial ratios

[381]:
```
fin.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1886 entries, 0 to 1885
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Ticker        1886 non-null   object
 1   Name          1886 non-null   object
 2   RetYTD        1886 non-null   float64
 3   B/P           1886 non-null   float64
 4   E/P           1886 non-null   float64
 5   ebit/P        1886 non-null   float64
 6   sale/P        1886 non-null   float64
 7   rent_ratio    1886 non-null   float64
 8   quick_ratio   1886 non-null   float64
 9   debt_ratio    1886 non-null   float64
 10  ROA           1886 non-null   float64
 11  ROE           1886 non-null   float64
 12  ATR           1886 non-null   float64
dtypes: float64(11), object(2)
memory usage: 270.8+ KB
```

- Create a dataframe containing only 10 financial ratios

[382]:
```
df_model = fin.iloc[:, 3:]
df_model
```

[382]:
```
           B/P       E/P    ebit/P    sale/P  rent_ratio  quick_ratio  \
0     0.111695  0.025079  0.029204  0.130970    0.054545     1.738290
1     0.425940  0.039111  0.189357  1.107882  -11.438228     0.952529
2    -0.630953 -0.171320 -0.473988  2.568684    7.315605     0.817689
3     0.940491  0.143967  0.206699  2.416822    0.889688     2.044556
4     0.111730  0.014083  0.017648  0.128111    6.540488     0.945752
…          …         …         …         …           …            …
1881  0.038577 -0.017636 -0.011941  0.105556    5.138318     1.556646
1882  0.733347  0.117891  0.188062  0.310130    4.512843     0.307874
1883  0.430106 -0.014402  0.032535  0.387066   24.118042     1.054497
1884  0.039394  0.017664  0.024618  0.067515    3.152676     2.786311
1885  0.586912  0.080961  0.108059  1.052171    4.997809     2.051261
```

```
      debt_ratio       ROA       ROE       ATR
0       0.496590  0.113031  0.224531  0.590285
1       0.581764  0.028552  0.091824  0.808785
2       1.110431 -0.029985  0.271526  0.449576
3       0.501711  0.076276  0.153076  1.280475
4       0.283014  0.090372  0.126044  0.822106
…            …         …         …         …
1881    0.800423 -0.091236 -0.457146  0.546083
1882    0.919925  0.012114  0.151280  0.031867
1883    0.510623 -0.016386 -0.033484  0.440406
1884    0.673094  0.146547  0.448382  0.560144
1885    0.446498  0.076352  0.137943  0.992275

[1886 rows x 10 columns]
```

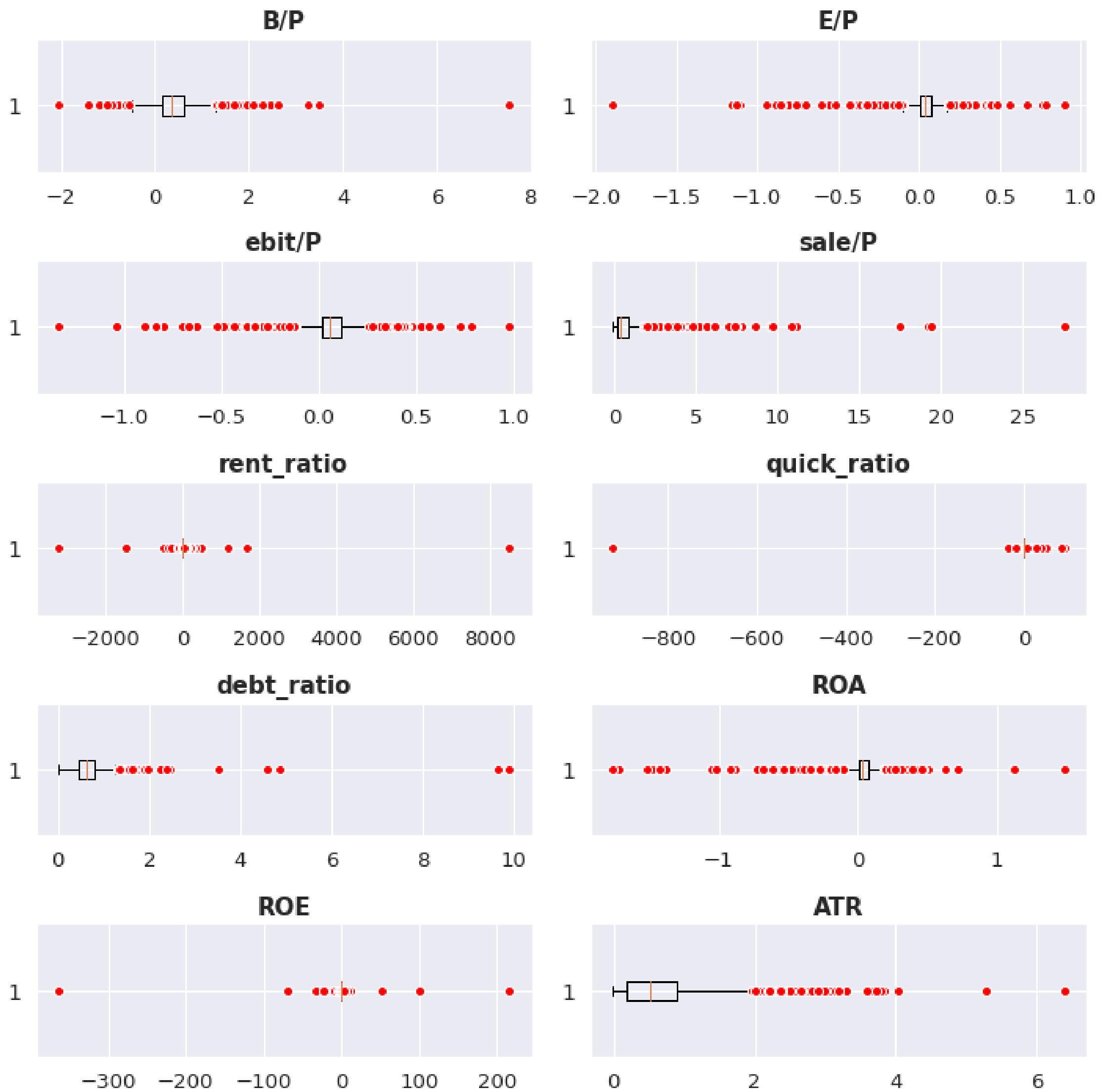
[383]:
```python
df_model.describe()
```

[383]:
```
               B/P          E/P       ebit/P        sale/P   rent_ratio  \
count  1886.000000  1886.000000  1886.000000  1886.000000  1886.000000
mean      0.422120     0.027009     0.060538     0.765997     7.065172
std       0.424100     0.136353     0.130324     1.439881   220.549098
min      -2.056645    -1.891725    -1.334204    -0.087004 -3248.673575
25%       0.161112     0.008113     0.019641     0.176223     0.439140
50%       0.337004     0.036704     0.056821     0.350729     3.695886
75%       0.613868     0.076322     0.113822     0.814565     6.611482
max       7.547555     0.898517     0.974888    27.543662  8515.124088

       quick_ratio   debt_ratio          ROA          ROE          ATR
count  1886.000000  1886.000000  1886.000000  1886.000000  1886.000000
mean      1.650150     0.646549     0.023572     0.009034     0.642005
std      21.836019     0.425294     0.171422    10.339557     0.616868
min    -924.116120     0.008474    -1.759213  -364.326147    -0.009803
25%       0.765277     0.460233     0.007584     0.024152     0.185607
50%       1.213342     0.633081     0.034187     0.112438     0.511923
75%       2.122198     0.802777     0.079409     0.212665     0.900111
max      91.177254     9.901970     1.484070   216.142857     6.389003
```

- below are the boxplots for all those finance ratios.

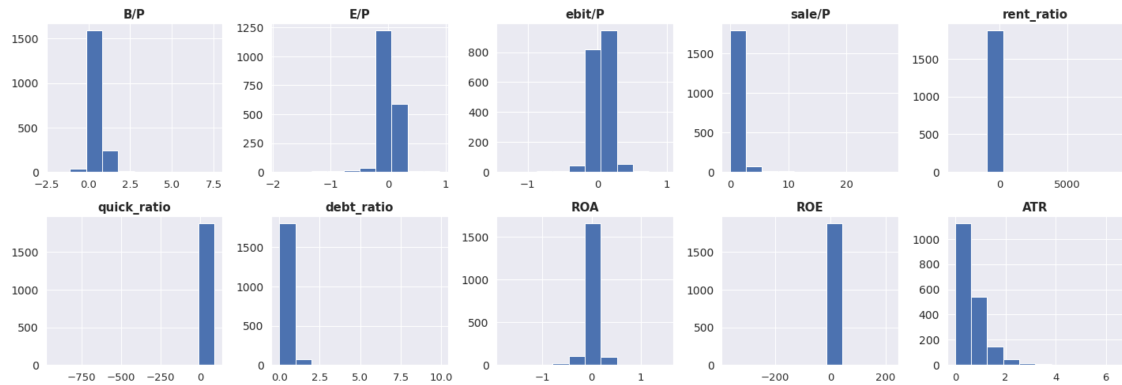
[384]:
```python
#Creating boxplot of each column with its own scale
red_circle = dict(markerfacecolor='red', marker='o', markeredgecolor='white')
mean_shape = dict(markerfacecolor='green', marker='D', markeredgecolor='green')

fig, axs = plt.subplots(5,2, figsize=(10,10))

for i, ax in enumerate(axs.flat):
```

```
    ax.boxplot(df_model.iloc[:,i], flierprops=red_circle, meanprops=mean_shape,␣
↪vert = False)
    ax.set_title(df_model.columns[i], fontsize=15, fontweight='bold')
    ax.tick_params(axis='y', labelsize=14)

plt.tight_layout()
```



[385]:
```
#Check the outlier using histogram
fig, axs = plt.subplots(2,5, figsize=(20,7))

for i, ax in enumerate(axs.flat):
    ax.hist(df_model.iloc[:,i])
    ax.set_title(df_model.columns[i], fontsize=15, fontweight='bold')
```

```
    ax.tick_params(axis='y', labelsize=14)

plt.tight_layout()
```



5.5.1 Summary I

- According to the multiple box plots above, every column has several outliers above 75 percentile and below 25 percentile. However, **sale / price**, **debt ratio** and **Asset Turnover Ratio** only have outliers which are above 75 percentile. In other words, the distribution of thoes three ratios are right-skewed. Especially, **rent_ratio** has extreme values compares to its 25 and 75 percentile values
- Since, we only have 1886 data points, dropping some outliers may decrease the model predictive power
- I will use Winsorization method to deal with outliers before establishing the linear regression models

5.5.2 Winsorization Method

[386]:
```
from scipy.stats.mstats import winsorize
```

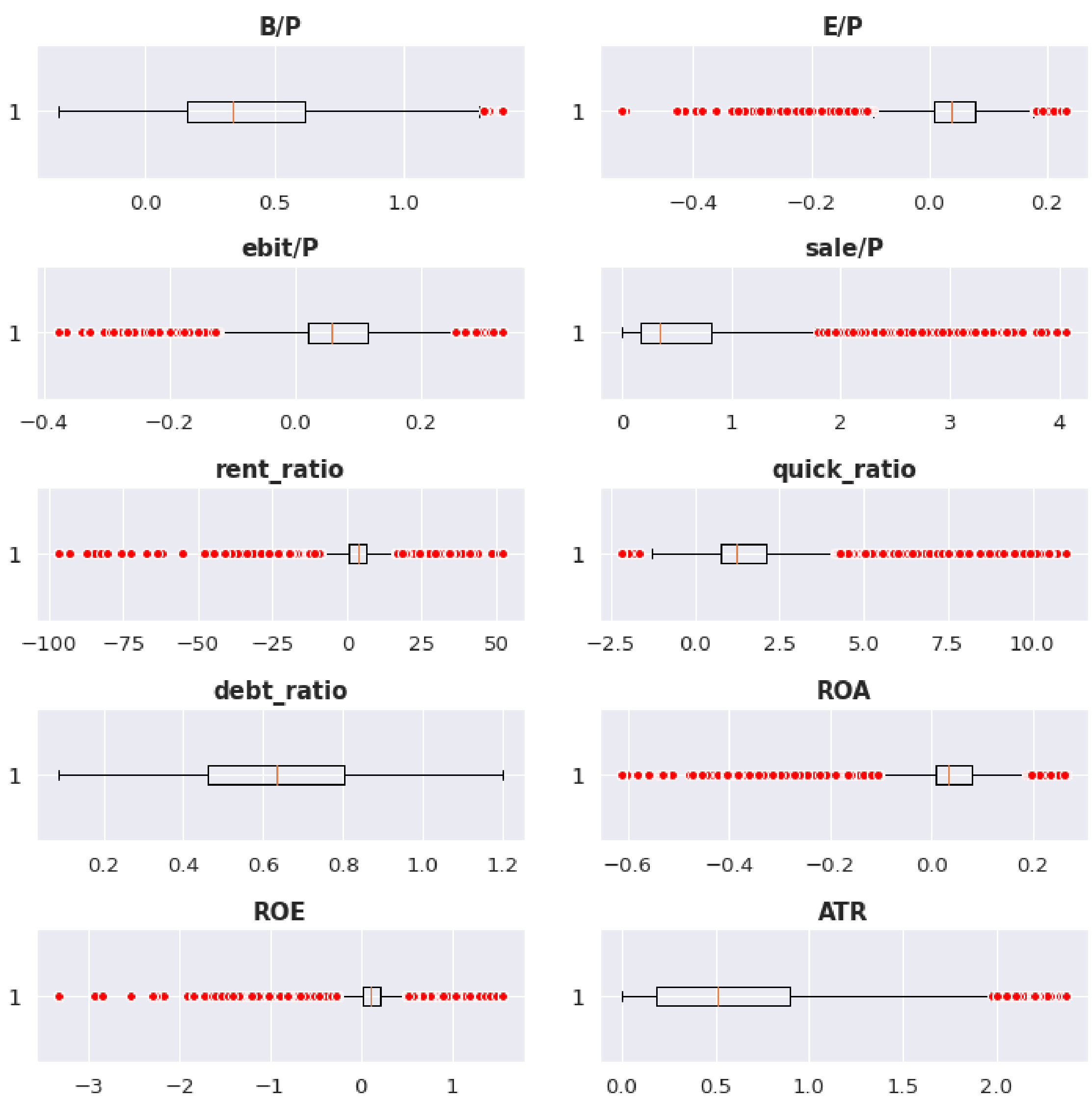
[387]:
```
df_win = df_model.copy()
for i in df_model.columns:
  df_win[i] = winsorize(df_model[i], (0.01, 0.02))

df_win.describe()
```

[387]:
```
                B/P          E/P       ebit/P       sale/P   rent_ratio  \
count   1886.000000  1886.000000  1886.000000  1886.000000  1886.000000
mean       0.414033     0.028158     0.060454     0.687720     3.018709
std        0.343580     0.102961     0.107562     0.854932    16.455807
```

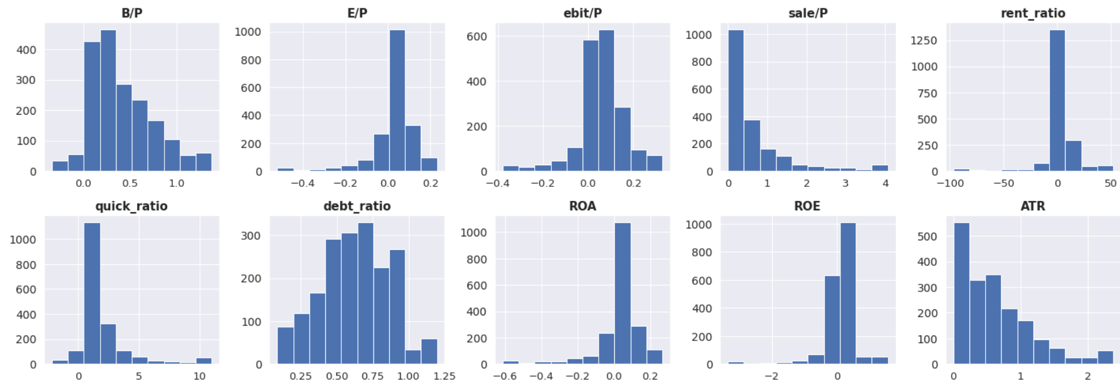
```
min      -0.337273   -0.521213   -0.378436    0.000060  -96.808868
25%       0.161112    0.008113    0.019641    0.176223    0.439140
50%       0.337004    0.036704    0.056821    0.350729    3.695886
75%       0.613868    0.076322    0.113822    0.814565    6.611482
max       1.377565    0.231730    0.329366    4.063661   52.040147

       quick_ratio   debt_ratio         ROA         ROE         ATR
count  1886.000000  1886.000000  1886.000000  1886.000000  1886.000000
mean      1.906921     0.626141     0.024837     0.075062     0.626616
std       2.180792     0.241628     0.128383     0.546444     0.549811
min      -2.169569     0.086737    -0.612289    -3.335815     0.000132
25%       0.765277     0.460233     0.007584     0.024152     0.185607
50%       1.213342     0.633081     0.034187     0.112438     0.511923
75%       2.122198     0.802777     0.079409     0.212665     0.900111
max      10.989332     1.198789     0.266337     1.555145     2.370510
```

- Check the outliers after winsorization

[388]:
```python
#Check the outlier after Log Transformation
red_circle = dict(markerfacecolor='red', marker='o', markeredgecolor='white')
mean_shape = dict(markerfacecolor='green', marker='D', markeredgecolor='green')

fig, axs = plt.subplots(5,2, figsize=(10,10))

for i, ax in enumerate(axs.flat):
    ax.boxplot(df_win.iloc[:,i], flierprops=red_circle, meanprops=mean_shape, vert = False)
    ax.set_title(df_win.columns[i], fontsize=15, fontweight='bold')
    ax.tick_params(axis='y', labelsize=14)

plt.tight_layout()
```



[389]:
```python
#Check the outlier after Log Transformation
fig, axs = plt.subplots(2,5, figsize=(20,7))

for i, ax in enumerate(axs.flat):
    ax.hist(df_win.iloc[:,i])
    ax.set_title(df_win.columns[i], fontsize=15, fontweight='bold')
    ax.tick_params(axis='y', labelsize=14)

plt.tight_layout()
```



5.5.3 Summary II

- After Winsorization, **sale / price** and **Asset Turnover Ratio** are still right-skewed and **quick ratio** and **rent_ratio** are converted to a normal distribution. The good news is that the distributions of the rest 6 ratios are more close to normal, which is proved by the above 6 histograms and 6 boxplots.

[390]:
```
fin.iloc[:,:3]
```

[390]:
```
      Ticker                      Name  RetYTD
0          A      Agilent Technologies -0.2080
1         AA                Alcoa Corp  0.4731
2        AAL      American Airlines Gp  0.0579
3        AAN  Aarons Holdings Company -0.1327
4       AAON                  Aaon Inc -0.3456
…        …                         …       …
1881     ZEN              Zendesk Inc  0.2002
1882    ZION            Zions Bancorp -0.0038
1883    ZNGA           Zynga Inc Cl A  0.3969
1884     ZTS          Zoetis Inc Cl A -0.2325
1885    ZUMZ              Zumiez Inc -0.1792

[1886 rows x 3 columns]
```

5.6 Export Fin-Ratios.csv file

[391]:
```
fin_ratios = pd.concat([fin.iloc[:,:3], df_win], axis = 1)
```

[392]:
```
fin_ratios.to_csv('Fin-Ratios.csv', index = False)
```

6 Industry Indicators

6.1 Data Merging (Project-2021-Sector.csv & ProjectTickers.csv)

[393]:
```
industry = pd.merge(sec_21, ticker, how = 'outer', left_on= 'tic', right_on=␣
↪'Ticker', indicator= True)
industry._merge.value_counts()
```

[393]:
```
both           1886
left_only         0
right_only        0
Name: _merge, dtype: int64
```

- No missing value for this merged dataset

[394]:
```
industry.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1886 entries, 0 to 1885
Data columns (total 15 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   gvkey     1886 non-null   int64
 1   datadate  1886 non-null   int64
 2   fyear     1886 non-null   int64
 3   indfmt    1886 non-null   object
 4   consol    1886 non-null   object
 5   popsrc    1886 non-null   object
 6   datafmt   1886 non-null   object
 7   tic       1886 non-null   object
 8   curcd     1886 non-null   object
 9   costat    1886 non-null   object
 10  ggroup    1886 non-null   int64
 11  Ticker    1886 non-null   object
 12  Name      1886 non-null   object
 13  RetYTD    1886 non-null   float64
 14  _merge    1886 non-null   category
dtypes: category(1), float64(1), int64(4), object(9)
memory usage: 223.0+ KB
```

6.2 Export Industry.csv file

[395]:
```
industry[['Ticker','Name ','ggroup', 'RetYTD']].to_csv('Industry.csv', index =␣
↪False)
```

6.3 Create Industry Indicator (category) variables for the 24 Industry categories

- Select useful columns for the downstream tasks

[396]:
```
industry = industry[['Ticker', 'ggroup', 'RetYTD']]
industry.head()
```

[396]:
```
  Ticker  ggroup  RetYTD
0    AIR    2010  0.2944
1    AAL    2030  0.0579
2    PNW    5510  0.0985
3    ABT    3510 -0.1638
4    AMD    4530 -0.3533
```

[397]:
```
industry.groupby('ggroup')[['RetYTD']].agg('mean').sort_values("RetYTD")
```

[397]:
```
          RetYTD
ggroup
4530   -0.252813
2520   -0.208598
2510   -0.207992
4520   -0.172065
3520   -0.163937
3030   -0.153771
2550   -0.151223
4020   -0.122515
4510   -0.108644
2010   -0.090548
4010   -0.087832
5020   -0.070871
2030   -0.067454
3510   -0.066432
2530   -0.062583
6010   -0.035826
2020   -0.035157
4030    0.003779
5010    0.011838
3020    0.038780
5510    0.040759
1510    0.057846
3010    0.078312
1010    0.515035
```

[398]:
```
industry_new = pd.get_dummies(industry, columns=['ggroup'])
industry_new
```

[398]:
```
      Ticker  RetYTD  ggroup_1010  ggroup_1510  ggroup_2010  ggroup_2020  \
0        AIR  0.2944            0            0            1            0
1        AAL  0.0579            0            0            0            0
2        PNW  0.0985            0            0            0            0
3        ABT -0.1638            0            0            0            0
4        AMD -0.3533            0            0            0            0
…        …       …            …            …            …            …
1881     KRG  0.0275            0            0            0            0
1882     LYB  0.1664            0            1            0            0
1883     FRO  0.3380            1            0            0            0
1884    ALLE -0.1888            0            0            1            0
1885     LPG  0.2427            1            0            0            0

      ggroup_2030  ggroup_2510  ggroup_2520  ggroup_2530  …  ggroup_4010  \
0               0            0            0            0  …            0
1               1            0            0            0  …            0
2               0            0            0            0  …            0
3               0            0            0            0  …            0
4               0            0            0            0  …            0
…             …            …            …            …  …          …
1881            0            0            0            0  …            0
1882            0            0            0            0  …            0
1883            0            0            0            0  …            0
1884            0            0            0            0  …            0
1885            0            0            0            0  …            0

      ggroup_4020  ggroup_4030  ggroup_4510  ggroup_4520  ggroup_4530  \
0               0            0            0            0            0
1               0            0            0            0            0
2               0            0            0            0            0
3               0            0            0            0            0
4               0            0            0            0            1
…             …            …            …            …            …
1881            0            0            0            0            0
1882            0            0            0            0            0
1883            0            0            0            0            0
1884            0            0            0            0            0
1885            0            0            0            0            0

      ggroup_5010  ggroup_5020  ggroup_5510  ggroup_6010
0               0            0            0            0
1               0            0            0            0
2               0            0            1            0
3               0            0            0            0
4               0            0            0            0
…             …            …            …            …
1881            0            0            0            1
```

```
1882            0            0            0            0
1883            0            0            0            0
1884            0            0            0            0
1885            0            0            0            0

[1886 rows x 26 columns]
```

7 Run OLS explanatory for 4 categories

7.1 a. Risk Regressions:

7.1.1 Ret(i) = a + b1 * MktExposure(i) + b2 * SizeExposure(i) + b3 * ValueExposure(i) + e

[399]:
```python
ffexp
```

[399]:
```
                         Name   RetYTD TICKER  R-squared  Adj. R-squared  \
0        Agilent Technologies  -0.2080      A   0.498110        0.471223
1                  Alcoa Corp   0.4731     AA   0.539152        0.514464
2        American Airlines Gp   0.0579    AAL   0.528181        0.502905
3     Aarons Holdings Company  -0.1327    AAN   0.388654        0.355308
4                    Aaon Inc  -0.3456   AAON   0.167367        0.122761
…                           …        …      …          …               …
1881              Zendesk Inc   0.2002    ZEN   0.489536        0.462189
1882            Zions Bancorp  -0.0038   ZION   0.795943        0.785011
1883           Zynga Inc Cl A   0.3969   ZNGA   0.176268        0.132140
1884          Zoetis Inc Cl A  -0.2325    ZTS   0.377264        0.343903
1885               Zumiez Inc  -0.1792   ZUMZ   0.443239        0.413412

         const     mktrf       smb       hml
0     0.007241  1.014152 -0.253674 -0.143608
1     0.014194  1.984149  0.527862  1.924844
2    -0.017850  1.315550  0.612825  1.248123
3     0.009753  1.648965  0.242525  0.815613
4     0.008716  0.516779  0.422130 -0.117068
…          …         …         …         …
1881  0.010584  1.026803  1.363839 -0.774104
1882  0.004964  1.084739  0.869374  1.151468
1883  0.012054  0.101034  1.200987 -0.714916
1884  0.014726  0.728418 -0.560814 -0.179065
1885  0.009920  1.249317  2.336902  0.499124

[1886 rows x 9 columns]
```

7.1.2 Interpret and explain your findings (focus on R2, Adj R2 and coefficients)

[400]:
```python
ffexp['constant'] = 1 # intercept
Xff = ffexp[['mktrf', 'smb', 'hml'        ,'constant']]
yff = ffexp['RetYTD']

modelff = sm.OLS(yff, Xff)
resultsff = modelff.fit()
print(resultsff.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 RetYTD   R-squared:                       0.095
Model:                            OLS   Adj. R-squared:                  0.093
Method:                 Least Squares   F-statistic:                     65.53
Date:                Thu, 28 Apr 2022   Prob (F-statistic):           2.66e-40
Time:                        02:02:37   Log-Likelihood:                 79.981
No. Observations:                1886   AIC:                            -152.0
Df Residuals:                    1882   BIC:                            -129.8
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
mktrf          0.0082      0.009      0.952      0.341      -0.009       0.025
smb           -0.0120      0.004     -2.739      0.006      -0.021      -0.003
hml            0.1068      0.008     13.818      0.000       0.092       0.122
constant      -0.0981      0.011     -8.780      0.000      -0.120      -0.076
==============================================================================
Omnibus:                      642.277   Durbin-Watson:                   2.005
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3600.892
Skew:                           1.493   Prob(JB):                         0.00
Kurtosis:                       9.076   Cond. No.                         4.56
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

7.1.3 Summary

- Based on the **R2 and AdjR2**, we found that this model using **Risk Exposure Betas** indicates only **9.5% / 9.3%** of RetYTD could be explained by those 3 independent variables.
- The model summary shows that the coefficients of **Size Risk Exposure Beta** and **Value Risk Exposure Beta** are **statistically significant at 5%**. In addition, **|t-values|** for two independent variables are **|-2.739| and |13.818|**, respectively and both are **more than**

1.96, which also proves that coefficients for those 2 variables are **significant different from 0 at 5% level**.

- **On average, for 0.5 increase in *Market Risk Exposure Beta*, the RetYTD will increase 0.0082.95% of coefficients of Market Risk Exposure Beta** are within [-0.009, 0.025].
- **On average, for 0.5 increase in *Size Risk Exposure Beta*, the RetYTD will decrease by 0.0120.95% of coefficients of Size Risk Exposure Beta** are within [-0.021, -0.003]. In 1886 company stocks, **76% are small company stocks** which indicates that **positive Size Risk Exposure Beta** will have a corresponding **negative cofficients, which drop RetYTD**
- **On average, for 0.5 increase in *Value Risk Exposure Beta*, the RetYTD will increase by 0.1068**. **95% of coefficients of Value Risk Exposure Beta** are within [0.092, 0.122]. In 1886 company stocks **73% are value company stocks** which indicates that **positive Value Risk Exposure Beta** will have a corresponding **positive cofficients, which increase RetYTD**
- In summary, the 25th-75th range of each risk exposure beta is around 0.7 to 1.0. **If we assume the same 0.5 increase in each of 3 risk exposure betas, *Value Risk Exposure Beta* will cause the most impact and a increase in RetYTD by 0.1068.**

7.2 b. Financial Characteristics:

7.2.1 Ret(i) = a + c1 * Ratio1(i) + c2 * Ratio2(i) + … + c10 * Ratio10 (i) + e

[401]:
```
fin_ratios
```

[401]:

	Ticker	Name	RetYTD	B/P	E/P	ebit/P	\
0	A	Agilent Technologies	-0.2080	0.111695	0.025079	0.029204	
1	AA	Alcoa Corp	0.4731	0.425940	0.039111	0.189357	
2	AAL	American Airlines Gp	0.0579	-0.337273	-0.171320	-0.378436	
3	AAN	Aarons Holdings Company	-0.1327	0.940491	0.143967	0.206699	
4	AAON	Aaon Inc	-0.3456	0.111730	0.014083	0.017648	
…	…	…	…	…	…	…	
1881	ZEN	Zendesk Inc	0.2002	0.038577	-0.017636	-0.011941	
1882	ZION	Zions Bancorp	-0.0038	0.733347	0.117891	0.188062	
1883	ZNGA	Zynga Inc Cl A	0.3969	0.430106	-0.014402	0.032535	
1884	ZTS	Zoetis Inc Cl A	-0.2325	0.039394	0.017664	0.024618	
1885	ZUMZ	Zumiez Inc	-0.1792	0.586912	0.080961	0.108059	

	sale/P	rent_ratio	quick_ratio	debt_ratio	ROA	ROE	\
0	0.130970	0.054545	1.738290	0.496590	0.113031	0.224531	
1	1.107882	-11.438228	0.952529	0.581764	0.028552	0.091824	
2	2.568684	7.315605	0.817689	1.110431	-0.029985	0.271526	
3	2.416822	0.889688	2.044556	0.501711	0.076276	0.153076	
4	0.128111	6.540488	0.945752	0.283014	0.090372	0.126044	

```
…         …           …           …          …        …        …
1881  0.105556    5.138318    1.556646    0.800423 -0.091236 -0.457146
1882  0.310130    4.512843    0.307874    0.919925  0.012114  0.151280
1883  0.387066   24.118042    1.054497    0.510623 -0.016386 -0.033484
1884  0.067515    3.152676    2.786311    0.673094  0.146547  0.448382
1885  1.052171    4.997809    2.051261    0.446498  0.076352  0.137943

           ATR
0     0.590285
1     0.808785
2     0.449576
3     1.280475
4     0.822106
…          …
1881  0.546083
1882  0.031867
1883  0.440406
1884  0.560144
1885  0.992275

[1886 rows x 13 columns]
```

7.2.2 Interpret and explain your findings (focus on R2, Adj R2 and coefficients)

[402]:
```
fin_ratios.describe().T
```

[402]:
```
             count      mean        std        min       25%       50%  \
RetYTD      1886.0 -0.063484   0.243801  -0.787400 -0.199450 -0.086750
B/P         1886.0  0.414033   0.343580  -0.337273  0.161112  0.337004
E/P         1886.0  0.028158   0.102961  -0.521213  0.008113  0.036704
ebit/P      1886.0  0.060454   0.107562  -0.378436  0.019641  0.056821
sale/P      1886.0  0.687720   0.854932   0.000060  0.176223  0.350729
rent_ratio  1886.0  3.018709  16.455807 -96.808868  0.439140  3.695886
quick_ratio 1886.0  1.906921   2.180792  -2.169569  0.765277  1.213342
debt_ratio  1886.0  0.626141   0.241628   0.086737  0.460233  0.633081
ROA         1886.0  0.024837   0.128383  -0.612289  0.007584  0.034187
ROE         1886.0  0.075062   0.546444  -3.335815  0.024152  0.112438
ATR         1886.0  0.626616   0.549811   0.000132  0.185607  0.511923

                 75%        max
RetYTD      0.038750   1.735700
B/P         0.613868   1.377565
E/P         0.076322   0.231730
ebit/P      0.113822   0.329366
sale/P      0.814565   4.063661
rent_ratio  6.611482  52.040147
```

```
quick_ratio  2.122198  10.989332
debt_ratio   0.802777   1.198789
ROA          0.079409   0.266337
ROE          0.212665   1.555145
ATR          0.900111   2.370510
```

[403]:
```python
pd.DataFrame((fin_ratios.describe().T.iloc[:,[6]].values - fin_ratios.
    ↪describe().T.iloc[:,[4]].values), index = fin_ratios.describe().T.index,␣
    ↪columns=['25th - 75th / IQR']).sort_values('25th - 75th / IQR')
```

[403]:

	25th - 75th / IQR
E/P	0.068209
ROA	0.071826
ebit/P	0.094181
ROE	0.188513
RetYTD	0.238200
debt_ratio	0.342543
B/P	0.452756
sale/P	0.638342
ATR	0.714504
quick_ratio	1.356921
rent_ratio	6.172342

[404]:
```python
fin_ratios['constant'] = 1 # intercept
Xfr = fin_ratios.drop(columns=['RetYTD','Ticker', 'Name '])
yfr = fin_ratios['RetYTD']

modelfr = sm.OLS(yfr, Xfr)
resultsfr = modelfr.fit()
print(resultsfr.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 RetYTD   R-squared:                       0.089
Model:                            OLS   Adj. R-squared:                  0.085
Method:                 Least Squares   F-statistic:                     18.41
Date:                Thu, 28 Apr 2022   Prob (F-statistic):           1.59e-32
Time:                        02:02:37   Log-Likelihood:                 74.610
No. Observations:                1886   AIC:                            -127.2
Df Residuals:                    1875   BIC:                            -66.26
Df Model:                          10
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
B/P             0.1360      0.021      6.396      0.000       0.094       0.178
E/P            -0.3917      0.107     -3.675      0.000      -0.601      -0.183
```

```
ebit/P          0.1209       0.093       1.305       0.192       -0.061        0.303
sale/P          0.0432       0.011       4.081       0.000        0.022        0.064
rent_ratio      0.0003       0.000       0.926       0.354       -0.000        0.001
quick_ratio    -0.0031       0.003      -1.002       0.317       -0.009        0.003
debt_ratio      0.0306       0.028       1.082       0.279       -0.025        0.086
ROA             0.3862       0.069       5.621       0.000        0.251        0.521
ROE            -0.0038       0.011      -0.338       0.736       -0.026        0.018
ATR            -0.0525       0.017      -3.142       0.002       -0.085       -0.020
constant       -0.1364       0.028      -4.793       0.000       -0.192       -0.081
==============================================================================
Omnibus:                      637.806   Durbin-Watson:                   2.037
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3721.527
Skew:                           1.467   Prob(JB):                         0.00
Kurtosis:                       9.225   Cond. No.                         409.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

7.2.3 Summary

- Based on the **R2 and AdjR2**, we found that this model using **10 Financial Ratios** indicates only **8.9% / 8.5%** of RetYTD could be explained by those 10 independent variables.
- The model summary shows that the coefficients of **B/P*, E/P*, Sale/P*, ROA*, ATR** are **statistically significant at 5%**. In addition, **|t-values|** for five independent variables are **|6.396|, |-3.675|, |4.081|, |5.621|, |-3.142|**, respectively and both are **more than 1.96**, which also proves that coefficients for those 5 variables are **significant different from 0 at 5% level**.
- **On average, for 0.1 increase in *B/P*, the RetYTD will increase 0.0136**. The higher the *B/P* is, the more YTD stock return will be increased during early 2022. **In other words, undervalued stocks in 2021 will increase RetYTD in 2022**. **95% of coefficients of B/P** are within [0.094, 0.178].
- **On average, for 0.1 increase in *E/P*, the RetYTD will decrease 0.0392** . The higher the *E/P* is, the more YTD stock return will be decreased during early 2022. **In other words, investors are pessimistic regarding future earnings from the stock due to the higher E/P**. **95% of coefficients of E/P** are within [-0.601, -0.183].
- **On average, for 0.1 increase in *Sales/P*, the RetYTD will increase 0.0043**. The higher the *Sales/P* is, the more YTD stock return will be increased during early 2022. **In other words, the investment to the stocks with higher Sales / P becomes more attactive**. **95% of coefficients of Sales/P** are within [0.022, 0.064].
- **On average, for 0.1 increase in *ROA*, the RetYTD will increase 0.039**. The higher the *ROA* is, the more YTD stock return will be increased during early 2022. In other words, the higher assets management's effectiveness will have negatively impact on YTD stock return

during early 2022 .**95% of coefficients of ROA** are within [0.251, 0.521].

- **On average, for 0.1 increase in *ATR*, the RetYTD will decrease 0.0052**. The higher the *ATR* is, the more YTD stock return will be decreased during early 2022. In other words, the higher assets management's effectiveness to generate revenue is, the more YTD stock return will be decreased during early 2022 .**95% of coefficients of ROA** are within [-0.085, -0.020].
- In sum, **E/P and ROA* will have more significant effect on the valuations of stocks in 2022.**, since the main reason is that E/P has directly relationship to the company's value in the market. Furthermore, ROA could be a good indicator of companies' profitbaility in the future. For instance, high ROA suggests tha the more profit is being generated from each dollar invested in assets.

7.3 c. Industry Dummies:

7.3.1 Ret(i) = a + coefficients * IndustryDummies + e

[405]:
```python
# add constant column to the original dataframe
industry_new['constant'] = 1
industry_new
```

[405]:
```
      Ticker  RetYTD  ggroup_1010  ggroup_1510  ggroup_2010  ggroup_2020  \
0        AIR  0.2944            0            0            1            0
1        AAL  0.0579            0            0            0            0
2        PNW  0.0985            0            0            0            0
3        ABT -0.1638            0            0            0            0
4        AMD -0.3533            0            0            0            0
…      …     …          …          …          …          …
1881     KRG  0.0275            0            0            0            0
1882     LYB  0.1664            0            1            0            0
1883     FRO  0.3380            1            0            0            0
1884    ALLE -0.1888            0            0            1            0
1885     LPG  0.2427            1            0            0            0

       ggroup_2030  ggroup_2510  ggroup_2520  ggroup_2530  …  ggroup_4020  \
0                0            0            0            0  …            0
1                1            0            0            0  …            0
2                0            0            0            0  …            0
3                0            0            0            0  …            0
4                0            0            0            0  …            0
…          …          …          …          …  …   …
1881             0            0            0            0  …            0
1882             0            0            0            0  …            0
1883             0            0            0            0  …            0
1884             0            0            0            0  …            0
1885             0            0            0            0  …            0
```

```
      ggroup_4030  ggroup_4510  ggroup_4520  ggroup_4530  ggroup_5010  \
0               0            0            0            0            0
1               0            0            0            0            0
2               0            0            0            0            0
3               0            0            0            0            0
4               0            0            0            1            0
…             …          …          …          …          …
1881            0            0            0            0            0
1882            0            0            0            0            0
1883            0            0            0            0            0
1884            0            0            0            0            0
1885            0            0            0            0            0

      ggroup_5020  ggroup_5510  ggroup_6010  constant
0               0            0            0         1
1               0            0            0         1
2               0            1            0         1
3               0            0            0         1
4               0            0            0         1
…             …          …          …          …
1881            0            0            1         1
1882            0            0            0         1
1883            0            0            0         1
1884            0            0            0         1
1885            0            0            0         1

[1886 rows x 27 columns]
```

7.3.2 Interpret and explain your findings (focus on R2, Adj R2 and coefficients)

- The reason I choose to drop ggroup_1010 is that its mean value of RetYTD in ggroup_1010 is much different from that mean values of RetYTD in other industries, which could help model generate all statistically significant cofficients of other industries.

[406]:
```python
# Define x as a subset of original dataframe
# only keep industy dummy variables and drop one industry indicator (let's␣
↪choose "ggroup_1010")
Xind = industry_new.drop(columns=['Ticker', 'RetYTD', 'ggroup_1010'])
# Define y as a series
yind = industry_new['RetYTD']
# pass x as a dataframe, while pass y as a series
sm.OLS(yind, Xind).fit().summary()
```

[406]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

OLS Regression Results

Dep. Variable:	RetYTD	R-squared:	0.326
Model:	OLS	Adj. R-squared:	0.317
Method:	Least Squares	F-statistic:	39.07
Date:	Thu, 28 Apr 2022	Prob (F-statistic):	6.29e-141
Time:	02:02:37	Log-Likelihood:	357.63
No. Observations:	1886	AIC:	-667.3
Df Residuals:	1862	BIC:	-534.2
Df Model:	23		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
ggroup_1510	-0.4572	0.031	-14.692	0.000	-0.518	-0.396
ggroup_2010	-0.6056	0.028	-21.924	0.000	-0.660	-0.551
ggroup_2020	-0.5502	0.034	-15.980	0.000	-0.618	-0.483
ggroup_2030	-0.5825	0.040	-14.646	0.000	-0.660	-0.504
ggroup_2510	-0.7230	0.047	-15.540	0.000	-0.814	-0.632
ggroup_2520	-0.7236	0.034	-21.018	0.000	-0.791	-0.656
ggroup_2530	-0.5776	0.034	-16.849	0.000	-0.645	-0.510
ggroup_2550	-0.6663	0.032	-20.932	0.000	-0.729	-0.604
ggroup_3010	-0.4367	0.054	-8.070	0.000	-0.543	-0.331
ggroup_3020	-0.4763	0.038	-12.449	0.000	-0.551	-0.401
ggroup_3030	-0.6688	0.054	-12.359	0.000	-0.775	-0.563
ggroup_3510	-0.5815	0.030	-19.639	0.000	-0.640	-0.523
ggroup_3520	-0.6790	0.028	-24.154	0.000	-0.734	-0.624
ggroup_4010	-0.6029	0.028	-21.756	0.000	-0.657	-0.549
ggroup_4020	-0.6375	0.031	-20.342	0.000	-0.699	-0.576
ggroup_4030	-0.5113	0.035	-14.719	0.000	-0.579	-0.443
ggroup_4510	-0.6237	0.030	-20.963	0.000	-0.682	-0.565
ggroup_4520	-0.6871	0.032	-21.346	0.000	-0.750	-0.624
ggroup_4530	-0.7678	0.036	-21.470	0.000	-0.838	-0.698
ggroup_5010	-0.5032	0.061	-8.314	0.000	-0.622	-0.384
ggroup_5020	-0.5859	0.039	-15.090	0.000	-0.662	-0.510
ggroup_5510	-0.4743	0.035	-13.464	0.000	-0.543	-0.405
ggroup_6010	-0.5509	0.029	-18.884	0.000	-0.608	-0.494
constant	0.5150	0.023	22.140	0.000	0.469	0.561

Omnibus:	403.265	Durbin-Watson:	2.008
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2314.656
Skew:	0.877	Prob(JB):	0.00
Kurtosis:	8.136	Cond. No.	26.1

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly

```
specified.
"""
```

7.3.3 Summary

- In terms of the explanatory power of these regressions, **32.6%/31.7% of variation in Year to date Stock Return has been explained by the variation in 24 industry indicators**. Although the model with 0.326 R-squared usually couldn't be considered as having a high explanatory power, this value in the stock market which is unstable and erratic could be deemed as a reasonbale value.

- The model summary shows that the coefficients of **all 24 industries** are **statistically significant at 5%**. In addition, **|t-values|** for 24 independent variables are **more than 1.96**, which also proves that coefficients for those 24 variables are **significant different from 0 at 5% level**.

- **The lowest average stock returns** is around **-0.25** and its corresponding industry is **Semiconductors & Semiconductor Equipment(4530)**. Despite the continuous chip shortage starting from the outbreak of COVID 19, The 2022 stock price for the semiconductor industry appears to go down. Several leading stocks like Nvidia, Taiwan Semiconductor and Intel are **all down over 20% year to date** due to **downgrades** and the **increased risk of consumer spending decreasing**. For instance, On April 11th, Baird downgraded Nvidia to neutral and reduced its price target from **$360 to $225**, because there exists a slowdown in consumer spending, especially in China's consumer market, the smartphone market shifts and headwinds caused by the Russian embargo. Furthermore, South Korea, a hub for semiconductor production, **relies on Neon, xenon and krypton** used in the production of advanced chips and **imported from Russia and Ukraine**. However, **the recent tension between Russia and Ukraine and accompanying sanctions on Russia have cut off its supply**.

- **The highest average stock returns is around 0.52** and its corresponding industry is **Energy(1010)**. In general, the Energy sector includes energy equipment and services, and oil, gas and consumable fuels. With the gradual reopening of the global economy, the demand in oil has incredibly increased. In the other hand, **the total oil inventories have declined due to some cautious producers like OPEC and U.S. Especially, rising tensions amid Russia/Ukraine war exacerbated the oil shortage, which in turn highly increase the oil prices**. In addition, **Valuations in the Energy sector are attractive relative to the other sectors**. Although there are the strong gains in energy stock prices, they have not kept up with rapidly rising earnings expectations.

7.4 d. Combined Regressions:

7.4.1 Interpret and explain your findings (focus on R2, Adj R2 and coefficients)

[407]:
```
fin_ratios
```

[407]:

```
      Ticker                     Name  RetYTD       B/P       E/P    ebit/P  \
0          A     Agilent Technologies -0.2080  0.111695  0.025079  0.029204
1         AA               Alcoa Corp  0.4731  0.425940  0.039111  0.189357
2        AAL     American Airlines Gp  0.0579 -0.337273 -0.171320 -0.378436
3        AAN  Aarons Holdings Company -0.1327  0.940491  0.143967  0.206699
4       AAON                 Aaon Inc -0.3456  0.111730  0.014083  0.017648
…        …                        …     …         …         …         …
1881     ZEN              Zendesk Inc  0.2002  0.038577 -0.017636 -0.011941
1882    ZION            Zions Bancorp -0.0038  0.733347  0.117891  0.188062
1883    ZNGA           Zynga Inc Cl A  0.3969  0.430106 -0.014402  0.032535
1884     ZTS          Zoetis Inc Cl A -0.2325  0.039394  0.017664  0.024618
1885    ZUMZ               Zumiez Inc -0.1792  0.586912  0.080961  0.108059

        sale/P  rent_ratio  quick_ratio  debt_ratio       ROA       ROE  \
0     0.130970    0.054545     1.738290    0.496590  0.113031  0.224531
1     1.107882  -11.438228     0.952529    0.581764  0.028552  0.091824
2     2.568684    7.315605     0.817689    1.110431 -0.029985  0.271526
3     2.416822    0.889688     2.044556    0.501711  0.076276  0.153076
4     0.128111    6.540488     0.945752    0.283014  0.090372  0.126044
…          …           …           …          …         …         …
1881  0.105556    5.138318     1.556646    0.800423 -0.091236 -0.457146
1882  0.310130    4.512843     0.307874    0.919925  0.012114  0.151280
1883  0.387066   24.118042     1.054497    0.510623 -0.016386 -0.033484
1884  0.067515    3.152676     2.786311    0.673094  0.146547  0.448382
1885  1.052171    4.997809     2.051261    0.446498  0.076352  0.137943

           ATR  constant
0     0.590285         1
1     0.808785         1
2     0.449576         1
3     1.280475         1
4     0.822106         1
…          …        …
1881  0.546083         1
1882  0.031867         1
1883  0.440406         1
1884  0.560144         1
1885  0.992275         1

[1886 rows x 14 columns]
```

[408]:

```
df_combine= pd.
 ↪concat([ffexp[['mktrf','smb'        ,'hml']],fin_ratios[['Ticker','B/P',␣
 ↪'sale/P', 'debt_ratio', 'ROA', 'ATR']]], axis = 1 )

df_combine = pd.merge(df_combine,industry_new,on = 'Ticker',how = 'inner')
```

```python
# Define x as a subset of original datafroame
# only keep industy dummy variables and drop one industry indicator (let's choose "gsector_10")
Xcom = df_combine.drop(columns = ['Ticker', 'RetYTD',
       'ggroup_1010'])
# Define y as a series
ycom = df_combine['RetYTD']
# pass x as a dataframe, while pass y as a series
sm.OLS(ycom, Xcom).fit().summary()
```

[408]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 RetYTD   R-squared:                       0.386
Model:                            OLS   Adj. R-squared:                  0.376
Method:                 Least Squares   F-statistic:                     37.65
Date:                Thu, 28 Apr 2022   Prob (F-statistic):          1.00e-171
Time:                        02:02:38   Log-Likelihood:                 446.72
No. Observations:                1886   AIC:                            -829.4
Df Residuals:                    1854   BIC:                            -652.1
Df Model:                          31
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
mktrf          -0.0165      0.008     -2.025      0.043      -0.033      -0.001
smb            -0.0166      0.004     -3.904      0.000      -0.025      -0.008
hml             0.0605      0.009      6.909      0.000       0.043       0.078
B/P             0.0944      0.020      4.678      0.000       0.055       0.134
sale/P          0.0183      0.009      1.989      0.047       0.000       0.036
debt_ratio      0.0776      0.025      3.065      0.002       0.028       0.127
ROA             0.1801      0.046      3.945      0.000       0.091       0.270
ATR            -0.0362      0.015     -2.372      0.018      -0.066      -0.006
ggroup_1510    -0.4322      0.031    -13.809      0.000      -0.494      -0.371
ggroup_2010    -0.5561      0.028    -19.724      0.000      -0.611      -0.501
ggroup_2020    -0.5071      0.035    -14.539      0.000      -0.576      -0.439
ggroup_2030    -0.5717      0.039    -14.568      0.000      -0.649      -0.495
ggroup_2510    -0.6929      0.045    -15.262      0.000      -0.782      -0.604
ggroup_2520    -0.6808      0.034    -19.841      0.000      -0.748      -0.613
ggroup_2530    -0.5306      0.035    -15.355      0.000      -0.598      -0.463
ggroup_2550    -0.6187      0.033    -18.936      0.000      -0.683      -0.555
ggroup_3010    -0.3902      0.057     -6.897      0.000      -0.501      -0.279
ggroup_3020    -0.4374      0.039    -11.111      0.000      -0.515      -0.360
ggroup_3030    -0.6080      0.054    -11.337      0.000      -0.713      -0.503
ggroup_3510    -0.4807      0.032    -15.253      0.000      -0.542      -0.419
ggroup_3520    -0.5177      0.032    -16.322      0.000      -0.580      -0.456
```

```
ggroup_4010    -0.6662      0.031    -21.745      0.000      -0.726      -0.606
ggroup_4020    -0.6461      0.032    -20.200      0.000      -0.709      -0.583
ggroup_4030    -0.5627      0.036    -15.437      0.000      -0.634      -0.491
ggroup_4510    -0.5171      0.032    -16.251      0.000      -0.579      -0.455
ggroup_4520    -0.6200      0.033    -19.057      0.000      -0.684      -0.556
ggroup_4530    -0.6710      0.036    -18.527      0.000      -0.742      -0.600
ggroup_5010    -0.5021      0.059     -8.449      0.000      -0.619      -0.386
ggroup_5020    -0.5329      0.039    -13.706      0.000      -0.609      -0.457
ggroup_5510    -0.4738      0.038    -12.570      0.000      -0.548      -0.400
ggroup_6010    -0.5323      0.030    -17.549      0.000      -0.592      -0.473
constant        0.4001      0.037     10.897      0.000       0.328       0.472
==============================================================================
Omnibus:                      439.518   Durbin-Watson:                   2.026
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2553.076
Skew:                           0.964   Prob(JB):                         0.00
Kurtosis:                       8.364   Cond. No.                         61.9
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

- In terms of the explanatory power of these regressions, **38.6%/37.6% of variation in Year to date Stock Return has been explained by the variation in 3 risk exposures + 5 financial ratios + 23 industry indicators**. **The Adj R squared increased from 8.5% to 37.6%**.
- Although the model with 0.376 Adj R-squared usually couldn't be considered as having a high explanatory power, this value in the stock market which is unstable and erratic could be deemed as a reasonbale value.
- The model summary shows that the coefficients of **all 31 independent variables** are **statistically significant at 5%**. In addition, **|t-values|** for 31 independent variables are **more than 1.96**, which also proves that coefficients for those 31 variables are **significant different from 0 at 5% level**.

[ ]:
```
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic
```

[410]:
```
!jupyter nbconvert --to pdf '/content/drive/MyDrive/BA870_Individual␣
↪Project_Ji_Qi.ipynb'
```

```
[NbConvertApp] Converting notebook /content/drive/MyDrive/BA870_Individual
Project_Ji_Qi.ipynb to pdf
[NbConvertApp] Support files will be in BA870_Individual Project_Ji_Qi_files/
[NbConvertApp] Making directory ./BA870_Individual Project_Ji_Qi_files
[NbConvertApp] Making directory ./BA870_Individual Project_Ji_Qi_files
[NbConvertApp] Making directory ./BA870_Individual Project_Ji_Qi_files
```

```
[NbConvertApp] Making directory ./BA870_Individual Project_Ji_Qi_files
[NbConvertApp] Making directory ./BA870_Individual Project_Ji_Qi_files
[NbConvertApp] Making directory ./BA870_Individual Project_Ji_Qi_files
[NbConvertApp] Writing 198435 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', './notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', './notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 377306 bytes to /content/drive/MyDrive/BA870_Individual
Project_Ji_Qi.pdf
```