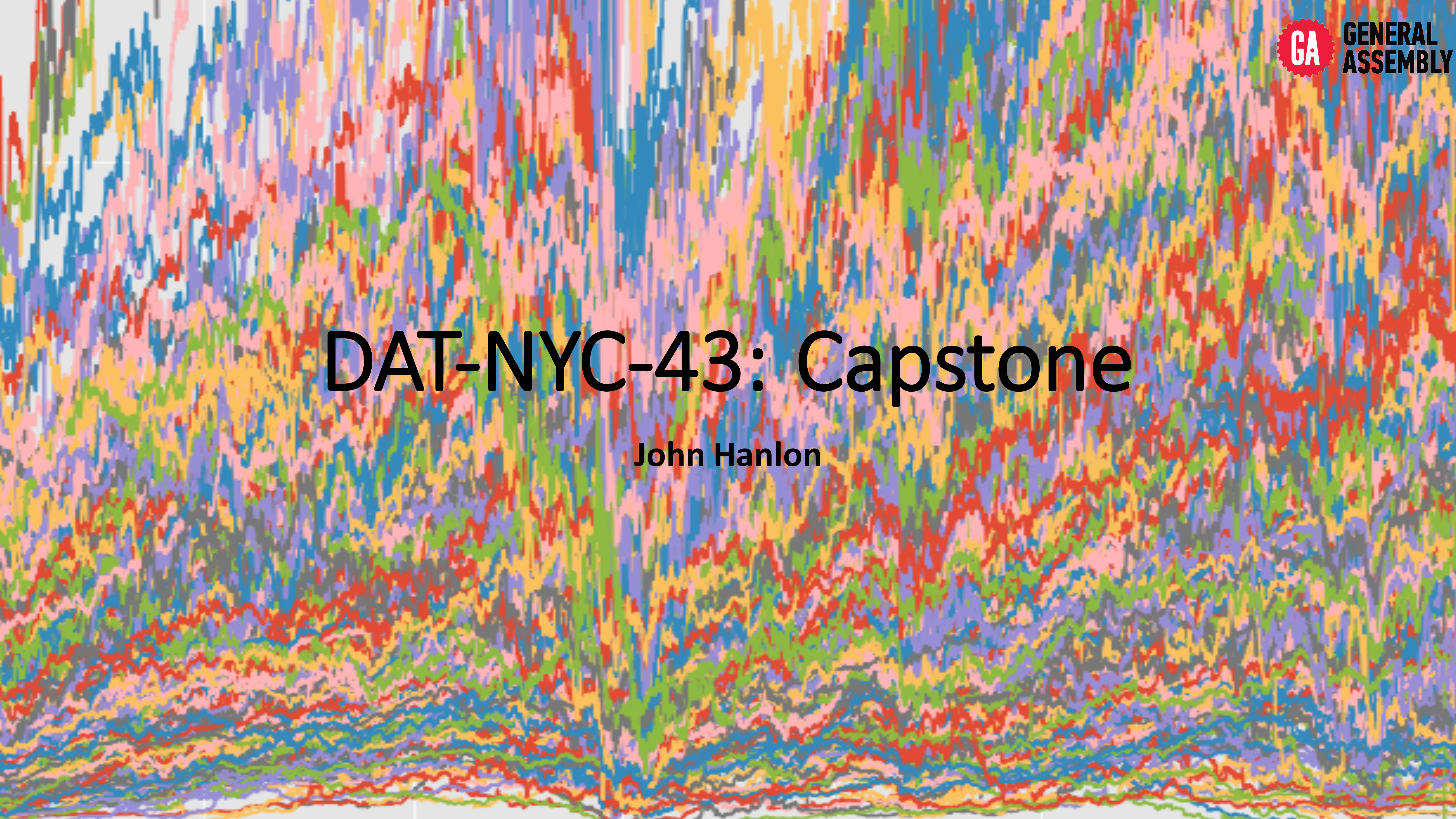


DAT-NYC-43: Capstone

John Hanlon

Executive Summary

- Aim and Goals
- Datasets
- Methods, Models, and Analysis
- Risks and Assumptions
- Conclusion

Aim and Goals

- The original goal of this exercise was to create a functioning mean reversion trading strategy
  - Similar to our session on time series
  - This became complex very quickly
- After reviewing the datasets in scope, the reassessed goal of this exercise was to develop a machine learning classification algorithm to identify which securities might outperform the S&P 500
- Classification features were financial performance metrics and ratios
- Labels were assigned to a binary variable ‘status’
  - 0: Underperform
  - 1: Outperform

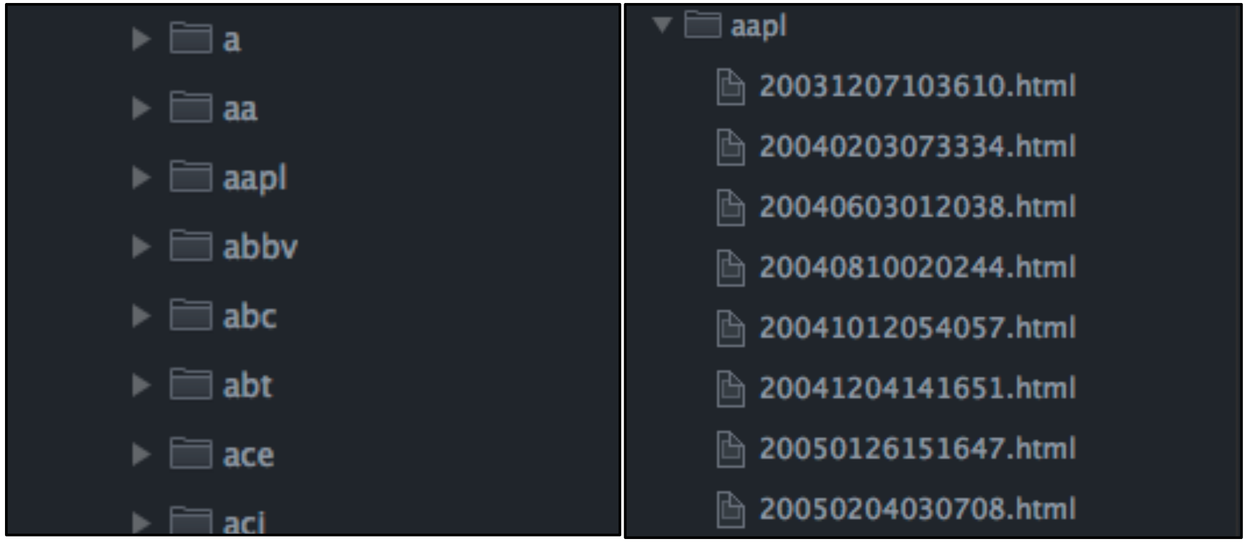
Executive Summary

- Aim and Goals
- Datasets
- Methods, Models, and Analysis
- Risks and Assumptions
- Conclusion

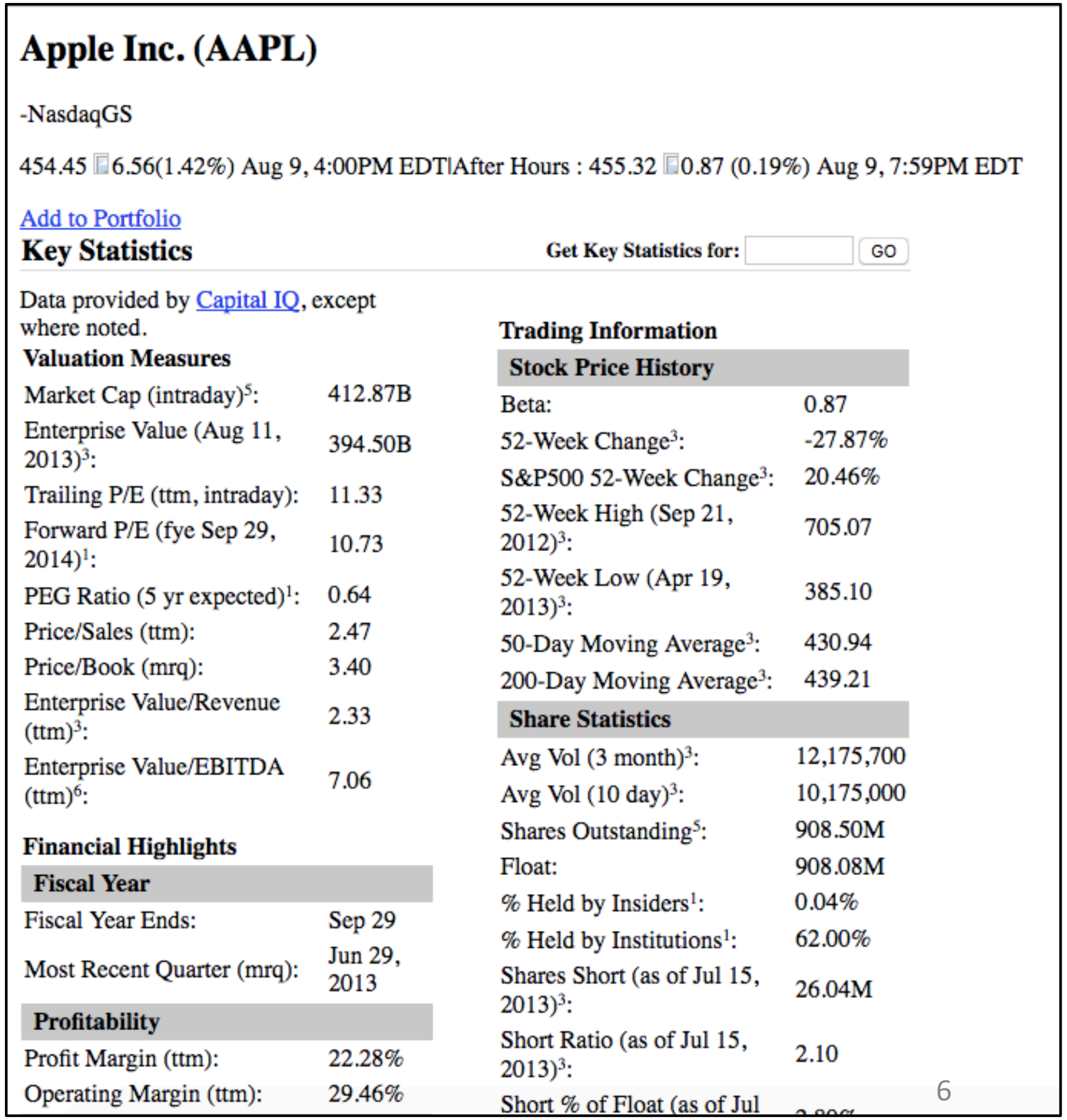
Datasets

- Several datasets were used to perform this analysis:
  - a.) Historical fundamental data and key metrics from S&P 500 listed companies between 2003-2013
  - b.) Historical SPY adjusted close price data (2003-2016)
  - c.) Historical price data for all S&P 500 listed companies (2003-2016)
- These raw dataset were then joined into two data frames:
  - a.) Daily adjusted close prices for all of the securities in scope
  - b.) Merged dataset of all adjusted close prices and features for the securities in scope

Datasets: Fundamental Data




```
Total Debt/Equity (mrq):</td><td class="yfnc_tabledata1">13.75</td></tr><tr><td
class="yfnc_tablehead1" width="74%">Current Ratio (mrq):</td><td
class="yfnc_tabledata1">1.88</td></tr><tr><td class="yfnc_tablehead1"
width="74%">Book Value Per Share (mrq):</td><td
class="yfnc_tabledata1">135.79</td></tr></table></td></tr></table><table
class="yfnc_datamodoutline1" width="100%" cellpadding="0" cellspacing="0"
border="0"><tr><td><table width="100%" cellpadding="2" cellspacing="1"
border="0"><tr><td colspan="2" style="padding: 3px 8px;background-
color:#CCC;"><b>Cash Flow Statement</b></td></tr><tr><td
class="yfnc_tablehead1" width="74%">Operating Cash Flow (ttm):</td><td
class="yfnc_tabledata1">52.89B</td></tr><tr><td class="yfnc_tablehead1"
width="74%">Levered Free Cash Flow (ttm):</td><td
class="yfnc_tabledata1">31.00B</td></tr></table></td></tr></table><table
```



Apple Inc. (AAPL)

-NasdaqGS

454.45 6.56(1.42%) Aug 9, 4:00PM EDT|After Hours : 455.32 0.87 (0.19%) Aug 9, 7:59PM EDT

Add to Portfolio

Key Statistics Get Key Statistics for: GO

Data provided by Capital IQ, except where noted.

Valuation Measures

Market Cap (intraday)[5]:	412.87B
Enterprise Value (Aug 11, 2013)[3]:	394.50B
Trailing P/E (ttm, intraday):	11.33
Forward P/E (fye Sep 29, 2014)[1]:	10.73
PEG Ratio (5 yr expected)[1]:	0.64
Price/Sales (ttm):	2.47
Price/Book (mrq):	3.40
Enterprise Value/Revenue (ttm)[3]:	2.33
Enterprise Value/EBITDA (ttm)[6]:	7.06

Financial Highlights

Fiscal Year	
Fiscal Year Ends:	Sep 29
Most Recent Quarter (mrq):	Jun 29, 2013
Profitability	
Profit Margin (ttm):	22.28%
Operating Margin (ttm):	29.46%

Trading Information

Stock Price History	
Beta:	0.87
52-Week Change[3]:	-27.87%
S&P500 52-Week Change[3]:	20.46%
52-Week High (Sep 21, 2012)[3]:	705.07
52-Week Low (Apr 19, 2013)[3]:	385.10
50-Day Moving Average[3]:	430.94
200-Day Moving Average[3]:	439.21
Share Statistics	
Avg Vol (3 month)[3]:	12,175,700
Avg Vol (10 day)[3]:	10,175,000
Shares Outstanding[5]:	908.50M
Float:	908.08M
% Held by Insiders[1]:	0.04%
% Held by Institutions[1]:	62.00%
Shares Short (as of Jul 15, 2013)[3]:	26.04M
Short Ratio (as of Jul 15, 2013)[3]:	2.10
Short % of Float (as of Jul	[illegible]

Datasets: Historical Price & Adj. Close Stock Data

Date	a	aa	aapl	abbv	abc	abt
11/21/03	17.6455689		1.31942819		12.9936481	12.9396825
11/24/03	17.978237		1.37603088		13.1716432	13.2070708
11/25/03	18.0277833		1.34545242		13.1341706	13.178422
11/26/03	18.1905784		1.34805484		13.0123844	13.2166204
11/28/03	18.3533735		1.36041635		13.0685934	13.1020254
12/1/03	18.3533735		1.4124648		13.4152156	13.3312153
12/2/03	18.0419394		1.4014045		13.6025789	13.407612
12/3/03	18.3675296		1.36822362		13.4245838	13.4171616
12/4/03	18.1622662		1.37603088		13.4245838	13.4362607
12/5/03	17.4686177		1.35651271		13.4245838	13.3216658
12/8/03	17.82252		1.36952483		13.4245838	13.4267112
12/9/03	17.4332275		1.33048849		13.1716432	13.5031078
12/10/03	17.2208861		1.32593425		12.9842799	13.4458103
12/11/03	17.7092713		1.37993452		12.7313394	13.5604053
12/12/03	17.6880371		1.35911514		12.7688121	13.5795045
12/15/03	17.5960225		1.31227153		13.0685934	13.5317566
12/16/03	17.82252		1.3090185		13.0123844	13.5986036
12/17/03	17.6384908		1.29340397		12.9842799	13.5126574
12/18/03	18.0560955		1.30381366		13.1341706	13.7991448
12/19/03	18.1622662		1.28169307		13.1154342	13.7513969
12/22/03	18.4099978		1.29145215		13.1716432	13.7227482
12/23/03	18.3462954		1.28884973		13.1622751	13.7800457
12/24/03	18.2188906		1.32788607		13.1716432	13.8850911
12/26/03	18.1410321		1.35195847		13.1622751	13.8755415

```
def stock_prices():
    ''' Searches the quandl api for the adjusted close price of each security in
    our local dataset

    Parameters
    ==========

    Returns
    =======
    df : pandas dataframe
    '''
    df = pd.DataFrame()
    statspath = '{}{}'.format(path, stocklist)
    stock_list = [x[0] for x in os.walk(statspath)]
    print('Fetching stock prices...')

    for each_dir in stock_list[1:]:
      try:
        ticker = each_dir.split('/')[-1]
        query = '{}{}'.format(q, ticker)
        data = quandl.get(query, trim_start='2003-01-01',
                          trim_end='2016-11-30',
                          authtoken=api_key)
        data[ticker] = data['Adj_Close']
        df = pd.concat([df, data[ticker]], axis=1)
      except Exception as e:
        print(str(e), ticker)

    df.to_csv('{}{}'.format(path, op))
    print('Query completed...')
```

Datasets: Final Appended Data Frame

Date	Unix	Ticker	Price	stock p chang	sp500	sp500 p char	difference	status	DE Ratio	Trailing P/E	Price/Sales	Price/Book
8/2/11 6:45	1312281937	a	26.82	-1.45	112.564667	11.96	-13.41	underperforr	54.36	16.18	2.38	3.71
5/14/13 0:28	1368505738	a	30.68	28.26	153.925584	16.73	11.53	outperform	44.12	13.75	2.19	2.83
9/6/13 6:09	1378462192	a	33.18	23.57	155.49613	23.13	0.44	underperforr	56.39	17.74	2.32	3.31
4/13/04 4:20	1081844410	aa			86.5049554	5.65		underperforr	0.595	24.92	1.33	2.41
9/9/04 16:15	1094760945	aa			86.2687579	12.98		underperforr	0.561	20.88	1.26	2.34
11/24/04 10:37	1101310640	aa			91.4839977	7.99		underperforr	0.53	21.15	1.26	2.34
12/5/04 0:36	1102225002	aa			92.1736943	7.88		underperforr	0.53	20.69	1.25	2.33
1/13/05 7:18	1105618712	aa			91.276144	11.33		underperforr	0.457	18.43	1.1	1.95
2/7/05 3:48	1107766117	aa			93.1846194	6.35		underperforr	0.457	18.39	1.08	1.92
9/3/11 9:21	1315056083	aa			109.631473	17.38		underperforr	48.26	13.78	0.57	0.85
11/2/11 22:04	1320285887	aa			111.791136	17.65		underperforr	50.93	11.23	0.45	0.74
11/5/11 21:29	1320542964	aa			111.791136	17.65		underperforr	50.93	11.47	0.47	0.78
1/2/12 19:42	1325551326	aa			113.869262	15.51		underperforr	50.93	9.08	0.37	0.62
5/19/12 11:51	1337442712	aa			121.046383	27.25		underperforr	55.28	23.1	0.36	0.64
4/23/13 20:02	1366761777	aa			146.985285	21.2		underperforr	53.21	36.21	0.37	0.65
4/26/13 20:55	1367024115	aa			147.413814	20.85		underperforr	53.21	36.92	0.38	0.67
6/3/04 1:20	1086240038	aapl	1.85	169.19	85.6546446	9.29	159.9	outperform	2.921	63.7	1.49	2.33
8/10/04 2:02	1092117764	aapl	2.05	175.12	83.1226079	16.06	159.06	outperform	2.917	54.4	1.51	2.38

Executive Summary

- Aim and Goals
- Datasets
- Methods, Models, and Analysis
- Risks and Assumptions
- Conclusion

Methods, Models, and Analysis

- First, the historical stock price data needed to be pulled and appended into a single data frame for analysis

Methods, Models, and Analysis: Historical Adjusted Close Stock Data

Date	a	aa	aapl	abbv	abc	abt
11/21/03	17.6455689		1.31942819		12.9936481	12.9396825
11/24/03	17.978237		1.37603088		13.1716432	13.2070708
11/25/03	18.0277833		1.34545242		13.1341706	13.178422
11/26/03	18.1905784		1.34805484		13.0123844	13.2166204
11/28/03	18.3533735		1.36041635		13.0685934	13.1020254
12/1/03	18.3533735		1.4124648		13.4152156	13.3312153
12/2/03	18.0419394		1.4014045		13.6025789	13.407612
12/3/03	18.3675296		1.36822362		13.4245838	13.4171616
12/4/03	18.1622662		1.37603088		13.4245838	13.4362607
12/5/03	17.4686177		1.35651271		13.4245838	13.3216658
12/8/03	17.82252		1.36952483		13.4245838	13.4267112
12/9/03	17.4332275		1.33048849		13.1716432	13.5031078
12/10/03	17.2208861		1.32593425		12.9842799	13.4458103
12/11/03	17.7092713		1.37993452		12.7313394	13.5604053
12/12/03	17.6880371		1.35911514		12.7688121	13.5795045
12/15/03	17.5960225		1.31227153		13.0685934	13.5317566
12/16/03	17.82252		1.3090185		13.0123844	13.5986036
12/17/03	17.6384908		1.29340397		12.9842799	13.5126574
12/18/03	18.0560955		1.30381366		13.1341706	13.7991448
12/19/03	18.1622662		1.28169307		13.1154342	13.7513969
12/22/03	18.4099978		1.29145215		13.1716432	13.7227482
12/23/03	18.3462954		1.28884973		13.1622751	13.7800457
12/24/03	18.2188906		1.32788607		13.1716432	13.8850911
12/26/03	18.1410321		1.35195847		13.1622751	13.8755415

```
def stock_prices():

    df = pd.DataFrame()
    statspath = '{}{}'.format(path, stocklist)
    stock_list = [x[0] for x in os.walk(statspath)]
    print('Fetching stock prices...')

    for each_dir in stock_list[1:]:
        try:
            ticker = each_dir.split('/')[-1]
            query = '{}{}'.format(q, ticker)
            data = quandl.get(query, trim_start='2003-11-21',
                              trim_end='2016-11-30',
                              authtoken=api_key)
            data[ticker] = data['Adj_Close']
            df = pd.concat([df, data[ticker]], axis=1)
        except Exception as e:
            print(str(e), ticker)

    df.to_csv('{}{}'.format(path, op))
    print('Query completed...')
```

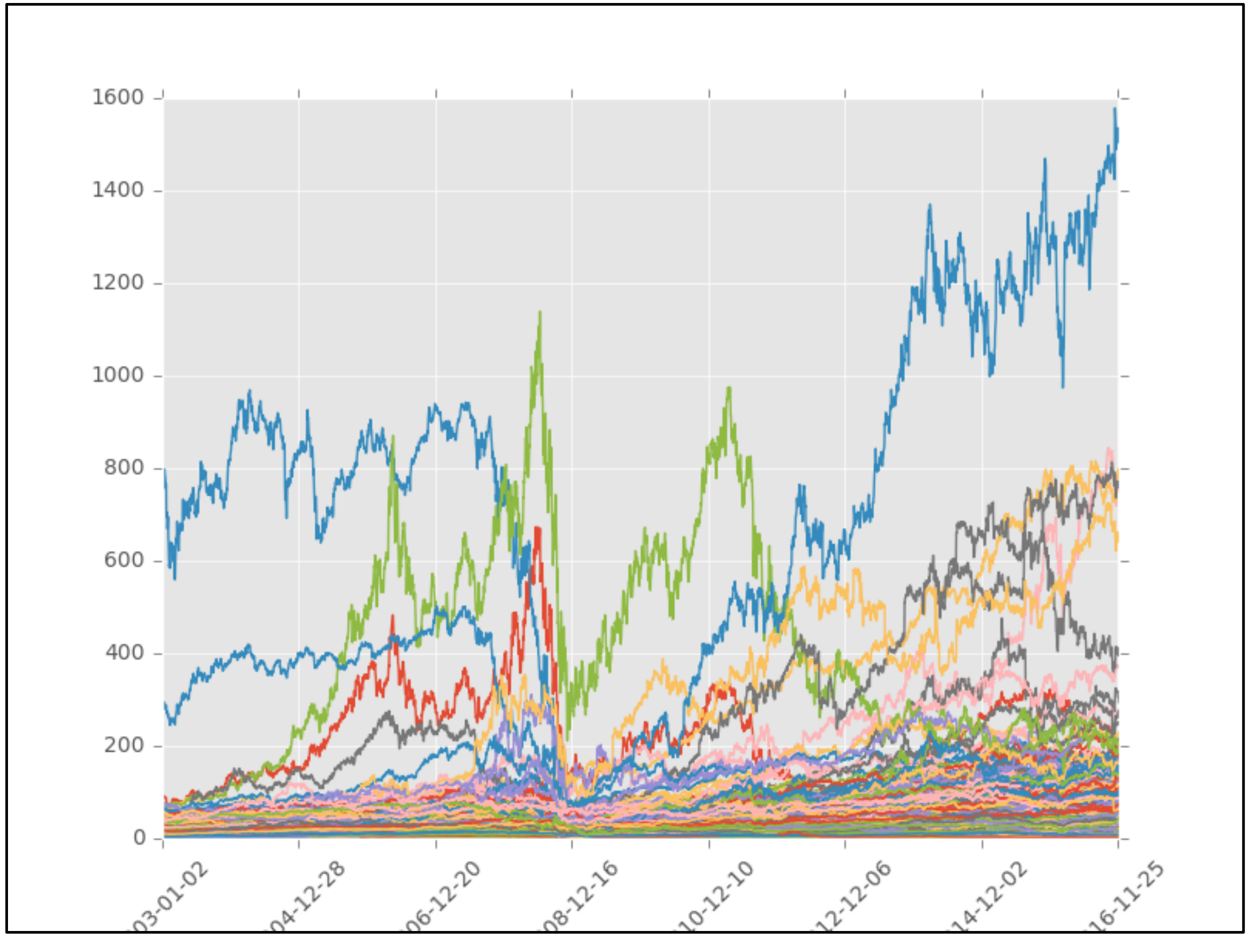
Methods, Models, and Analysis

- Analyzing the chart helps to show if the query was successful

```
def plot_df():
   df = pd.read_csv('{}{}'.format(path, op))
   df.plot(x=df['Date'], legend=None)
   plt.xticks(rotation=45)
   plt.show()

plot_df()
```

- In this case, it does
  - No glaring data gaps
  - Most securities trade < $100/share



Methods, Models, and Analysis

- This dataset was queried from archived finance.yahoo.com data
- Thus, historical fundamental data set was provided in html code
- By using ‘gather’ and importing re (regular expressions), we were able to parse the fundamental metrics in each file

```
def Key_Stats(gather=['Total Debt/Equity',
                      'Trailing P/E',
                      'Price/Sales',
                      'Price/Book',
                      'Profit Margin',
                      'Operating Margin',
                      'Return on Assets',
                      'Return on Equity',
                      'Revenue Per Share',
                      'Market Cap',
                      'Enterprise Value',
                      'Forward P/E',
                      'PEG Ratio',
                      'Enterprise Value/Revenue',
                      'Enterprise Value/EBITDA',
                      'Revenue',
                      'Gross Profit',
                      'EBITDA',
                      …
                      'Shares Short (prior ']):
```

```
rx = '.*?(\d{1,8}\.\d{1,8}M?B?|N/A)%?'
regex = re.escape(each_data) + r'{}'.format(rx)
value = re.search(regex, source)
value = (value.group(1))
```

Methods, Models, and Analysis: Fundamental Data

GENERAL ASSEMBLY

Apple Inc. (AAPL)

-NasdaqGS

454.45 6.56(1.42%) Aug 9, 4:00PM EDT|After Hours : 455.32 0.87 (0.19%) Aug 9, 7:59PM EDT

[Add to Portfolio](#)

Key Statistics Get Key Statistics for: GO

Data provided by [Capital IQ](#), except where noted.

Valuation Measures

Market Cap (intraday)[5]:	412.87B
Enterprise Value (Aug 11, 2013)[3]:	394.50B
Trailing P/E (ttm, intraday):	11.33
Forward P/E (fye Sep 29, 2014)[1]:	10.73
PEG Ratio (5 yr expected)[1]:	0.64
Price/Sales (ttm):	2.47
Price/Book (mrq):	3.40
Enterprise Value/Revenue (ttm)[3]:	2.33
Enterprise Value/EBITDA (ttm)[6]:	7.06

Financial Highlights

Fiscal Year	
Fiscal Year Ends:	Sep 29
Most Recent Quarter (mrq):	Jun 29, 2013
Profitability	
Profit Margin (ttm):	22.28%
Operating Margin (ttm):	29.46%
Management Effectiveness	

Trading Information

Stock Price History	
Beta:	0.87
52-Week Change[3]:	-27.87%
S&P500 52-Week Change[3]:	20.46%
52-Week High (Sep 21, 2012)[3]:	705.07
52-Week Low (Apr 19, 2013)[3]:	385.10
50-Day Moving Average[3]:	430.94
200-Day Moving Average[3]:	439.21
Share Statistics	
Avg Vol (3 month)[3]:	12,175,700
Avg Vol (10 day)[3]:	10,175,000
Shares Outstanding[5]:	908.50M
Float:	908.08M
% Held by Insiders[1]:	0.04%
% Held by Institutions[1]:	62.00%
Shares Short (as of Jul 15, 2013)[3]:	26.04M
Short Ratio (as of Jul 15, 2013)[3]:	2.10
Short % of Float (as of Jul 15, 2013)[3]:	2.80%

aapl

- 20031207103610.html
- 20040203073334.html
- 20040603012038.html
- 20040810020244.html
- 20041012054057.html
- 20041204141651.html
- 20050126151647.html
- 20050204030708.html

```
Total Debt/Equity (mrq):</td><td class="yfnc_tabledata1">13.75</td></tr><tr><td
class="yfnc_tablehead1" width="74%">Current Ratio (mrq):</td><td
class="yfnc_tabledata1">1.88</td></tr><tr><td  class="yfnc_tablehead1"
width="74%">Book Value Per Share (mrq):</td><td
class="yfnc_tabledata1">135.79</td></tr></table></td></tr></table><table
class="yfnc_datamodoutline1" width="100%" cellpadding="0" cellspacing="0"
border="0"><tr><td><table  width="100%" cellpadding="2" cellspacing="1"
border="0"><tr><td  colspan="2" style="padding: 3px 8px;background-
color:#CCC;"><b>Cash Flow Statement</b></td></tr><tr><td
class="yfnc_tablehead1"  width="74%">Operating Cash Flow (ttm):</td><td
class="yfnc_tabledata1">52.89B</td></tr><tr><td  class="yfnc_tablehead1"
width="74%">Levered Free Cash Flow (ttm):</td><td
class="yfnc_tabledata1">31.00B</td></tr></table></td></tr></table><table
```

Methods, Models, and Analysis

- This dataset was queried from archived finance.yahoo.com
- Thus, historical fundamental data set was provided in html code
- By using ‘gather’ and importing re (regular expressions), we were able to parse the fundamental metrics in each file and return it to a single data frame

```
def Key_Stats(gather=['Total Debt/Equity',
                      'Trailing P/E',
                      'Price/Sales',
                      'Price/Book',
                      'Profit Margin',
                      'Operating Margin',
                      'Return on Assets',
                      'Return on Equity',
                      'Revenue Per Share',
                      'Market Cap',
                      'Enterprise Value',
                      'Forward P/E',
                      'PEG Ratio',
                      'Enterprise Value/Revenue',
                      'Enterprise Value/EBITDA',
                      'Revenue',
                      'Gross Profit',
                      'EBITDA',
                      …
                      'Shares Short (prior ']):
```

```
rx = '.*?(\d{1,8}\.\d{1,8}M?B?|N/A)%?’
regex = re.escape(each_data) + r'{}'.format(rx)
value = re.search(regex, source)
value = (value.group(1))
```

Methods, Models, and Analysis

- Now that we have all our raw data, we'll need to start to manipulate prior to analysis
- In order to test the model and see if a company's fundamentals are predictive, there has to be a metric to evaluate
  - 1-year future price

```
try:
    stock_price_1y = datetime.fromtimestamp(one_year_later).strftime('%Y-%m-%d')
    row = stock_df[(stock_df['Date'] == stock_price_1y)][ticker]
    stock_1y_value = round(float(row), 2)

    except Exception as e:
        try:
            stock_price_1y = datetime.fromtimestamp(
                one_year_later - 259200).strftime('%Y-%m-%d')
            row = stock_df[(stock_df['Date'] == stock_price_1y)][ticker]
            stock_1y_value = round(float(row), 2)
                except Exception as e:
                    print('ERROR IN SECTION 3: ', ticker, str(e))

try:
    stock_price = datetime.fromtimestamp(unix_time).strftime('%Y-%m-%d')
    row = stock_df[(stock_df['Date'] == stock_price)][ticker]
    stock_price = round(float(row), 2)
except Exception as e:
    try:
        stock_price = datetime.fromtimestamp(unix_time - 259200).strftime('%Y-%m-%d')
        row = stock_df[(stock_df['Date'] == stock_price)][ticker]
        stock_price = round(float(row), 2)
    except Exception as e:
        print('ERROR IN SECTION 4: ', ticker, str(e))
```

Methods, Models, and Analysis

- A few variables must now be addressed…
- p_change was defined as the price delta of the security or SPY 1 year in the future
  - So, if this security was purchased and sold after a year, how much would money would be made or lost?
- Classify the dataset with this variable
  - The adjacent figure shows that in order to be classified as ‘outperform’ the security must have returned 5% more than the SPY

```
stock_p_change = round(((stock_1y_value - stock_price) /
                                   stock_price * 100.0), 2)

sp500_p_change = round(((sp500_1y_value - sp500_value) /
                                   sp500_value * 100.0), 2)

difference = stock_p_change - sp500_p_change

if difference > 5:
    status = 'outperform’
else:
    status = 'underperform'

if value_list.count('N/A') > 0:
    pass
```

Methods, Models, and Analysis

- Three Linear SVC scenarios were tested:
- The first test set the outperformance threshold to 0%
- From a universe of 446 securities, the model found 367 unique companies to invest in, and outperformed the market by 6%

```
def Build_Data_Set():
    data_df = pd.read_csv('{}{}'.format(path, key_stats_csv))

    data_df =
data_df.reindex(np.random.permutation(data_df.index))
    data_df = data_df.replace('NaN', 0).replace('N/A', 0)

    X = np.array(data_df[FEATURES].values)
    y = (data_df['status'].replace('underperform',
0).replace('outperform', 1)
                                        .values.tolist())
    X = preprocessing.scale(X)

    Z = np.array(data_df[['stock p change', 'sp500 p change',
'Price',
                  'Ticker']])
```

```
X, y, Z = Build_Data_Set()

   test_size = int(len(X) * 0.75)
   clf = svm.SVC(kernel='linear',C=1.0)
   clf.fit(X[:test_size], y[:test_size])

   correct_count = 0
   ticker_list = []

   for x in range(1, test_size + 1):
      if clf.predict(X[-x])[0] == y[-x]:
         correct_count += 1

      if clf.predict(X[-x])[0] == 1:
         invest_return = invest_amount +
(invest_amount * (Z[-x][0] / 100))
```

```
Our outperformance threshold to define "status" was 0%
The total length of the dataset was 2957
The total length of the test set was 74.97% or 2217 samples
We were 59% accurate with our predictions
We performed 1512 total trades
And if we invested $100 each trade we might see the following results...
Ending with Strategy: 184628
Ending with Market: 173490
And if we did nothing: 151200
Compared to Market, we earned: 6.42% more
Average Market return: 14.74%
Average Strategy return: 22.11%
We have a list of 367 unique securities to consider
```

Methods, Models, and Analysis

- Three Linear SVC scenarios were tested:
- The second test set the outperformance threshold to 5%
- From a universe of 446 securities, the model found 47 unique companies to invest in, and outperformed the market by 32%

```
def Build_Data_Set():
    data_df = pd.read_csv('{}{}'.format(path, key_stats_csv))

    data_df =
data_df.reindex(np.random.permutation(data_df.index))
    data_df = data_df.replace('NaN', 0).replace('N/A', 0)

    X = np.array(data_df[FEATURES].values)
    y = (data_df['status'].replace('underperform',
0).replace('outperform', 1)
                                    .values.tolist())
    X = preprocessing.scale(X)

    Z = np.array(data_df[['stock p change', 'sp500 p change',
'Price',
                    'Ticker']])
```

```
X, y, Z = Build_Data_Set()

    test_size = int(len(X) * 0.75)
    clf = svm.SVC(kernel='linear',C=1.0)
    clf.fit(X[:test_size], y[:test_size])

    correct_count = 0
    ticker_list = []

    for x in range(1, test_size + 1):
        if clf.predict(X[-x])[0] == y[-x]:
            correct_count += 1

        if clf.predict(X[-x])[0] == 1:
            invest_return = invest_amount +
(invest_amount * (Z[-x][0] / 100))
```

```
Our outperformance threshold to define "status" was 5%
The total length of the dataset was 2957
The total length of the test set was 74.97% or 2217 samples
We were 61% accurate with our predictions
We performed 96 total trades
And if we invested $100 each trade we might see the following results...
Ending with Strategy: 14319
Ending with Market: 10877
And if we did nothing: 9600
Compared to Market, we earned: 31.64% more
Average Market return: 13.31%
Average Strategy return: 49.16%
We have a list of 47 unique securities to consider
```

Methods, Models, and Analysis

- Three Linear SVC scenarios were tested:

- The final test set the outperformance threshold to 15%

- From a universe of 446 securities, the model found 8 unique companies to invest in, and outperformed the market, by 62%

```
def Build_Data_Set():
    data_df = pd.read_csv('{}{}'.format(path, key_stats_csv))

    data_df =
data_df.reindex(np.random.permutation(data_df.index))
    data_df = data_df.replace('NaN', 0).replace('N/A', 0)

    X = np.array(data_df[FEATURES].values)
    y = (data_df['status'].replace('underperform',
0).replace('outperform', 1)
                                                  .values.tolist())
    X = preprocessing.scale(X)

    Z = np.array(data_df[['stock p change', 'sp500 p change',
'Price',
                          'Ticker']])
```

```
X, y, Z = Build_Data_Set()

   test_size = int(len(X) * 0.75)
   clf = svm.SVC(kernel='linear',C=1.0)
   clf.fit(X[:test_size], y[:test_size])

   correct_count = 0
   ticker_list = []

   for x in range(1, test_size + 1):
      if clf.predict(X[-x])[0] == y[-x]:
         correct_count += 1

      if clf.predict(X[-x])[0] == 1:
         invest_return = invest_amount +
(invest_amount * (Z[-x][0] / 100))
```

```
Our outperformance threshold to define "status" was 15%
The total length of the dataset was 2957
The total length of the test set was 74.97% or 2217 samples
We were 74% accurate with our predictions
We performed 10 total trades
And if we invested $100 each trade we might see the following results...
Ending with Strategy: 1625
Ending with Market: 1134
And if we did nothing: 1000
Compared to Market, we earned: 43.25% more
Average Market return: 13.42%
Average Strategy return: 62.47%
We have a list of 8 unique securities to consider
```

Executive Summary

- Aim and Goals
- Datasets
- Methods, Models, and Analysis
- Risks and Assumptions
- Conclusion

Risks and Assumptions

- How the fundamental dataset was populated
  - Only training on high performing securities?

- Temporal
  - Fundamental dataset is from 2003-2013
  - Need input from 2014-2016

- No indicator entry / exit indicators

Executive Summary

- Aim and Goals
- Datasets
- Methods, Models, and Analysis
- Risks and Assumptions
- Conclusion

Conclusion

- Model is a screener for identifying which securities to research further
- Need to investigate, standardize, and aggregate more data
- Address risk/assumption control points
- Taking it a step further, will look into creating a model to systematically value these companies with new data

Acknowledgements

- Stefan and Phillippa – thank you!
- DAT-NYC-43 class
- Stack Overflow, Google, Sentdex, and online community