PEAD.txt Investment Thesis

Prepared by the Undergraduate Student Investment Management Fund
Presented by Aiden O'Connor and Evan Treger

Under the designation of Dr. Wahal |
Friday December 6th, 2024

Team Introduction

ASU W. P. Carey School of Business
Arizona State University

Mukul Anand

Nicholas Beeter

Vidhit Jitendra Jain

Aiden O’Connor

Kara Sierka

Evan Treger

Daniel Winkler

Brendan Weinberg
Portfolio Manager

Table of Contents

I. Investment Thesis

- Post Earnings Announcement Drift
- PEAD.txt

II. Methods of Implementation

- Trained Model and Earnings Call Analyzer
- Future Enhancements

III. Trading Decisions

- Investable Universe
- Investment Decisions

Investment Thesis

Post-Earnings Announcement Drift (PEAD)

Earnings Announcement

Positive Earnings Surprise

Stock Price Increases

- **Investors fail to price in earnings surprises immediately**
- **Assertions have been made that the magnitude of PEAD has diminished**

Published Paper

Meursault, V., Liang, P. J., Routledge, B. R., & Scanlon, M. M. (2022). PEAD.txt: Post-earnings-announcement drift using text. *Journal of Financial and Quantitative Analysis*, *58*(6), 2299–2326. https://doi.org/10.1017/s0022109022001181

PEAD.txt

Earnings Announcement

Positive Textual Surprise

Stock Price Increases

- **Measures unexpected information from EC transcripts**
- **PEAD.txt is consistently larger than PEAD**

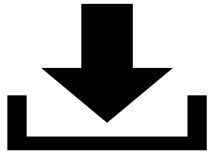
PEAD versus PEAD.txt

Model Implementation

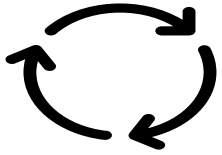
Model Outline

Purpose

- Train model and vectorizer



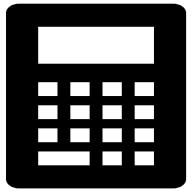
- Saves outputs to be used in analyzer



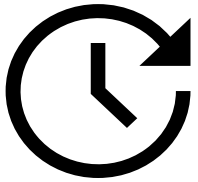
- Automates processes for future use in analyzer

Key Methods

- Analyzes earnings calls for key features



- Logistic regression converts raw text into numerical surprise measures



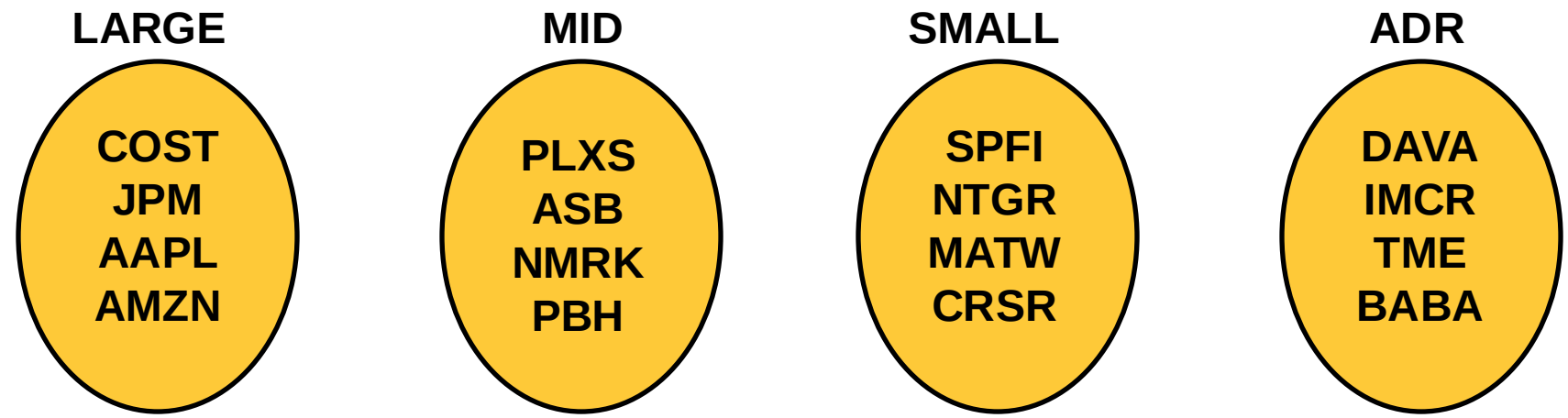
- Outputs features list, SUE.txt values, and numerical measures for future use

Model Dataset

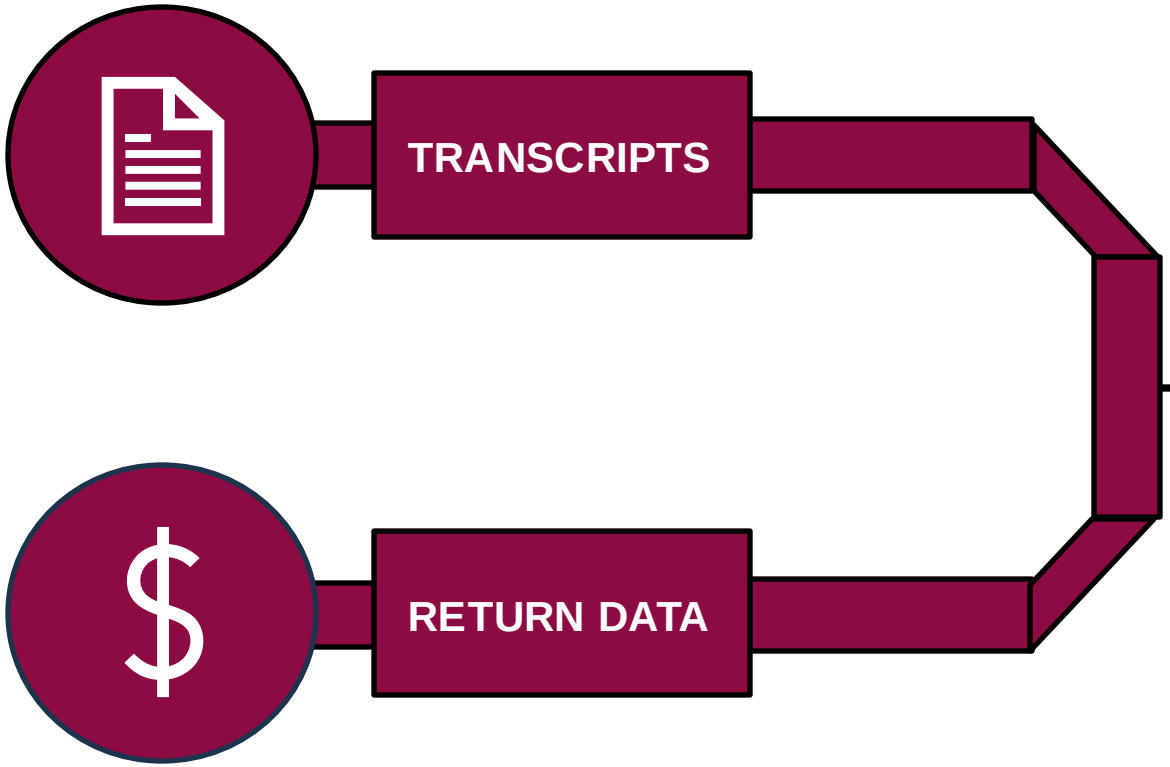
ASU W. P. Carey School of Business
Arizona State University

135 Total Stocks Included

8 Quarters of Historical Return Data and Earnings Calls



Model Process



- Read and process transcripts
- Perform sentiment analysis
- Calculate textual earnings surprise
- Save model, vectorizer, and outputs

Issues and Solutions

ASU W. P. Carey School of Business
Arizona State University

ISSUES

Pattern Recognition

- Features list included nonsensical words/patterns

Miscellaneous Errors

- Model complexity contributed to issues identifying coding errors

Model Runtime

- Initial versions were hardware and software intensive

1

2

3

SOLUTIONS

Implement Filters

- Filter out obstructive words and maintain pattern recognition

Log/Track Progress

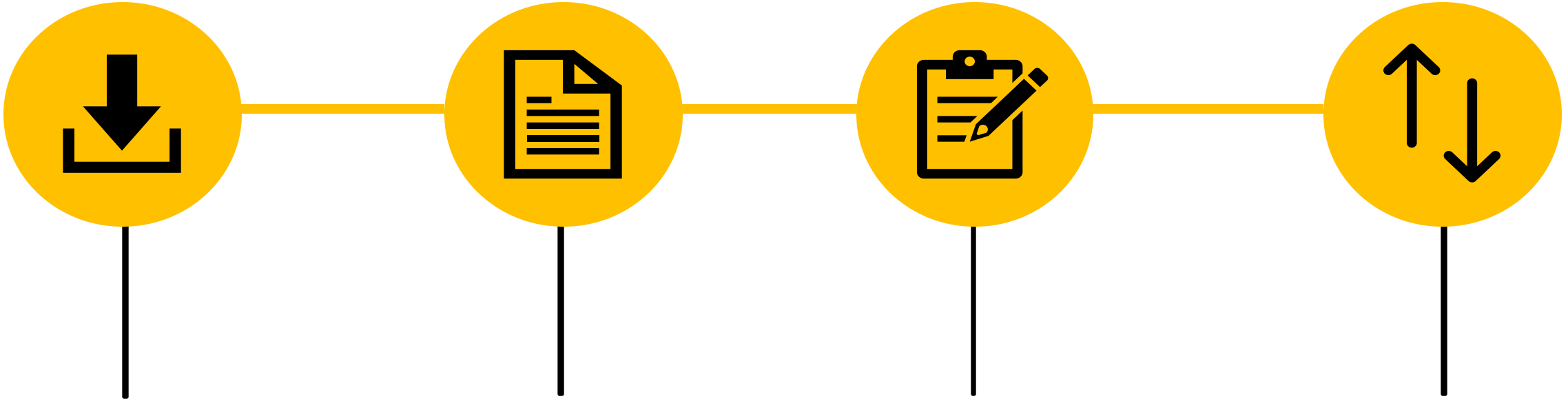
- Model logs errors and track progress as it runs

Preprocessing

- Transcripts are processed and saved in a cache

Earnings Call Analyzer

ASU W. P. Carey School of Business
Arizona State University



Import

Trained Outputs
Trained Processes

Read

Earnings Calls

Create

Keyword Analysis
Earnings Call Statistics

Group

Earnings Calls Based
On SUE.txt Value

Example Output


```
2024-11-30 20:36:40,532 - INFO - Initialized analyzer with 3 classes
2024-11-30 20:36:40,552 - INFO - Number of pages in PDF: 15
2024-11-30 20:36:42,343 - INFO - Found content start marker: 'Presentation' at position 552
2024-11-30 20:36:42,346 - INFO - Found Q&A start marker: '
Q - Marie Thibault' at position 36607
2024-11-30 20:36:42,350 - INFO - Total valid paragraphs found: 7
2024-11-30 20:36:42,352 - INFO - Total valid paragraphs found: 15
2024-11-30 20:36:42,353 - INFO -
Final Statistics:
2024-11-30 20:36:42,353 - INFO - Presentation section length: 36055
2024-11-30 20:36:42,354 - INFO - Q&A section length: 17026
2024-11-30 20:36:42,354 - INFO - Number of presentation paragraphs: 7
2024-11-30 20:36:42,355 - INFO - Number of Q&A paragraphs: 15

Analysis Results:
SUE.txt Score: 2.117
Quintile: Q4
```

Future Improvements

ASU W. P. Carey School of Business
Arizona State University

Earnings Call Analysis

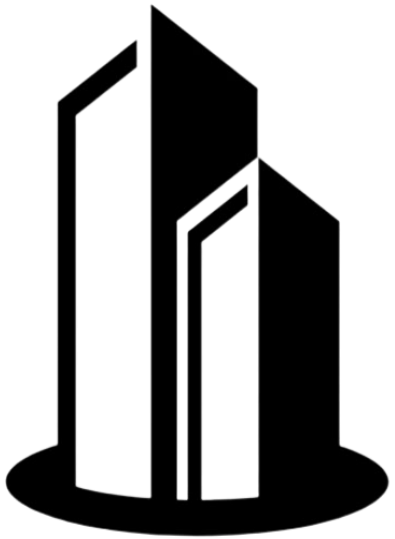
- Text processing improvements
- Text cleaning for special characters and company specific terms

Training Data Updates

- Update dataset monthly with new earnings calls and corresponding returns
- Improve sentiment analysis

Trading Decisions

Investable Universe



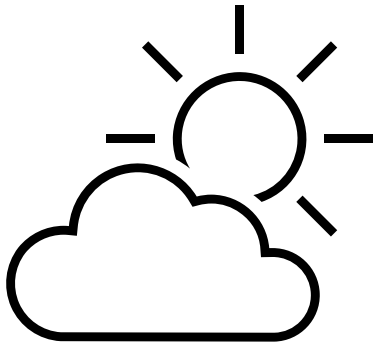
No REITs

No Utilities

>500mm Market Cap

>$10 Share Price

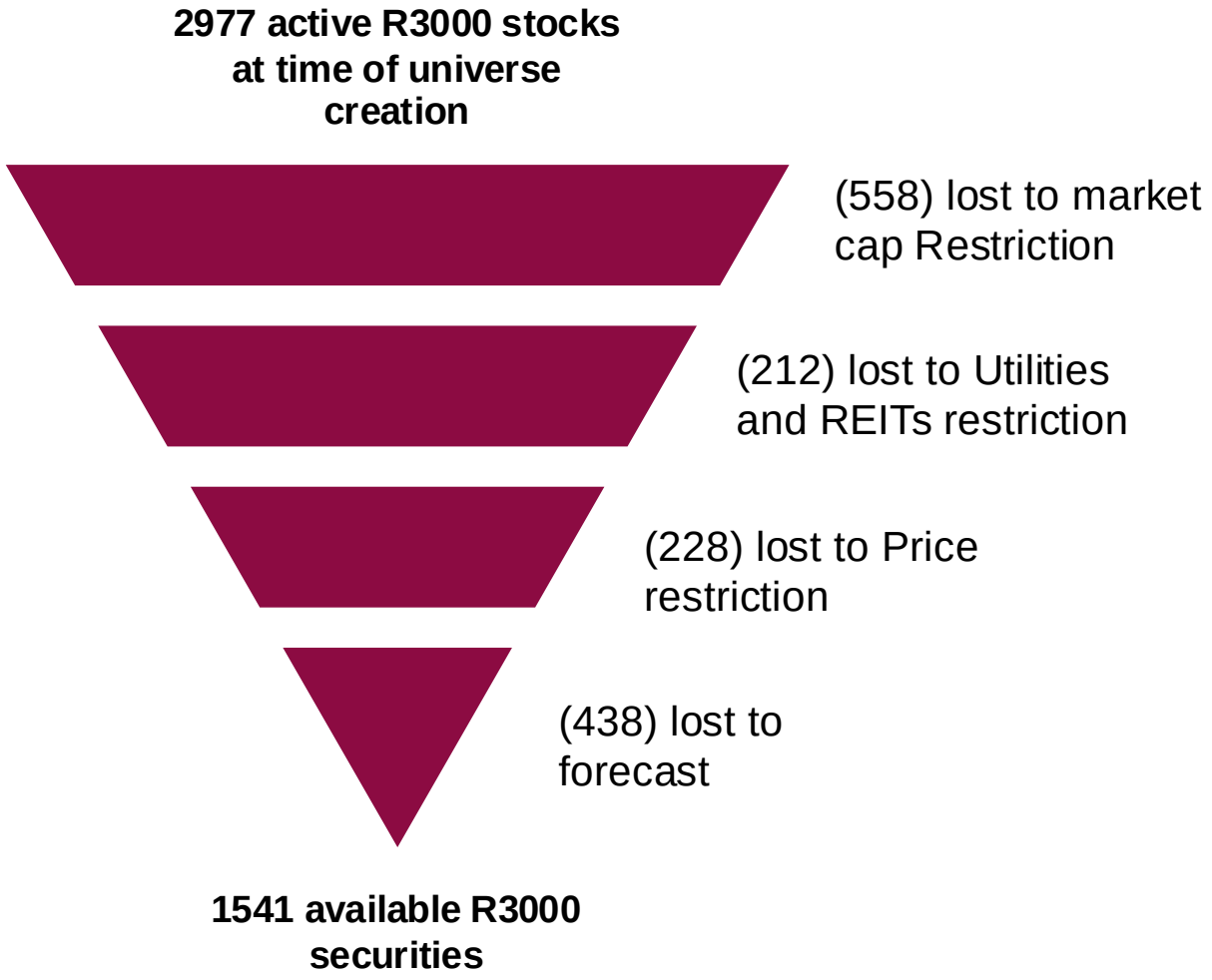
Investable Universe



# of EPS Forecasts >= 3

Initial Public Offering Date > 1 Year Ago (for ADRs)

Investable Universe



Gantt Chart

Month	Nov-24	Dec-24	Jan-25	Feb-25	Mar-25	Apr-25	May-25	TOTAL
Total Events	**76**	**94**	**407**	**809**	**36**	**4**	**8**	**1434**
Communication Services	0	0	10	40	1	0	0	51
Consumer Discretionary	8	16	56	93	1	0	2	176
Consumer Staples	4	6	20	39	1	0	0	70
Energy	4	3	21	48	2	0	0	78
Financials	14	19	79	177	10	1	2	302
Health Care	19	15	77	115	10	0	2	238
Information Technology	9	15	60	112	6	2	1	205
Industrials	15	17	62	145	4	0	1	244
Materials	3	3	20	35	1	1	0	63
Real Estate	0	0	2	5	0	0	0	7

Data Collection

Surprise Benchmark

Industry	Median of Positive Surprise (%)
Information Technology	19.30
Communication Services	24.47
Consumer Discretionary	12.28
Consumer Staples	9.00
Financials	8.27
Health Care	10.83
Energy	17.22
Industrials	8.57
Real Estate	40.22
Materials	10.23

Process

6:00pm

Data Collection

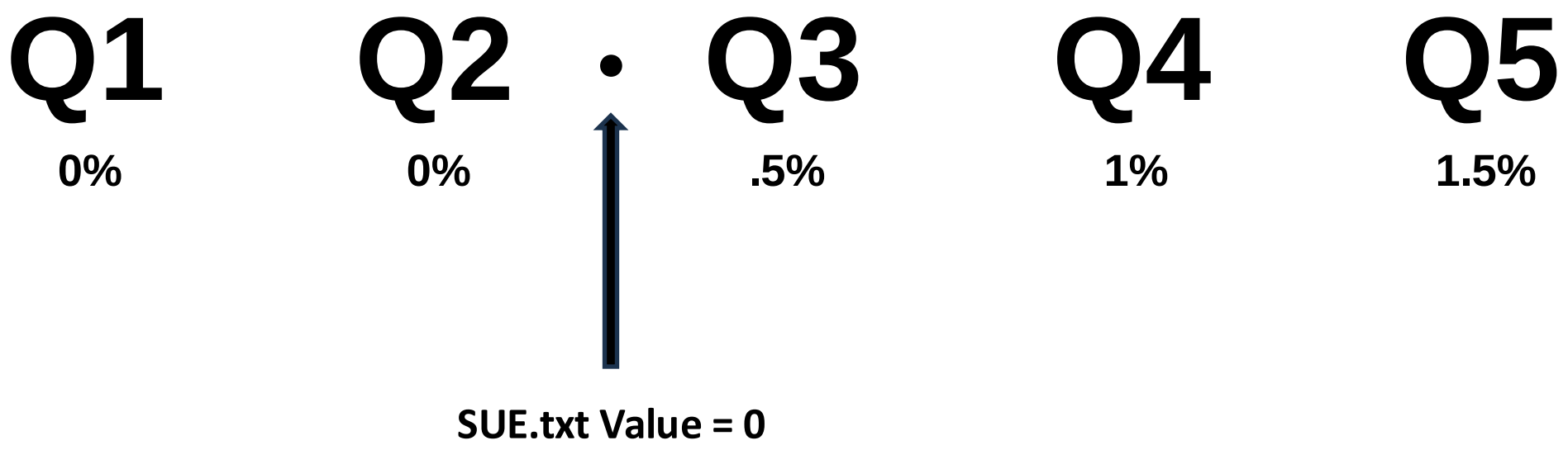
Surprise – Benchmark Analysis

Model Evaluation

Model Output

ASU W. P. Carey School of Business
Arizona State University

Percent of Portfolio Value by Quintile



Portfolio Breakdown / Current Investments

ASU W. P. Carey School of Business
Arizona State University

Company	Ticker	Shares	Price	Total Transaction Amount
iShares: Russell 3000	IWV	4,347	$341.63	$1,485,069.96
Northern Treasury Portfolio	NITXX	26,494	$1.00	$26,493.00
KEYSIGHT TECHNOLOGIES, INC.	KEYS	95	$169.14	$16,068.30
BJ'S WHOLESALE CLUB HOLDINGS, INC.	BJ	259	$96.34	$24,952.06
PALO ALTO NETWORKS, INC.	PANW	65	$379.54	$24,670.10
POWELL INDUSTRIES, INC.	POWL	55	$293.75	$16,156.25
UNIVERSAL TECHNICAL INSTITUTE, INC.	UTI	659	$24.56	$16,185.04
PAGERDUTY, INC.	PD	836	$19.70	$16,480.49
WORKDAY, INC.	WDAY	66	$249.47	$16,473.67
Embecta Corp.	EMBC	850	$19.38	$16,488.90
URBAN OUTFITTERS, INC.	URBN	355	$46.35	$16,463.73
AUTODESK, INC.	ADSK	84	$294.61	$24,756.19
iShares: Russell 3000	IWV	-265	$344.35	($91,243.60)

Company	Ticker	Industry	Gain / Loss
KEYSIGHT TECHNOLOGIES, INC.	KEYS	Electronic Equipment & Parts	$161.50
BJ'S WHOLESALE CLUB HOLDINGS, INC.	BJ	Diversified Retail	($10.36)
PALO ALTO NETWORKS, INC.	PANW	Software & IT Services	$538.20
POWELL INDUSTRIES, INC.	POWL	Machinery, Equipment & Components	($1,450.35)
UNIVERSAL TECHNICAL INSTITUTE, INC.	UTI	Professional & Business Education	$863.29
PAGERDUTY, INC.	PD	Software & IT Services	$1,276.15
WORKDAY, INC.	WDAY	Software & IT Services	$25.67
Embecta Corp.	EMBC	Healthcare Equipment & Supplies	$1,216.60
URBAN OUTFITTERS, INC.	URBN	Specialty Retailers	$835.42
AUTODESK, INC.	ADSK	Software & IT Services	($236.59)

Questions?

Appendix – Live Tracker

ASU W.P. Carey School of Business
Arizona State University

(as of 12/3)	Portfolio Value	$ Day Change	Gain / Loss	Total Value	% Gain (Loss)	(Gain/Loss) - Russell
26588.64	$1,646,927.59	$ 12,868.09	$ 39,729.30	$ 1,673,516.23	$ 1.72	$9,398.46

Current Holdings

Company	Ticker	Industry	GICS Sector	Quantity	Current Price	Market Value	% Breakdown	$ Day Change	Gain (Loss)	52 Week High	52 Week Low	% Gain (Loss)
iShares:Russell 3000 (ARCX:IWV)	IWV	#FIELD!	#N/A	4009	$348.94	$ 1,398,900.46	84.94%	$ 8,619.35	$ 30,330.84	$349.20	$260.49	
VALVOLINE INC. (XNYS:VVV)	VVV	Personal & Household Products & Services	Consumer Discretionary	648	$38.89	$ 25,200.72	1.53%	$ (168.48)	$ 236.00	$48.27	$33.86	0.01
KEYSIGHT TECHNOLOGIES, INC. (XNYS:	KEYS	Electronic Equipment & Parts	Information Technology	95	$172.26	$ 16,364.70	0.99%	$ 119.70	$ 296.40	$175.39	$119.72	0.02
BJ'S WHOLESALE CLUB HOLDINGS, INC	BJ	Diversified Retail	Consumer Staples	259	$97.17	$ 25,167.03	1.53%	$ 233.10	$ 214.97	$99.91	$63.73	0.01
PALO ALTO NETWORKS, INC. (XNAS:PA	PANW	Software & IT Services	Information Technology	65	$404.58	$ 26,297.70	1.60%	$ 779.35	$ 1,627.60	$408.53	$260.09	0.07
POWELL INDUSTRIES, INC. (XNAS:POW	POWL	Machinery, Equipment & Components	Industrials	55	$291.76	$ 16,046.80	0.97%	$ 1,255.65	$ (109.45)	364.9799	75.05	-0.01
UNIVERSAL TECHNICAL INSTITUTE, INC	UTI	Professional & Business Education	Consumer Discretionary	659	$24.82	$ 16,356.38	0.99%	$ 428.35	$ 171.34	26.64	10.4599	0.01
PAGERDUTY, INC. (XNYS:PD)	PD	Software & IT Services	Information Technology	836	$21.56	$ 18,024.16	1.09%	$ 275.88	$ 1,543.67	26.7	16.46	0.09
WORKDAY, INC. (XNAS:WDAY)	WDAY	Software & IT Services	Information Technology	66	$268.87	$ 17,745.42	1.08%	$ 779.46	$ 1,271.75	311.28	199.81	0.08
Embecta Corp. (XNAS:EMBC)	EMBC	Healthcare Equipment & Supplies	Health Care	850	$20.50	$ 17,425.00	1.06%	$ 59.50	$ 936.10	21.48	9.925	0.06
URBAN OUTFITTERS, INC. (XNAS:URBN	URBN	Specialty Retailers	Consumer Discretionary	355	$52.55	$ 18,655.25	1.13%	$ 656.75	$ 2,191.52	52.68	33.86	0.13
AUTODESK, INC. (XNAS:ADSK)	ADSK	Software & IT Services	Information Technology	84	$304.23	$ 25,555.32	1.55%	$ 442.68	$ 799.13	326.62	195.32	0.03
Credo Technology Group Holding Ltd (	CRDO	Semiconductors & Semiconductor Equipment	Information Technology	365	$69.01	$ 25,188.65	1.53%	$ (613.20)	$ 219.43	75.3	16.82	0.01

SIM Fund - Team 2

Insider Investment Horizon

SIM Fund Team 2

Maxwell Fields, Carson Stork, Antonio Nguyen, Ilya Illiashenko, Anushayana Pant, Tung-Lin Pai

Team Introduction

Antonio Nguyen

Carson Stork

Maxwell Fields

Anushayana Pant

Vlada Vaska

(Fund Manager)

Tung-Lin Pai

Ilya Illiashenko

Insider Investment Horizon Strategy

Arizona State University

Insider Trading Scenarios

Insiders trade for informational reasons.

Insiders trade for non-informational reasons.

Trade Categorization

Insider Investment Horizon: Categorized by average annual net order flow

- Long-Horizon Insiders repeatedly buy or sell stock
  - *Net Order Flow < -0.8 or Net Order Flow > 0.8*
- Short-Horizon Insiders alternate between buying and selling stock
  - *-0.5 < Net Order Flow < 0.5*

$$\frac{P_{i,j,y} - S_{i,j,y}}{P_{i,j,y} + S_{i,j,y}}$$

Source: Akbas, Ferhat, et al. "Insider Investment Horizon." *The Journal of Finance*, LXXV, no. 3, June 2020.

Trade Categorization

Note: The graph depicts insider trading activity of an anonymous short horizon insider (Mr. A).

Source: Akbas, Ferhat, et al. "Insider Investment Horizon." *The Journal of Finance*, LXXV, no. 3, June 2020.

Investment Horizon Findings

ASU W. P. Carey School of Business
Arizona State University

★ Short Horizon insiders perform better than typical long horizon insiders.

- SH purchases returned **2.44%** in the following month (outperformed LH purchases by 0.81%)

★ Long Horizon insiders who execute unexpected trades perform better than short horizon insiders.

- LH unexpected purchases returned **4.10%** in the following month (outperformed SH purchases by 1.66%)

$$UNEXP_{i,j,t} = CURRENT_{i,j,t} - (Mean\ IOF)_{i,j,t}$$

Source: Akbas, Ferhat, et al. "Insider Investment Horizon." *The Journal of Finance*, LXXV, no. 3, June 2020.

Trade Categorization

Routine Trades are classified as insider trades executed in the same month for the previous three years.

Opportunistic Trades are classified as insider trades that do not follow consistent calendar trading patterns (not traded in the same month for the previous three years).

Source: COHEN, LAUREN, et al. "Decoding Inside Information." *The Journal of Finance*, LXVII, no. 3, 2012.

Opportunistic Trading Findings

ASU W. P. Carey School of Business
Arizona State University

★ Opportunistic trades perform better than routine trades.

- Opportunistic purchases returned **1.79%** monthly on average over an 18 year period in a value-weighted portfolio (outperformed routine purchases by .68%)

Source: COHEN, LAUREN, et al. "Decoding Inside Information." *The Journal of Finance*, LXVII, no. 3, 2012.

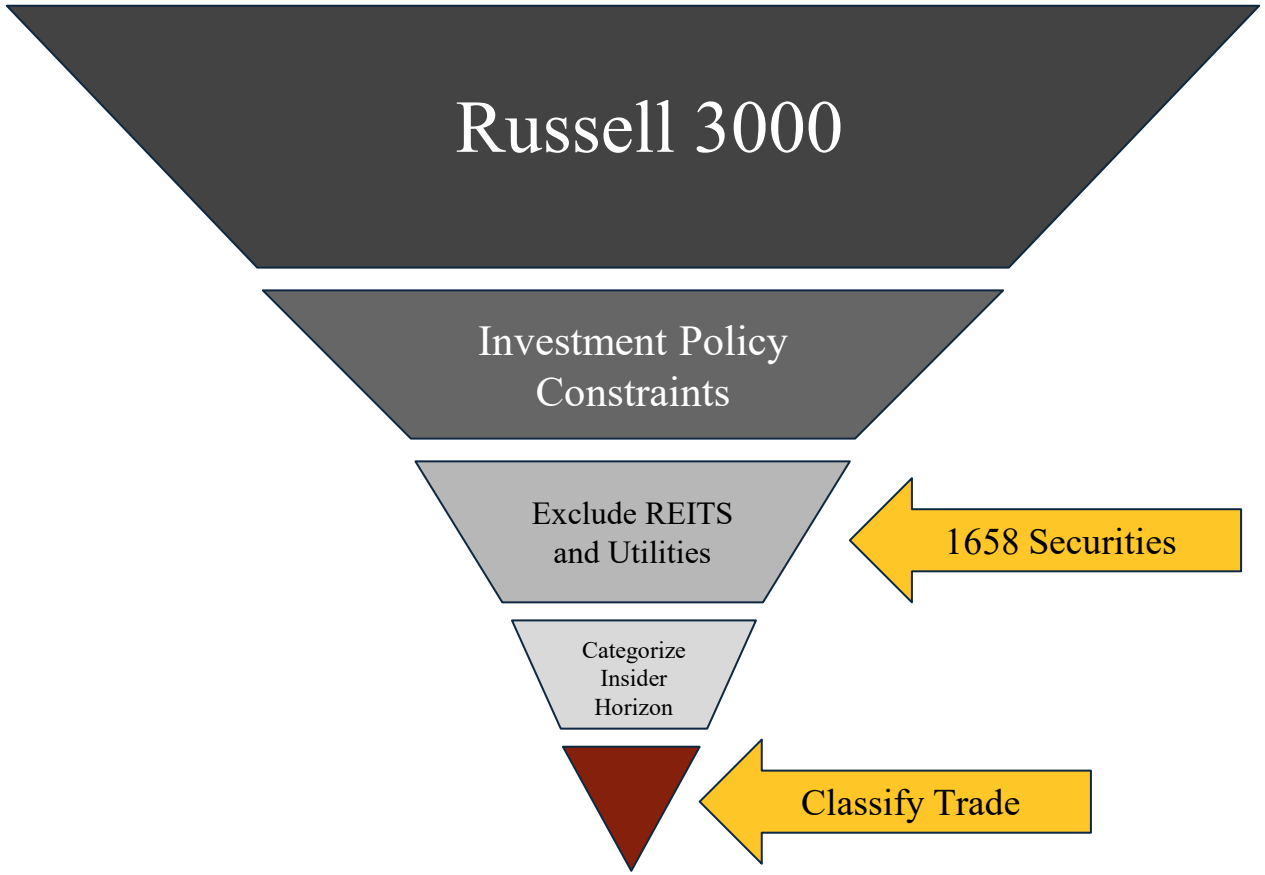
Strategy Implementation

Arizona State University

Strategy Plan

1. Filter investable universe.
2. Collect historical insider data.
3. Categorize SH and LH insiders.
4. Develop an automated SEC scraping program.
5. Distinguish routine vs. opportunistic trades.
6. Generate a daily .csv of insider trades.

Investment Universe



Insider Categorization

```
In [10]: categorized = cate_strength(insider_names, clean_df)
         categorized
```

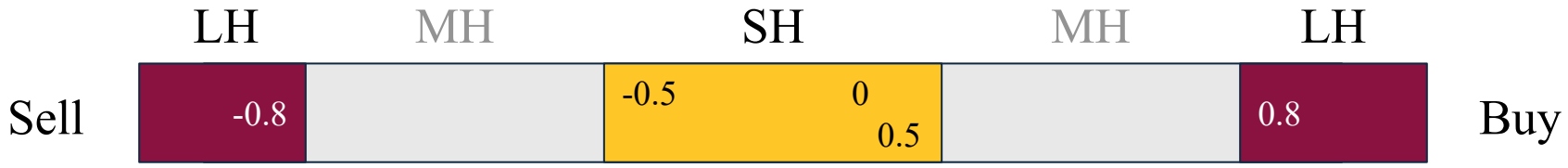
Out[10]:

	Insider name	cusip	symbol	trading strength	avg_ts	historical yrs	yrs_traded
0	SOTOK FREDERICK A	371901	GNTX	-1.000000	-1.000000	6.280630	6
1	COLTHARP DOUGLAS E	904311	UA	-1.000000	-1.000000	10.045175	5
2	CONROY KEVIN T	30063P	EXAS	-0.968663	-0.986522	10.327173	11
3	LIDGARD GRAHAM PETER	30063P	EXAS	-1.000000	-1.000000	7.186858	6
4	DOLBY DAGMAR	25659T	DLB	-1.000000	-1.000000	5.555099	6
...	...	...	...	...	...	...	...
5127	BROAD MATTHEW R	237194	DRI	-0.985399	-0.666667	5.152635	6
5128	SCHRADER ROBERT L.	704326	PAYX	-1.000000	-1.000000	5.207392	5
5129	ARIAN MARK D.	500643	KFY	-0.990050	0.000000	5.768652	2
5130	GUERTIN TIMOTHY E	880770	TER	-1.000000	-1.000000	7.104723	2
5131	SCHMIDT JOHN W	129500	CAL	-0.949367	0.000000	5.125257	4

5132 rows × 7 columns

```
In [11]: categorized.to_csv('categorized.csv', index=False)
```

- ★ Short Horizon insiders have net order flows between -0.50 and 0.50.
- ★ Long Horizon insiders have net order flows less than -0.80 or greater than 0.80.



SEC Scraper

ASU W. P. Carey School of Business
Arizona State University

In [16]:
```
finished_df['Opportunistic'] = [check_opp(a, b, c, d) for a, b, c, d in zip(finished_df['month'], finished_df['year'], finished_d
finished_df
```

Out[16]:

	insider	symbol	company_name	cik	date	code	formtype	shares	pricepershare	sharesownedafter	...	trading strength	avg_ts	historical yrs	yrs_tra
0	FROST PHILLIP MD ET AL	OPK	OPKO HEALTH, INC.	944809	2024-11-29	P	4	199072.0	1.5340	211911477.0	...	1.0	1.0	10.387406	
1	LEE SPENCER S	CHE	CHEMED CORP	19584	2024-11-27	S	4	732.0	575.6230	18287.0	...	-1.0	-1.0	10.464066	
2	PREVOZNIK MICHAEL E	DGX	QUEST DIAGNOSTICS INC	1022079	2024-11-29	S	4	3009.0	162.2588	70615.0	...	-1.0	-1.0	10.524298	
3	PREVOZNIK MICHAEL E	DGX	QUEST DIAGNOSTICS INC	1022079	2024-11-29	S	4	32137.0	163.1473	38478.0	...	-1.0	-1.0	10.524298	
4	Doherty Catherine T.	DGX	QUEST DIAGNOSTICS INC	1022079	2024-11-27	S	4	4803.0	162.1435	96712.0	...	-1.0	-1.0	9.560575	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
118	Banon Shay	ESTC	Elastic N.V.	1707753	2024-11-27	S	4	55675.0	108.7200	2549303.0	...	-1.0	-1.0	5.018480	
119	Banon Shay	ESTC	Elastic N.V.	1707753	2024-11-27	S	4	74066.0	109.7000	2475237.0	...	-1.0	-1.0	5.018480	
120	Banon Shay	ESTC	Elastic N.V.	1707753	2024-11-27	S	4	4496.0	110.5700	2470741.0	...	-1.0	-1.0	5.018480	
121	Banon Shay	ESTC	Elastic N.V.	1707753	2024-11-27	S	4	12762.0	111.6200	2457979.0	...	-1.0	-1.0	5.018480	
122	Banon Shay	ESTC	Elastic N.V.	1707753	2024-11-27	S	4	3001.0	112.3600	2454978.0	...	-1.0	-1.0	5.018480	

123 rows × 21 columns

Daily .csv Generation

```
def check_opp(month,year,name,symbol):
    yrlist = [year-1,year-2]
    dfpastyear1 = clean_hist.loc[(clean_hist['first_name'] == name) & (clean_hist['TICKER'] == symbol) & (clean_hist['year'] == y
    dfpastyear2 = clean_hist.loc[(clean_hist['first_name'] == name) & (clean_hist['TICKER'] == symbol) & (clean_hist['year'] == y
    if (len(dfpastyear1) != 0) and (len(dfpastyear2) != 0):
        return False
    else:
        return True
```

```
finished_df['Opportunistic'] = [check_opp(a, b, c, d) for a, b, c, d in zip(finished_df['month'], finished_df['year'], finished_d
finished_df
```

pe	shares	pricepershare	sharesownedafter	...	trading strength	avg_ts	historical yrs	yrs_traded	shorthorizon	longhorizon	first_name	month	year	Opportunistic
4	10000.0	51.6639	81882.0	...	-1.000000	-1.000000	6.822724	3	False	True	hoffman	11	2024	True
4	2000.0	572.7700	101735.0	...	-1.000000	-1.000000	10.600958	11	False	True	mcnamara	11	2024	True
4	500.0	675.5200	7911.0	...	-1.000000	-1.000000	9.691992	5	False	True	teske	11	2024	True
4	1300.0	295.4696	103515.0	...	-0.998556	-0.818182	9.864476	11	False	True	moret	11	2024	True
4	1150.0	296.4167	102365.0	...	-0.998556	-0.818182	9.864476	11	False	True	moret	11	2024	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4	18026.0	101.4400	43916.0	...	-1.000000	-1.000000	5.251198	2	False	True	flowers	11	2024	True
4	1000.0	102.1300	42916.0	...	-1.000000	-1.000000	5.251198	2	False	True	flowers	11	2024	True
4	5000.0	19.5100	715207.0	...	-1.000000	-1.000000	5.629021	4	False	True	stock	11	2024	True

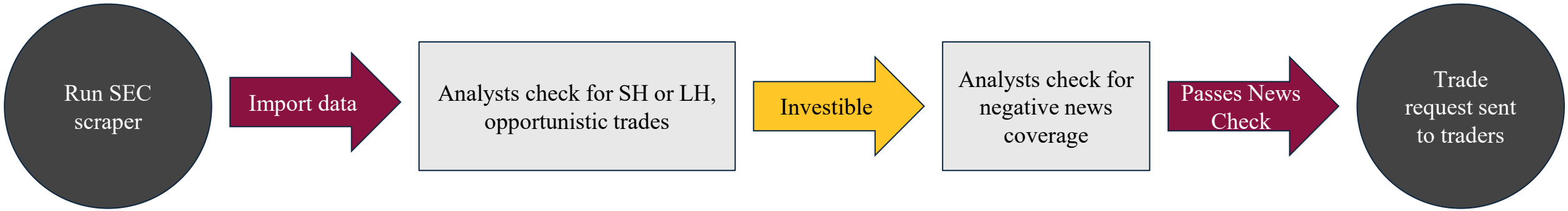
Trade Decision Making

```
def check_opp(month,year,name,symbol):
    yrlist = [year-1,year-2]
    dfpastyear1 = clean_hist.loc[(clean_hist['first_name'] == name) & (clean_hist['TICKER'] == symbol) & (clean_hist['year'] == y
    dfpastyear2 = clean_hist.loc[(clean_hist['first_name'] == name) & (clean_hist['TICKER'] == symbol) & (clean_hist['year'] == y
    if (len(dfpastyear1) != 0) and (len(dfpastyear2) != 0):
        return False
    else:
        return True
```

```
finished_df['Opportunistic'] = [check_opp(a, b, c, d) for a, b, c, d in zip(finished_df['month'], finished_df['year'], finished_d
finished_df
```

'pe	shares	pricepershare	sharesownedafter	...	trading strength	avg_ts	historical yrs	yrs_traded	shorthorizon	longhorizon	first_name	month	year	Opportunistic
4	10000.0	51.6639	81882.0	...	-1.000000	-1.000000	6.822724	3	False	True	hoffman	11	2024	True
4	2000.0	572.7700	101735.0	...	-1.000000	-1.000000	10.600958	11	False	True	mcnamara	11	2024	True
4	500.0	675.5200	7911.0	...	-1.000000	-1.000000	9.691992	5	False	True	teske	11	2024	True
4	1300.0	295.4696	103515.0	...	-0.998556	-0.818182	9.864476	11	False	True	moret	11	2024	True
4	1150.0	296.4167	102365.0	...	-0.998556	-0.818182	9.864476	11	False	True	moret	11	2024	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4	18026.0	101.4400	43916.0	...	-1.000000	-1.000000	5.251198	2	False	True	flowers	11	2024	True
4	1000.0	102.1300	42916.0	...	-1.000000	-1.000000	5.251198	2	False	True	flowers	11	2024	True
4	5000.0	19.5100	715207.0	...	-1.000000	-1.000000	5.629021	4	False	True	stock	11	2024	True

Daily Trading Process



Questions?

Appendix

Portfolio Progress

insider	symbol	company_ı	code	shares	trading strength	shorthorizon	longhorizon	Opportunistic
HORIZON	TPL	Texas Paci	P	3	1	FALSE	TRUE	TRUE

★ 1% Portfolio Allocation to Texas Pacific Land Corp as of December 5

Initial Column Sorting

```
clean_df = df[['PERSONID', 'CUSIP6','OWNER','TICKER','TRANDATE', 'TRANCODE','TPRICE', 'OWNERSHIP', 'SHARES', 'SHARESHELD', 'TPRIC
clean_df['TRANDATE'] = pd.to_datetime(clean_df['TRANDATE'])
clean_df = clean_df.loc[(clean_df['TRANCODE'] == 'P')| (clean_df['TRANCODE'] == 'S')]
clean_df
```

	PERSONID	CUSIP6	OWNER	TICKER	TRANDATE	TRANCODE	TPRICE	OWNERSHIP	SHARES	SHARESHELD	TPRICE_ADJ
2	13129904	371901	SOTOK FREDERICK A	GNTX	2014-07-28	S	29.02	D	6000.0	13500.0	14.5081
6	12220927	904311	COLTHARP DOUGLAS E	UA	2014-07-28	S	69.17	D	8400.0	NaN	69.1700
11	16138352	30063P	CONROY KEVIN T	EXAS	2014-07-25	S	15.99	D	11336.0	145563.0	15.9900
13	12063059	247361	HIRST RICHARD B	DAL	2014-07-28	S	38.32	D	132000.0	356030.0	38.3228
16	16116348	30063P	ARORA MANEESH K	EXAS	2014-07-25	S	15.99	D	9291.0	84893.0	15.9900
...	...	...	...	...	...	...	...	...	...	...	...
1341524	16290090	144200	WATSON JIMMY R.	CART	2015-07-28	P	5.50	I	270.0	21831.0	5.5000
1341526	16290125	144200	RHYNE JOHNATHAN L JR	CART	2015-10-26	P	5.67	D	200.0	130469.0	5.6700
1341527	16299206	144200	PASCHALL NANCY BORDERS	CART	2015-11-17	P	5.60	I	366.0	5617.0	5.6000
1341528	16157759	144200	OCHELTREE JERRY L	CART	2015-11-17	P	5.60	D	366.0	6490.0	5.6000
1341529	16299206	144200	PASCHALL NANCY BORDERS	CART	2015-05-06	P	5.29	D	744.0	NaN	5.2900

414223 rows × 11 columns

SEC Scraper

ASU W. P. Carey School of Business
Arizona State University

In [5]:
```
def sec_scraper(cik, start=0, count=2):
    # Base URL for fetching insider transactions
    base_url = f"https://www.sec.gov/cgi-bin/browse-edgar?action=getcompany&CIK={cik}&type=4&owner=only&start={start}&count={cour

    # SEC headers (User-Agent is required)
    headers = {"User-Agent": "Antonio Nguyen antonio.nguyen203@gmail.com"}

    response = requests.get(base_url, headers=headers)
    if response.status_code != 200:
        raise Exception("Failed to fetch data from SEC EDGAR.")

    # Parse the XML response using BeautifulSoup
    soup = BeautifulSoup(response.content, "xml")
    entries = soup.find_all("entry")

    transactions = []

    # Loop through each entry to fetch transaction data
    for entry in entries[0:count]:
        base_link = 'https://www.sec.gov'
        # Fetch basic entry data
        filing_href = entry.find("link")['href']
        new_url = filing_href
        response_new = requests.get(new_url, headers=headers)
        new_soup = BeautifulSoup(response_new.content, "html.parser")
        outer_div = new_soup.find_all('div',id='formDiv')[1]
        xml_link = outer_div.find('a',href=True)['href']
        if xml_link not in lines:
            #clean xml_link
            l = xml_link.split('/')
            cleanxml_link = f"/{l[1]}/{l[2]}/{l[3]}/{l[4]}/{l[5]}/{l[7]}"
            # Create xml transaction link
            new_link = f'https://www.sec.gov{cleanxml_link}'
            # Final response
            response_final = requests.get(new_link, headers=headers)
            final_soup = BeautifulSoup(response_final.content, "xml")
            owner_name = final_soup.find('rptOwnerName').text
            transaction_div = final_soup.find_all('nonDerivativeTransaction')
            for code in transaction_div:
                transaction_data = {}
                tc = code.find('transactionCode').text
                if (tc =='S') or (tc == 'P'):
                    transaction_data['insider'] = get_text_safe1(final_soup, 'rptOwnerName')
                    transaction_data['symbol'] = get_text_safe1(final_soup, 'issuerTradingSymbol')
                    transaction_data['company_name'] = get_text_safe1(final_soup, 'issuerName')
                    transaction_data['cik'] = get_text_safe1(final_soup, 'issuerCik')
                    transaction_data['date'] = get_text_safe2(code, 'transactionDate')
                    transaction_data['code'] = get_text_safe1(code, 'transactionCode')
                    transaction_data['formtype'] = get_text_safe1(code, 'transactionFormType')
                    transaction_data['shares'] = get_text_safe2(code, 'transactionShares')
                    transaction_data['pricepershare'] = get_text_safe2(code, 'transactionPricePerShare')
                    transaction_data['sharesownedafter'] = get_text_safe2(code, 'sharesOwnedFollowingTransaction')
                    # there is also 'transactionAquiredDisposedCode' and 'directOrIndirectOwnership'
                    transactions.append(transaction_data)

            lines.append(xml_link)


    # Convert the transactions to a DataFrame
    df = pd.DataFrame(transactions)
    return df
```

In [6]:
```
def freshinsiderdata(universe):
    insider_data = pd.DataFrame(columns=['insider', 'symbol', 'company_name', 'cik', 'date', 'code', 'formtype',
       'shares', 'pricepershare', 'sharesownedafter'])
    cik_list = stocklist_cik(universe)
    for x in cik_list:
        print(x)
        try:
            df = sec_scraper(x, start=0, count=3)
            insider_data = pd.concat([insider_data, df], ignore_index=True)
        except:
            print(f'Failed to fetch Data for: {x}')
            continue

    # Write updated list back to the file
    with open('cache.txt', 'w') as file:
        file.write('\n'.join(lines))  # Join list items with newline characters

    return insider_data
```

SEC Column Sorting

```
hist = pd.read_csv('ptpuzdukcjydedar.csv', encoding='latin1')
clean_hist = hist[['PERSONID', 'CUSIP6','OWNER','TICKER','TRANDATE', 'TRANCODE','TPRICE', 'OWNERSHIP', 'SHARES', 'SHARESHELD', 'T
clean_hist['TRANDATE'] = pd.to_datetime(clean_hist['TRANDATE'])
clean_hist = clean_hist.loc[(clean_hist['TRANCODE'] == 'P')| (clean_hist['TRANCODE'] == 'S')]
clean_hist['month'] = clean_hist['TRANDATE'].dt.month
clean_hist['year'] = clean_hist['TRANDATE'].dt.year
clean_hist['first_name'] = [safe_process(x) for x in clean_hist['OWNER']]
clean_hist
```

	PERSONID	CUSIP6	OWNER	TICKER	TRANDATE	TRANCODE	TPRICE	OWNERSHIP	SHARES	SHARESHELD	TPRICE_ADJ	month	year	first_
2	13129904	371901	SOTOK FREDERICK A	GNTX	2014-07-28	S	29.02	D	6000.0	13500.0	14.5081	7	2014	
6	12220927	904311	COLTHARP DOUGLAS E	UA	2014-07-28	S	69.17	D	8400.0	NaN	69.1700	7	2014	c
11	16138352	30063P	CONROY KEVIN T	EXAS	2014-07-25	S	15.99	D	11336.0	145563.0	15.9900	7	2014	
13	12063059	247361	HIRST RICHARD B	DAL	2014-07-28	S	38.32	D	132000.0	356030.0	38.3228	7	2014	
16	16116348	30063P	ARORA MANEESH K	EXAS	2014-07-25	S	15.99	D	9291.0	84893.0	15.9900	7	2014	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1341524	16290090	144200	WATSON JIMMY R.	CART	2015-07-28	P	5.50	I	270.0	21831.0	5.5000	7	2015	
1341526	16290125	144200	RHYNE JOHNATHAN L JR	CART	2015-10-26	P	5.67	D	200.0	130469.0	5.6700	10	2015	
1341527	16299206	144200	PASCHALL NANCY BORDERS	CART	2015-11-17	P	5.60	I	366.0	5617.0	5.6000	11	2015	p
1341528	16157759	144200	OCHELTREE JERRY L	CART	2015-11-17	P	5.60	D	366.0	6490.0	5.6000	11	2015	oc
1341529	16299206	144200	PASCHALL NANCY BORDERS	CART	2015-05-06	P	5.29	D	744.0	NaN	5.2900	5	2015	p

414223 rows × 14 columns

Insider Horizon Categorization

Arizona State University W. P. Carey School of Business

Trading Strength = (Shares Bought - Shares sold)/Total trading volume of the stock

In [8]:
```
def trading_str(insider_name):
    df = clean_df.loc[(clean_df['OWNER'] == insider_name)]
    shares_bought = df.loc[df['TRANCODE'] == 'P']['SHARES'].sum()
    shares_sold = df.loc[df['TRANCODE'] == 'S']['SHARES'].sum()
    total_trading = shares_bought + shares_sold
    trading_str = (shares_bought - shares_sold)/total_trading
    return trading_str

def avg_tradestrength(insider_name):
    df = clean_df.loc[(clean_df['OWNER'] == insider_name)]
    years = df['TRANDATE'].dt.year.unique().tolist()
    data = []
    for y in years:
        # Filter rows where the year in 'TRANDATE' matches the current year in the loop
        yearly_df = df[df['TRANDATE'].dt.year == y]
        shares_bought = yearly_df.loc[yearly_df['TRANCODE'] == 'P', 'SHARES'].sum()
        shares_sold = yearly_df.loc[yearly_df['TRANCODE'] == 'S', 'SHARES'].sum()
        total_trading = shares_bought + shares_sold
        trading_str = (shares_bought - shares_sold)/total_trading
        # Do something with shares_bought or append to data
        data.append(trading_str)

    avg = sum(data)/len(data)
    return avg
```

In [9]:
```
def cate_strength(names, df):
    categorized_id = pd.DataFrame(columns=['Insider name','cusip','symbol','trading strength','avg_ts','historical yrs','yrs_trac
    for n in names:
        idf = df.loc[(df['OWNER'] == n)]
        idf['FDATE'] = pd.to_datetime(idf['TRANDATE'])
        insider = n
        hist_yrs = (idf['TRANDATE'].max() - idf['TRANDATE'].min()).days/365.25
        if hist_yrs >= 5:
            symbol = list(idf['TICKER'])[0]
            trading_strg = trading_str(n)
            cusip = list(idf['CUSIP6'])[0]
            avg_ts = avg_tradestrength(n)
            yrs_traded = len(idf['TRANDATE'].dt.year.unique().tolist())
            new_row = {'Insider name':insider,'cusip':cusip,'symbol': symbol, 'trading strength': trading_strg,'avg_ts':avg_ts,'h
            categorized_id.loc[len(categorized_id)] = new_row
        else:
            continue

    return categorized_id
```

Additional Functions/Tools

ASU W. P. Carey School of Business
Arizona State University

In [4]:
```python
def stocklist_cik(stocks):
    new_list = []
    mapper = StockMapper()
    for ticker in stocks:
        cik = mapper.ticker_to_cik.get(ticker.upper())
        new_list.append(cik)

    return new_list

def get_text_safe1(tag, child_tag_name):
    try:
        # Attempt to find the child tag and return its text
        return tag.find(child_tag_name).text
    except (AttributeError, TypeError):
        # If tag or text not found, return "N/A"
        return "N/A"

def get_text_safe2(tag, child_tag_name):
    try:
        # Attempt to find the child tag and return its text
        return tag.find(child_tag_name).find('value').text
    except (AttributeError, TypeError):
        # If tag or text not found, return "N/A"
        return "N/A"
```

Tools Used: Jupyter Notebook, Python, Pandas df, Microsoft Excel, SEC EDGAR Database