ACF Project

Yihao Li and Christopher Barry

Question 1 Data

Data Sets:

Name	Type	Observations
Microsoft	Daily Stock Price Data	252
Eastman Chemical	Daily Stock Price Data	1259
Renewable Energy Group	Daily Stock Price Data	1258
NRG Energy Inc	Daily Stock Price Data	1259
Option OTM (only extrinsic value)	Different options call/put	8095
Option 2000	Different options call/put	2000

Question 1 Data (Cont)

Data Sets:

Name	Type	Observations
Facebook	Minute HFT	17160
APPLE	Minute HFT	5850
Google 2018	Minute HFT	29025
Google 2019	Minute HFT	17550
Music Data	Sound	6705

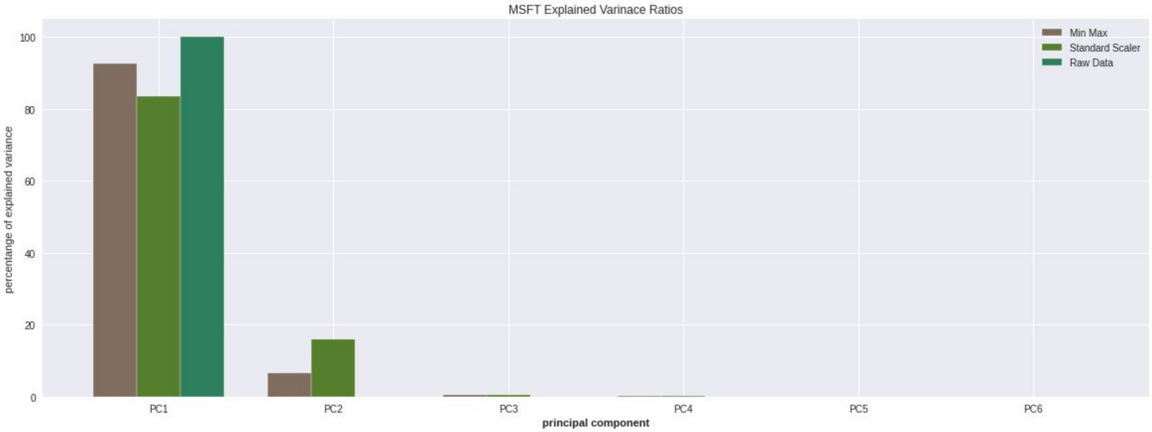
Variables

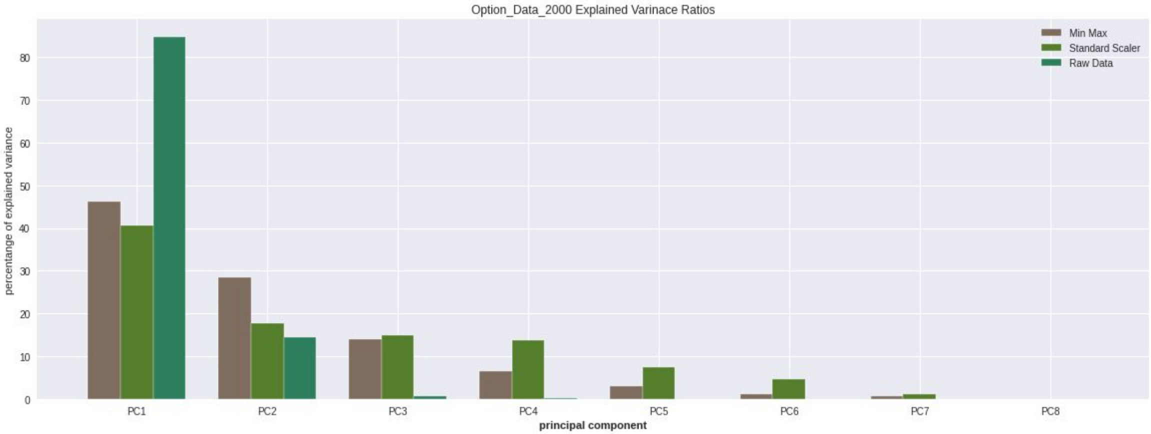
DataSet	Variables
Microsoft	Open,High,Low,Close,Adj Close,Volume
Eastman Chemical	Open,High,Low,Close,Adj Close,Volume
Renewable Energy Group	Open,High,Low,Close,Adj Close,Volume
NRG Energy Inc	Open,High,Low,Close,Adj Close,Volume
Option OTM (only extrinsic value)	Option_type,Ask,Bid,Option_price,Stock_price,Strike_price,Volatility,Volume,Time_to_maturity,Implied_volatility
Option 2000	Stock_Price,Strike_Price,Time_to_Maturity,Interest_Rate,Option_Price,Option_Type,Volatility,Implied Volatility

Variables

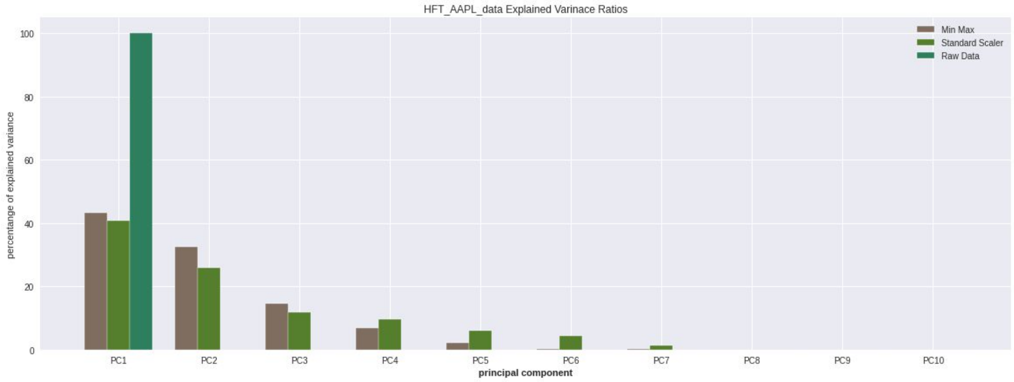
DataSet	Variables
Facebook	average,changeOverTime,close,date,high,label,low,marketAverage,marketChangeOverTime,marketClose,marketHigh,marketLow,marketNotional,marketNumberOfTrades,marketOpen,marketVolume,minute,notional,numberOfTrades,open,volume
APPLE	Date,marketAverage,marketChangeOverTime,marketClose,marketHigh,marketLow,marketNotional,marketNumberOfTrades,marketOpen,marketVolume
Google 2018	Open,High,Low,Close,Adj Close,Volume
Google 2019	average,changeOverTime,close,date,high,label,low,marketAverage,marketChangeOverTime,marketClose,marketHigh,marketLow,marketNotional,marketNumberOfTrades,marketOpen,marketVolume,minute,notional,numberOfTrades,open,volume
Music Data	Sound Frequency at different times/interval

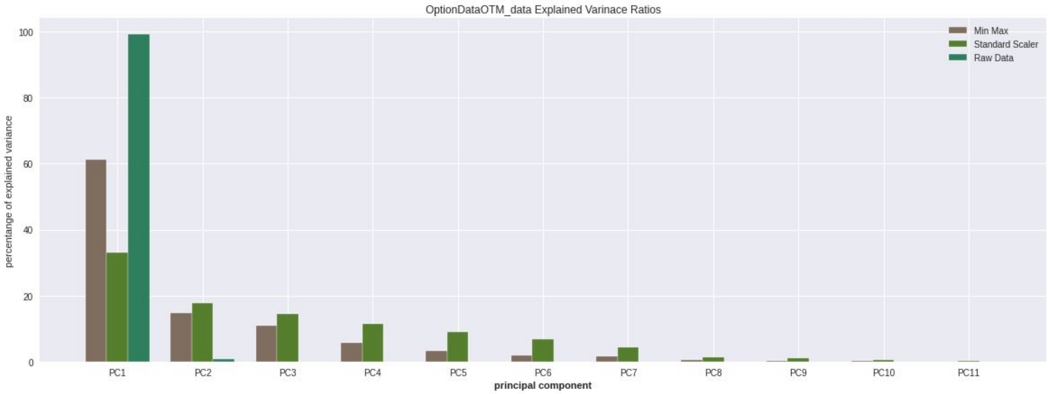
PCA with Difference VCR



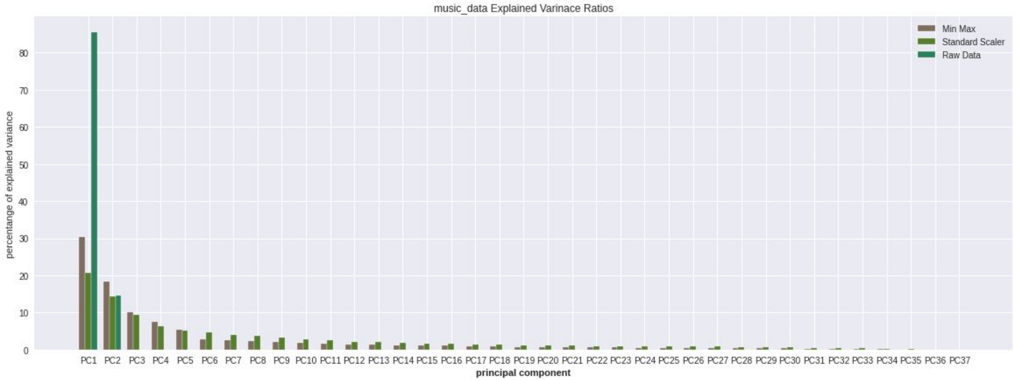


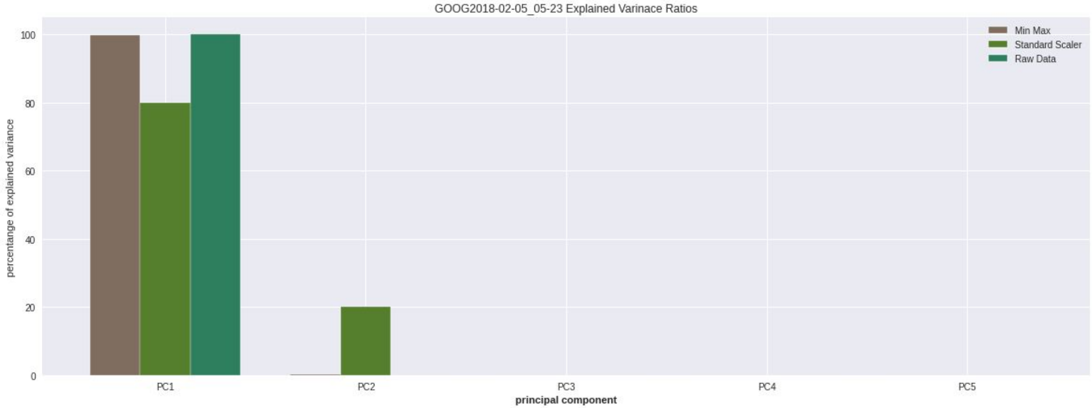
PCA with Difference VCR (Cont)



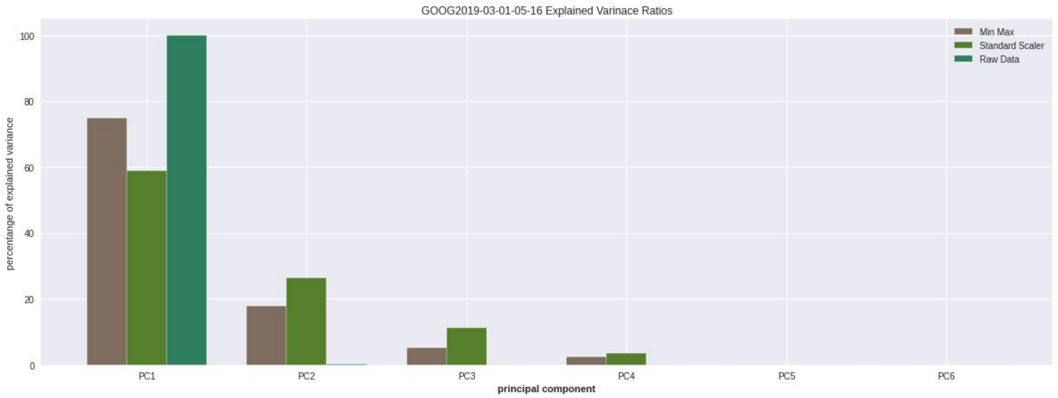


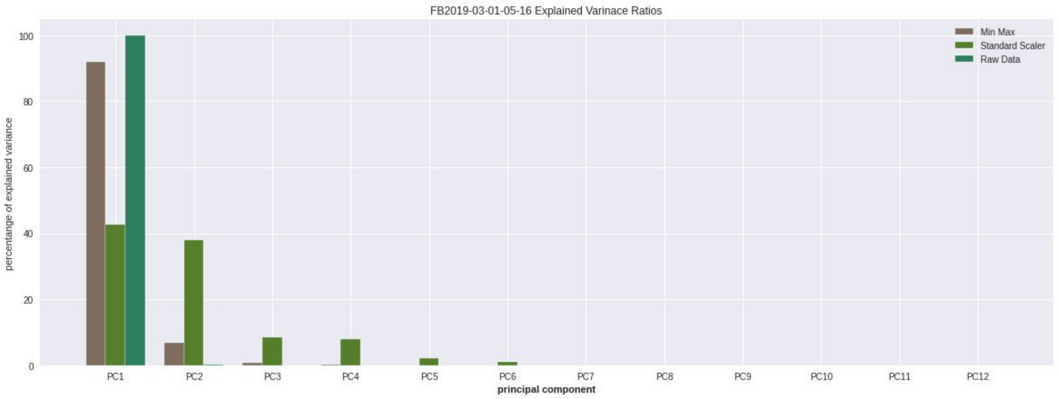
PCA with Difference VCR (Cont)



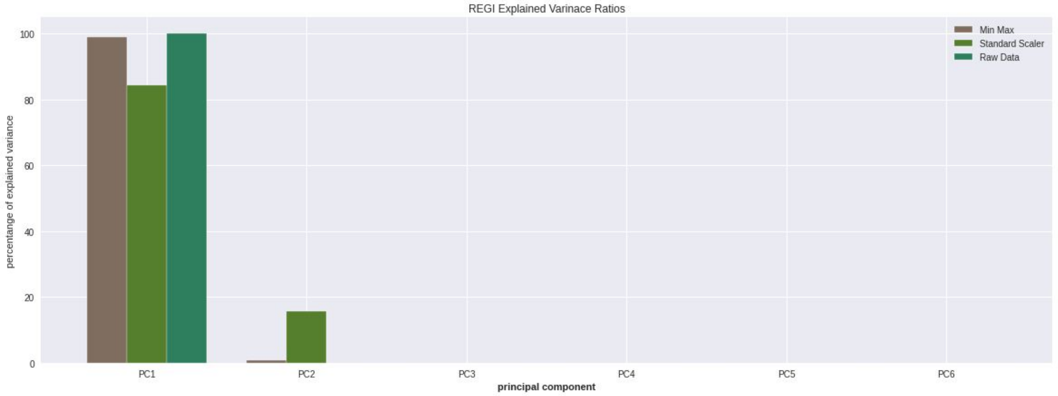


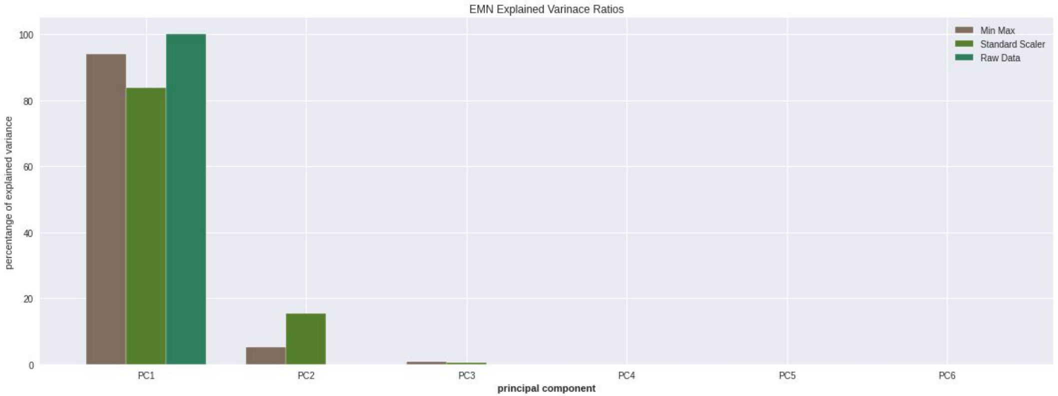
PCA with Difference VCR (Cont)



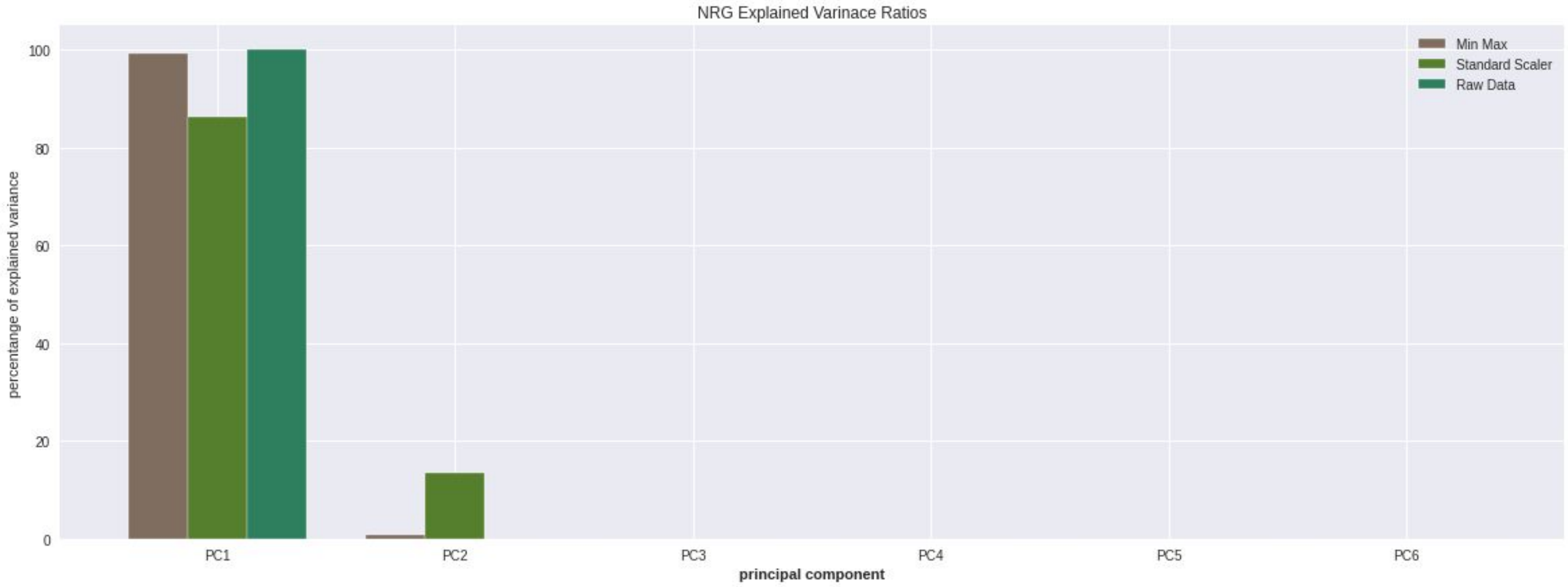


PCA with Difference VCR (Cont)





PCA with Difference VCR (Cont)



Conclusions

PCA 1 is typically highest with raw data

PCA 1 is typically 2nd highest with min max scaling

PCA 1 is lowest when using standard scaler

Difference becomes much more apparent with more variables between raw and transformed data

Transformed data should be used

Question 2 - Data Brief

The Apple Stock High Frequency Trading Data

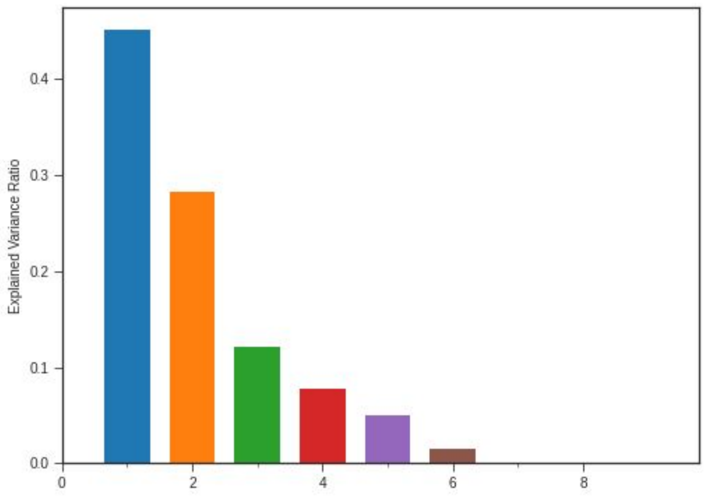
5850 rows, 10 columns.

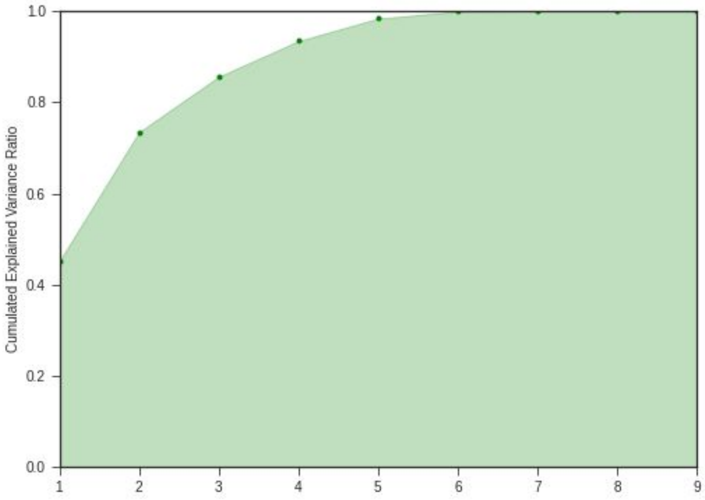
#	Column	Non-Null Count	Dtype
0	Date	5850 non-null	object
1	marketAverage	5850 non-null	float64
2	marketChangeOverTime	5850 non-null	float64
3	marketClose	5850 non-null	float64
4	marketHigh	5850 non-null	float64
5	marketLow	5850 non-null	float64
6	marketNotional	5850 non-null	float64
7	marketNumberOfTrades	5850 non-null	int64
8	marketOpen	5850 non-null	float64
9	marketVolume	5850 non-null	int64

	Date	marketAverage	marketChangeOverTime	marketClose	marketHigh	marketLow	marketNotional	marketNumberOfTrades	marketOpen	marketVolume
0	2/1/2019 9:30	167.058	0.000000	167.150	167.55	166.67	1.319461e+08	1766	166.930	789821
1	2/1/2019 9:31	167.182	0.000742	167.175	167.42	166.80	2.853055e+07	1143	167.210	170656
2	2/1/2019 9:32	167.051	-0.000042	166.910	167.20	166.88	2.232776e+07	1016	167.170	133658
3	2/1/2019 9:33	166.945	-0.000676	166.958	167.11	166.77	2.014459e+07	839	166.970	120666
4	2/1/2019 9:34	167.045	-0.000078	167.180	167.19	166.92	1.707804e+07	843	166.960	102236

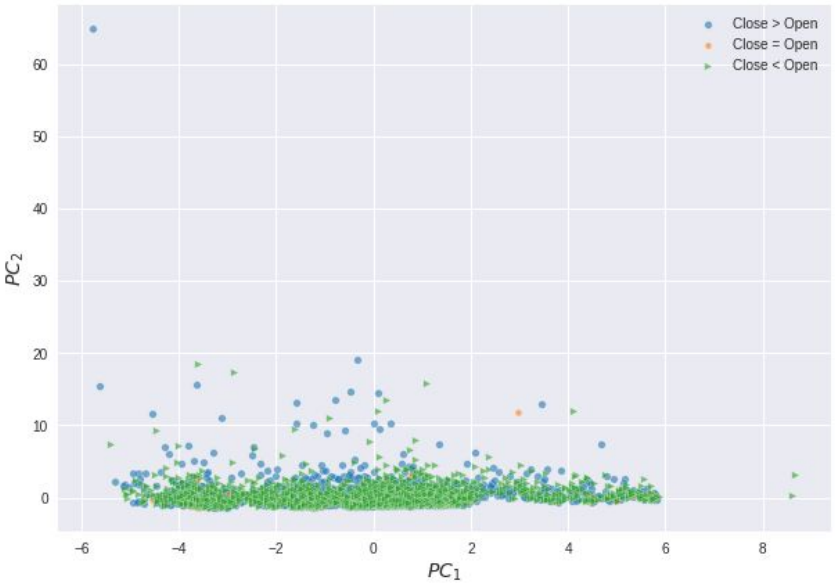
Question 2 - Data Brief

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Exp_Ratio	0.451853	0.282853	0.121564	0.078259	0.049272	0.015553	0.000574	0.000069	0.000004
Exp_Ratio_Cu	0.451853	0.734705	0.856269	0.934528	0.983800	0.999353	0.999927	0.999996	1.000000

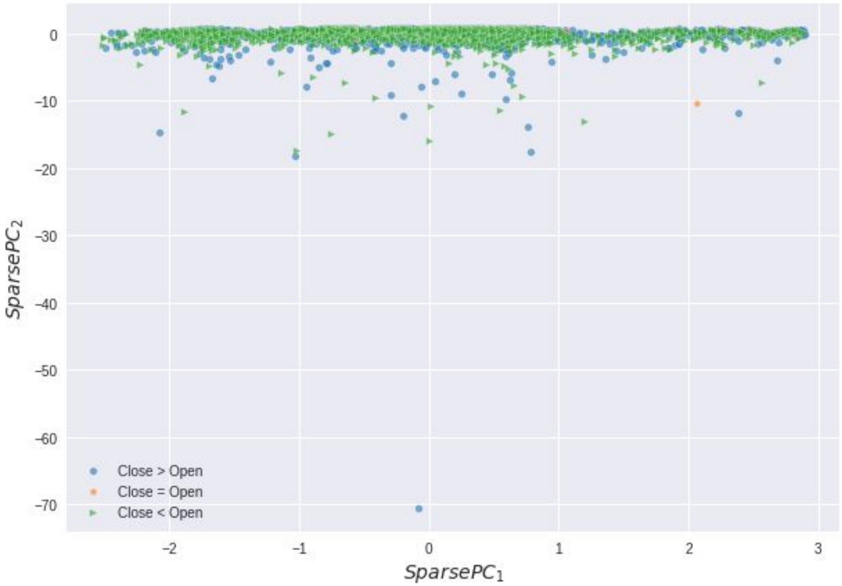




Question 2 - PCA & Manifold Learning

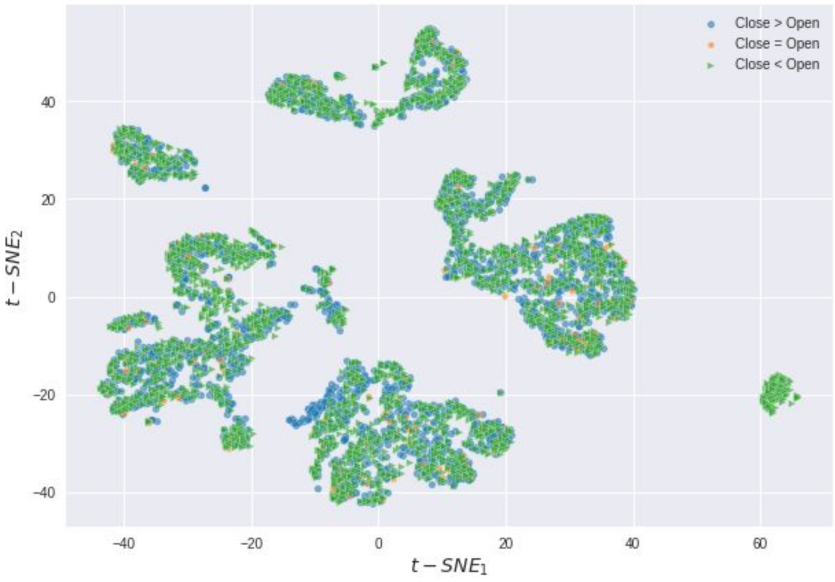


PCA

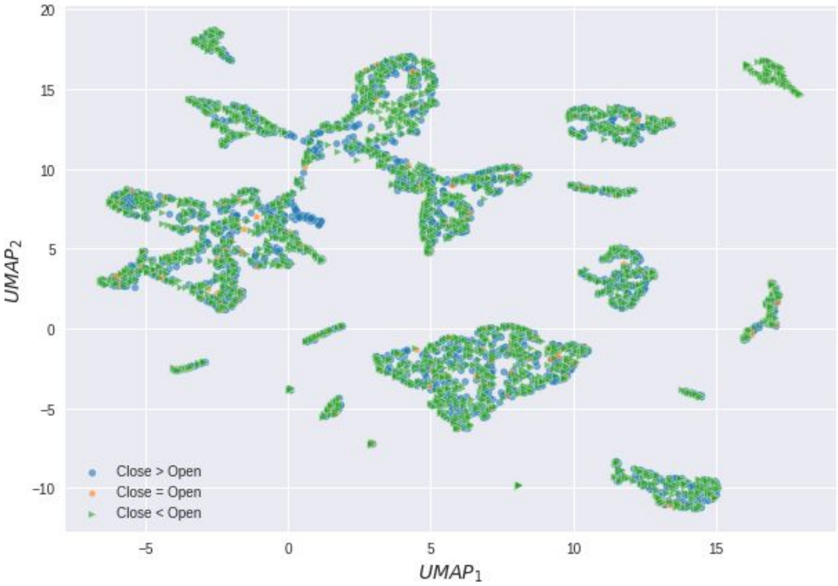


Sparse PCA: Sparseness = 0.8

Question 2 - PCA & Manifold Learning

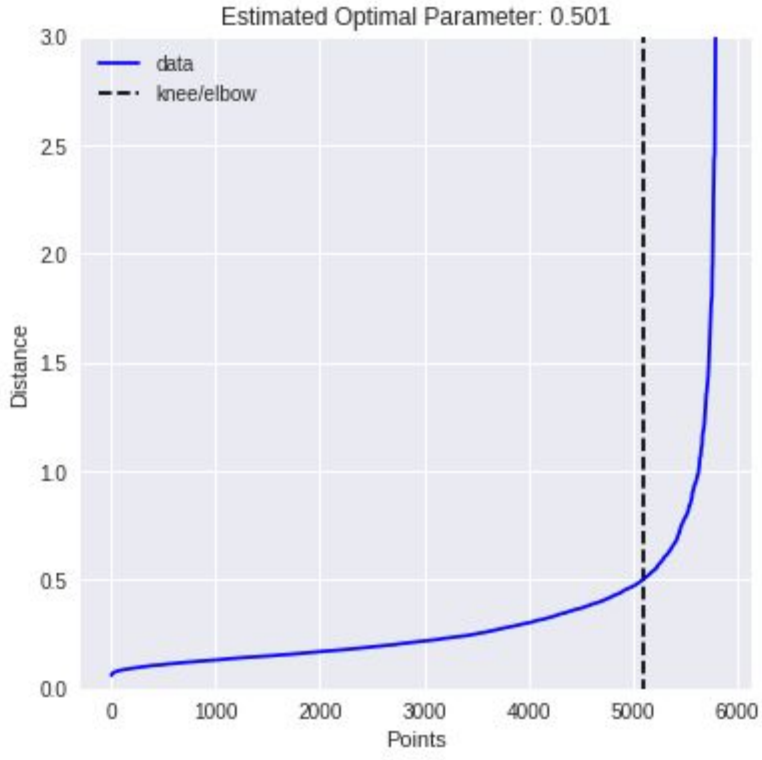


t-SNE: Perplexity = 100, init = ‘pca’



UMAP: n_neighbors = 15, min_dist = 0.2

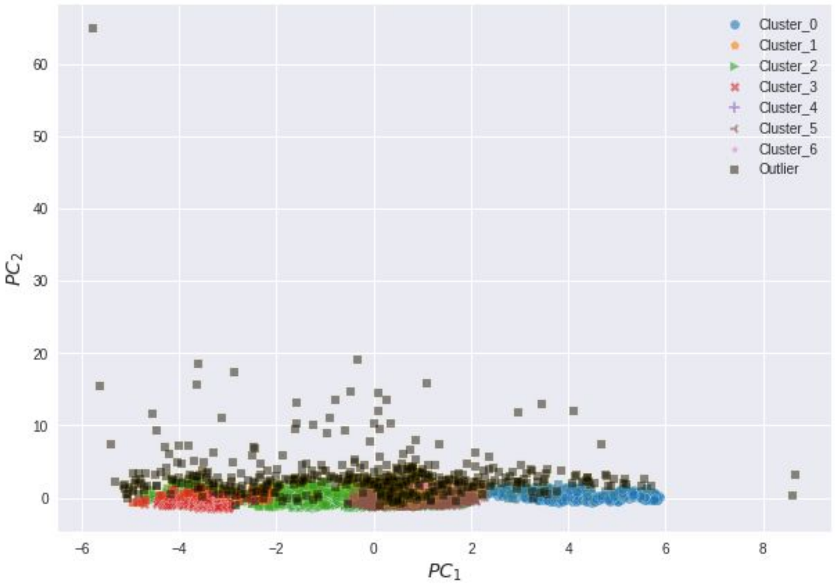
Question 2 - Elbow for DBSCAN



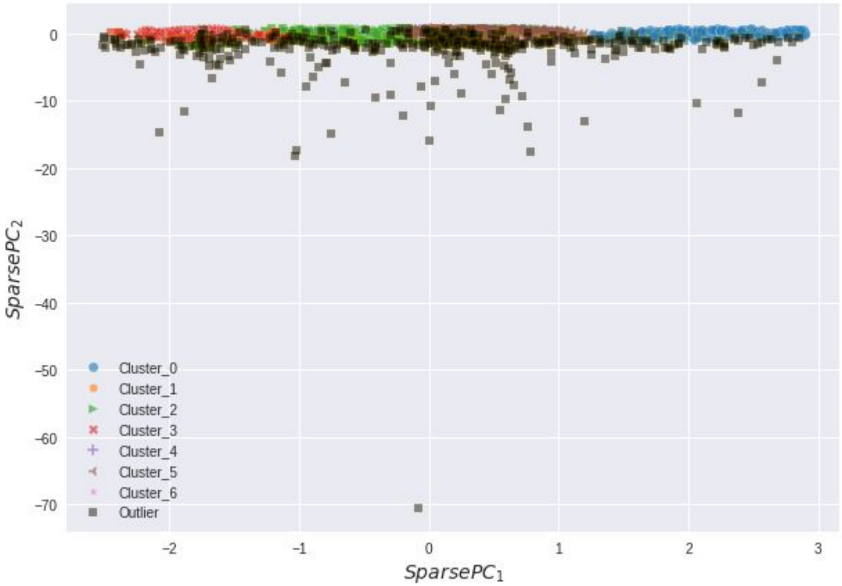
eps = 0.5 , min_samples = 10

```
Estimated number of clusters: 7
Estimated number of noise points: 464
```

Question 2 - DBSCAN

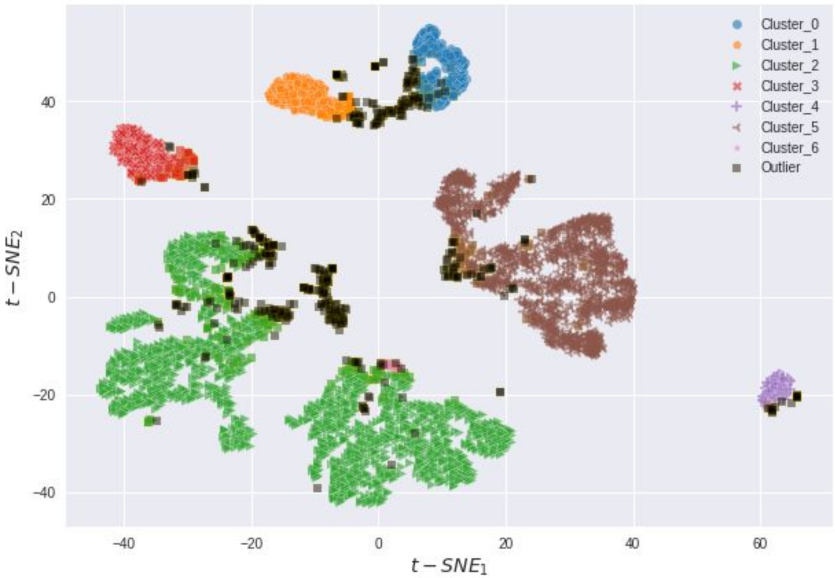


PCA

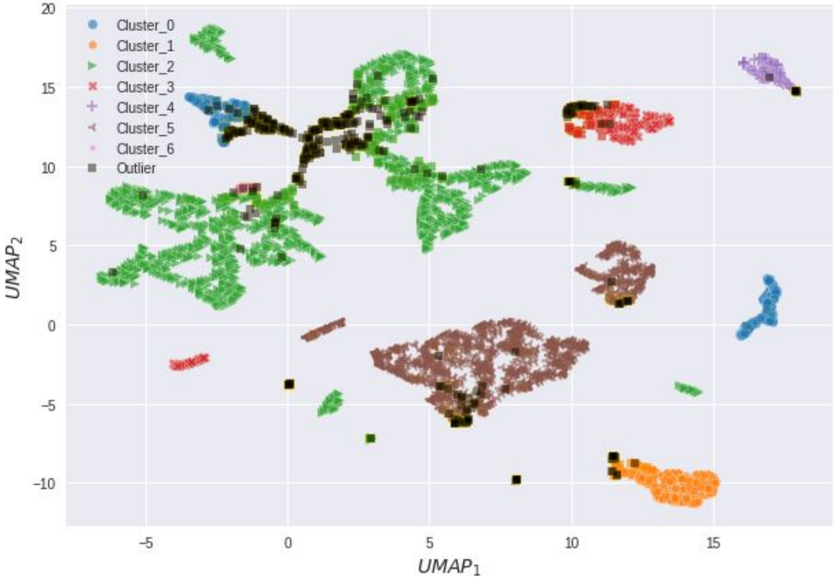


Sparse PCA: Sparseness = 0.8

Question 2 - DBSCAN

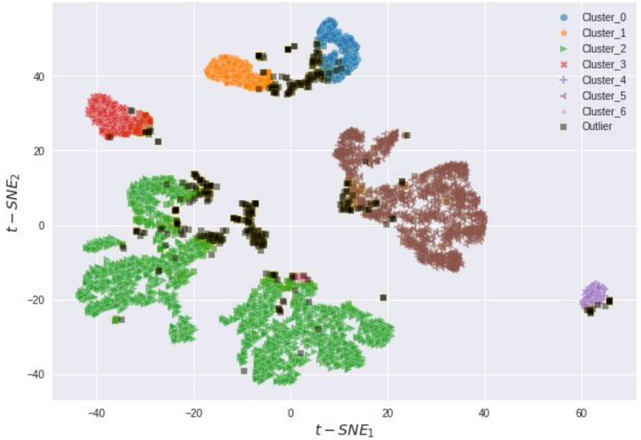


t-SNE: Perplexity = 100, init = ‘pca’

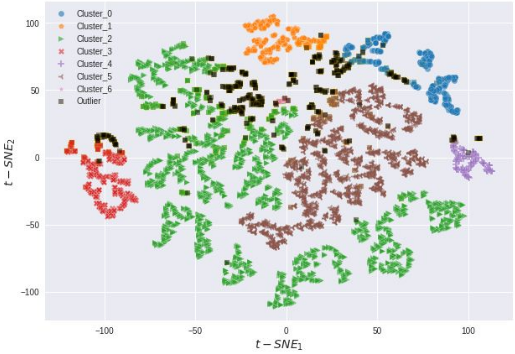


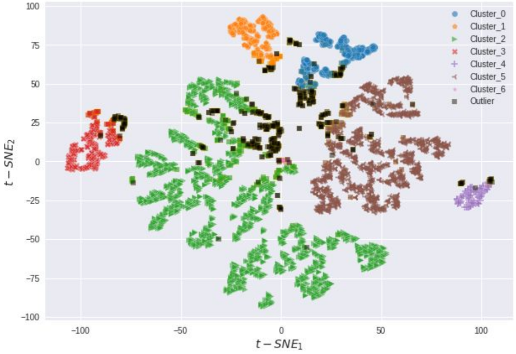
UMAP: n_neighbors = 15, min_dist = 0.2

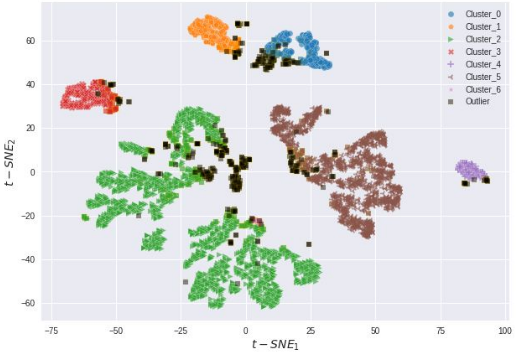
Question 2 - Parameter Tuning

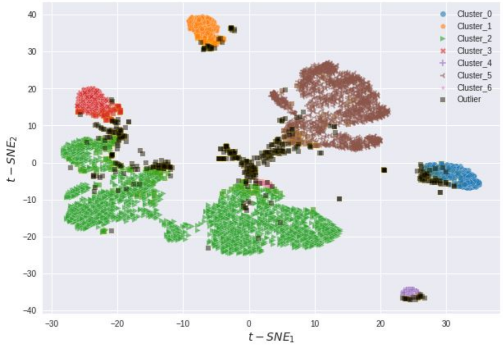


t-SNE: Perplexity = 100, init = ‘pca’







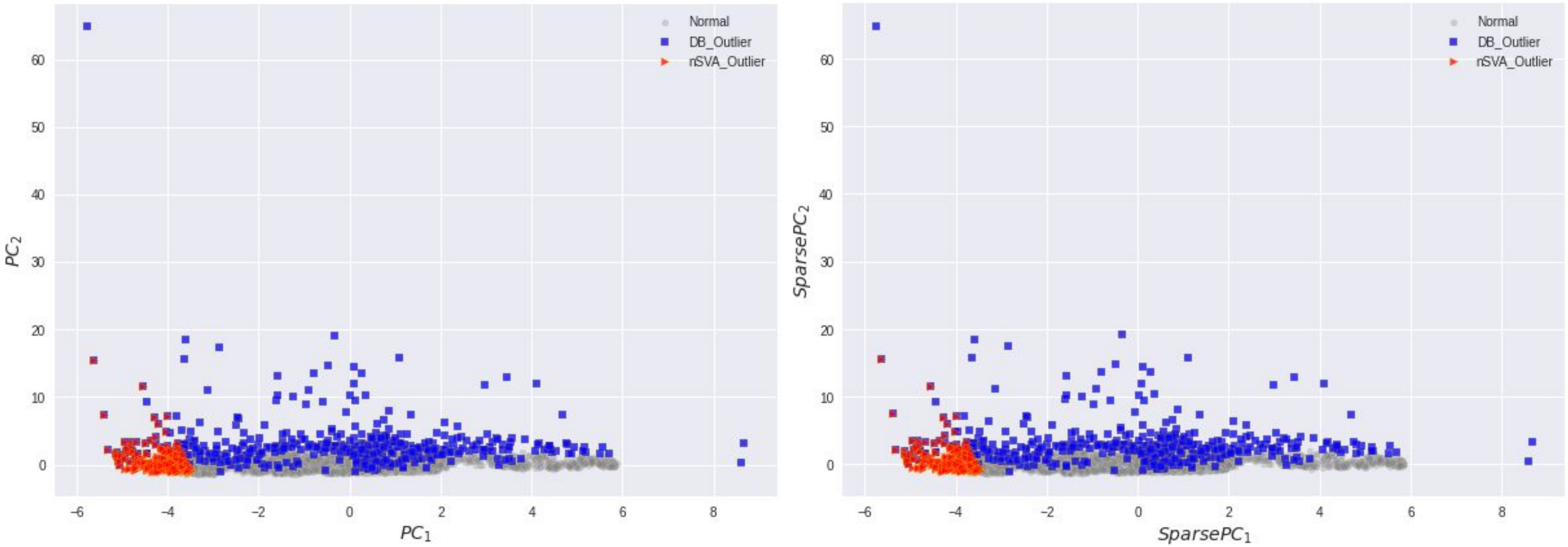


Question 2 - Outliers

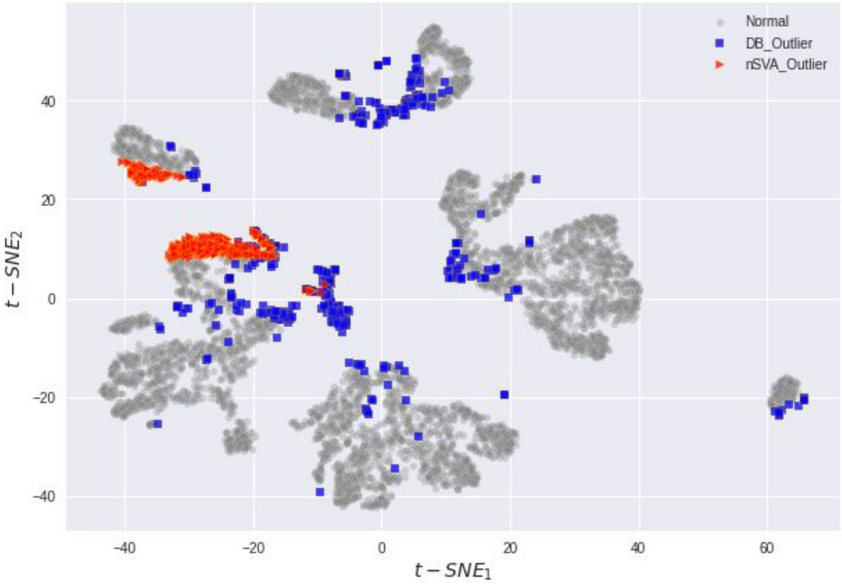
```python
# nSVA Outliers
def nsva_ranking(data):
    minmax_data = MinMaxScaler().fit_transform(data)
    if np.sum(np.sum(minmax_data<0))>0:
        print(" \nError: Data must be a nonnegative matrix\n")
        ranking_score='Negative'
    else:
        u,s,v =np.linalg.svd(minmax_data, full_matrices=False)
        ranking_score=-u[:,0]
    return ranking_score

def nsva_outlier(data, outlier_n=100):
    ranking_score = nsva_ranking(data)
    ranking = pd.Series(ranking_score.copy())
    r0 = np.argsort(ranking_score)[::-1]
    for i in range(len(ranking)):
        ranking[r0[i]] = int(i)
    nsva_outlier_label = ranking < outlier_n
    return nsva_outlier_label
```

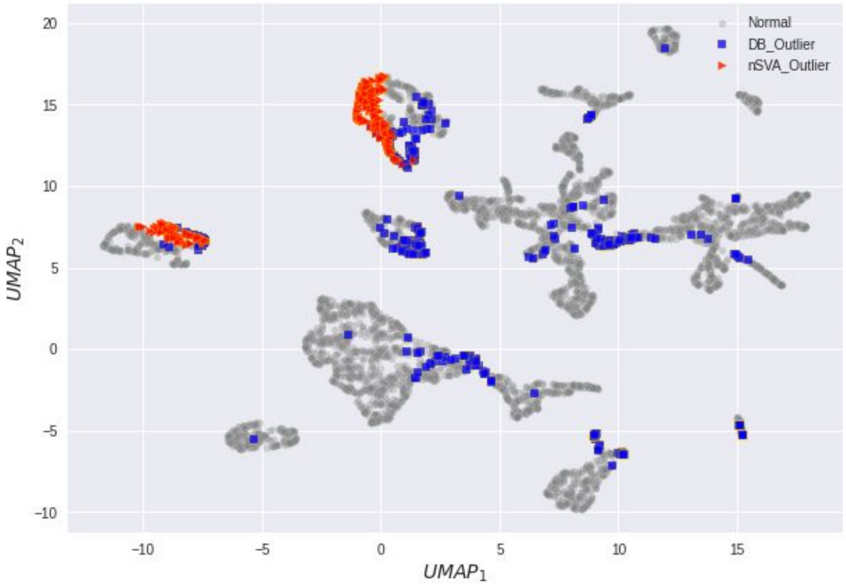
	marketAverage	marketChangeOverTime	marketClose	marketHigh	marketLow	marketNotional	marketNumberOfTrades	marketOpen	marketVolume
786	174.129	0.007306	173.990	174.300	172.930	6.058726e+07	2356	174.030	347945
792	174.246	0.007983	174.447	174.480	173.790	5.828713e+07	2336	173.940	334510
793	174.363	0.008660	174.400	174.480	173.995	4.442417e+07	1372	174.430	254780
794	174.325	0.008440	174.339	174.430	174.230	2.757585e+07	1057	174.415	158186
795	174.298	0.008284	174.265	174.430	174.180	2.195587e+07	920	174.360	125967
...	...	...	...	...	...	...	...	...	...
1348	174.481	-0.001882	174.420	174.570	174.410	1.777959e+07	490	174.500	101900
1360	174.518	-0.001670	174.540	174.580	174.410	1.196410e+07	418	174.411	68555
1361	174.502	-0.001762	174.490	174.563	174.460	6.463910e+06	274	174.520	37042
1363	174.546	-0.001510	174.560	174.598	174.410	9.545554e+06	391	174.410	54688
1364	174.565	-0.001402	174.490	174.620	174.480	1.399139e+07	466	174.560	80150

Question 2 - Outliers in PCA/Manifold Space

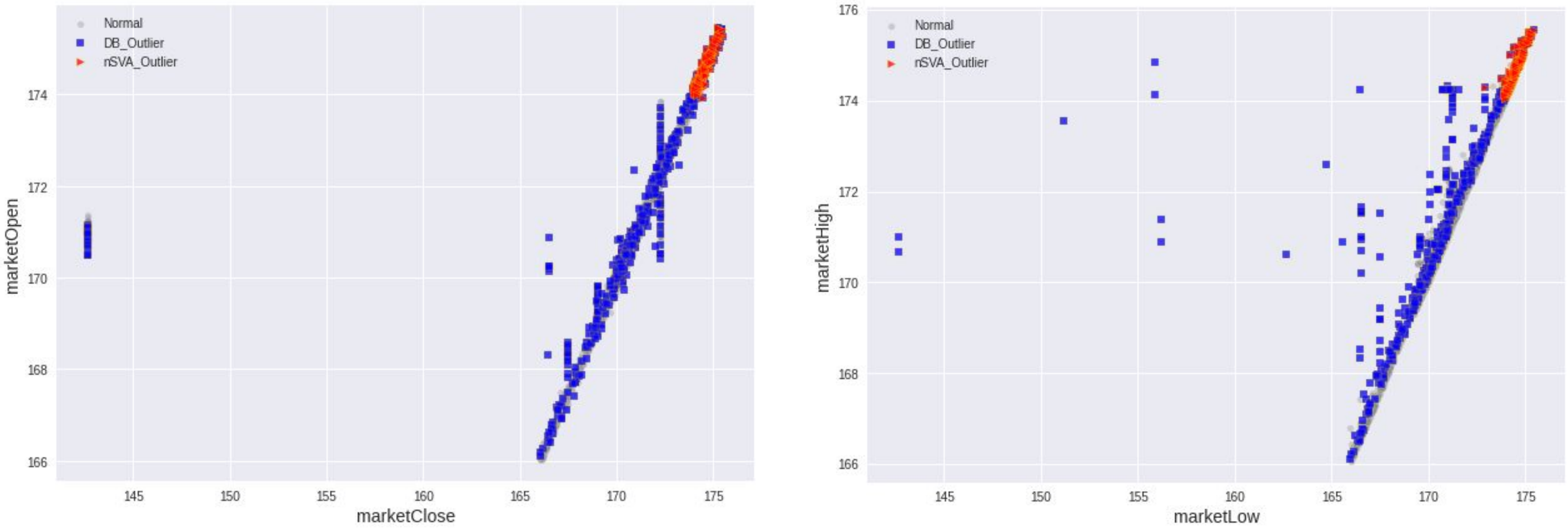


Question 2 - Outliers in PCA/Manifold Space

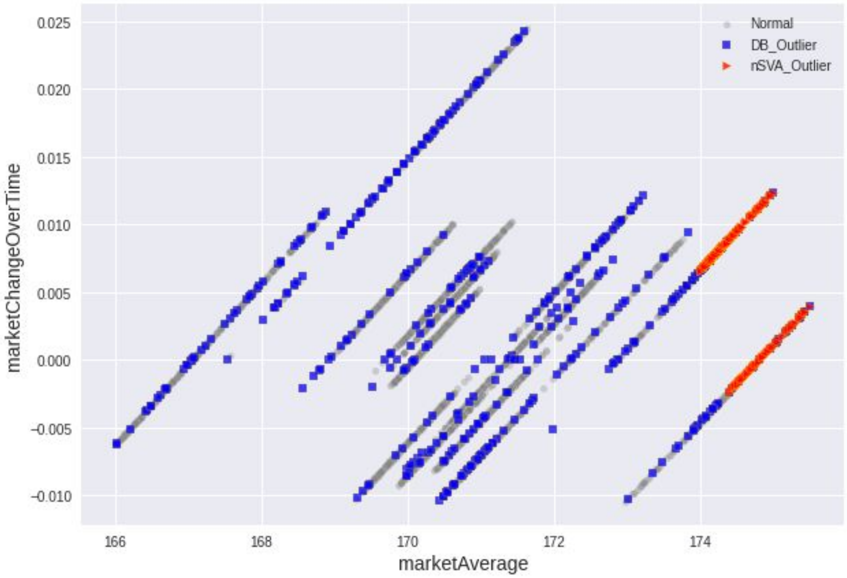


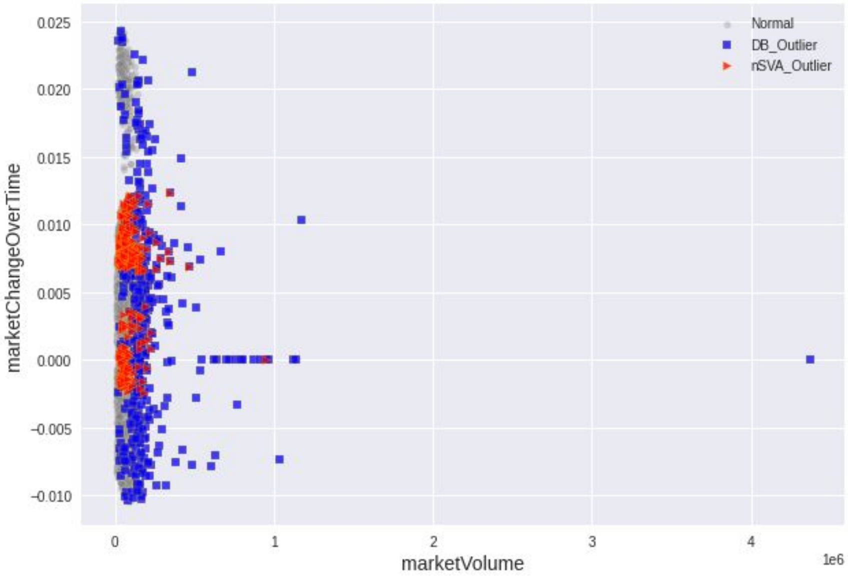


Question 2 - Outliers in Price Space



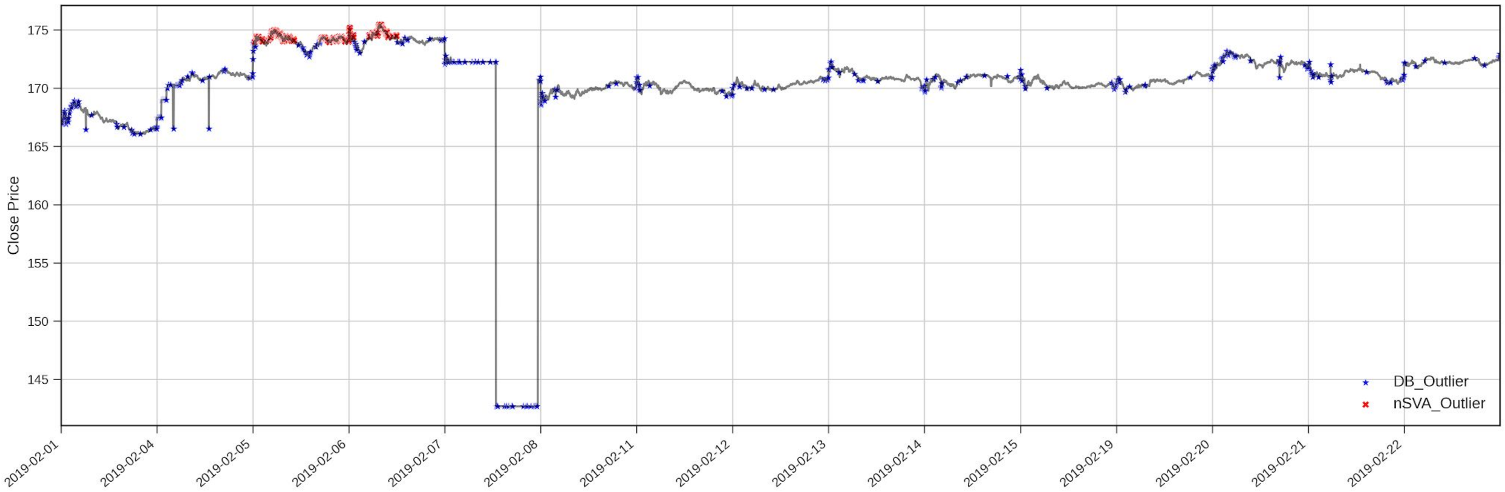
Question 2 - Outliers in Price Space



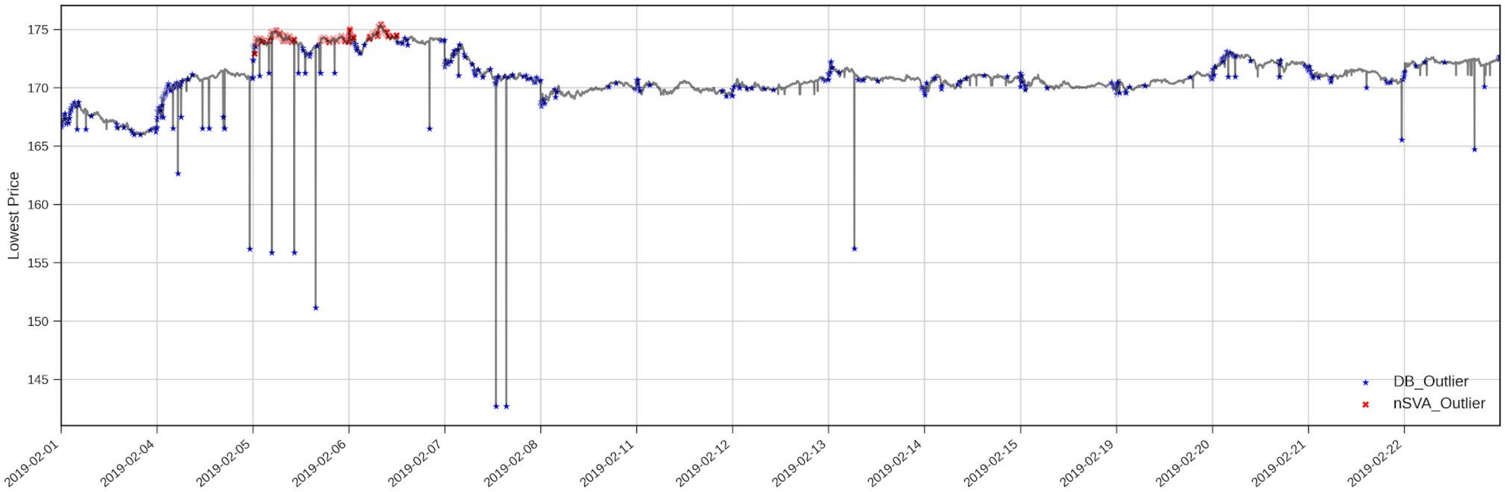


Question 2 - Outliers in Price Space (Revised)

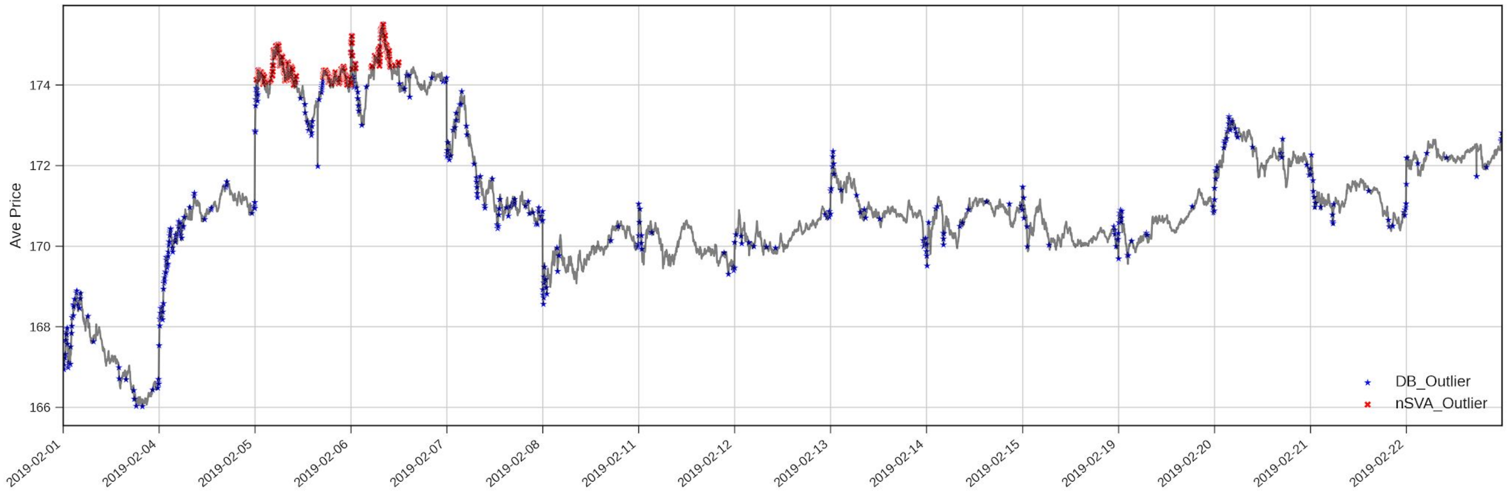
Question 2 - Outliers in Price Space (Revised)



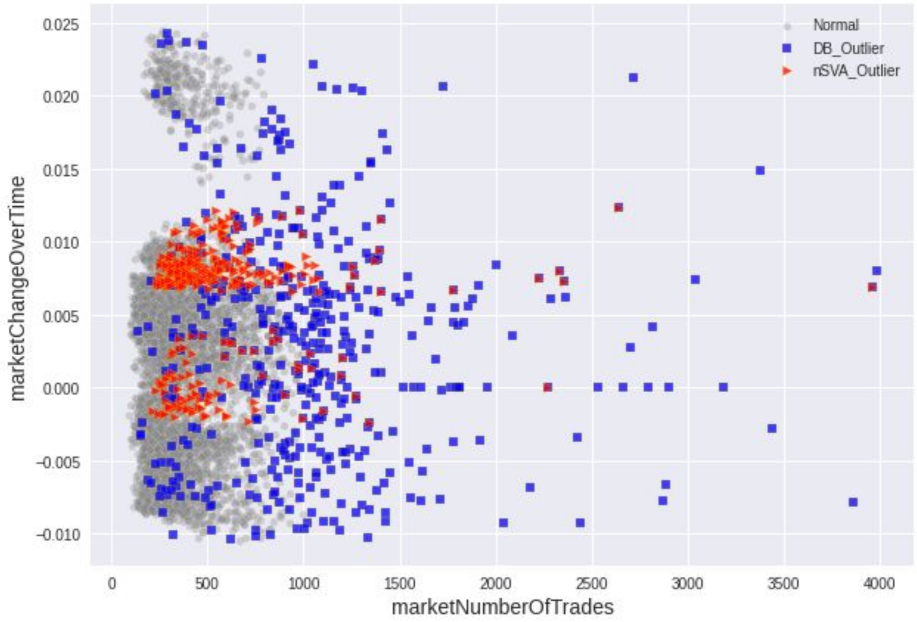
Question 2 - Outliers in Price Space (Revised)



Question 2 - Outliers in Price Space (Revised)



Question 2 - Outliers in Price Space



Question 2 - Polygon HFT data API

https://polygon.io

Custom Data Solutions for
Universities, Educators, and Students

Enabling students to analyze real-time and historical market data at an affordable price.

Accurate to One Billionth of a Second
Nanosecond Timestamps

999,999 units of accuracy more than the competition

Polygon.io Timestamp

Aug 8, 2019 01:17:57.682285707

Timestamp: 1565053186822857O7

- SIP Timestamp
- Participant/Exchange Timestamp
- Trade Reporting Facility (TRFs, Darkpools) Timestamp

Polygon provides all the timestamps to know exactly where and when the trade occurred. This combined with nanosecond timestamp accuracy provides unmatched tick details.

Competitor's Timestamp

Aug 8, 2019 01:17:57.682

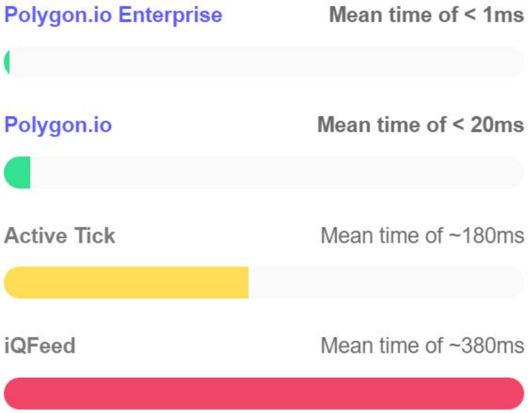
Timestamp: 156505318682

- SIP Timestamp Only
- Participant/Exchange Timestamp
- Trade Reporting Facility (TRFs, Darkpools) Timestamp

Our competitors only offer millisecond timestamps, as well as only 1 timestamp attribute. This leaves much to be desired.

Low Latency
Institutional Level Data

We are located in the same datacenters with NYSE, NASDAQ, BATS, IEX and the other top exchanges. We connect directly to the exchanges for an institutional level feed. This level of quality feed has been out of reach for end users.. until now.



Question 2 - Source: Polygon.io

~$199/month for US Stocks minute level data

7 day free trial

Unlimited Pulls - we pulled aggregate level data, but trade level data is available

https://github.com/polygon-io/client-python/blob/master/polygon/rest/client.py

https://polygon.io/docs/#get_v2_aggs_ticker__ticker__range__multiplier___timespan___from___to__anchor

Question 2 - Data

Minute Level Data

NA values were dropped

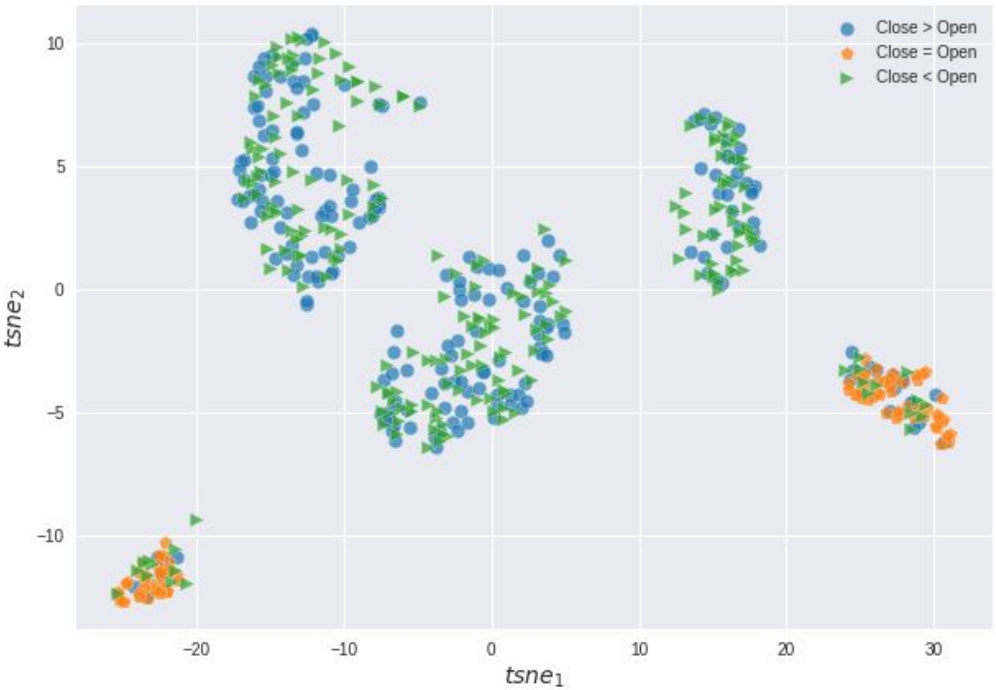
Minute level data from 2020-06-12 - 2020-06-15

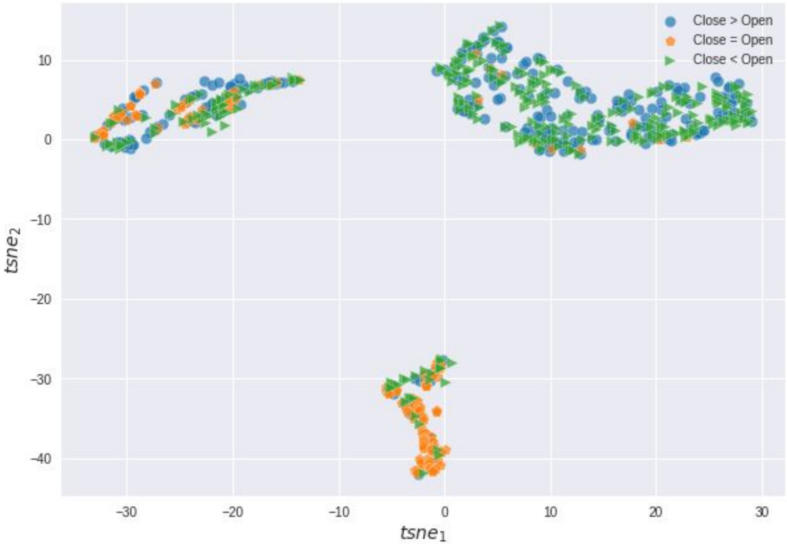
Roughly 1000 observations 13 and 14 were a weekend.

8 hours * 60 minutes a day 2 days = 960

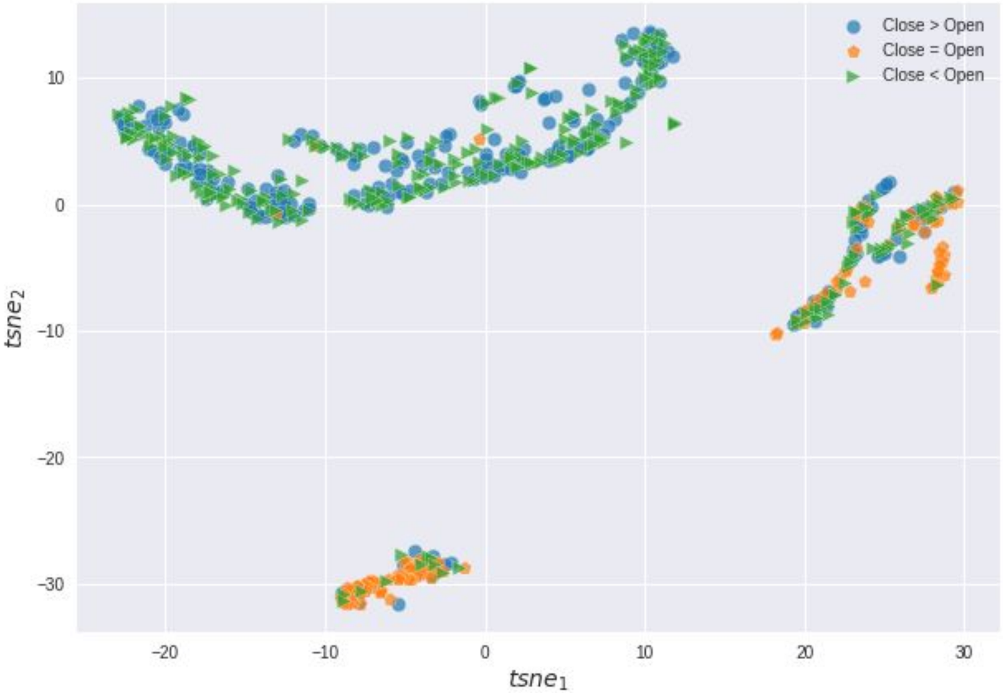
For AMZN NA values were ~50% of the data

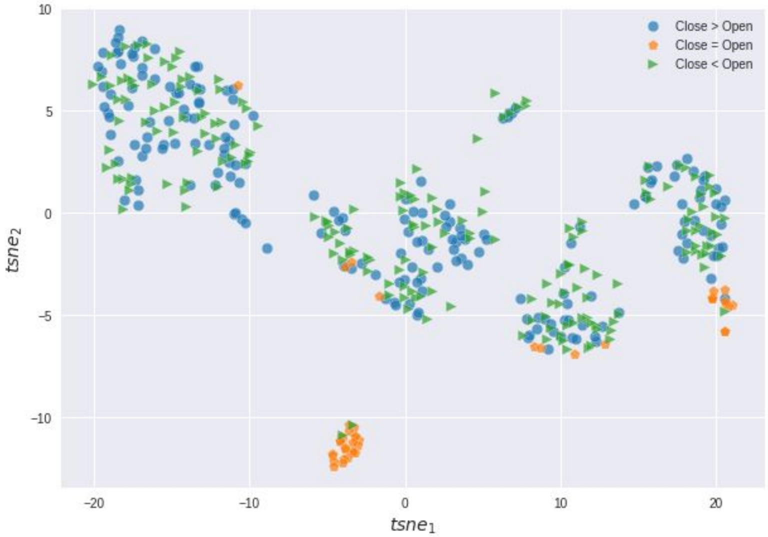
Question 2 - TSNE on HFT Amazon and Bank of America



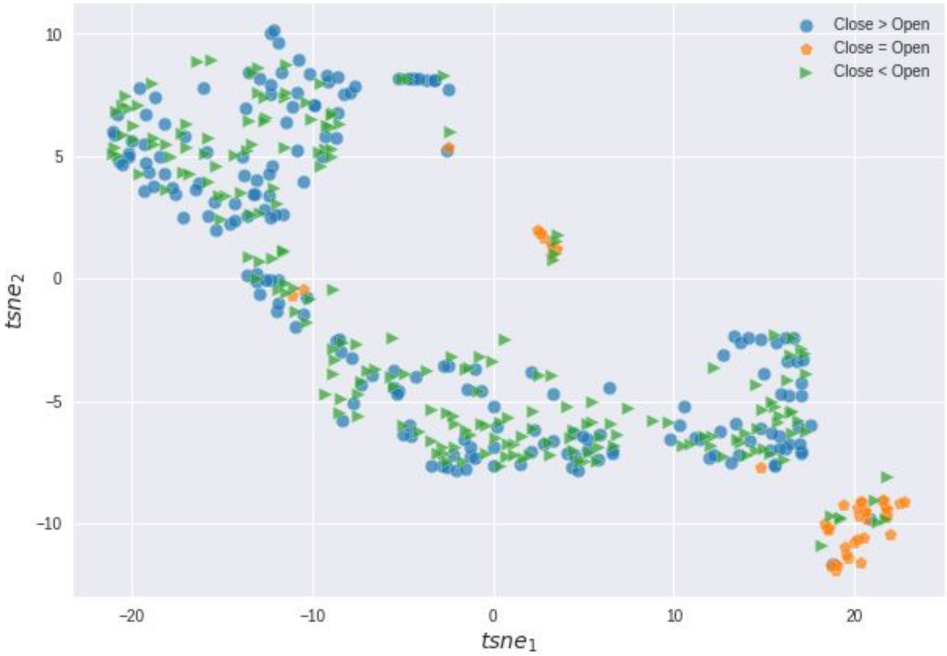


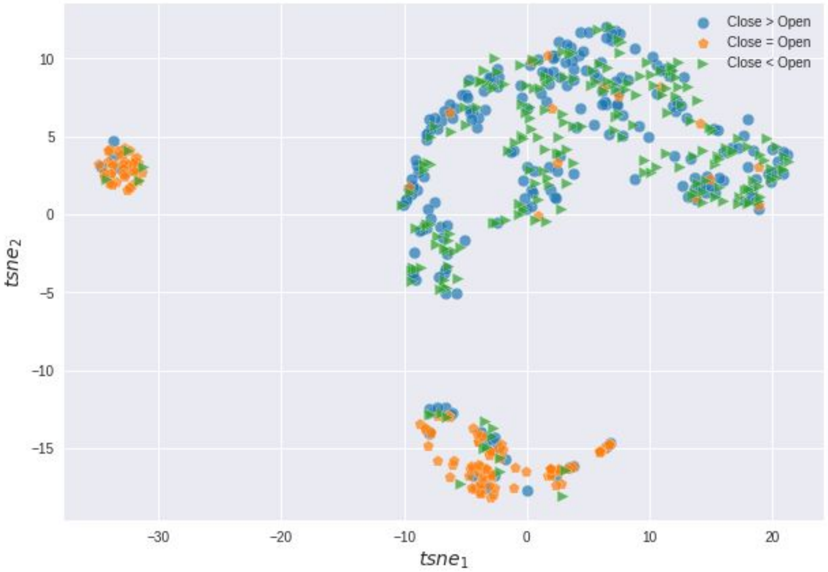
Question 2 - TSNE (Cont) Google and Facebook



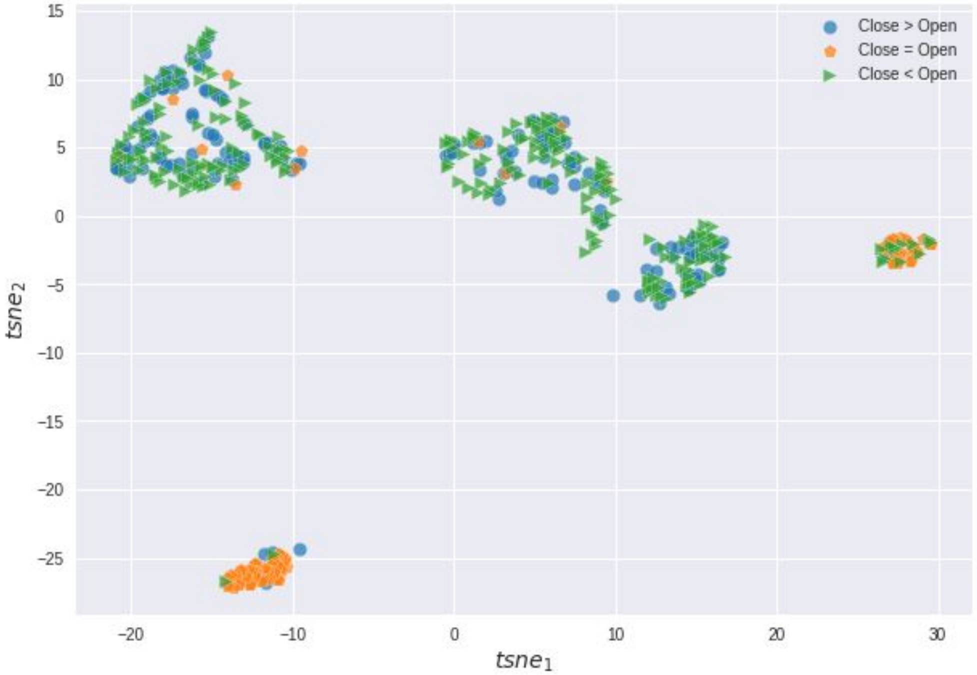


Question 2 - TSNE (Cont) Home Depot and Kohl's





Question 2 - TSNE on Walmart



Question 3

Evaluate the locality of dimension reduction methods for credit risk data

Credit Risk Data

Variables (11): Delinquence, Revolving Credit Percentage,Capital ReservesNum Late 60,Debt Ratio,Monthly Income,Num Credit Lines,Num Late Past 90,Num Real Estate,Num Late 90,Num Employees

Observations: 120270

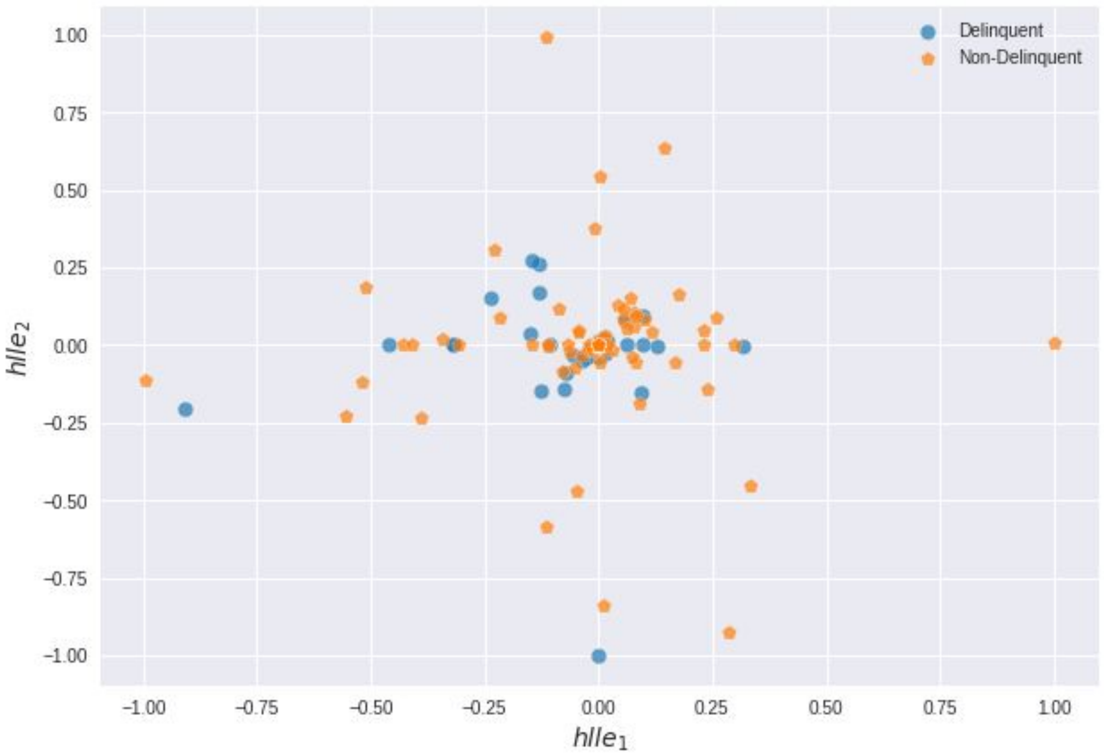
Run PCA,SPCA,TSNE,UMAP,MLLE,HLLE

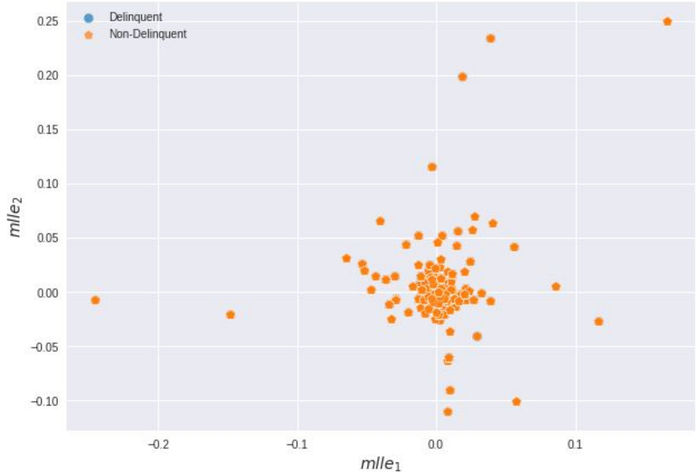
MLLE and HLLE could not be run on the entire dataset

Was broken into 6 equal parts to allow for RAM space

Results we put back together → better ways to do this with more time

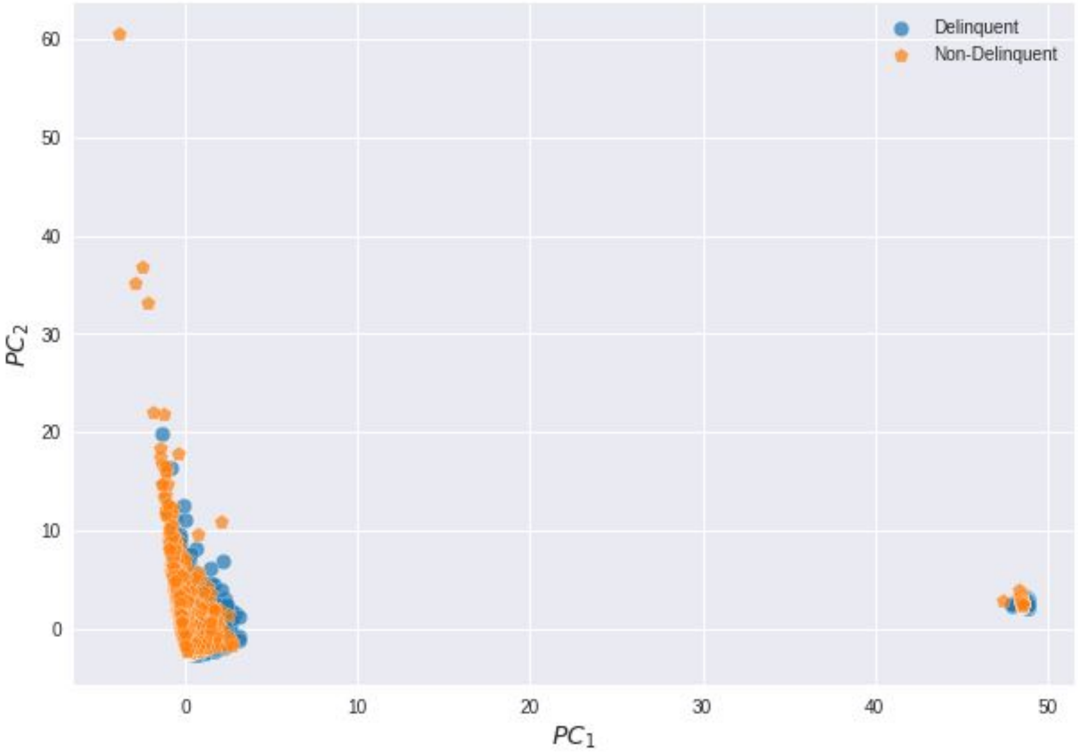
MLLE and HLLE

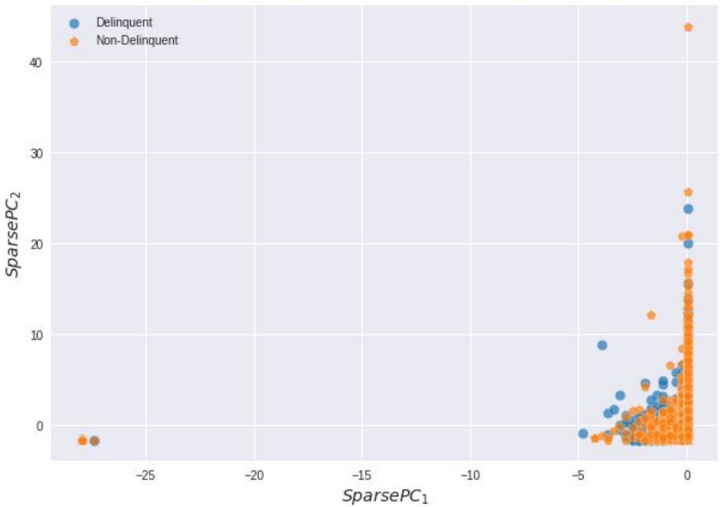




Neither very useful, most likely due to running the test on individual samples of the test and aggregating

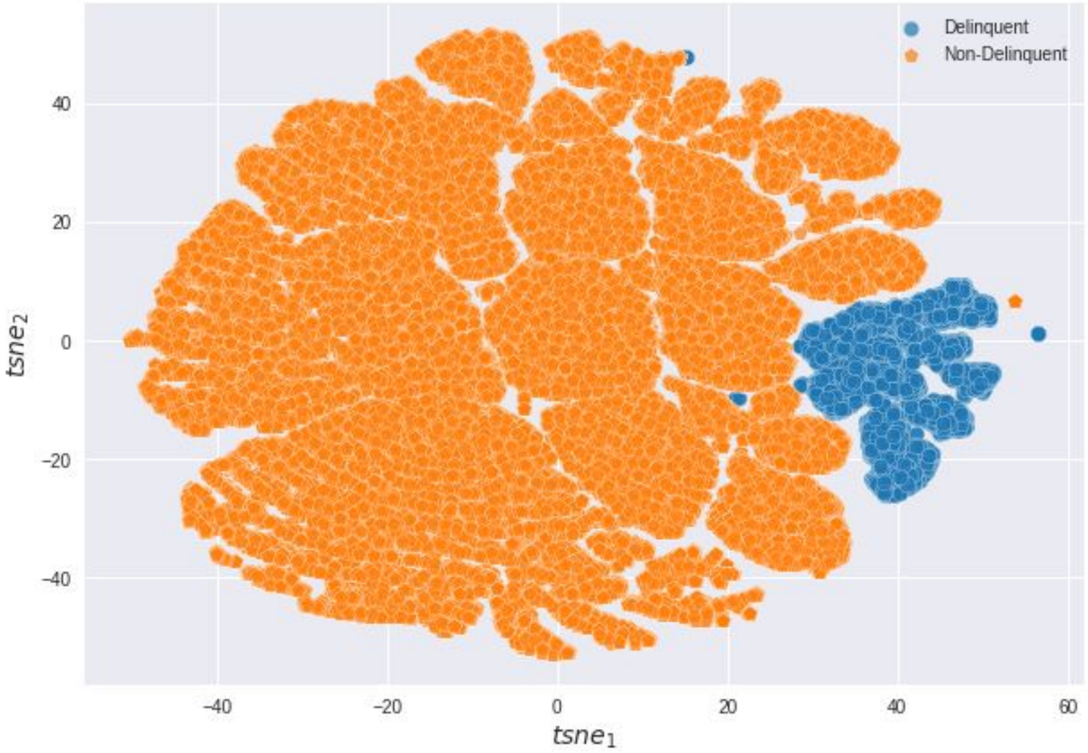
PCA and SPCA

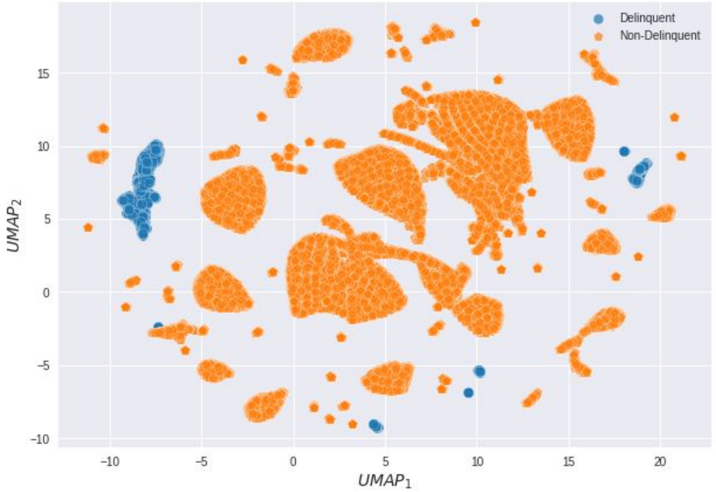




Delinquent scores grouped on edge of triangle cluster

TSNE and UMAP





Clusters seem to work well with both methods → best with TSNE

Random Sampling and kNN

```
def random_sample(data, size):
    if size > len(data):
        print(f'The required sample size is too large, the data only has {len(data)} records!')
        quit
    else:
        idx = np.random.choice(len(data),size,replace=False)
        idx.sort()
        return idx
```

```
def get_neighbors(data, n_neighbors=10, normalize=True):
    if normalize:
        data_ = StandardScaler().fit_transform(data)
    else:
        data_ = data
    nbrs = NearestNeighbors(n_neighbors=(10+1))
    neighbors = nbrs.fit(data_)
    distances, indices = neighbors.kneighbors(data_)
    nb = indices[:,1:]
    return nb
```

```
def overlap_rate(n1, n2, idx=None):
    if len(n1) != len(n2):
        print('Two data should have same length!')
        quit
    else:
        rates = []
        if idx is None:
            idx = range(len(n1))
        for i in idx:
            rate = len(set(n1[i])&set(n2[i]))/len(set(n1[i]))
            rates.append(rate)
        ave_rate = np.mean(rates)
        return ave_rate
```

Random Sampling and kNN

	Neighbor_1	Neighbor_2	Neighbor_3	Neighbor_4	Neighbor_5	Neighbor_6	Neighbor_7	Neighbor_8	Neighbor_9	Neighbor_10
1404	86079	1969	56237	56282	38143	85812	105742	27347	39730	73192
1883	37734	97678	12352	27600	93106	10751	104749	97345	89880	119098
2111	12661	35786	8471	25822	40101	45231	6384	33429	20028	4572
2374	90472	106864	119447	18655	109982	60248	13523	40199	43152	3069
2388	67201	84077	66154	20421	57700	57711	23076	111149	30803	33127
...	...	...	...	...	...	...	...	...	...	...
117697	72871	85776	114983	94045	44735	113132	101430	70862	118836	118814
117975	33707	50652	690	111186	45752	54949	82209	85661	33059	119809
118643	28924	93048	83754	6617	88959	87849	16818	16946	79246	22414
118885	24345	22188	29544	54437	111451	102611	116751	47802	88456	14937
119898	17966	40102	65252	71599	63897	98716	37032	103554	49837	66537

200 rows × 10 columns

Overlap rates

```
The 10-nearest neighbor overlap rate for original space and PCA space is 0.074.
The 10-nearest neighbor overlap rate for original space and SPCA space is 0.074.
The 10-nearest neighbor overlap rate for original space and TSNE space is 0.504.
The 10-nearest neighbor overlap rate for original space and UMAP space is 0.377.
```