Final Paper – Not just another asset management firm (Finding profitable cointegrated pairs for implementing a high frequency trading strategy using S&P 500 companies)

Author – *Archana Singh Parihar*

Data sourcing into Big data platform and Data Cleaning

Data sourcing is undoubtedly the most crucial aspect of data warehousing. If the data source systems are not understood correctly, then the dimensional modelling can be incorrect when we store the data into our Data warehouse. This may also result in incorrect ETL specifications. Having a complete understanding of the data source ensures data quality and integrity in the system. This leads to reliable analysis for decision making.

Data quality can increase our efficiency, reduce the cost of rework and improves the success rate of enterprise initiatives like Business Intelligence.

We have three data sources with a different data structure. The need is to collate data from all three sources into a big data platform through an ETL tool and a data management tool and clean it before performing a commit.

The benefit of performing ETL on the data during the sourcing stage instead of during the visualization stage is to better the performance. I am choosing to use 'Informatica' to collate, combine and compare the data from these different sources so that it can be made to work as a seamless whole. Informatica allows for data integration from these different databases.

I have decided to use Informatica as the ETL tool because it is the industry leader in Data integration as per the Gartner report of 2017.

Data sources:

1. CSV file from Kaggle containing all the stock data for five years. (number of rows = 60k+)
2. S&P 500 company information from **Wikipedia** (**https://en.wikipedia.org/wiki/List_of_S%26P_500_companies**) (Sector and Sub-Industry)
3. S&P 500 company information from **https://www.suredividend.com/**

Data flow Sourcing -> ETL -> Ingest in Azure Warehouse -

Web scraping from two sources will be done using Python libraries like (Selenium and Beautiful soup). This information, along with the five years of stock market data from Kaggle will be inserted in a shared repository like 'SharePoint' or 'One Drive' belonging to the company (for security purposes).

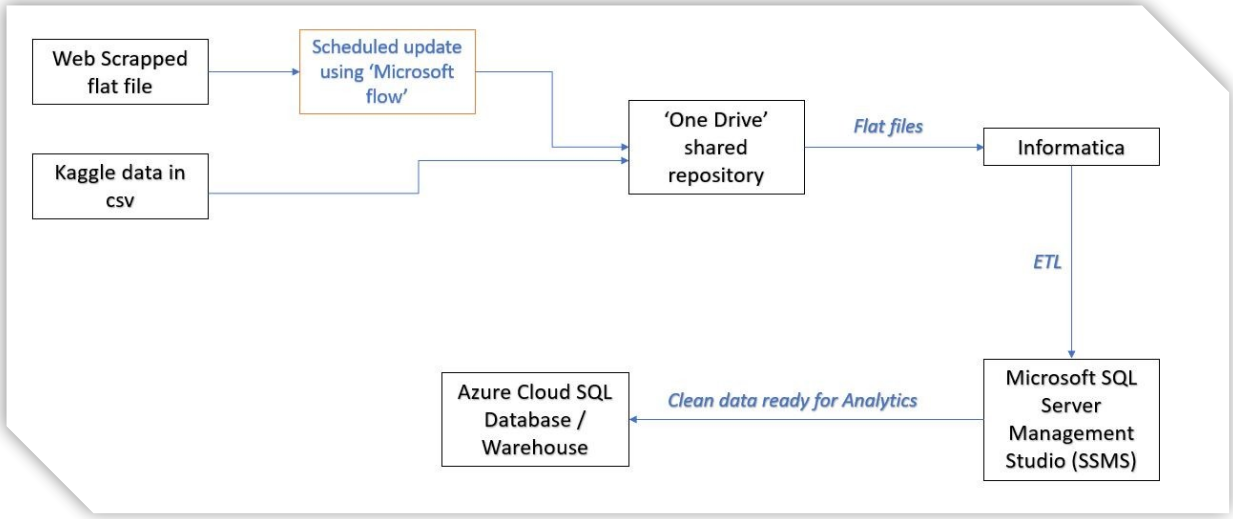

Figure 1: Data Flow Diagram

The flat files will then be loaded into 'Informatica Cloud' by using the 'Informatica Power Designer Tool.' SQL tasks can be executed in a sequence in Informatica on the respective Source tables. For example, the 'Sector' and 'Sub-industry' information from Wikipedia can be inserted as columns on the S&P 500 data from suredividend.com. Since suredividend.com data gets refreshed monthly, this update can be scheduled monthly using Informatica Cloud's 'Synchronized tasks'.

'Send email' task on Informatica Cloud can be created every time the data refresh is done so that the data analysts can keep track of the frequency of updates.

After 'Source' and 'Target' tables are configured and identified in the Informatica workspace, ETL will be performed on the 'Target' data, and then it would be pushed into Microsoft SSMS.

In SSMS a T-SQL procedure will be created to update the table dimensions. Fact table also created and updated into SSMS.

Server connection at the SSMS end will be set up with Azure SQL database, and the cleaned tables will be pushed into Azure's Big data Management system.

Data Cleaning:

1) The S&P 500 dataset with company details had 505 companies and the stock price dataset from Kaggle which contained close price of each stock for five years had 505 companies.

But in the stock price dataset from Kaggle, there were some companies that had insufficient amount of data. While there were 470 companies out of the 505 that had 730 observations for three years, 35 of the 505 companies had less than 730 observations.

A list of these companies is :

Companies with incomplete data : 35 ['BMY', 'DHR', 'ES', 'ICE', 'ORCL', 'O', 'IQV', 'COTY', 'FOXA', 'FOX', 'NWSA', 'NWS', 'ALLE', 'GOOG', 'NAVI', 'INFO', 'SYF', 'CFG', 'QRVO', 'WRK', 'KHC',

'PYPL', 'HPE', 'HPQ', 'CSRA', 'WLTW', 'UA', 'FTV', 'EVHC', 'HLT', 'DXC', 'BHGE', 'BHF', 'DWDP', 'APTV']

These companies had to be removed from our analysis and both datasets.

2) In the stock price, dataset two companies were named differently when compared to the S&P 500 company dataset. And this was the reason null exception was getting created every time a dataframe of close prices and company name was made. To remove this errors, the company, 'BRK.B' was renamed as 'BRKB,' and the company 'BF.B' was renamed as 'BFB.'

Architecture diagram and data storage

After ingesting data into Azure SQL Database into two tables, an Azure Jupyter notebook containing python code will be run by keeping the Azure SQL data as the source. The 'pyodbc' library will be used to connect to the Azure SQL server.

The clustering algorithm will be written using Python libraries and the clustering algorithm used will be K-Means. The cointegration level of each cluster will be computed by using the Augmented Dickey-Fuller test.

The results of the same would be inserted into a result table which will be exported to a Business Intelligence tool like 'Microsoft Power BI.'

On the Business Intelligence tool, dashboards will be created and published onto 'SharePoint.' The advantage of using 'SharePoint' as the front end tool for viewing the dashboards is that it allows for role-based administration and a security layer for authorized users.

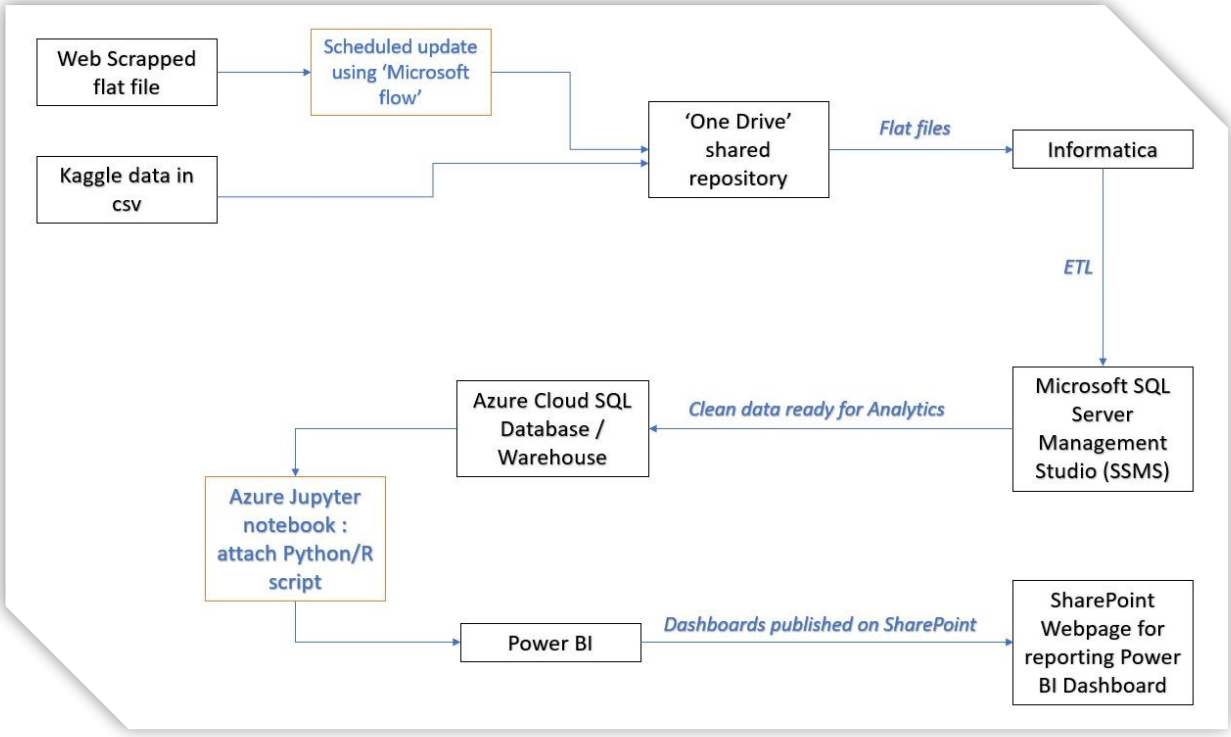

Figure 1: Architecture diagram of solution

The data will be stored in the Azure SQL database in the form of two tables. One table will have all the stock prices of all the S&P 500 companies for the past 5 years, and the second table will have the company information of S&P500 companies which includes information like the company sector, the sub-industry, the PE Ratio, etc.

Table Characteristics:

Table 1 :

Symbo	GICS Sector	GICS Sub Industry	Location	Name	Price	Dividend Yield (Forwar	PE Ratio (Forward	Price to Book Valu	Market Cap (millions	Beta (3Y)	Return on Equity (TTN
A	Health Care Equipment	Health Care Equipment	Santa Clara, California	Agilent Technologies	$ 64	1.0%	23.8	4.4	$ 20,323	1.49	5%
AAL	Industrials	Airlines	Fort Worth, Texas	American Airlines Group	$ 44	0.9%	8.4	#N/A	$ 20,688	1.32	67%
AAP	Consumer Discretionary	Automotive Retail	Roanoke, Virginia	Advance Auto Parts	$ 130	0.2%	19.2	2.7	$ 9,622	0.95	#N/A
AAPL	Information Technology	Technology Hardware, Storag	Cupertino, California	Apple	$ 193	1.5%	16.8	7.5	$ 9,50,145	1.17	40%
ABBV	Health Care	Pharmaceuticals	North Chicago, Illinois	AbbVie	$ 99	3.9%	12.7	44.2	$ 1,56,942	1.63	121%
ABC	Health Care	Health Care Distributors	Chesterbrook, Pennsylvania	AmerisourceBergen	$ 85	1.8%	13.1	5.8	$ 18,631	1.18	32%
ABMD	Health Care	Health Care Equipment	Danvers, Massachusetts	Abiomed	$ 406	#N/A	113.3	26.2	$ 18,067	0.01	19%
ABT	Health Care	Health Care Equipment	North Chicago, Illinois	Abbott Laboratories	$ 63	1.8%	22.0	3.5	$ 1,10,486	1.69	2%

Figure 2: Company Table

Field name	**Field Type**	**Constraint**
Symbol	Varchar	PK
GICS Sector	Varchar	Not null
GICS Sub Industry	Varchar	Not null
Location	Varchar	
Name	Varchar	
Price	float	
Dividend Yield	float	
PE Ratio	float	Not null
Market Cap	float	Not null
Beta (3Y)	float	
Return on Equity	float	Not null

Table 2 :

Name	Date	open	high	low	close	volume
AAL	08-02-2013	15.07	15.12	14.63	14.75	8407500
AAL	11-02-2013	14.89	15.01	14.26	14.46	8882000
AAL	12-02-2013	14.45	14.51	14.1	14.27	8126000
AAL	13-02-2013	14.3	14.94	14.25	14.66	10259500
AAL	14-02-2013	14.94	14.96	13.16	13.99	31879900
AAL	15-02-2013	13.93	14.61	13.93	14.5	15628000
AAL	19-02-2013	14.33	14.56	14.08	14.26	11354400
AAL	20-02-2013	14.17	14.26	13.15	13.33	14725200

Figure 3: Stock Price Table

Field name	**Field Type**	**Constraint**
Name	Varchar	PK, FK(CompanyTable)

Date	Datetime	
Open	Float	
High	Float	
Low	Float	
Close	Float	Not Null
Volume	Float	

Data Science and Analytics techniques to derive results :

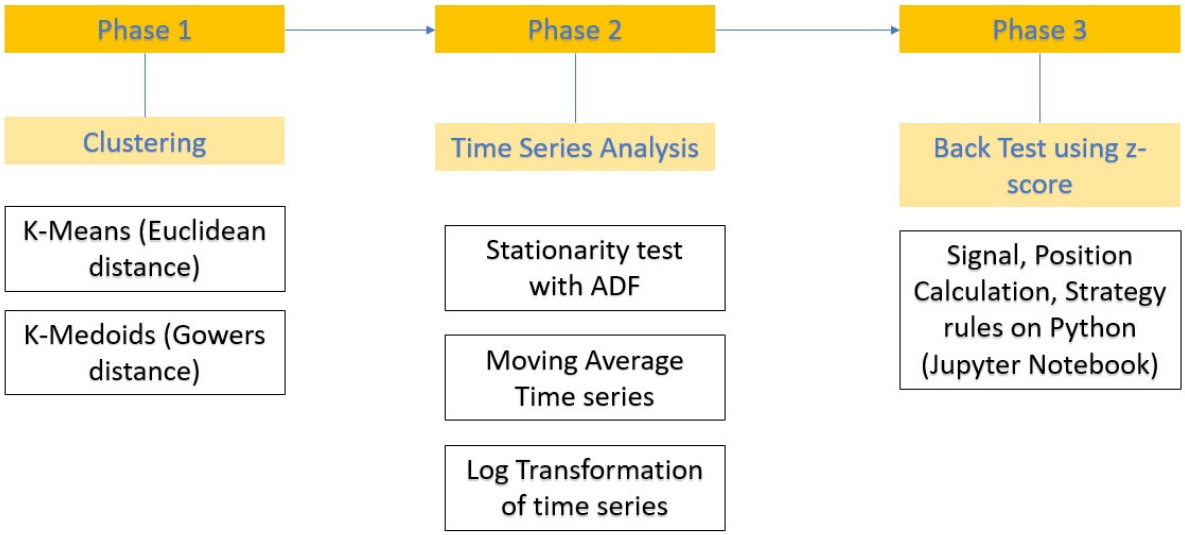

Figure 1 : Data Science techniques distributed in phases

Phase 1: Clustering

I have done clustering using two algorithms, K-means and K-medoid.

My dataset used for clustering contained two categorical variables (Sector name, Sub-Industry name) and three numerical variables (Market Capitalization, PE Ratio, Return on Equity).

Since K-means takes only Euclidean distances in its algorithm, I could not use categorical variables as features in my clustering. So the clustering was done with three numerical variables. It was found that the optimal number of clusters is 5. This was identified by using the 'Elbow method' of

plotting the optimal number of clusters.

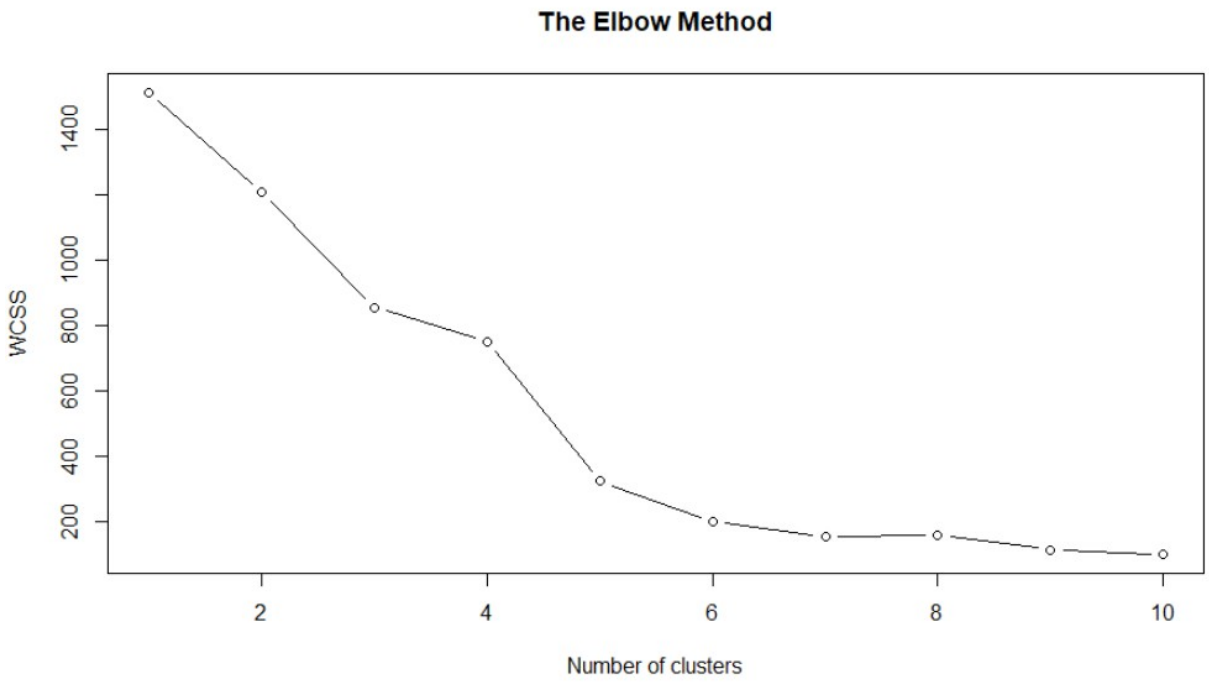

Figure 2: Optimal number of clusters using K-means is 5 for our dataset

Clustering of the dataset using K-means was done with Euclidean distances as the distance measure, and five clusters were created of the 500 S&P companies.

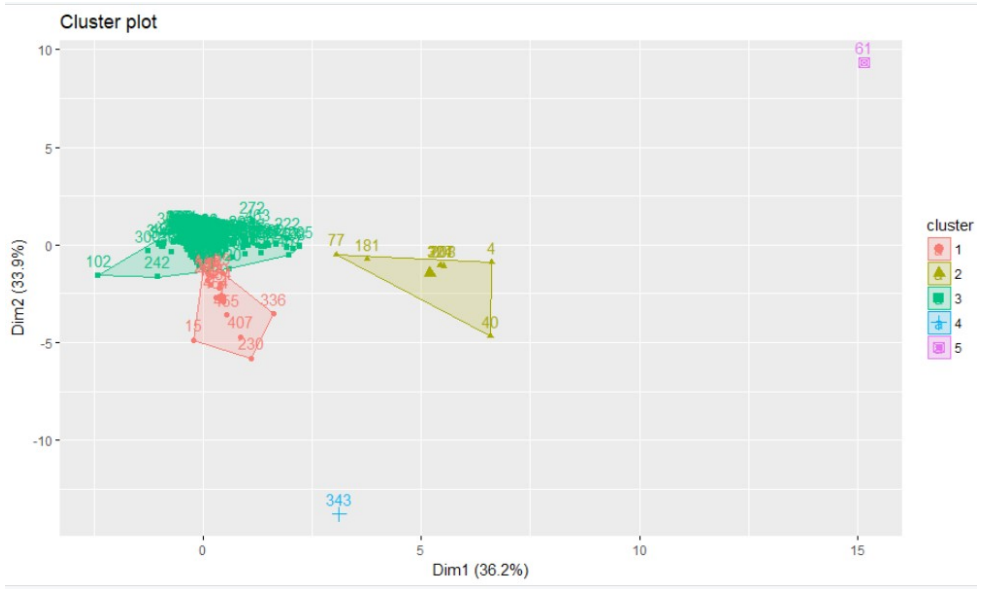

Figure 3: Five clusters of S&P 500 companies

K-Medoid Clustering :

K-Means – Value of k is first decided, and then the mean is calculated. Data points are plotted around it, and this is continued for several iterations till final clusters are made.

K-Medoid clustering is a modified form of K-Means clustering, and it is more robust to noise than K-Means clustering.

K-Medoid is less sensitive to noise, because it uses medoids as cluster centers instead of means (used in k-means). This makes it more robust to outliers.

To calculate the dissimilarity matrix, 'Gower's Distance' is being used because then, our categorical variables can also be employed in the clustering algorithm. To find the trends behind why a cluster has higher cointegration than others, it was essential to also include the industry sector and sub industry in the clustering.

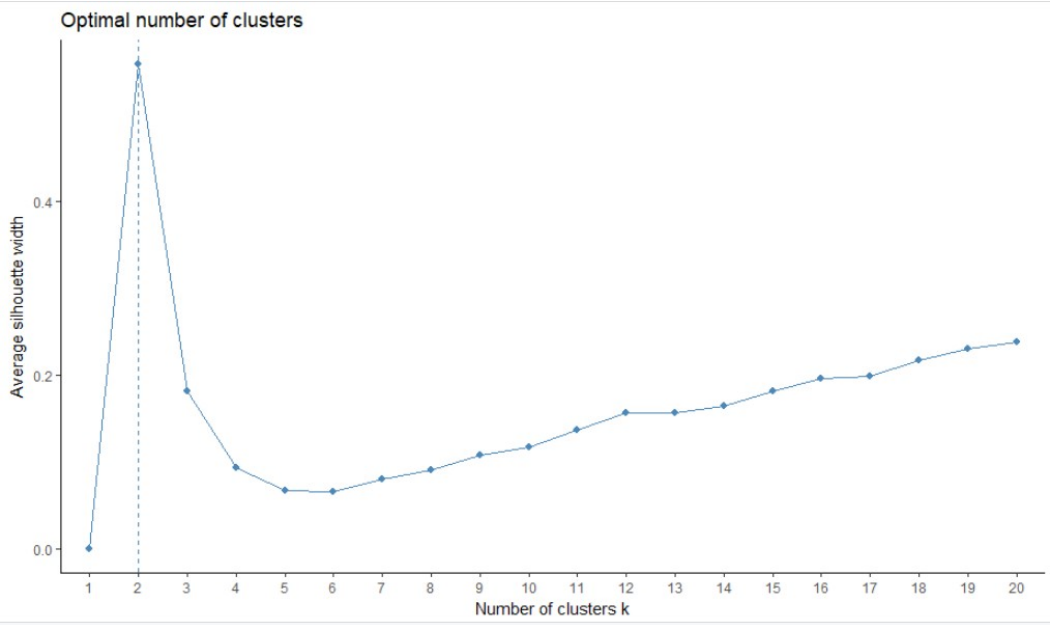

Figure 4: Optimal number of clusters are found in K-Medoids method using Silhouette method

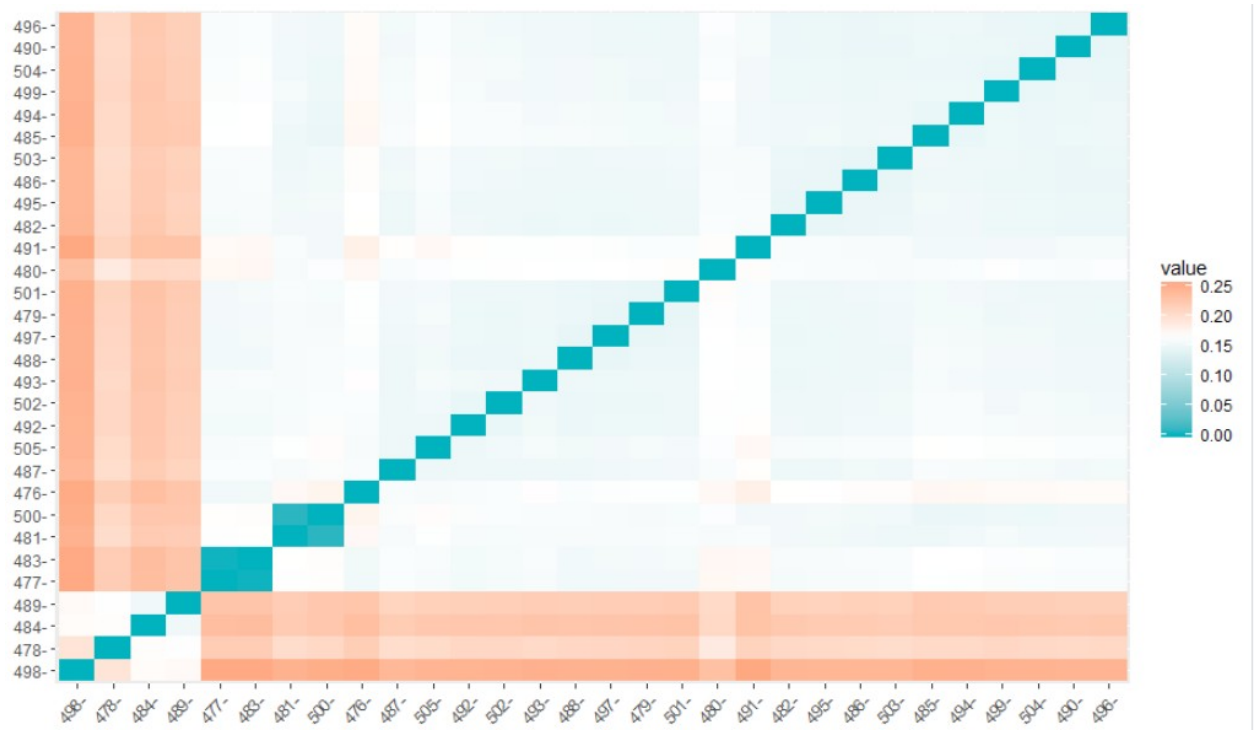

Figure 5: Dissimilarity matrix showing the similarities (blue) between 25 companies (subset of the matrix taken for ease of readability)

The clusters were made using PAM (Partitioning around Medoids), and the result was saved for Phase 2.

Phase 2:

Phase two involves the analysis of 3 years of stock market data for 500 companies. This dataset has around 3 lac rows. From the clusters received from phase 1 consisting of company names, all possible pairs were made in Python. These pairs for each cluster were stored in a list.

Now, the analysis was done pair by pair.

Then time series was created for both the companies as shown below (This series was created using the close price of the stock for each trading day for 3 years)

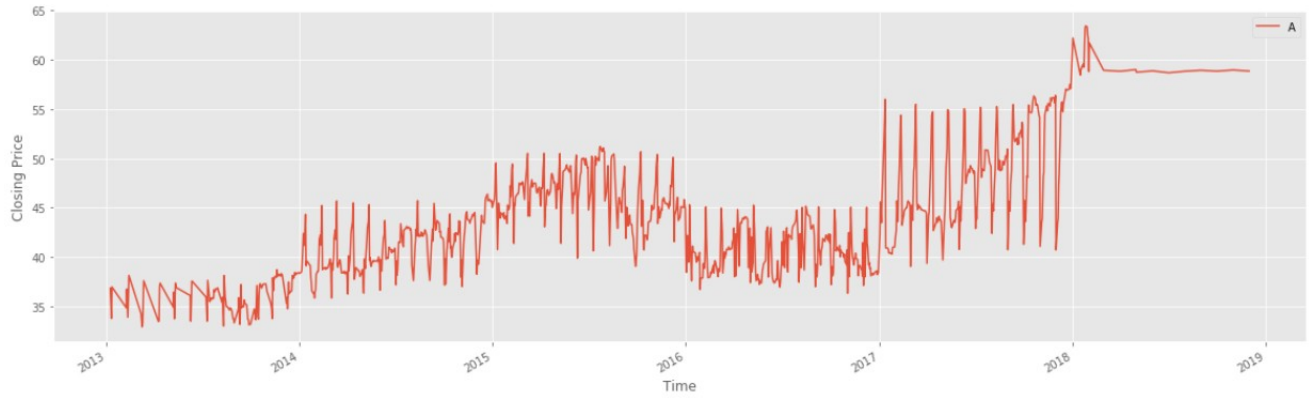

Figure 6 : Closing price time series for Company – ABT

Then the log and moving average of the time series is found and subtracted for removing noise and building a more reliable stationarity test.

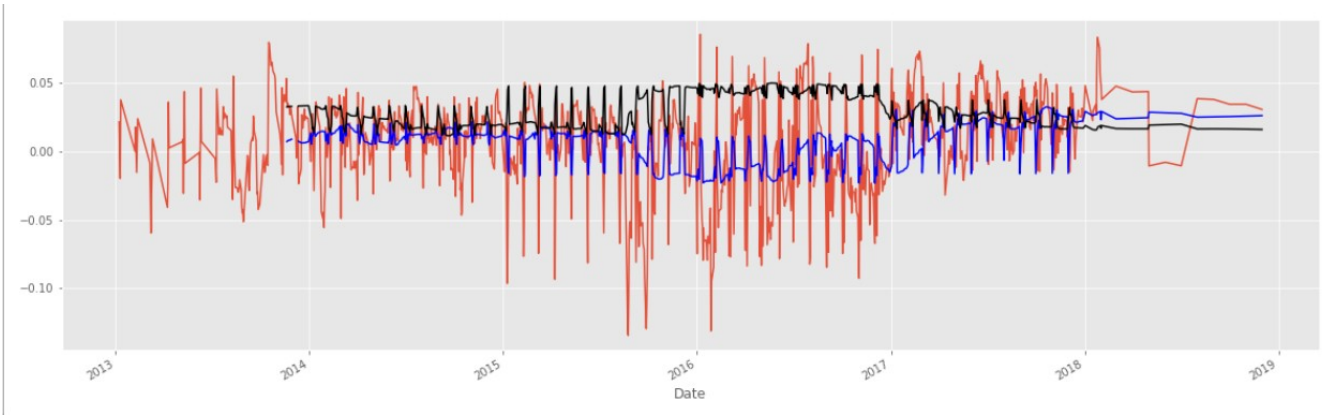

Figure 7: Log minus Moving average of Company ABT (red), Mean of the series (blue), Standard Deviation(black)

After this is done for both the companies in the pair, the linear combination of both the series is found such that this linear combination is stationary. Stationarity is tested by using the Augmented Dickey Fuller test. And the linearly combined time series is found by regression analysis.

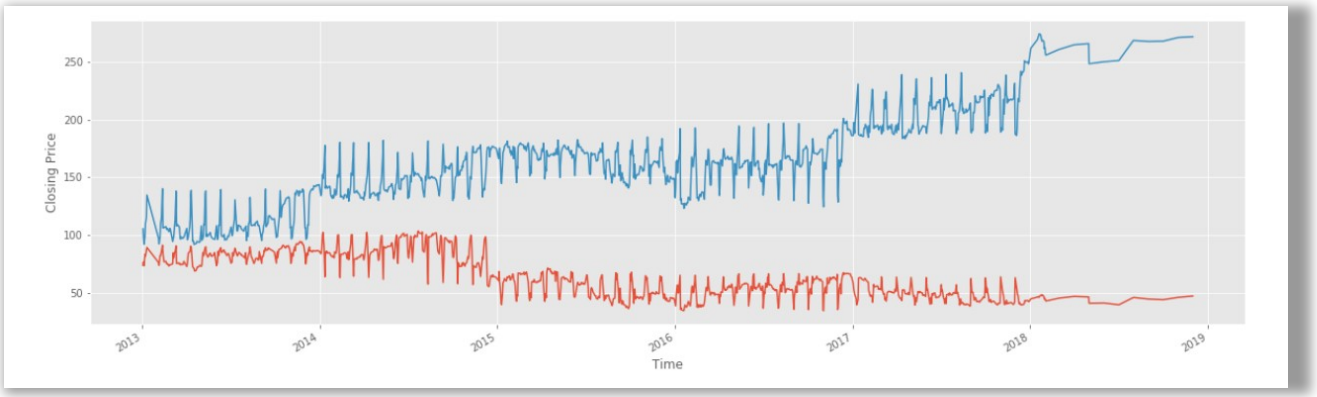

Figure 8.1: Cointegrated pair ABT and FDX – Graph of their closing prices

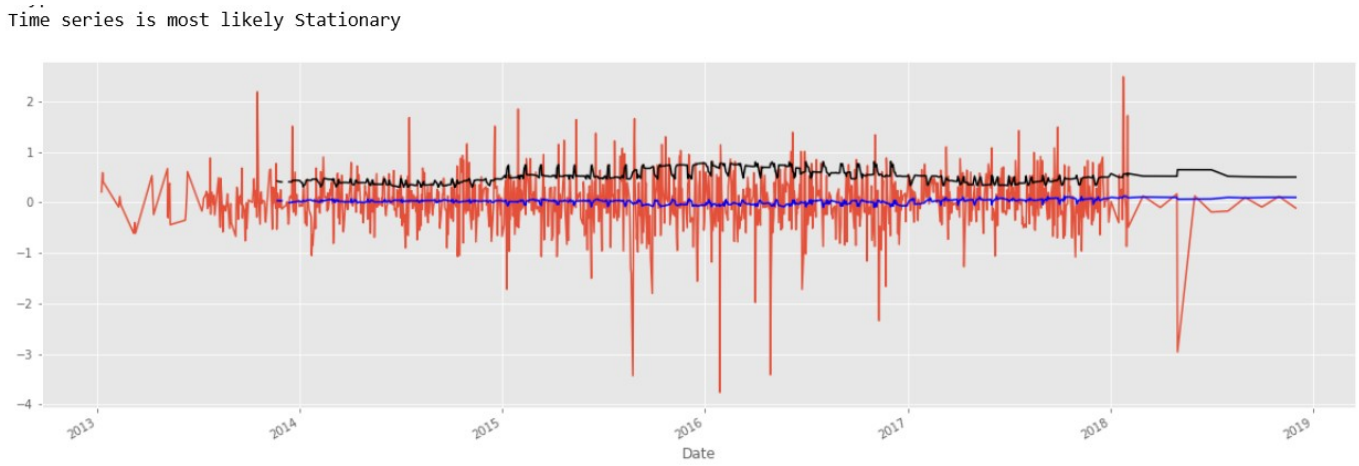

Figure 8 : Graph showing stationarity of linearly combined series for companies ABT and FDX

Once we find all the cointegrated pairs in each cluster, they will be sent to phase 3 where we calculate the cumulative return for each pair with 2 years of stock data.

Pair Based trading and the strategy adopted:

Hedged Position

A short sale makes money when the security sold, loses value.

And a long purchase makes money if the security purchased, gains value. In the case of two securities, we call it a hedged position when we go long on one security and go short on the other.

If both securities go up together or both go down together, we make no money. But we make money if both of them are at some point reverting to the mean.

Testing for cointegration for pair based trading:

For two time series (x1, x2), if they are of I(1), and some linear combination of them (Y) is I(0), then we can say that this set of time series is Cointegrated.

[where linear combination means Y=A*x1 + B*x2]

This means that there is some solid relation between x1 and x2 because its linear combination is stationary.

It shows a deep economic link between two stocks. This linear combination (Y) will vary around a mean and at all points in time, the combination between them is related to the same probability distribution.

Testing for Cointegration using Regression

Y=A*x1 + B*x2

We are trying to find the value of A and B such that Y is I(0) -> or stationary.

A standard way to do this in time series is to use linear regression to estimate B' in the following model :

X2 = A' + B'*x1 + c

The idea is that if these two are cointegrated, then we can remove x2's dependency on x1, leaving behind stationary noise.

X2 - B'*x1 = A' + c (where Y = A'+c)

After phase 2 where 35,108 pairs were found to be cointegrated from a total of 1,76,464 pairs from all clusters combined, phase 3 had to be initiated where all these 35 k pairs would be back tested on stock price data from 2016 – 2018.

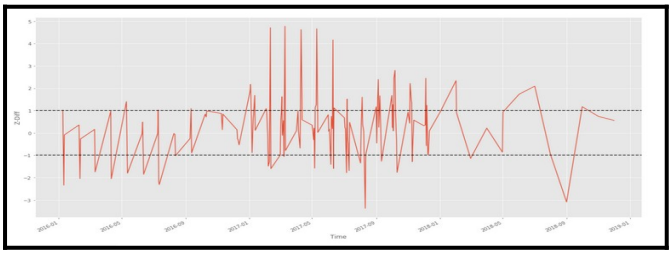

Fig 9: Z-diff score of combined series

- Whenever the z_diff was less than -1, the signal was created to go long in the market, and whenever the z_diff was more than +1, the signal was created to go short in the market. Forward return was calculated, and the cumulative return of this strategy was found by acting on the signals.

  Final Results:

  Found 609 profitable cointegrated pairs by Med-10.

[10,756 pairs in total when seen from all clusters]

- The most profitable cointegrated pairs belonged to following industries (clusters):

1. Healthcare (53% are profitable)
2. Information Technology (58% are profitable)
3. Real Estate (58% are profitable)
4. Energy (53% are profitable)

- The best clustering algorithm from pair based trading perspective is K-Medoid clustering using Gowers Distance giving more weightage to sector and sub sector instead of prices.
- More positive cointegrated pairs were found when the test for cointegration was changed from a p-value threshold of 0.05 to 0.01
- P-Value is inversely proportional to the number of cointegrated pairs found.
- The pair with maximum cumulative return is (NWL , SCG) with a return of 2.51

References:

(n.d.). Retrieved from https://www.quantopian.com/posts/tag/pairs-trading/newest

(n.d.). Retrieved from https://machinelearningmastery.com/moving-average-smoothing-for-time-series-forecasting-python/

(n.d.). Retrieved from https://www.r-bloggers.com/the-ultimate-guide-to-partitioning-clustering/

(n.d.). Retrieved from https://github.com/kartikeyathakur/pairtrading

(n.d.). Retrieved from https://cran.r-project.org/web/packages/kmed/vignettes/kmedoid.html

Udemy Course– Time Series Analysis & Forecasting. (n.d.). Retrieved from Udemy.