



---

# Visualise & Predict Financial Data

## Application Studio A:

**Meghana Vanapamula (13372596)**

**Uha Dommaraju (13453840)**

## Fundamental Studio A:

**Harilaos Papas (13534767)**

**Anubhav Jetley (13890278)**

**Raj Kulkarni (13890292)**

## Table of Contents

### 1. Introduction

About project.....	3
Scope.....	3
SWOT analysis.....	4
Team Dynamic .....	5
Team Experience .....	6

### 2. Clustering Stocks and PCA

Clustering stock with Python.....	7
Libraries .....	8
Visualise and Scale the Data .....	9
Principal Component Analysis.....	13
Build and run the model.....	17
Model Evaluation.....	17
Chart the clusters.....	18

### 3. Data Visualisation Models

Line Plot .....	20
Candlestick.....	20
Simple moving average.....	21
Exponential moving average.....	22
Moving Average Convergence Divergence (MACD).....	24
Relative strength index (RSI).....	25

### 4. Dash Plotly

Frontend.....	26
Backend.....	27
Dash Packages.....	28
App Callback.....	30

### 5. Development Progress Flowchart.....31

### 6. Enhancements and future endeavours

Steps to PCA.....	33
Improving dash and visualisations.....	38

### 7. Reflection.....40

### 8. References.....41

## **Introduction:**

### **About the project:**

The brief of this product was 'directed at providing a framework for analysis, visualisation and prediction of financial, stock market and related data'. Our products focus was on being able to provide users with an edge over other investors by providing them with the ability to make wiser trades. Due to the number of variables to consider when placing a trade, stock market data is often complex to analyse. With this in mind, through python we were able to visualise high dimensional data by compacting it into low dimensional data, whilst providing interactive tools and graphs to aid any level of investor.

### **Scope:**

This proposal is an investigation into the world of financial markets and how data science can be used to visualise the large volumes of stock market data in order to make better investment decisions. Data visualisation has been displayed through several different mediums in a clear and efficient way in order to communicate large sums of dense data available to the public. All mediums have integrated graphical user capabilities (GUI's) that can be manipulated by the user in order to alter the information displayed according to their individual needs. For example, the ability to rotate and view the graphs differently and the drop-down boxes for; dates, ticker, open, high, close, volume are various functions the user is able to manipulate.

The stock market provides traders with the opportunity to generate wealth and profits. How a stock market is behaving and what profits its producing, provide a benchmark for a country's industrial health and overall wealth. These benchmarks are often compared to the industries history to provide investors with an expected outlook of how the stocks and market generally will perform. This information is important as financial markets provide businesses and governmental entities access to capital whilst also providing employment to those in the industry.

This notion has been fundamental in this project's development with the main objective of the project undertaken is to explore and understand various financial conditions. The project thrives on its ability to compare stocks, visualise them in alternative ways that have not been done before in order to provide the user with an alternative medium to visualise and establish trends. Our alternative medium to visualise this information is what distinguishes us from other existing platforms. Currently, there are platforms such as Commsec, Trading View and the ASX which display generic historical figures, but we are unaware of any established platforms that intend to do what we have achieved.

Initially we wanted to apply the concepts of prediction and analysis of data to derive meaningful models which may assist to predict the future trends. However, we realised that due to an array of factors, but mainly the volatility of the market and the impact of unexpected events or external

influences such as Covid, it makes it difficult to reliably predict stocks. As a result, our project will not include any recommendations or information regarding to predictions on stock prices.

### SWOT Analysis:

<p><b><u>STRENGTHS</u></b></p> <ul style="list-style-type: none"> <li>• 3D Virtual environments have become an increasingly normalized medium for communicating spatial information</li> <li>• 3D Visualization model acts as a 'walk through' of the financial numbers</li> <li>• Large amounts of financial data available for analysis</li> <li>• Unique and fresh data analysis methodology within finance</li> <li>• Easier method of understanding financial trends for users inexperienced in financial markets</li> </ul>	<p><b><u>WEAKNESSES</u></b></p> <ul style="list-style-type: none"> <li>• Users may not be able to understand the visualisations and data displayed.</li> <li>• Not every single stock available in all markets will be able to be displayed</li> <li>• Some trends may be due to an unknown factor and mislead the user</li> <li>• Cannot detect social and economic events that can influence financial data</li> </ul>
<p><b><u>OPPORTUNITIES</u></b></p> <ul style="list-style-type: none"> <li>• An easier way to understand trends in financial data</li> <li>• Greater amounts of investors due to greater understanding of numbers</li> <li>• Greater potential for ROI for investors</li> <li>• A better understanding of whether certain factors influence financial trends and numbers</li> </ul>	<p><b><u>THREATS</u></b></p> <ul style="list-style-type: none"> <li>• Established platforms Sponsored by CHESS</li> <li>• Major capital losses due to overinvestment and confidence in financial predictions</li> </ul>

### Team Dynamic:

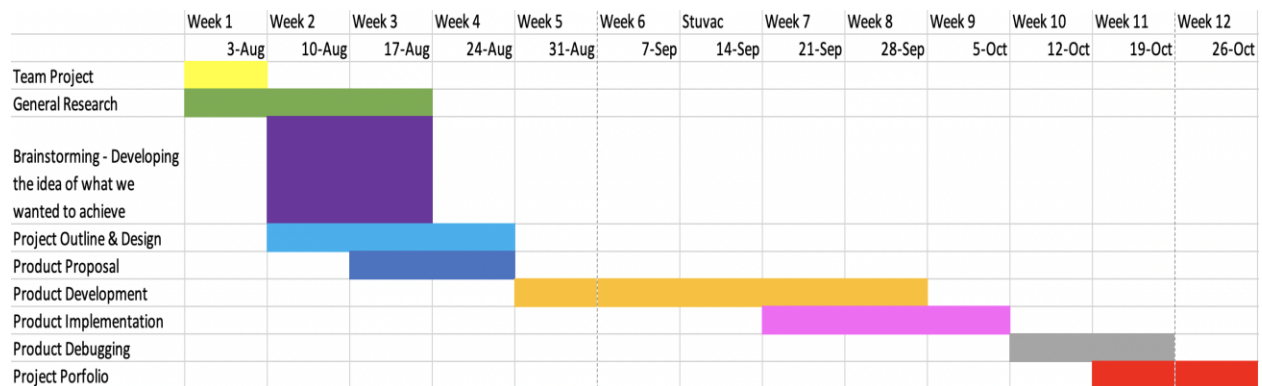
The team comprised of five members of varying knowledge and relevant experience for the project. None of the team had any experience in 3D visualisation prior to this project and thus would be a learning experience for everyone.

Due to two of the members -Meghana and Uha - having more experience being Applications students and also more exposure to project development and programming languages, naturally they took a more leadership role within the group. This allowed for faster dispersion of information

related to the data engineering tools and techniques necessary to complete the project prior to submission deadlines.

Because we had direct guidance and a more efficient channel of information being shared and communicated we were able to identify what we would like to identify what we would like to achieve from this project. As a result, more time was being directed towards research and methods in achieving what we set out to do and how we could develop these further.

Our meetings were organised from the beginning of the semester to take place every Monday either prior or following our meeting with Don, as per the subject requirements. If members were unable to attend or events such as the plenary affected our usual meeting time, contingency plans were in place to have another meeting during a mutually agreed general time. Our meetings would generally involve up-dating other members and Don on our progress or information that we found, seeking advice for issues we had encountered and agreeing on tasks to complete prior to the next meeting in addition to a general update on the project 'direction'. If there were any other significant developments, issues identified or questions that needed to be asked, this would be communicated to the group and an impromptu meeting would be organised.



*Gantt Chart of Development Process*

**Team experience:**

<u><b>Meghana Vanapamula</b></u> <i>(team co-leader)</i>	<ul style="list-style-type: none"> <li>• Applications studio A</li> <li>• Worked on a similar projects - Ericsson's weather prediction and Analysis and Twitter Mood Detection Analysis</li> <li>• Technical skills - Python, IDE Java, MATLAB, C Programming, HTML, XML, CSS, R Studios, Knime, MySQL</li> </ul>
<u><b>Uha Dommaraju</b></u> <i>(team co-leader)</i>	<ul style="list-style-type: none"> <li>• Application Studio A</li> <li>• Worked on a similar projects - Ericsson's weather prediction and Analysis and Twitter Mood detection Analysis</li> <li>• Technical skills: Python, C Programming, MATLAB, R Studios, Database programming, Knime, HTML</li> <li>• Data analyst intern</li> </ul>
<u><b>Raj Kulkarni</b></u>	<ul style="list-style-type: none"> <li>• Fundamentals Studio Student.</li> <li>• Technical skills: Python, C++</li> </ul>
<u><b>Anubhav Jetley</b></u>	<ul style="list-style-type: none"> <li>• Fundamentals Studio Student.</li> <li>• Technical skills: Python, C++</li> <li>• Data Analytics – Machine Learning prediction models</li> </ul>
<u><b>Harris Papas</b></u>	<ul style="list-style-type: none"> <li>• Fundamentals Studio A Student.</li> <li>• Brief/Limited experience in Python</li> <li>• Limited experience in MATLAB (GUI's)</li> <li>• Above average understanding of financial data, stock market and Technical Analysis</li> <li>• Basic Data Analytics</li> </ul>

## **Development Process (PCA/CLUSTERING):**

### Clustering stock with Python

Often, we are given data sets that contain ungrouped or uncategorized data; for this project we wanted to find underlying structure that might not be apparent to users. When grouping objects into similar sets, it is known as 'clustering'. The idea is to use dimensionality reduction for data that is similar in nature would be categorised into the same group of stocks, by using different groups based on special financial metric measures of similarity to determine whether an object is similar to another. While clustering as a concept is easy to grasp, there were different implementations in order to achieve this. Some applications use different algorithms or use various measures to define similarity. Clustering is used extensively in the financial industry to do a wide range of tasks, spanning from portfolio construction, outlier detection, or stock selection.

Historical price data can be used by investors and analysts to back-test pricing models or investment strategies, to mine data for patterns that have occurred in the past, or to detect technical indicators for day traders, among other uses. However, to do this, they must choose stocks that are not correlated with each other or in some cases to find stocks that are similar in nature to give them adequate exposure to a particular segment of the market.

There are many clustering algorithms we can use, for this project, we intended to use K-means which is an unsupervised machine learning algorithm which is used on unlabeled data (data without defined categories or groups). The goal of K-Means algorithm is to divide  $n$  data points into  $k$  partitions, where the sum of the distances is minimized.


#### **Broken into steps, the algorithm is executed in the following order:**

1. Randomly select K-Centers, to be the cluster centers.
2. Calculate the distance of each point to a cluster and assign a cluster label where the Euclidean distance is smallest.
3. Recompute the centroids by taking the mean of all the data points assigned to that cluster.
4. Repeat these steps until one of the following two conditions are met:
  - a. *The Sum of the Distances is minimized.*
  - b. *The maximum number of iterations has been reached.*

The algorithm will converge to a result, but this result is not necessarily guaranteed to be the optimal result. K-Means is used heavily for exploratory data analysis and for its ability to take unstructured data and create structure from it. This is powerful when it comes to finding patterns in data that aren't necessarily apparent to the naked eye, especially at higher dimensions.

## Libraries

Luckily, performing cluster analysis in Python is easy due to built-in libraries that make running the algorithms efficient and fast. Pandas was used to take data collected from our data set of 3 tickers selected from Yahoo Finance (GOOG, FB, AAPL) and store it in a manner that makes manipulation manageable. To perform our cluster analysis, we used sklearn which has built-in functions to create instances of our clustering algorithms. Along with this, a model evaluation is performed with built-in metrics that will take our results and evaluate them. For some visualization tasks, the team used matplotlib and yellowbrick to create cluster graphs and silhouette graphs, respectively.

```
In [40]:  #IMPORTING DATA
#import pandas
import pandas as pd
#Load csv
#APPL, FB, GOOG
stock_df = pd.read_csv("/Users/Uha Dommaraju/Desktop/yahoo 3 ticker/3 ticker

# we have a character that will cause issues in our request so we have to re
stock_df['Ticker'] = stock_df['Ticker'].str.replace('^', '')

# display the number of rows.
display(stock_df.shape)
```

---

(3585, 7)

With over 3585, the stocks are broken into this data frame into chunks of 100 so that we can request multiple tickers at once. After removing some of these extreme values, a statistical summary of our dataset is most appropriate, a summary data frame and then a standard deviation matrix is calculated using the mean.

```

In [50]: ► #STATISTICAL SUMMARY
desc_df = indicators_df.describe()

# add the standard deviation metric
desc_df.loc['+3_std'] = desc_df.loc['mean'] + (desc_df.loc['std'] * 3)
desc_df.loc['-3_std'] = desc_df.loc['mean'] - (desc_df.loc['std'] * 3)

# display it
desc_df

```

Out[50]:

	Open	High	Low	Close	Volume
count	3585.000000	3585.000000	3585.000000	3585.000000	3.585000e+03
mean	411.944613	415.894110	408.092262	412.210931	5.175134e+07
std	458.537432	462.965269	454.345381	458.911746	6.806977e+07
min	22.500000	22.917500	22.367500	22.584999	3.475000e+05
25%	52.357498	52.660000	51.799999	52.252499	1.905900e+06
50%	166.130005	168.339996	164.210007	166.320007	1.817240e+07
75%	803.299988	806.205017	796.320007	801.489990	9.171760e+07
max	1586.989990	1586.989990	1554.280029	1568.489990	5.334788e+08
+3_std	1787.556909	1804.789918	1771.128405	1788.946170	2.559607e+08
-3_std	-963.667683	-973.001698	-954.943882	-964.524307	-1.524580e+08

### Visualise and Scale the Data

The data being in a good range, a few values which are outside three standard deviations, but it's expected given the variety of stocks we have. Since statistical summary is successful enough, the team was able to move onto the plotting process. In this example, a 3D scatter plot using *matplotlib* is used to view all the data points and create our visualisation.

```

In [52]: !pip install matplotlib
#3D SCATTER PLOT
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# define a figure and a 3D axis
fig = plt.figure()
ax = Axes3D(fig)

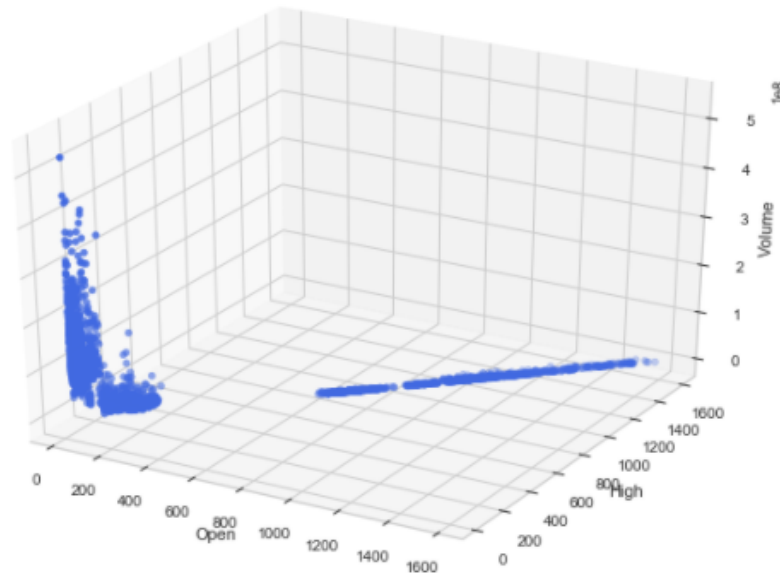
# define the x, y, & z of our scatter plot, this will just be the data from our data frame.
x = list(indicators_df.iloc[:,0])
y = list(indicators_df.iloc[:,1])
z = list(indicators_df.iloc[:,4])

# define the axis labels
column_names = indicators_df.columns
ax.set_xlabel(column_names[0])
ax.set_ylabel(column_names[1])
ax.set_zlabel(column_names[4])

# define the markers, and the color
ax.scatter(x, y, z, c='royalBlue', marker='o')

plt.show()

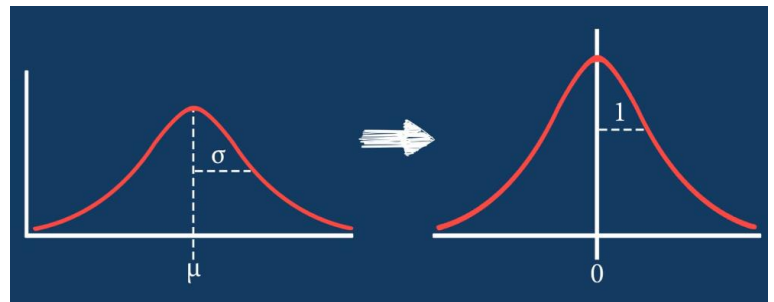
```



As mentioned, purposely limited the number of attributes in our data set. The reason for this is it's hard to visualize data that exceeds three dimensions. However, looking at this data it does seem to be clumped together with no definite spherical shape, this can cause issues when clustering because naturally, the algorithm looks for the precise structure to cluster the data around. There are steps we can take to help mitigate this issue, but this also a laity of data, there might not be clear clusters and in some cases, clustering might not be the right approach.

However, by normalizing our data, to help handle outliers and things of that nature. Normalizing our data as the model tends to perform better with normalized data vice the alternative. There are a few options to normalise the data:

Standard Scaler	$\frac{x_i - \text{mean}(\mathbf{x})}{\text{stdev}(\mathbf{x})}$
MinMax Scaler	$\frac{x_i - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}$
Robust Scaler	$\frac{x_i - Q_1(\mathbf{x})}{Q_3(\mathbf{x}) - Q_1(\mathbf{x})}$



1. **Standard Scaler:** Here we subtract the mean from each data point and divide it by the standard deviation. This method is sensitive to outliers; however, like computing the mean means taking the average of ALL data points, including the outliers.
2. **Min Max Scaler:** Here we scale the data so that it fits in a range between 0 and 1. Mathematically we take each data point, subtract the minimum from it, and then divide it by the difference of the maximum value and the minimum value. Again, this is sensitive to outliers as the maximum amount would be the outlier.
3. **Robust Scaler:** This method is a better choice if the data has outliers. With this method, we use the interquartile range instead of the minimum and maximum, which helps control for outliers.

In our example, because there might be a possibility for outliers, the team chose to use the Robust Scaling method.

```

In [30]: # ROBUST SCALING METHOD
import numpy as np
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler

# for demonstration purposes, I will be creating all three instances of the scalers.
min_max_scaler = MinMaxScaler()
std_scaler = StandardScaler()
robust_scaler = RobustScaler()

# scale the data
X_train_minmax = min_max_scaler.fit_transform(indicators_df)
X_train_standard = std_scaler.fit_transform(indicators_df)
X_train_robust = robust_scaler.fit_transform(indicators_df)

# create a new plot
fig = plt.figure()
ax = Axes3D(fig)

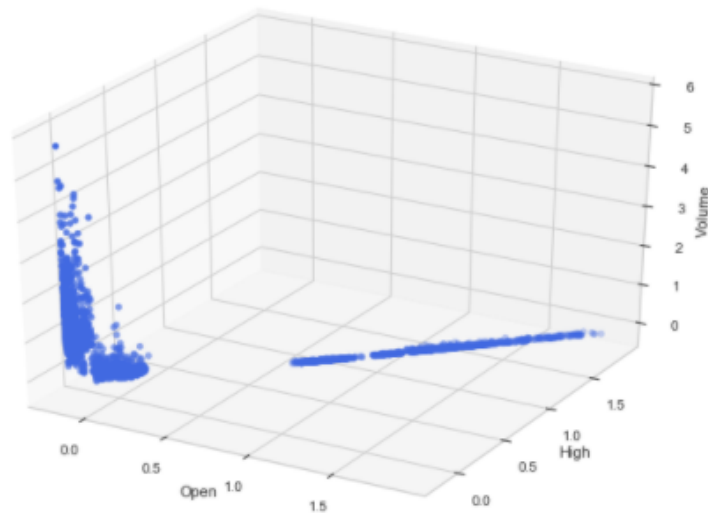
# take the scaled data in this example.
x = X_train_robust[:,0]
y = X_train_robust[:,1]
z = X_train_robust[:,4]

# define the axes labels
column_names = indicators_df.columns
ax.set_xlabel(column_names[0])
ax.set_ylabel(column_names[1])
ax.set_zlabel(column_names[4])

# create a new plot
ax.scatter(x, y, z, c='royalBlue')

plt.show()

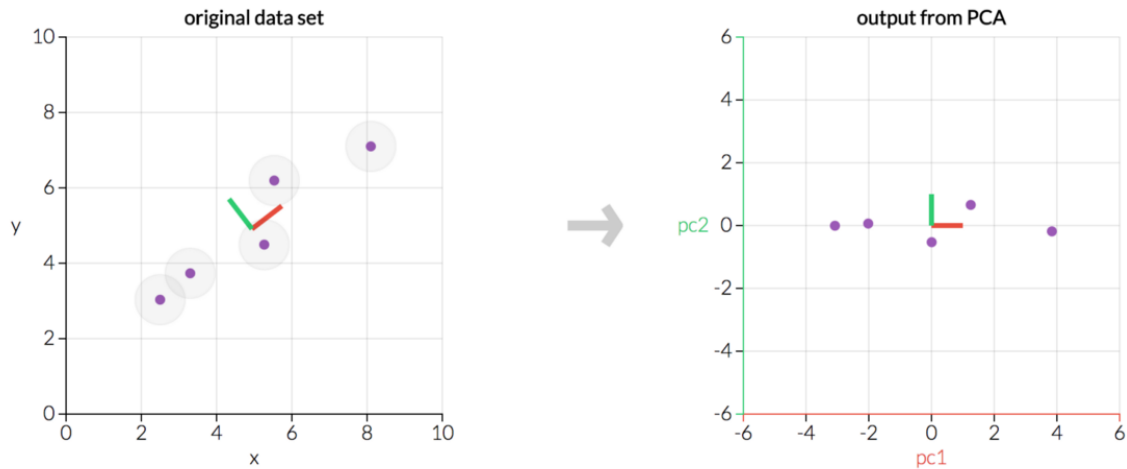
```



### Principal Component Analysis (PCA)

By scaling the data, we are allowing our algorithm to perform better with the data. Also with large data sets with multiple attributes, it will allow our model to run faster and have less redundant data.

To help conceptualise this concept:



→ ***Fundamental and Application students all enjoy the subject.***

Now imagine, I added more detail to the story to help describe the subject.

→ ***Fundamental and Application Students all enjoy the subject with a big smile***

For the most, many people would say that adding the extra detail didn't provide much more to the story. Leaving it out of the story, most people would understand the gist of the story. With PCA, trying to determine this additional info and removing it from the story hence why we remove these attributes from our data set.

The idea is that even though i might lose 3% of all information I still can describe 97% of the story and in most cases, that is more than enough to get the general theme of the story. We also gain the benefit of faster execution time when it comes to training our model. In a clustering algorithm like K-Means, while it is easy to understand and implement, it does suffer

from long training runs when there are too many attributes.

PCA attempts to remedy this problem by removing the attributes that don't contribute too much to the story and keep those attributes which contribute significantly to the story. Implementing PCA in *sklearn* is easy to do, but something we need to do as an initial step is to get an understanding of the number of components we need.

Visually this can be done by passing through our scaled data set into our PCA class object, and plotting the *explained\_variance\_ratio* which will tell us how much of the variance is explained at any given number of components

From the Chart above, we can see that we have 100% variance explained with only two

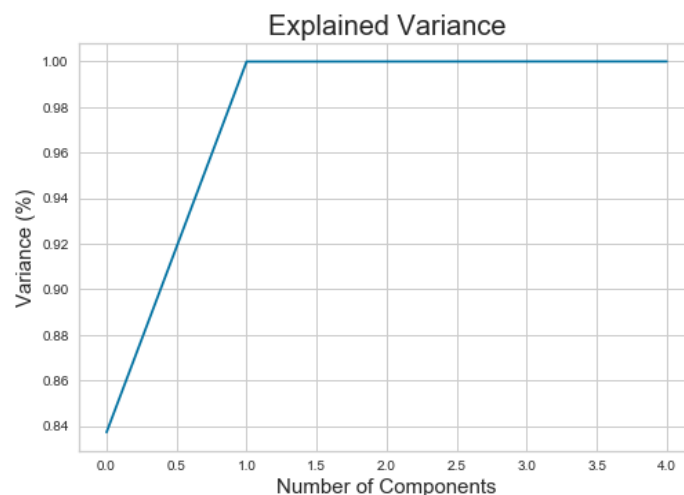
```
In [31]: #PRINCIPAL COMPONENT ANALYSIS
from sklearn.decomposition import PCA

# pass through the scaled data set into our PCA class object
pca = PCA().fit(X_train_robust)

# plot the Cumulative Summation of the Explained Variance
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))

# define the Labels & title
plt.xlabel('Number of Components', fontsize = 15)
plt.ylabel('Variance (%)', fontsize = 15)
plt.title('Explained Variance', fontsize = 20)

# show the plot
plt.show()
```



components. This means that if we were to implement a PCA, we would select our number of components to be 2. In most examples, we won't get 100% explained the variance, but the general rule is we chose the minimum number of components that demonstrates the highest amount of variance.

```

In [32]: #2 PCA
# create a PCA modified dataset
pca_dataset = PCA(n_components=2).fit(X_train_robust).transform(X_train_robust)

# store it in a new data frame
pca_dataset = pd.DataFrame(data = pca_dataset, columns = ['principal component 1', 'principal component 2'])

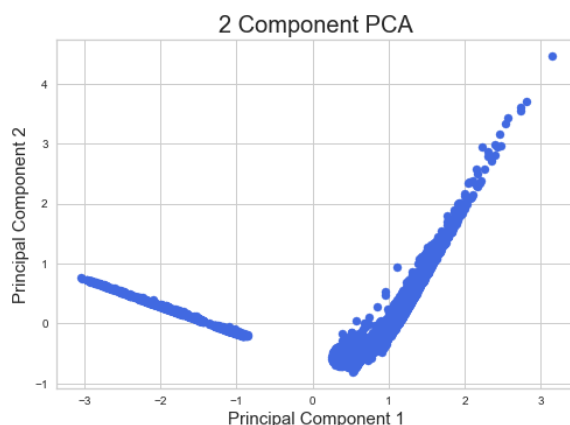
# define a figure
plt.figure()

# define the label and title
plt.xlabel('Principal Component 1', fontsize = 15)
plt.ylabel('Principal Component 2', fontsize = 15)
plt.title('2 Component PCA', fontsize = 20)

# plot the figure
plt.scatter(pca_dataset['principal component 1'], pca_dataset['principal component 2'], c='royalBlue', s = 50)

Out[32]: <matplotlib.collections.PathCollection at 0x2694e735e48>

```



By running PCA, we have reduced the number of dimensions in our data set from 3 to 2. This means if we graph our new data frame, we will only have two dimensions. Graphically our new data set would look like the above plot.

### Build and Run the Model

One of the biggest challenges with K-means is determining the optimum number of clusters, so we decided to explore some options we have, one type of analysis we can do is related to silhouette analysis. It can be used to study the separation distance between the resulting clusters. The silhouette plot displays a measure of how close each point in one cluster is to point in the neighboring clusters and this provides a way to assess parameters like the number of clusters visually. This measure has a range of (-1,1).

Silhouette coefficients (as these values are referred to as) near +1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster.

Creating an instance of our K-Means model and exploring how the silhouette score changes within a limited range of clusters. From this we took the maximum score to determine our cluster.

```
In [33]: #MODEL

from sklearn.cluster import KMeans
from sklearn import metrics

# define a dictionary that contains all of our relevant info.
results_dict = {}

# define how many clusters we want to test up to.
num_of_clusters = 10

# run through each instance of K
for k in range(2, num_of_clusters):
    print("-" * 100)

    # define the next dictionary to hold all the results of this run.
    results_dict[k] = {}

    # create an instance of the model, and fit the training data to it.
    kmeans = KMeans(n_clusters=k, random_state=0).fit(X_train_robust)

    # define the silhouette score
    sil_score = metrics.silhouette_score(X_train_robust, kmeans.labels_, metric='euclidean')

    # store the different metrics
    results_dict[k]['silhouette_score'] = sil_score
    results_dict[k]['inertia'] = kmeans.inertia_
    results_dict[k]['score'] = kmeans.score
    results_dict[k]['model'] = kmeans

    # print the results
    print("Number of Clusters: {}".format(k))
    print('Silhouette Score:', sil_score)

-----
Number of Clusters: 2
Silhouette Score: 0.702720724668849
-----
Number of Clusters: 3
Silhouette Score: 0.6511211450842931
-----
Number of Clusters: 4
Silhouette Score: 0.6819628826625381
-----
Number of Clusters: 5
Silhouette Score: 0.6634179764931054
-----
Number of Clusters: 6
Silhouette Score: 0.6521259023036475
-----
Number of Clusters: 7
Silhouette Score: 0.6474296886252918
-----
Number of Clusters: 8
Silhouette Score: 0.643495501083568
-----
Number of Clusters: 9
Silhouette Score: 0.638631582724426
-----
```

Once running through our clusters, the first thing we look at is the overall silhouette score, ideally the larger the number, the better the results. However, this score is just part of the project, we will have to examine the results to give ourselves our own visually represented proof. However, we can use the silhouette score as an indication of which ones we should target first.

Looking at the results above, 2 or 3 should be our target. When running our PCA data set to see what the output would look like. In this case, we get a higher silhouette score, which is a good indication and points to the same outcome, exploring a cluster of 2 or 3

## Model Evaluation

With the help of a graphical aid, we can also analyze the results of our cluster. If we use the *yellow brick* library, we get to access the Silhouette Visualizer which will help visualise the silhouette score for each point in that particular cluster. What we are looking for is that each cluster exceeds the red line or the above average silhouette score and that the clusters are as evenly distributed as possible.

```
In [35]: #MODEL EVALUATION
from yellowbrick.cluster import SilhouetteVisualizer

clusters = [2, 3]

for cluster in clusters:
    print('-' * 100)

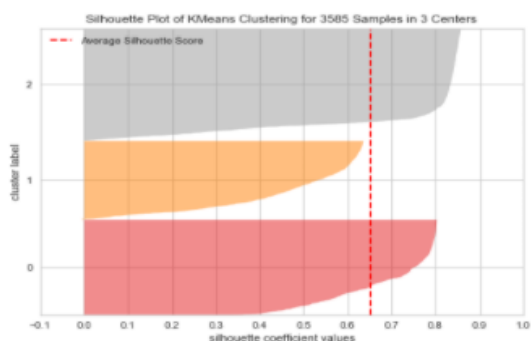
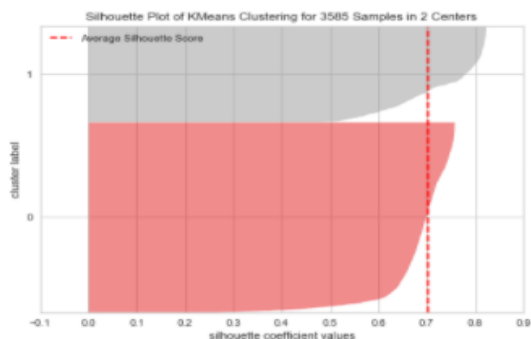
    # define the model for K
    kmeans = KMeans(n_clusters=cluster, random_state=0)

    # pass the model through the visualizer
    visualizer = SilhouetteVisualizer(kmeans)

    # fit the data
    visualizer.fit(X_train_robust)

    # show the chart
    visualizer.poof()

    clusters = [2, 3]
```



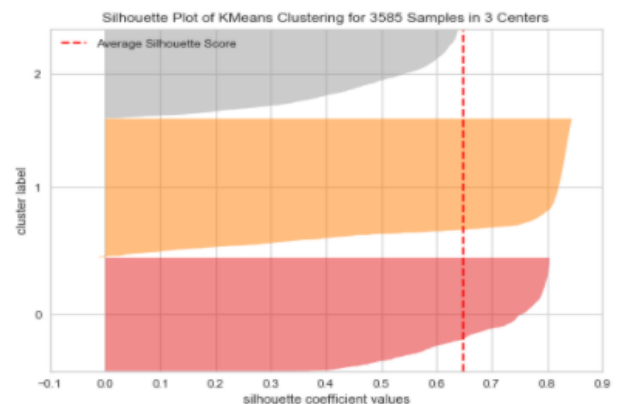
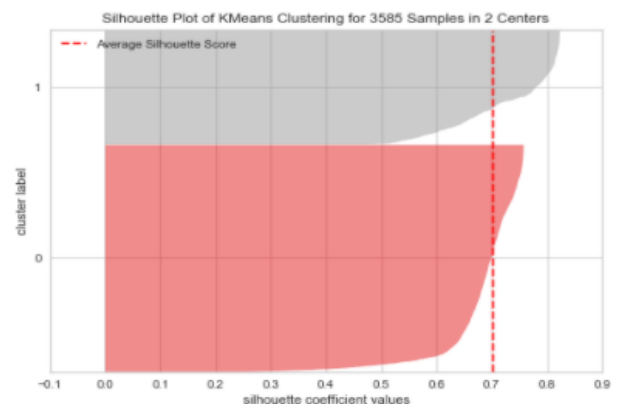
```
In [36]: for cluster in clusters:
    print('-' * 100)

    # define the model for K
    kmeans = KMeans(n_clusters=cluster, random_state=0)

    # pass the model through the visualizer
    visualizer = SilhouetteVisualizer(kmeans)

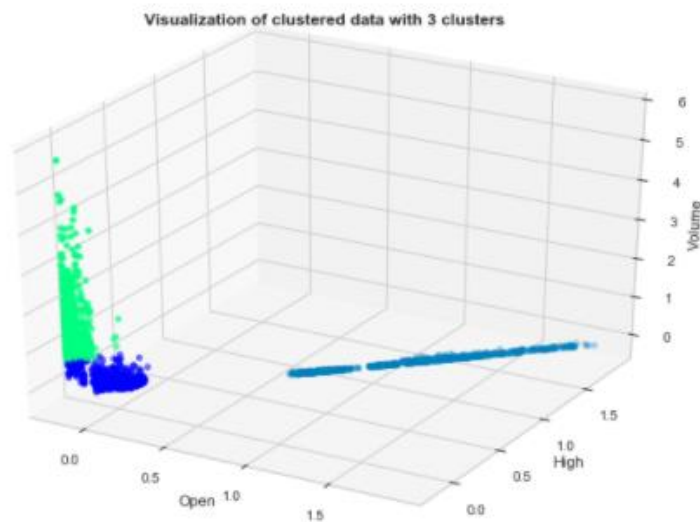
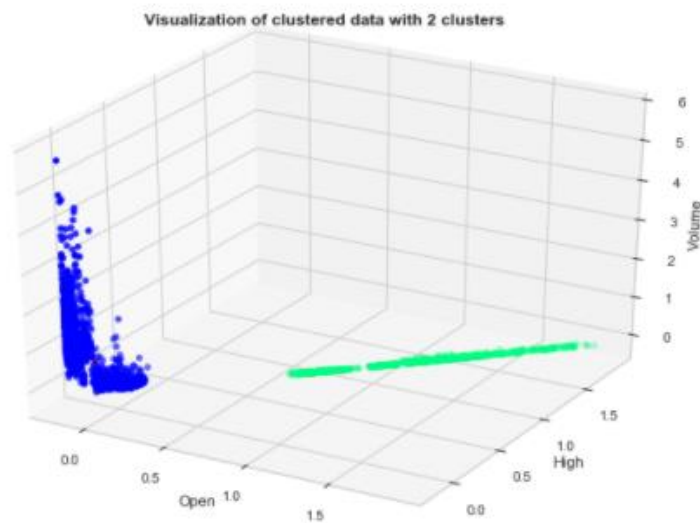
    # fit the data
    visualizer.fit(pca_dataset)

    # show the chart
    visualizer.poof()
```



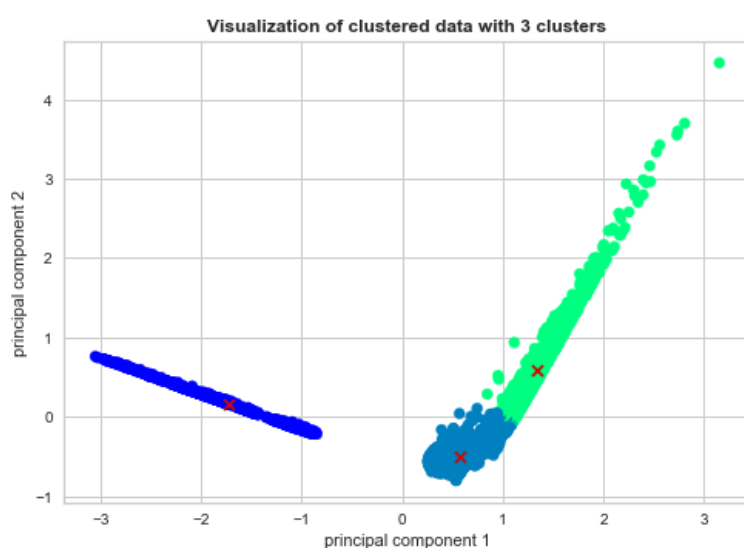
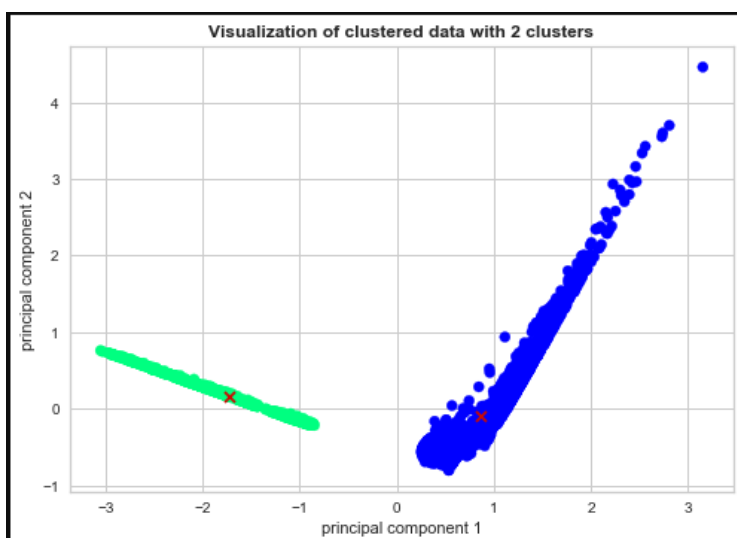
### Chart the Clusters

This step involved visualizing our cluster, this will involve in the non PCA case, creating another 3D scatter plot, but in this instance, we will define the c parameter of our scatter to equal our model labels, this will correctly create the cluster for us. The create will do this in the case of 2 and 3 clusters so we are able to visually see the difference.



From the predictions above, we have some companies who perform horribly across all three metrics, that would be the dark blue religion in the lower left-hand corner. We then have our average company, denoted by the colour green; these are companies who might beat on some metrics, lose on some, or do average. Finally we have our “show stopper”. These are companies denoted in the light blue and represent the companies who are crushing every metric.

Again, there might be different ways to interpret these results, but for our team it visually made sense, and while they might not have a nice spherical structure there is some type of grouping there that is weak in some instances.



## Data Visualisation Models:

### LINE CHART:

Line graphs are one of the most fundamental tools for data analysis and in stocks they can be found virtually everywhere, generally showing investors the price of a stock over time. Although this is widely used, it doesn't provide much insight into the reasons for the changes in stock market values, but more of a snapshot of the result.

We implemented the line graph in Python by employing the yahoo finance stock libraries which we used to import the data and display the closing price of each time interval.

```
start_date = datetime.now().date() - timedelta(days=5 * 365)
end_data = datetime.now().date()
df = yf.get_data(
    ticker, start_date=start_date, end_date=end_data, interval="1d"
)
stock = Sdf(df)
```

Using the line “yf.get\_data(#parameters)” we are able to import stock market data directly from the yahoo finance server and in the parameters, we can set the time period we wish to analyse too.

```
# line plot
if chart_name == "Line":
    fig = go.Figure(
        data=[
            go.Scatter(
                x=list(df.index), y=list(df.close),
            )
        ],
```

- This shows how we configured the settings of the line graph. Using the function defined before “df” we can set the values of the x and y axis to being the company name and the closing price of each interval, respectively.

### CANDLESTICK:

Moving on from the simplistic line graph, the candlestick chart is an indicator of the overall profit or loss of every single day, as displayed onto a time series 2-dimensional graph. This method is similar to the line graph in how it's represented but along with the actual closing price, the chart also shows the change in value in a single “candlestick”.

Some aspects to note:

- RED candlestick indicates a good SELLING opportunity since RED represents a lower closing price than the opening (loss).
- GREEN candlestick indicates a good BUYING opportunity since GREEN represents a higher closing price than the opening (profit).
- The tails or “wicks” of the candlestick represent the HIGH and LOW prices of the stock throughout the day.
- The size of the candlestick indicates the magnitude of change in price.

```
# Candelstick
if chart_name == "Candlestick":
    fig = go.Figure(
        data=[
            go.Candlestick(
                x=list(df.index),
                open=list(df.open),
                high=list(df.high),
                low=list(df.low),
                close=list(df.close),
                name="Candlestick",
            )
        ]
    )
```

- Once again loading the stock data from yahoo finance and injecting it into the in-built candlestick function within Python.

- This shows how we configured the structure of the candlestick chart by identifying the key characteristics (open, high, low and close) to the Python program environment.

## SIMPLE MOVING AVERAGE:

Most investors don't need a per-second snapshot of the current price every second to understand whether or not a stock is worth buying or selling; in fact it only complicates matters when everyday investors witness the violent fluctuations of share price on a daily or even weekly basis. This is where SMA (simple moving average) becomes useful. SMA provides a value which is the average of a range of closing prices over a selected time period.

Price(SMA)= (Sum of prices)/(number of time intervals)

For example, to analyse a company using a 10-day simple moving average, the program would average the prices over 10 days and on the 11th day, the earliest price would be removed and the latest price for the 11th day is added in.

$$\text{Price (after the first 10 days)} = \frac{22 + 21 + 19 + 23 + \dots + 25}{10}$$

$$\text{Price (after the first 10 days)} = \frac{21 + 19 + 23 + \dots + 25 + 24}{10}$$

Eventually a good understanding is attained of the overall fluctuation of the price. However outliers will definitely have a negative impact when using SMA to analyse a stock.

```
if chart_name == "SMA":
    close_ma_10 = df.close.rolling(10).mean()
    close_ma_15 = df.close.rolling(15).mean()
    close_ma_30 = df.close.rolling(30).mean()
    close_ma_100 = df.close.rolling(100).mean()
    fig = go.Figure(
        data=[
            go.Scatter(
                x=list(close_ma_10.index), y=list(close_ma_10), name="10 Days"
            ),
            go.Scatter(
                x=list(close_ma_15.index), y=list(close_ma_15), name="15 Days"
            ),
            go.Scatter(
                x=list(close_ma_30.index), y=list(close_ma_30), name="30 Days"
            ),
            go.Scatter(
                x=list(close_ma_100.index), y=list(close_ma_100), name="100 Days"
            ),
        ],
    )
```

- In Python we defined the different characteristics of a SMA for 10, 15, 30 and 100 days using the closing price for each day in the program's calculations.

### Exponential Moving Average:

Simple moving averages are great indicators for the long-term trend of a stock's value. However many investors miss out on shorter term profits simply because they can't keep up with the tsunami of media coverage that only provides an even more confusing picture of the stock market's volatility. This is where the exponential moving average (EMA) becomes useful.

A key aspect to note about EMA is that it provides a much greater significance to recent stock market data than SMA (simple moving average) does. When calculating the current EMA there are some steps to follow:

1. Calculate the weight. The weight essentially provides a number to indicate how significantly the previous day's stock price will affect the current price.

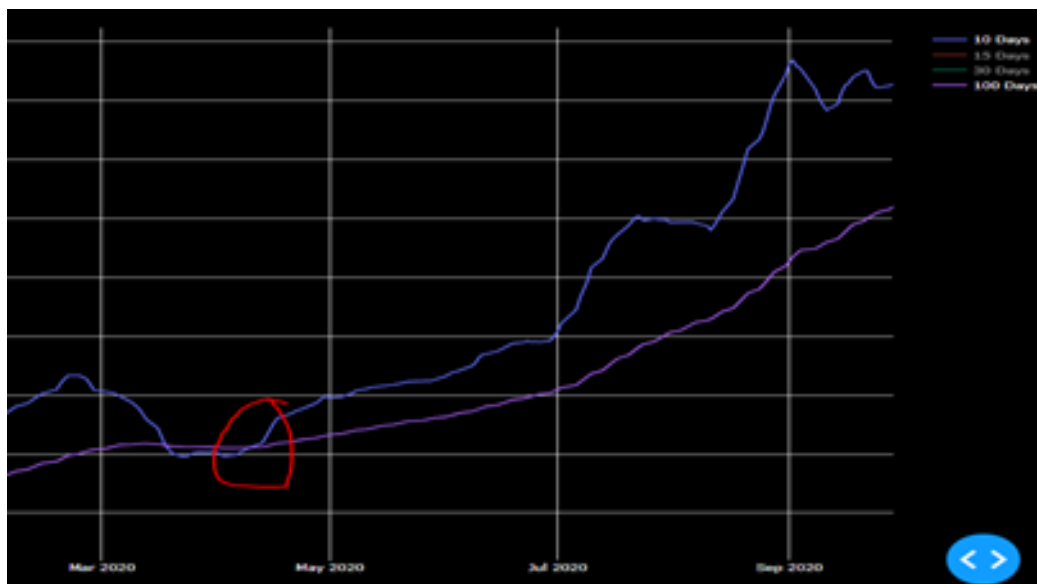
$$\text{Weight} = \frac{2}{\text{number of intervals} + 1}$$

The time interval indicates how many days the EMA is being analysed over. For example when describing the simple moving average previously, the moving average could be calculated over 10 days, 30 days or more.

## 2. Calculate EMA based off this formula

$$EMA = [Price(today) \times weight] + [EMA(yesterday) \times (1 - weight)]$$

However, the EMA of just one-time interval by itself isn't using the EMA analysis to its full potential. When comparing EMAs of different time intervals, investors can make accurate decisions to buy or sell based on how the EMAs interact with each other.



The blue line is the 10-day EMA.

The purple line is the 100-day EMA.

This shows some EMA analysis of Tesla stock using the 100 day and 10-day EMA overlayed on top of one another using the product our group has designed. In the circled red region, the 10 day and 100-day EMA intersect with each other as the 10-day EMA exceeds the 100-day EMA. This is a good indicator to **buy** Tesla stock due to the following reasons.

- The 10-day EMA is much more volatile than the 100-day EMA because it is much more similar to the actual price of the stock
- Therefore when the more volatile indicator surpasses the stable indicator, it is much more likely that the 10-day EMA will continue at a larger value than the 100-day EMA. This is evident in the rapid increase of Tesla stock after the intersection.

Conversely if it were the other way around (the 10-day EMA dropped below the 100-day EMA) this would be a good indicator to **sell** Tesla stock.

## Moving Average Convergence Divergence (MACD)

So far, the visualization techniques that have been explored discuss how to predict whether a stock should be bought or sold. However one of the greatest and unsolved problems for investors is predicting whether or not a stock will continue plummeting or skyrocketing; essentially figuring out the momentum of the market. This is where moving average convergence divergence (MACD) becomes a useful tool for analysis.

The theory of how MACD functions is dependent on understanding the theory of moving averages, mainly EMA. MACD is essentially the difference between the 26 day EMA and the 12 day EMA. However unlike EMA or SMA, MACD is never analysed on its own to determine the price movement of a stock. In fact, there are a couple of terms that need to be made familiar:

- Signal line. The signal line is a 9-day EMA performed on the MACD line.
- Histogram. The histogram is a popular form of data visualization, and in this case the histogram represents the difference between the MACD line and the signal line.



Figure source from Investopedia

- When the MACD line drops below the signal line, that is the indicator to **sell** (and vice versa).

In the figure above when the histogram resides mainly in the red (negative), this can show investors that a reversal in the market is probable to occur, and in this case, it means that the stock's value is likely to rise.

```
fig = go.Figure(
    data=[
        go.Scatter(x=list(df.index), y=list(df.MACD), name="MACD"),
        go.Scatter(x=list(df.index), y=list(
            df.signal), name="Signal"),
        go.Scatter(
            x=list(df.index),
            y=list(df["hist"]),
            line=dict(color="royalblue", width=2, dash="dot"),
            name="Histogram",
        ),
    ],
)
```

- Application of MACD, signal lines and a histogram in the Python environment

### Relative strength index (RSI)

Similar to MACD, the relative strength index (RSI) is also used to determine the momentum of a market. However, RSI is a more quantitative representation of the steepness of a rise or fall in stock prices. So, if the investor requires a more direct indicator to buy or sell a stock RSI can be very useful, although it must be noted that along with RSI's quantitative advantage, it can also be quite unpredictable in many cases. So it is recommended that if RSI is used to make decisions, it should be used in connection with other technical indicators such as MACD, EMA etc.

The RSI is visualized as a line graph valued between 0 and 100. The formula for calculating RSI is as follows:

$$RSI = 100 - \frac{100}{1 + \frac{\text{average gain}}{\text{average loss}}}$$

Generally speaking, it is accepted that:

- An RSI value above 70 means that the stock is overvalued, and it should be **sold**.
- An RSI value below 30 means that the stock is undervalued, and it should be **bought**.

This is because when the average gain is higher than average loss, the fraction in the formula will tend to be a smaller value, hence a higher RSI should be an indicator to **sell**. Conversely when average loss is greater than average gain the fraction in the formula will tend towards a larger value, hence a smaller RSI is an indicator to **buy**.

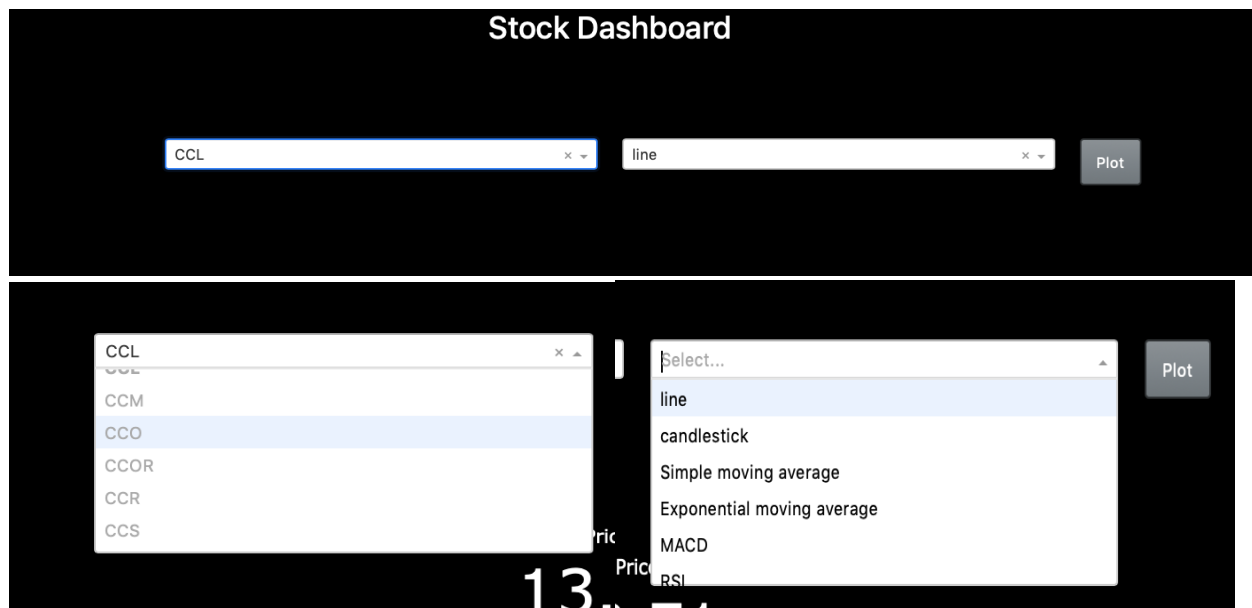
## Dash Plotly (Frontend/Backend):

The stock market stakeholders look for a high-level view of the stock market in order to analyze, identify and react to various changes in the stock market. A dashboard is one of the most interesting and fun ways to make decisions about investment, asset allocations along with balancing the risk against performance. It is a useful tool to show how the shares have performed in the past year, month or day allowing the customers to make an informative decision.

The final product is delivered by an interactive Stock Dashboard which is developed using Plotly dash and Dash Bootstrap CSS. The Dash is ideally built for data visualisation with a high customer user interface. It is built to render in any web browser and be shared around through URLs. A systematic python web framework written on React.js, Plotly.js, and flask is used to build a well responsive web application to display detailed interactive visualisations for better decision making.

### **User's View:**

On loading the dash, the application takes 2 inputs from the user: Stock company(ticker) and type of visualisation to be plotted. The company and visualisation dropdowns are pre-populated with a range of tickers and graphs that can be selected.



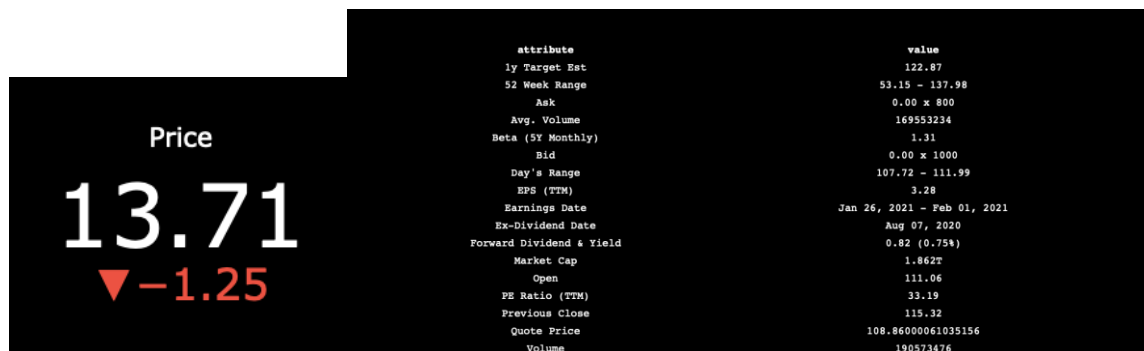
When a ticker is chosen by the user, the dashboard pulls the data from Live Yahoo Finance to produce information about the company's price history and financial market in a graphical representation. The range of interactive visualisations available as of now are:

- Line Graph
- Candlestick
- Simple Moving Average
- Exponential Moving Average

- Moving Average Convergence Divergence (MACD)
- Relative Strength Index (RSI)
- Open High Low Close (OHLC)
- 2D Scatter Plot

\*\* Detailed explanation of the visualisations is provided in the sections above

The dash also displays the tickers price rise or fall when a specific ticker is selected from the dropdown and other important attributes like volume, day's range, open, quote price , PE ratio etc.

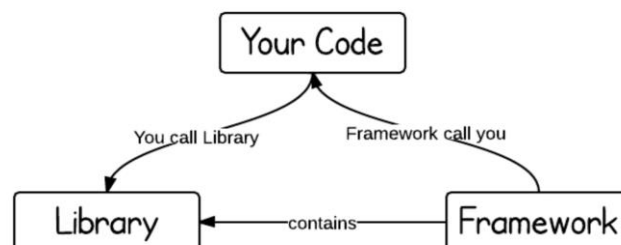


All the graphs are displayed with a time slider widget to allow the user to select which period of time or section of the graph to view. It is used to allow the user to focus on the details of a short range of time which is a great feature when analysing and comparing.



### Coder's View :

The dash is made up of layout and callback functions in python which describes the look of the application and makes the dashboard interactive respectively. Data Bootstrap components and dash HTML components allow for a lot of customization. The live data is loaded using the yahoo finance package available in python. CSS in combination with javascript is used for designing all the components of the dashboard while using plotly package in python to present interactive graphs. In order to start using dash several packages need to be downloaded.



The different types of packages used are as follows:

### 1. For dash:

```
import dash
import dash_core_components as dcc
import dash_html_components as html
import dash_bootstrap_components as dbc
import dash_table as dt
from dash.dependencies import Input, Output, State
import plotly.graph_objs as go
```

- **Dash:** This kickstarts the dashboard layout by calling the dash class of dash library
- **Dash\_core\_components:** Used to create graphs on the layout by using the graph class from the components library
- **Dash\_HTML Components:** It is used to generate HTML tags like H2, H1, background etc. Various styling tags and properties such as colour, font size, layout, text alignment are also used from this package.
- **Dash\_bootstrap\_components:** To make the designing of the app layout much easier. By installing this library, it allows it to use bootstrap components for front-end development from the bootstrap CSS framework. It is similar to HTML components.
- **Plotly:** High-end interactive features in graphs are given using the plotly package from python
- **Dash.dependencies:** on importing this package it allows to input text in a box in real time and updates it in my-div
- **Dash\_table:** This component is written using React.js. It is used to edit, view or explore large interactive spreadsheets or tables.

### 2. For live yahoo data:

```
from stockstats import StockDataFrame as Sdf
import yahoo_fin.stock_info as yf
from datetime import datetime, timedelta, date
```

- **Stockstats:** By initialising stock dataframe from stockstats, it converts pandas.dataFrame into StockDataFrame. This package considers that the data being used contains certain columns like open, high, low, close, volume etc and is timestamped. Stockstats is a package which is useful to calculate the indicators and statistics such as Exponential Moving Average (EMA), Moving Average Convergence Divergence(MACD), Relative Strength Index (RSI)

- **Yahoo\_fin:** Yahoo\_fin is a Python 3 package designed to scrape historical stock price data, as well as to provide current information on market caps, dividend yields, and which stocks comprise the major exchanges. Additional functionality includes scraping income statements, balance sheets, cash flows, holder information, and analyst data. The package includes the ability to scrape live (real-time) stock prices, capture cryptocurrency data, and get the most actively traded stocks on a current trading day. Yahoo\_fin also contains a module for retrieving option prices and expiration dates.
- **DateTime:** Datetime package is installed to work with dates as objects. This module provides access to classes that allows to manipulate date and time according to the problem statement.

### 3. For Data analysing:

```
import numpy as np
import pandas as pd
import random
import numpy
```

- **Pandas:** Pandas provides fast and expressive data models and analysis on large datasets. It is considered the most powerful data analysis and data manipulation tool ever available. It is suited for various kinds of data - statistical data, tabular data, arbitrary matrix data. Some of its best functions include handling missing values, group by, indexing, time series, data alignment, inserting and deleting columns. In this stock market visualization, pandas is commonly used to load large datasets from CSV files making it ready for manipulation and analysis.
- **NumPy:** This package is mainly used for working with powerful n-dimensional arrays. It provides functions to create arrays, array indexing, array math and broadcasting arrays,
- **Random:** By importing this standard python package it enables the system to generate random numbers.

### Layout:

The dashboard development is divided into two parts: Layout and interactivity. The app.layout describes how the app looks like. The `dash_core_components` library generates higher-level components like controls and graphs that are generated with HTML, CSS and javascript through the `react.js` library. The `dash_html_components` library provides classes for all of the HTML tags, and the keyword arguments describe the HTML attributes like `style`, `className`, and `id`. Styling such as changing the background, font colours, text and graphs alignment can be manipulated according to the requirement. Dropdowns are generated by calling the dropdown object from the `dash_core-components` library. The values are selected using the `values` attribute. The

dcc.dropdown object is used to create a dropdown option to select the desired ticker and type of graph the user wants to view.

```
app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
server = app.server
app.layout = html.Div(
    style={"backgroundColor": colors["background"]},
    children=[
        html.Div(
            [ # header Div
                dbc.Row(
                    [
                        dbc.Col(
                            html.Header(
                                [
                                    html.H1(
                                        "Stock Dashboard",
                                        style={
                                            "text-align": "center",
                                            "color": colors["text"],
                                        },
                                    ),
                                ],
                            ),
                        ),
                    ],
                ),
            ),
        ),
        html.Br(),
        html.Br(),
        html.Br(),
        html.Br(),
        html.Div(
            [ # Dropdown Div
                dbc.Row(
                    [
                        dbc.Col( # Tickers
                            dcc.Dropdown(
                                id="stock_name",
                                options=[
                                    {
                                        "label": str(ticker_list[i]),
                                        "value": str(ticker_list[i]),
                                    }
                                    for i in range(len(ticker_list))
                                ],
                                searchable=True,
                                value=str(
                                    random.choice(
                                        [
                                            "TSLA",
                                            "GOOGL",
                                            "F",
                                            "GE",
                                            "AAL",
                                            "DIS",
                                            "DAL",
                                            "AAPL",
                                            "MSFT",
                                            "CCL",
                                            "GPRO",
                                            "ACB",
                                            "PLUG",
                                            "AMZN",
                                        ]
                                    )
                                ),
                                placeholder="enter stock name",
                            ),
                        ),
                    ],
                ),
            ),
        ),
    ],
)
```

### App callbacks:

The designed layout in conjunction with the entered inputs becomes functional by adding the app callbacks. The callbacks are broken down as follows:

- Populating the charts by taking the ticker and type of graphs as inputs
- Populating the financial or stock market report
- Loading the selected companies data by passing the input ticker to yfinance
- Sorting and filtering
- Live date time

```
@app.callback(
    # output
    [Output("graph", "figure"), Output("live price", "figure")],
    # input
    [Input("submit-button-state", "n_clicks")],
    # state
    [State("stock_name", "value"), State("chart", "value")],
)
```

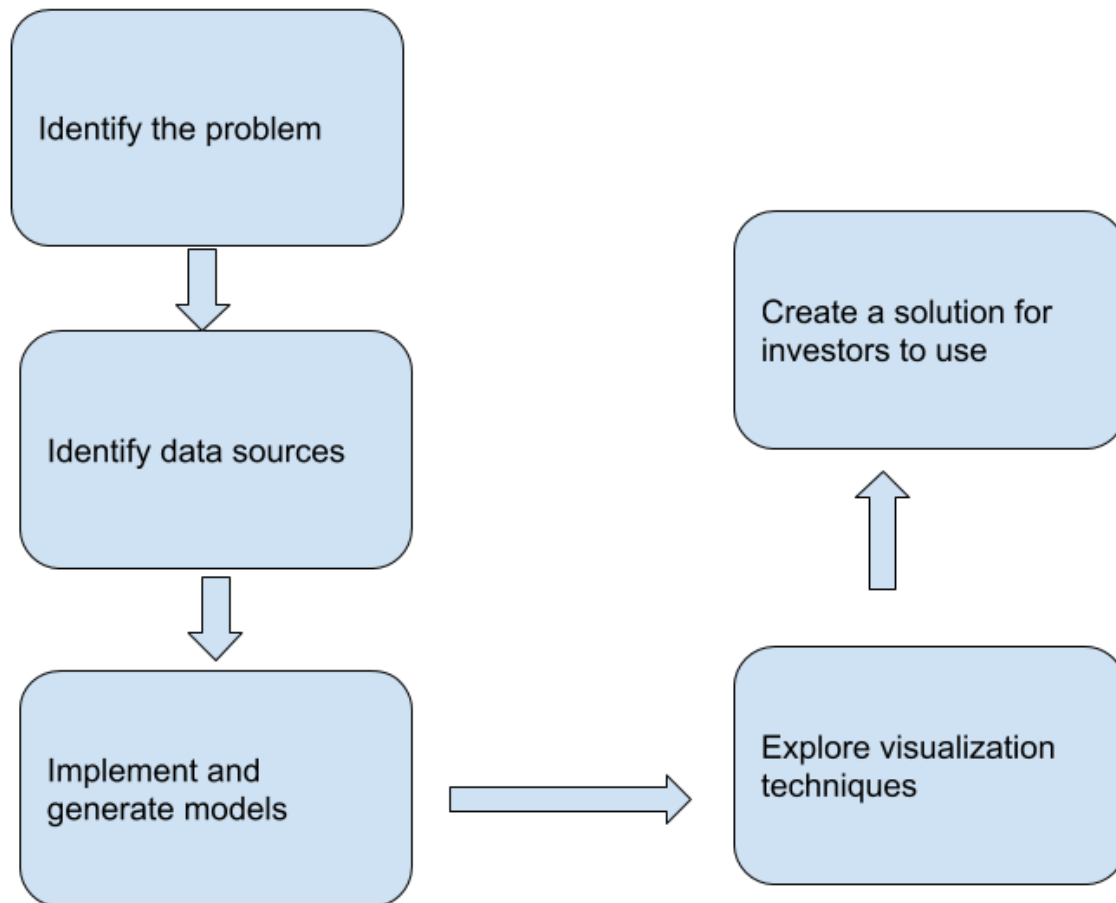
```
209         # loading data
210         start_date = datetime.now().date() - timedelta(days=5 * 365)
211         end_date = datetime.now().date()
212         df = yf.get_data(ticker, start_date=start_date, end_date=end_date, interval="1d")
213         stock = Sdf(df)
```

Just like in Flask we need to run the web server to view the dashboard and the graphed visualizations. The debug is set to true to avoid refreshing the browser each time a new input is entered, or new changes have been made.

```
if __name__ == "__main__":
    app.run_server(debug=True)
```

### THE DASH LINK:

<https://stockdashboardlive.herokuapp.com>

**Development Process Flowchart:****1. Identify the problem**

Our team reframed the problem at hand when we opted to focus on helping investors make trading decisions through useful visualization techniques.

**2. Identify the data sources**

At first, the team relied on the extensive Kaggle dataset. However this was outdated and also was not pre-processed data. As a result we found a solution by incorporating the in-built functions and libraries of Python that allow us to access filtered data from yahoo finance.

**3. Explore data visualization techniques**

The team explored techniques such as PCA, multiple variants of moving averages, RSI, MACD, scatter plots and more.

#### 4. Implement and generate models

Using the extensive Python environment, the team members created and tested visualization techniques.

#### 5. Create a solution for investors to use

Using the vast capabilities of Python, we created a dashboard that allows users to visualize any company they're interested in and analyse it using any one of a number of technical indicators ranging from as simple to line graphs to as sophisticated as the relative strength index (RSI) or MACD.

## **Enhancements & Future Endeavors:**

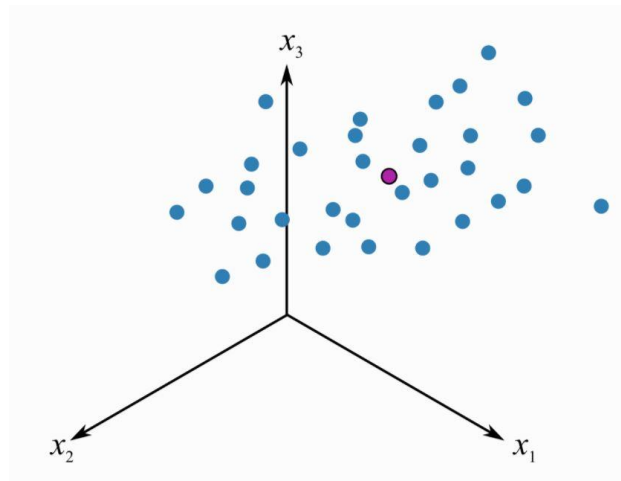
### **Dimension Reduction – PCA:**

PCA is a statistical procedure that uses a matrix model to summarize information in large datasets by means of a smaller “summary of indices”, it does this by transforming a number of correlated variables into a smaller number of uncorrelated variables called ‘principle components’. Hence, the main objective of principle component analysis is dimension reduction through which the original information, trends and variability are preserved within the reduced dataset. And hence this lower dimension level of stock data will provide an enhanced level of analysis, visualization and understanding of the information present for the users of our product.

### **PCA process:**

#### Step 1: Mean-centralisation and scaling of data points

The first step in PCA implementation is called mean-centering. Mean-centering involves moving the coordinate system to a new reference point, usually the origin of the coordinate system in K variables (i.e. in K-dimensional space). Figure 1.1 represents this pre-processing tool, and how it is utilized to remove arbitrary bias from measurements that don't show value for analysis. Standardization and scaling of data is also implemented to reduce the dominance and bias of attributes with greater variance against attributes with smaller variance.



*Figure 1.1: Centering and scaling of raw*

#### Step 2: Computation of covariance matrix

Covariance is the measure of how much two random variables vary together. If greater values of one variable correlate with the greater values of another variable, and vice versa than the covariance can be deemed as positive. Hence, a covariance matrix is computed to understand how variables from the original dataset vary from their mean, which aligns with identifying if there

is a relationship. The relationship is assessed of these attributes to determine the level of correlation between them, from which highly correlated variables that will produce redundant information can be removed from the dataset. Hence a covariance matrix is produced to compute the correlations.

The covariance matrix is a  $d \times d$  symmetric matrix (d represents the number of dimensions). Below *Figure 1.2* represents the covariance matrix of a 3-dimensional dataset with the attributes x, y and z.

$$\begin{bmatrix} Cov(x, x) & Cov(x, y) & Cov(x, z) \\ Cov(y, x) & Cov(y, y) & Cov(y, z) \\ Cov(z, x) & Cov(z, y) & Cov(z, z) \end{bmatrix}$$

*Figure 1.2: Covariance Matrix for 3-dimensional dataset*

### **What do the computed covariances in the matrix represent?**

The sign of the covariance between the two variables determines the correlation between the two variables. If the covariance between x and y was positive for example, then it can be identified that two variables data points increase or decrease together and hence shows that there is a correlation between the variables x and y. If the covariance between z and y for instance was negative, then it can be identified that there is an inverse correlation between the two variables in which one variables data increases the other variables data points will decrease.

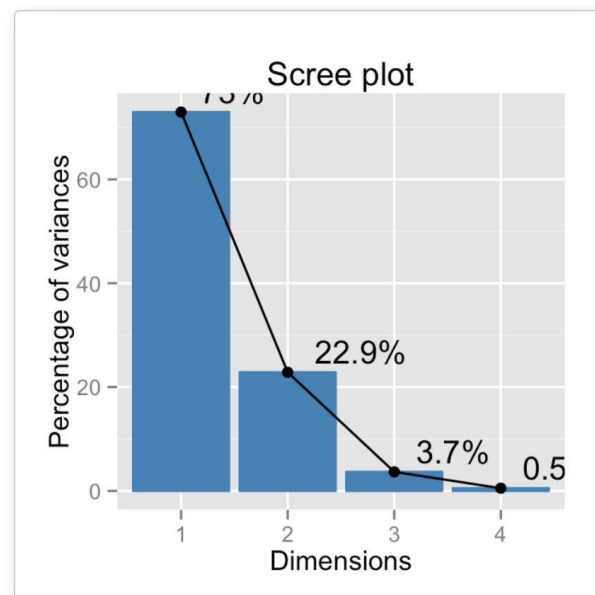
Hence, we can conclude that a covariance matrix is simply a table that summarises the correlations and relationships between all possible variables available in the dataset.

### **Step 3: Compute the eigenvectors and eigenvalues of the covariance matrix to identify principle components**

Eigenvectors and eigenvalues are linear algebraic concepts that must be computed using the covariance matrix in order to the identify principle components of the data.

### **What is a principle component?**

Principle components are newly constructed variables that are either linear combinations or mixtures of variables in the original dataset. These computed combinations are produced such that the new principal components formed are uncorrelated and as much data and information from the original variables is fitted into the first principal components. In express the theory as an application, if we were to have a 4-dimensional dataset which produces 4 principle components, PCA will apply the maximum possible data points into the first component. Followed by the remaining information into the second component and so forth, until a graph like below in *Figure 1.3* has been produced.

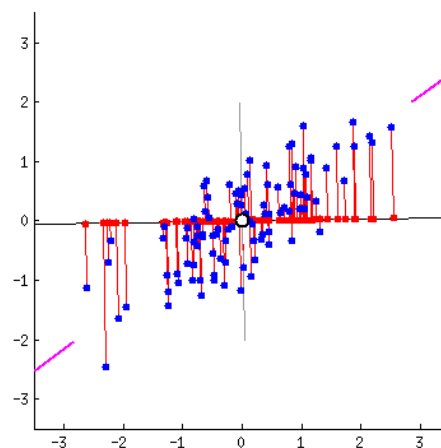


*Figure 1.3 – Scree plot shows the number of principle components to keep in a principle component analysis and the weighted allocation of data to each principle*

Hence, principal components are the key method of PCA, as they organize information so that there can be a reduction in dimensionality whilst preserving large amount of information. This is achieved by removing the components with low percentage of information assigned to them and simply attaining the components with larger information percentages as the new variables of the dimension reduced dataset. Geometrically, principal components can be expressed as directions or lines of data that explain the maximal amount of data and thus capture the most information of the dataset.

Now to put principle components into a PCA application, below *Figure 1.4* is a scatter plot that holds the information of our dataset. From our understanding of principle components, we are actually able to predict the first principle component. As we know that the first principal component accounts for the largest possible variance in the data set, we can predict that the first principal component is when the line matches with the purple lines because it firstly passes through the origin. It is also the line in which the projected points (red points) are the most spread out, and hence mathematically the line produces the greatest variance.

*Figure 1.4: Scatter plot showing the first principle*



Eigenvectors and eigenvalues are the vectors that are behind the formation of principle components. They act as a pair in which eigenvector points in the direction of linear transformation such as stretching for example, and eigenvalue is the factor affects the linear transformation. If for instance eigenvalue was negative for a stretching transformation than the direction is reversed.

Eigenvector and eigenvalues correspond to the number of dimensions within the dataset. Eigenvectors from the covariance matrix are the directions of the axes where there is most information or variance. Hence, the eigenvectors determine the direction of the principle components. Eigenvalues are the coefficients attached to the covariance eigenvectors, they effect the level of variance in the principle component.

Using eigenvalues and eigenvectors within a covariance matrix, the significance of the principle component can be determined. Below in *Figure 1.5* is the eigenvector and eigenvalues of the covariance vector from a 2-dimensional dataset.

$$v1 = \begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \quad \lambda_1 = 1.284028$$

$$v2 = \begin{bmatrix} -0.7351785 \\ 0.6778736 \end{bmatrix} \quad \lambda_2 = 0.04908323$$

*Figure 1.5 – Eigenvector and eigenvalues*

If we compare the eigenvalues of both variables, we can determine that  $\lambda_1 > \lambda_2$ , hence we can identify that the eigenvector that corresponds to the first principle component is v1 and v2 corresponds to the second principle component.

The eigenvectors and eigenvalues can also be used to calculate the percentage of variance computed by each principle component. To calculate this, the eigenvalue of each component is divided by the sum of the eigenvalues from both components together. If we apply this on the example above, we find that PC1 and PC2 carry respectively 96% and 4% of the variance of the data.

Step 4: Feature Vector

This PCA step involves discarding the principle components with less significance and hence less information (low eigenvalues) , and with the remaining components forming a matrix of vectors that is called a *feature vector*. The feature vector is a matrix format in which the columns are made up of the eigenvectors from the components chosen to preserve in the PCA. This step can be analysed as the initial phase of the dimension reduction within the dataset.

Continuing with our previous example, a feature vector with dimension reduction can be implemented by discarding one of the eigenvectors  $v_1$  and  $v_2$  and computing a feature vector with the remaining eigenvector.

$$\begin{bmatrix} 0.6778736 & -0.7351785 \\ 0.7351785 & 0.6778736 \end{bmatrix}$$

As we can identify from the example,  $v_2$  is the less significant eigenvector and can hence be discarded from the PCA process. Hence, a feature vector can be produced using  $v_1$ :

$$\begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix}$$

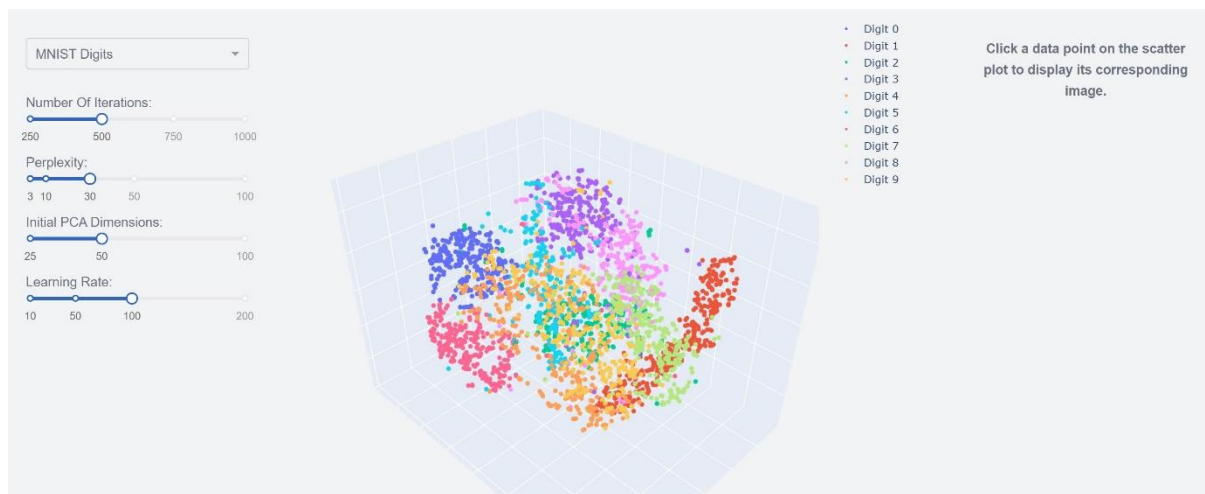
Discarding  $v_2$  will reduce the dimensionality in the dataset by 1 but will lose information from the dataset. However, as previously identified  $v_2$  only accounted for 4% of the total information in the dataset against the 96% carried by eigenvector  $v_1$ .

Step 5: Deriving the new dataset

The final step of the PCA process is to use the feature vector formed from the eigenvectors of the covariance matrix to reshape the original dataset axes to the axes represented by the principle component variables. This is achieved by taking the transpose of the feature vector and multiply it by the transpose of the original dataset. The transposed feature vector is now a row feature vector where the eigenvectors are now in rows such that the most significant are at the top of the multiplication process with the transposed original dataset.

## Improving dash and visualisations:

The team's discovery and research into the Dash plotly library and application was made late into project development. Hence, it is an area we would like to further develop and improve upon to further enhance the commercial aesthetic and functionality of financial data visualizations. One of the main goals we would like to set and achieve if we were to continue developing this product in the future is the application of PCA and 3D models into our dash app. As we scoped the app we have produced is a key platform for user personal analysis, as the user is allowed to select the features to research, however the opportunity to implement PCA and 3D visualisation models in the app would further enhance the user interaction and analysis gained by the benefits of a 3d models and dimension reduced models that can be zoomed, rotated and hovered over for specific data information. We have currently identified a demo dash UI in the plotly websites gallery that is a t-SNE explorer. As shown below the UI produced a 3D scatter plot that is running the t-SNE algorithm.



Although t-SNE differs from PCA, there are certain aspects of this model that can be useful to research and study of. The interactive scalar on the left for the PCA dimensions stands out as a very useful tool to implement in our own UI product through which a user can select the number of dimensions and help customise their information for a more suited analysis for them.

Another future development feature we would like to implement into the app is the ability to select multiple companies and plot a specific visualisation model on the app. This would enhance the users analytical understanding of the relationships or the lack of relationships between certain companies, which has the potential to provide the user with a more in-depth grasp historical stock movements and future financial forecasting.

Sentimental analysis is a key feature to our product that we believe can vastly improve the analytical substance and information available to the user. Sentimental analysis can be described as a polarity detector (e.g. positive or negative opinion) within text, whether a whole document, paragraph, sentence, or clause. When viewing sentimental analysis through a stock market

context, we already understand how volatile and sensitive stock movements and prices are to sentiments of traders and investors. Currently, there are numerous AI companies now using sentimental analysis to predict stock movements or market trends. And social media is the biggest platform to perform sentimental analysis on, where the simple accessibility in today's age has meant that a greater volume of sentiment in the form of posts and other forms of content that present particular views.

Sentiment analysis tool parses retrieved data for positive and negative keywords. This works by checking content against a bank of words and phrases pre-programmed to flag as positive, neutral, and negative. Next, the software generates a sentiment score. This is based on the density of the positive and negative terms surrounding the chosen topic. From the initial research made, we have identified that sentiment scores closer to 1 suggest "positive sentiment" whilst scores closer to 0 represent "negative sentiment".

With sentiment analysis, you have more information to base your ultimate decisions on. It adds another dimension to a user's financial and stock analysis and decisions for investing. Hence, we believe by adding a sentimental analysis of news articles, tweets etc. we can further commercialise our product to help investors with investment decisions, where that be to avoid missing out on an opportunity or to mitigate an approaching risk.

We have some initial research into sentimental analysis and its application on dash plotly, currently there are numerous resources online such as YouTube tutorials that will provide a better understanding of the process building sentimental analysis models and how to attain data that will help make stock market trading decisions.

### *Evidence of research into sentimental analysis*



Full Dash App - Data Visualization GUIs with Dash and Python p.12

82,666 views • Mar 16, 2018

803 20 SHARE SAVE ...

### 1. Import Libraries

First, we import the libraries that we need to store the data. 'BeautifulSoup' is needed to parse data from FinViz while 'requests' is needed to get data. 'Pandas' is used to store the data in DataFrames while 'Matplotlib' is used to plot the sentiment on a chart. Finally, the 'nlk.sentiment.vader' library is used to perform sentiment analysis on the news headlines!

```
1 # Import libraries
2 from urllib.request import urlopen, Request
3 from bs4 import BeautifulSoup
4 import os
5 import pandas as pd
6 import matplotlib.pyplot as plt
7 %matplotlib inline
8 # NLTK VADER for sentiment analysis
9 from nltk.sentiment.vader import SentimentIntensityAnalyzer
10
11 finviz_url = 'https://finviz.com/quote.ashx?t='
12
13 imports_sentiment.py hosted with ❤ by GitHub
```

view raw

## **Reflection**

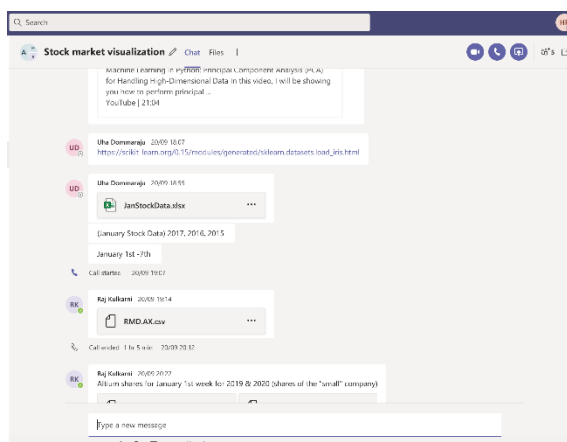
Reflecting upon the overall experience of the development of this application, all members of the group unanimously agree that we have all benefited and learnt more than we anticipated. We found it extremely beneficial working in a group in combination with both Fundamentals and Applications students. The Fundamentals students were able to question what it was possible to achieve whilst the Application students also provided guidance, resources and their expertise. This allowed for a great balance within the team dynamic as we all benefited from each-other to work united and complete our tasks.

Our meetings with the Product Owner (Don) were utilized effectively as we benefited from each session and able to update him on the project's progression. Don was available to help us as a team but also individually if we were seeking assistance. We felt that the resources he provided, especially at the beginning helped us significantly in understanding what we were trying to achieve and the projects direction.

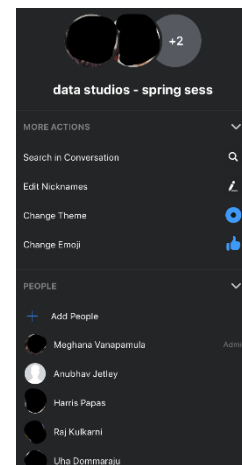
Whilst Microsoft Teams and Messenger were the primary platforms of communication, they also acted as a central place to send and retrieve our data and other information from. It allowed us to easily search for documents and track when the information had been shared and by who. Whilst this was extremely easy for us in the beginning, it eventually became a little more difficult as copies of documents were being duplicated/updated and resubmitted as some were too large to work on in the Microsoft Teams online Environment. In hindsight, we feel as though the group would have benefited from a third platform (such as Repl.it) where we could see other people's code and share code easier. It would also make it easier to update the team but also allow team members to help and edit each other's work without requesting them to send it or an updated version across. Regardless, the current processes were still effective and we still benefited from the platforms implemented.

We also feel that throughout the project's development, naturally elements have been changed which has inhibited us from achieving some of the initial requirements we set. As we evolved, we realised aspects such as predictions would be difficult to implement, however we were able to make the necessary amendments to continue on and as a result alter our project requirements. As a group we decided that we have fulfilled our amended requirements and proud to present the work we have produced, especially considering we had to work remotely.

*Evidence of Team Processes – Microsoft Teams*



*Evidence of Team Processes – Messenger Chat*



Stock market visualization Chat Files +

Share

✓	Type	Name	Shared on ▾	Sent by	
	File	1 visualise all the original dimensions.txt	26/10/2020	Uha Dommaraju	...
	File	2 visualise all the principle components.txt	26/10/2020	Uha Dommaraju	...
	File	3 2D PCA Scatter Plot.txt	26/10/2020	Uha Dommaraju	...
	File	4 visualise PCA with px.scatter_3d.txt	26/10/2020	Uha Dommaraju	...
	File	yahoodata.csv	06/10/2020	Uha Dommaraju	...
	File	2017-3ticks.csv	21/09/2020	Uha Dommaraju	...
	File	S&P500 2020.csv	20/09/2020	Raj Kulkarni	...
	File	S&P500 2019.csv	20/09/2020	Raj Kulkarni	...
	File	ALU.AX 2019.csv	20/09/2020	Raj Kulkarni	...
	File	ALU.AX 2020.csv	20/09/2020	Raj Kulkarni	...
	File	RMD.AX.csv	20/09/2020	Raj Kulkarni	...
	File	JanStockData.xlsx	20/09/2020	Uha Dommaraju	...

*Evidence of Team Processes – Microsoft Teams File*

## **References:**

1. Investopedia, Moving Average Convergence Divergence (MACD), 2020  
<<https://www.investopedia.com/terms/m/macd.asp>>
2. Investopedia, “*Relative Strength Index (RSI)*”, 2020  
<<https://www.investopedia.com/terms/r/rsi.asp>>
3. Investopedia, “*Simple Moving Average (SMA)*”, 2020  
<[https://www.investopedia.com/terms/s/sma.asp#:~:text=Key%20Takeaways-.A%20simple%20moving%20average%20\(SMA\)%20calculates%20the%20average%20of%20a,a%20bull%20or%20bear%20trend.>](https://www.investopedia.com/terms/s/sma.asp#:~:text=Key%20Takeaways-.A%20simple%20moving%20average%20(SMA)%20calculates%20the%20average%20of%20a,a%20bull%20or%20bear%20trend.>)>
4. Investopedia, “*How Is Exponential Moving Average (EMA) Calculated?*”, 2020  
<[https://www.investopedia.com/ask/answers/122314/what-exponential-moving-average-ema-formula-and-how-ema-calculated.asp#:~:text=The%20exponential%20moving%20average%20\(EMA\)%20is%20a%20technical%20chart%20indicator,importance%20to%20recent%20price%20data.>](https://www.investopedia.com/ask/answers/122314/what-exponential-moving-average-ema-formula-and-how-ema-calculated.asp#:~:text=The%20exponential%20moving%20average%20(EMA)%20is%20a%20technical%20chart%20indicator,importance%20to%20recent%20price%20data.>)>
5. Daily FX, “*What the MACD Indicator is and How it Works*”, 2019  
<<https://www.dailyfx.com/education/technical-analysis-tools/macd-indicator.html>>
6. Understanding Principle Component Analysis(PCA) step by step. 2020, Medium. viewed 3 November 2020, <https://medium.com/analytics-vidhya/understanding-principle-component-analysis-pca-step-by-step-e7a4bb4031d9>

7. Principal Components of PCA 2020, Medium. viewed 3 November 2020, <https://towardsdatascience.com/principal-components-of-pca-bea010cc1d33>
8. A Step by Step Explanation of Principal Component Analysis 2020, Built In. viewed 3 November 2020, <https://builtin.com/data-science/step-step-explanation-principal-component-analysis>.
9. Part 2. Layout | Dash for Python Documentation | Plotly 2020, Dash.plotly.com. viewed 3 November 2020, <https://dash.plotly.com/layout>
10. Dash for Beginners 2020, DataCamp Community. viewed 3 November 2020, <https://www.datacamp.com/community/tutorials/learn-build-dash-python>
11. Anon, 2020, Youtube.com. viewed 3 November 2020, <https://www.youtube.com/watch?v=rDodciQhfS8&t=92s>
12. Sentiment Analysis of Stocks from Financial News using Python 2020, Medium. viewed 3 November 2020, <https://towardsdatascience.com/sentiment-analysis-of-stocks-from-financial-news-using-python-82ebdcefb638>
13. How Sentiment Analysis in Stock Market Used for Right Prediction? 2020, Medium. viewed 3 November 2020, <https://medium.com/vsinghbisen/how-sentiment-analysis-in-stock-market-used-for-right-prediction-5c1bfe64c233>
14. areed1192/sigma\_coding\_youtube 2020, GitHub. viewed 3 November 2020, [https://github.com/areed1192/sigma\\_coding\\_youtube/blob/master/python/python-data-science/machine-learning/k-means/Clustering%20Stocks%20-%20KMeans.ipynb](https://github.com/areed1192/sigma_coding_youtube/blob/master/python/python-data-science/machine-learning/k-means/Clustering%20Stocks%20-%20KMeans.ipynb)