CMPE 257 – MID Term Exam

Group: Musketeers

Members: Raghava, Shiv, Kumuda

1. Use the yahoo finance link to get data, use other sources

We have used the following data sources:

1) For initial investigation and analysis, we will be using the data from [Kaggle stock Dataset](). This Dataset has around 5800+ stock symbols.

2) We pulled the stock cap details from Nasdaq stock screener [https://www.nasdaq.com/market-activity/stocks/screener].

3) We explored Yahoo finance API for historical stock data and subsided with Kaggle after comparing the output are the same.

4) We web scrapped yahoo financial site using BeautifulSoup to get dividend and earning details of the stocks

Web Scraping

```
from bs4 import BeautifulSoup
import requests

divArr =[]
sectorArr = []
for ticker in golden_cluster1['Ticker'].unique():
  response = requests.get('https://finance.yahoo.com/quote/'+ticker+'?p='+ticker)
  soup = BeautifulSoup(response.text, 'html.parser')
  k = soup.find_all("td", {"class":  "Ta(end) Fw(600) Lh(14px)"})[13].text.split('(')[0]
  divArr.append(k)

  try:
    response = requests.get('https://finance.yahoo.com/quote/'+ticker+'/profile?p='+ticker)
    soup = BeautifulSoup(response.text, 'html.parser')
    k = soup.find_all("p", {"class":  "D(ib) Va(t)"})[0]
    sector_kk = k.find_all("span", {"class": "Fw(600)"})[0].text
    sectorArr.append(sector_kk)
  except:
    sectorArr.append('NA')

scrappedDf = pd.DataFrame({'Ticker':golden_cluster1['Ticker'].unique(), 'Div_Val':divArr, 'Sector':sectorArr})
scrappedDf.head()
```

All the above datasets can be found in the following drive location:
https://drive.google.com/drive/u/1/folders/1SXxkjKv2zRg7ojx7XCqGY6Mwvi1jOlXN

Note: Due to the size of the dataset we have to mount the google drive before we can execute the collab.

The data wrangling and amalgamation were performed to get the final dataset on which we performed our analysis.

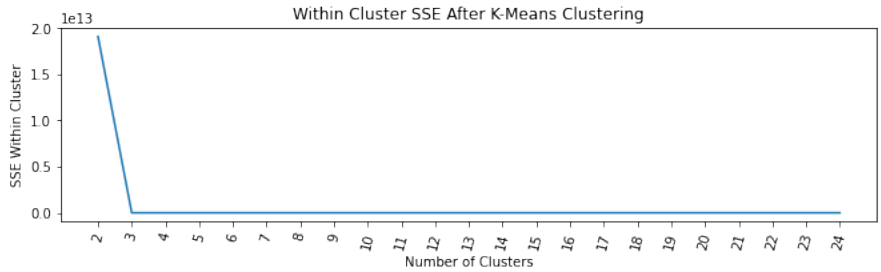
2. Use fractal clustering to Identify the golden cluster for investment of stock based on

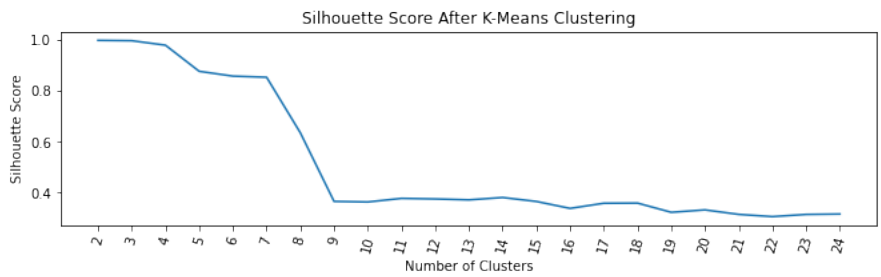
1. **Sharpe values**
2. **Risk**
3. **Return on investment**

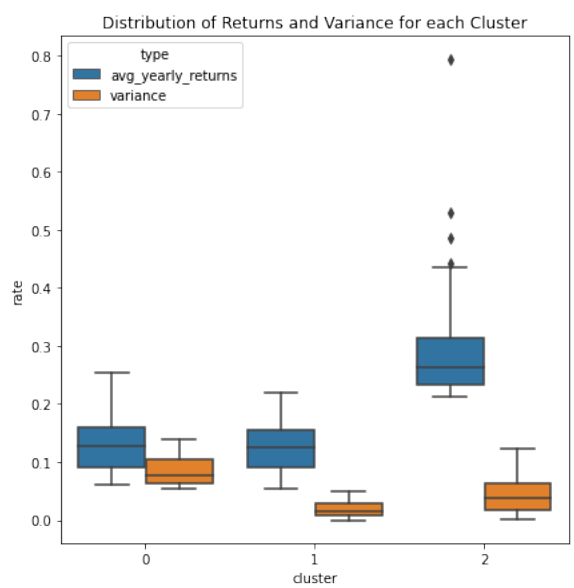
We generated the couple of latent variables

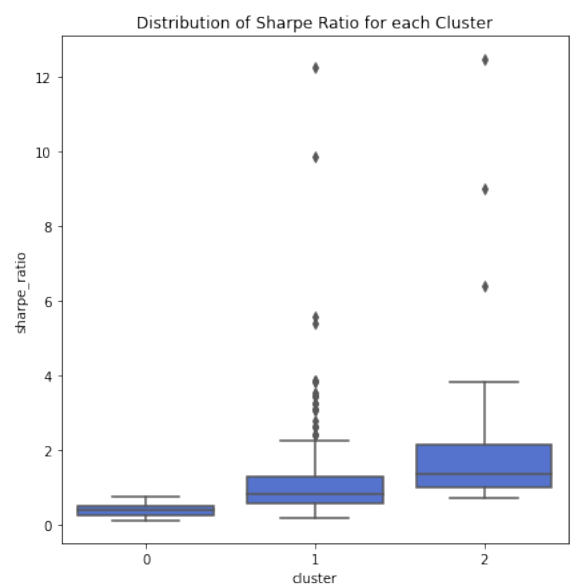
- Yearly Returns
- Average yearly return
- Overall variance

Using these we performed KMeans and calculated Silhouette scores, sharpe ratio for generating the **Golden Cluster.**





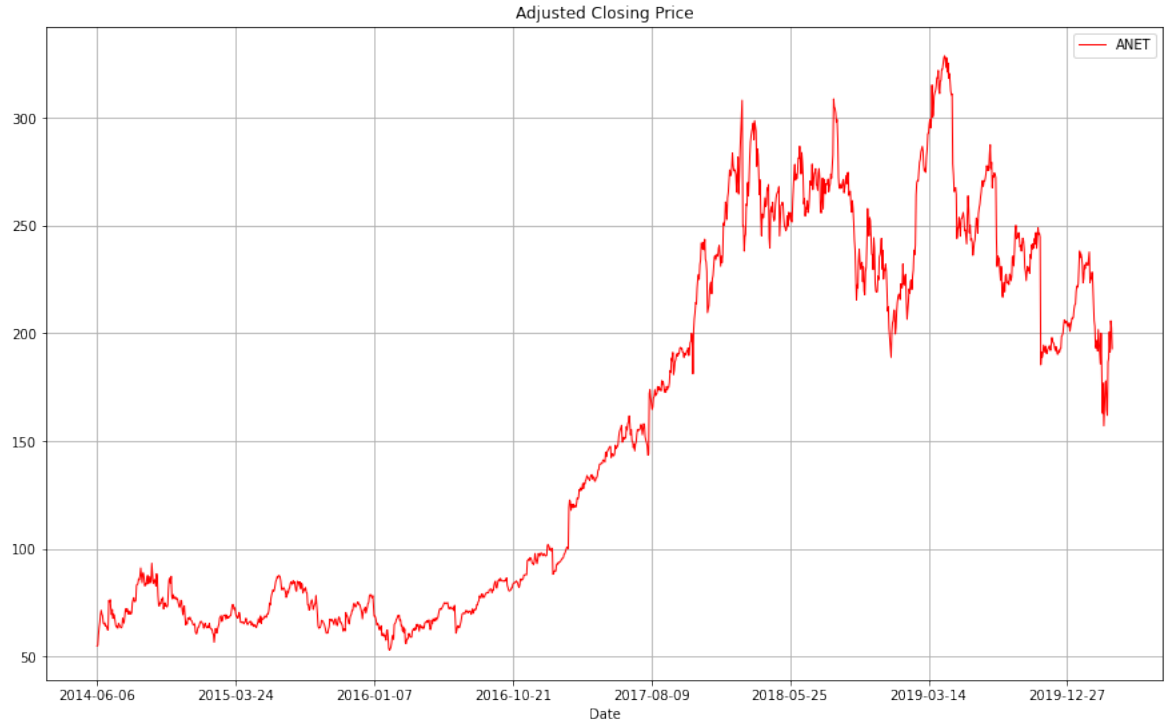


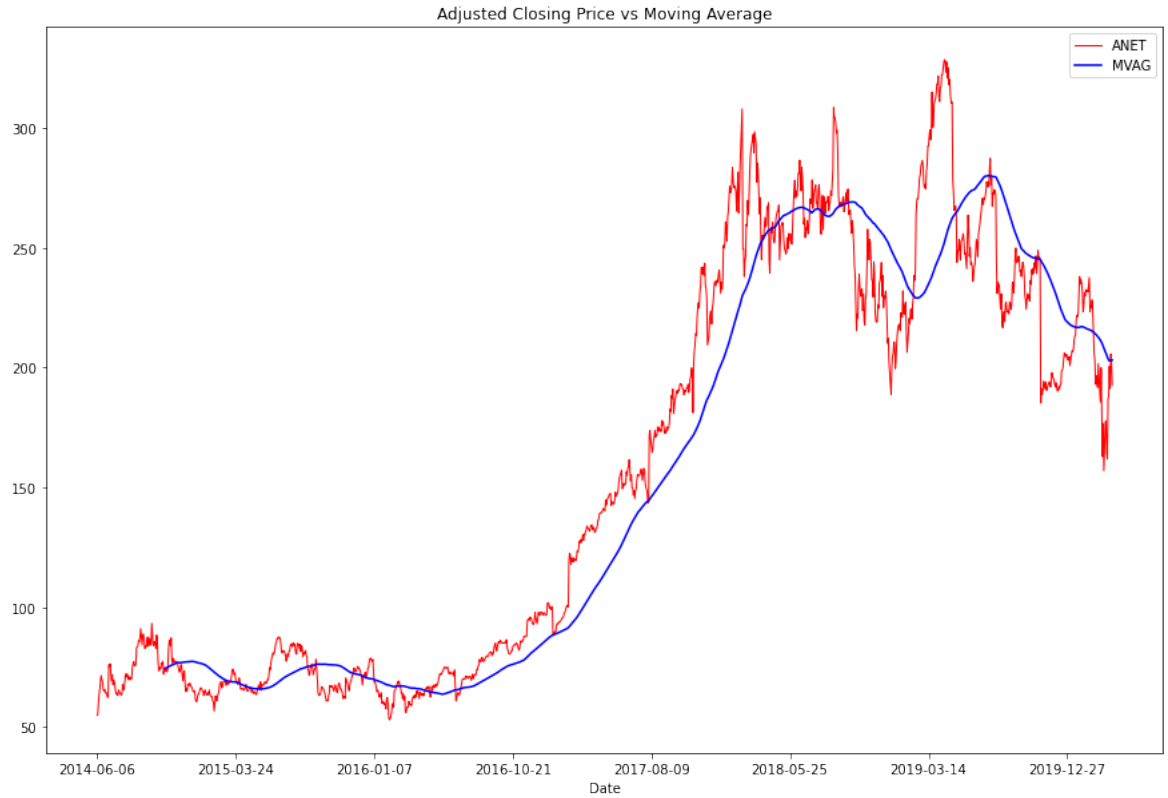


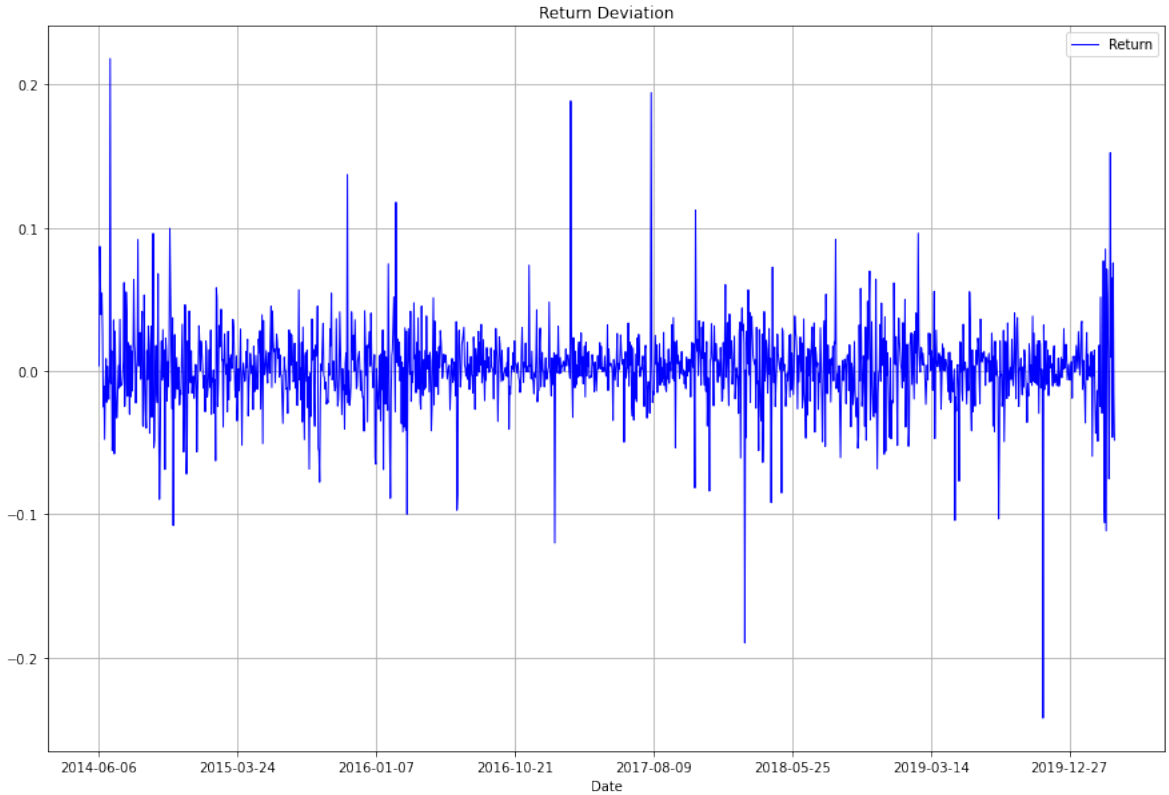
We had performed 3 iterations but decided with the second iteration cluster as our golden cluster.

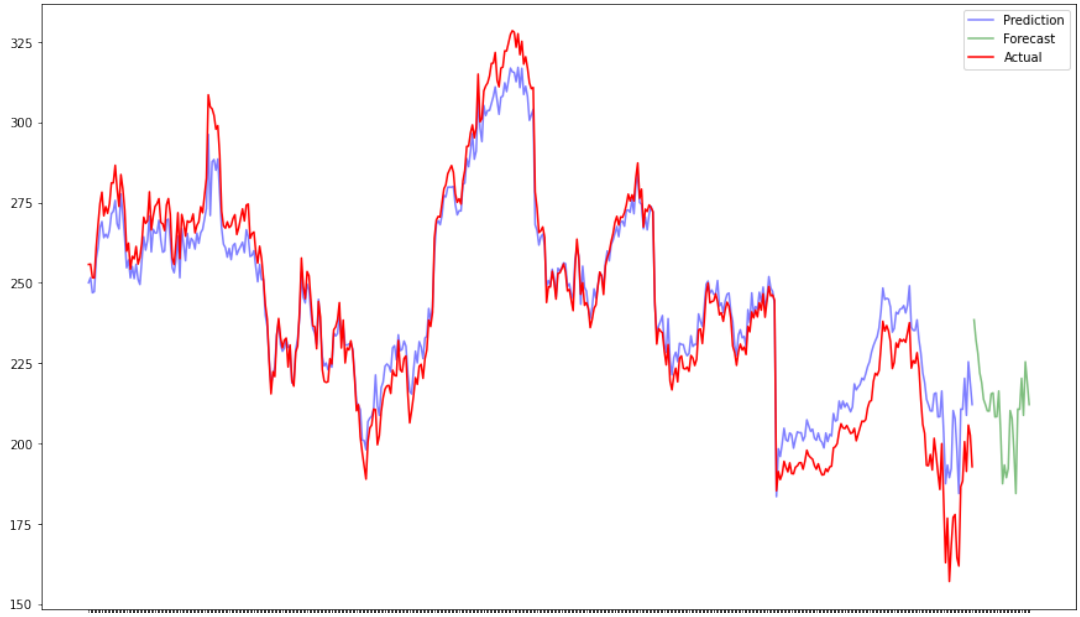
3. Predict the value of a investments in the golden cluster

We picked a stock from Golden Cluster and performed Regression (using Muller Loop) in order to determine the return over the investment for “EQIX” {Equinix Inc} which works in real estate sector and “ANET” {Arista Network} which is in Technology domain.









Muller Loop without Latent Variables:

	MODEL	R2_SCORE	MSE	MAE	TIME
0	Decision Tree Regressor	95.336211	344.057049	12.510127	0.011348
1	Random Forest	95.471560	334.072091	12.060435	0.043782
2	MLP Regressor	89.929573	742.915547	19.125206	0.728509
3	AdaBoost Regressor	94.618149	397.029934	14.248729	0.068027
4	Kernel Ridge	91.394216	634.865930	17.816332	0.107927
5	GradientBoostingRegressor	95.562819	327.339716	12.335880	0.310055
6	RandomForestRegressor	96.136038	285.052205	11.092476	0.643491
7	LinearRegression	91.725243	610.445416	17.272294	0.003845
8	LGBMRegressor	95.458362	335.045700	12.229442	0.181994

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=1, verbose=0, warm_start=False)
Regresssor with better R2 score= RandomForestRegressor, Score (test, accuracy) = 96.14
```

Muller Loop with Latent Variables:

	MODEL	R2_SCORE	MSE	MAE	TIME
0	Decision Tree Regressor	9.748127e+01	1.940232e+02	8.804992	0.020332
1	Random Forest	9.809689e+01	1.466005e+02	7.957862	0.096373
2	MLP Regressor	-1.168491e+06	9.001891e+07	5931.443165	2.335185
3	AdaBoost Regressor	9.796854e+01	1.564877e+02	9.477848	0.357950
4	Kernel Ridge	9.357071e+01	4.952614e+02	15.847667	0.664339
5	GradientBoostingRegressor	9.930949e+01	5.319143e+01	5.010679	0.812878
6	RandomForestRegressor	9.951567e+01	3.730877e+01	3.513178	1.561080
7	LinearRegression	9.496179e+01	3.881034e+02	14.278542	0.007741
8	LGBMRegressor	9.957934e+01	3.240437e+01	3.556725	0.410550

```
LGBMRegressor(bagging_fraction=0.8, boosting_type='gbdt', class_weight=None,
              colsample_bytree=1.0, feature_fraction=0.8,
              importance_type='split', learning_rate=0.17, max_depth=-1,
              metric='rmse', min_child_samples=20, min_child_weight=0.001,
              min_split_gain=0.0, n_estimators=100, n_jobs=-1, num_leaves=1200,
              objective='regression', random_state=None, reg_alpha=0.0,
              reg_lambda=0.9, silent=True, subsample=1.0,
              subsample_for_bin=200000, subsample_freq=0)
Regresssor with better R2 score= LGBMRegressor, Score (test, accuracy) = 99.58
```

We noticed that when we performed the regression using latent variables the performance increased.

Model Persistence:

Pickle files were generated for the best performing model in both muller loops, the same can be found in the following drive location:

https://drive.google.com/drive/u/1/folders/1UGA6PsWKIE151i5_5xwbz2sI7WwwVhHa

Pickle file for regression without latent variable:

```
regressor=load('/content/drive/MyDrive/MLSpring-2021/MLSpring-2021/teams/musketeers/Midterm/PickleModel/regressor_WOLV_RandomForestRegressor')
regressor.predict(X_test)
```

```
array([ 67.2051  , 189.502101, 239.729401, 232.517499, ..., 170.039604, 133.725702, 246.5369  ,  74.071702])
```

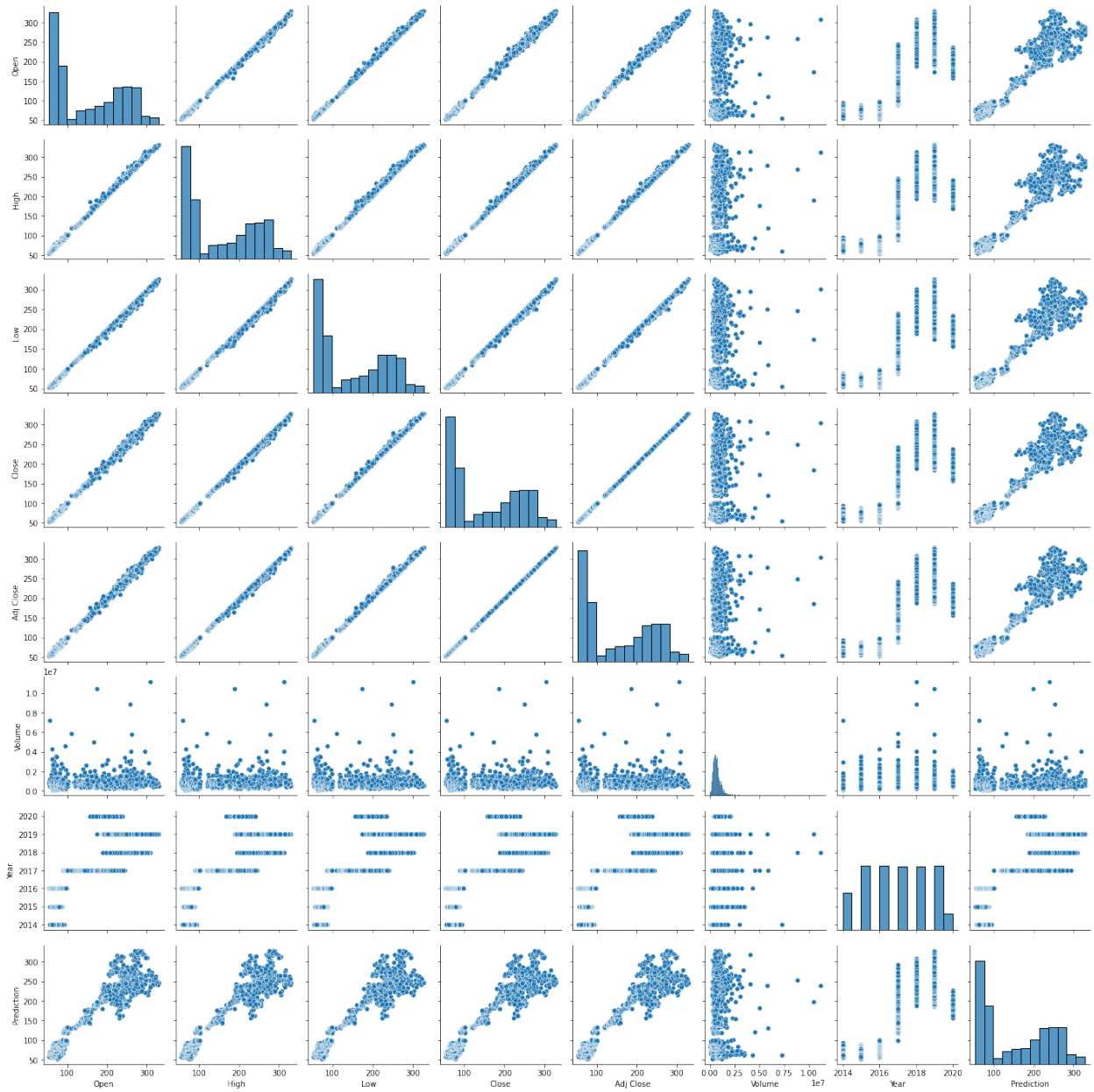
Pickle file for regression with latent variable:

```
regressorlv=load('/content/drive/MyDrive/MLSpring-2021/MLSpring-2021/teams/musketeers/Midterm/PickleModel/regressor_WLV_LGBMRegressor'
regressorlv.predict(X_test)
```

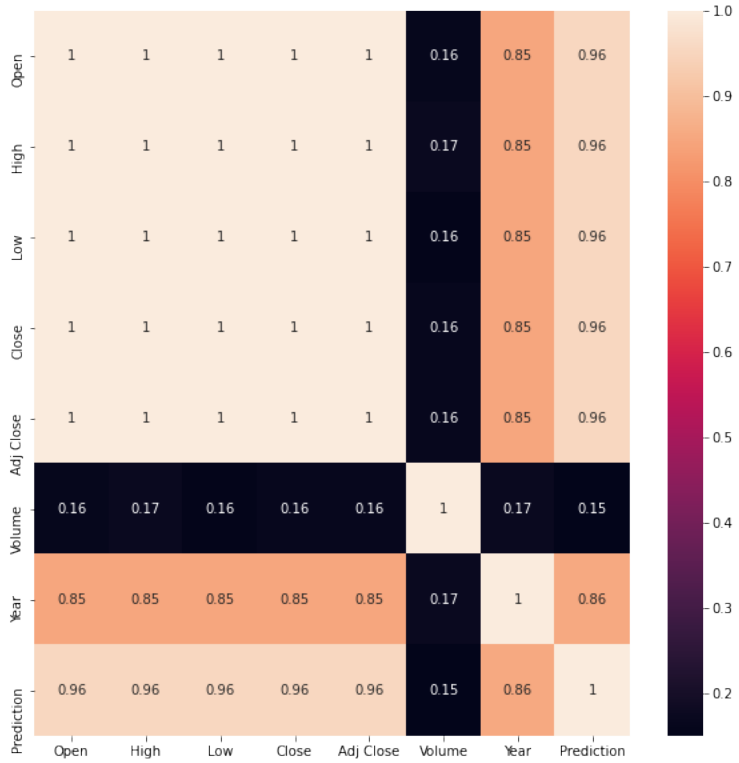
```
array([246.575075, 242.599456, 242.315397, 206.189105, ..., 260.202791, 240.34868 , 209.774981, 271.100447])
```

4. Data Visualization

Visualized the stock data before addition of latent variables

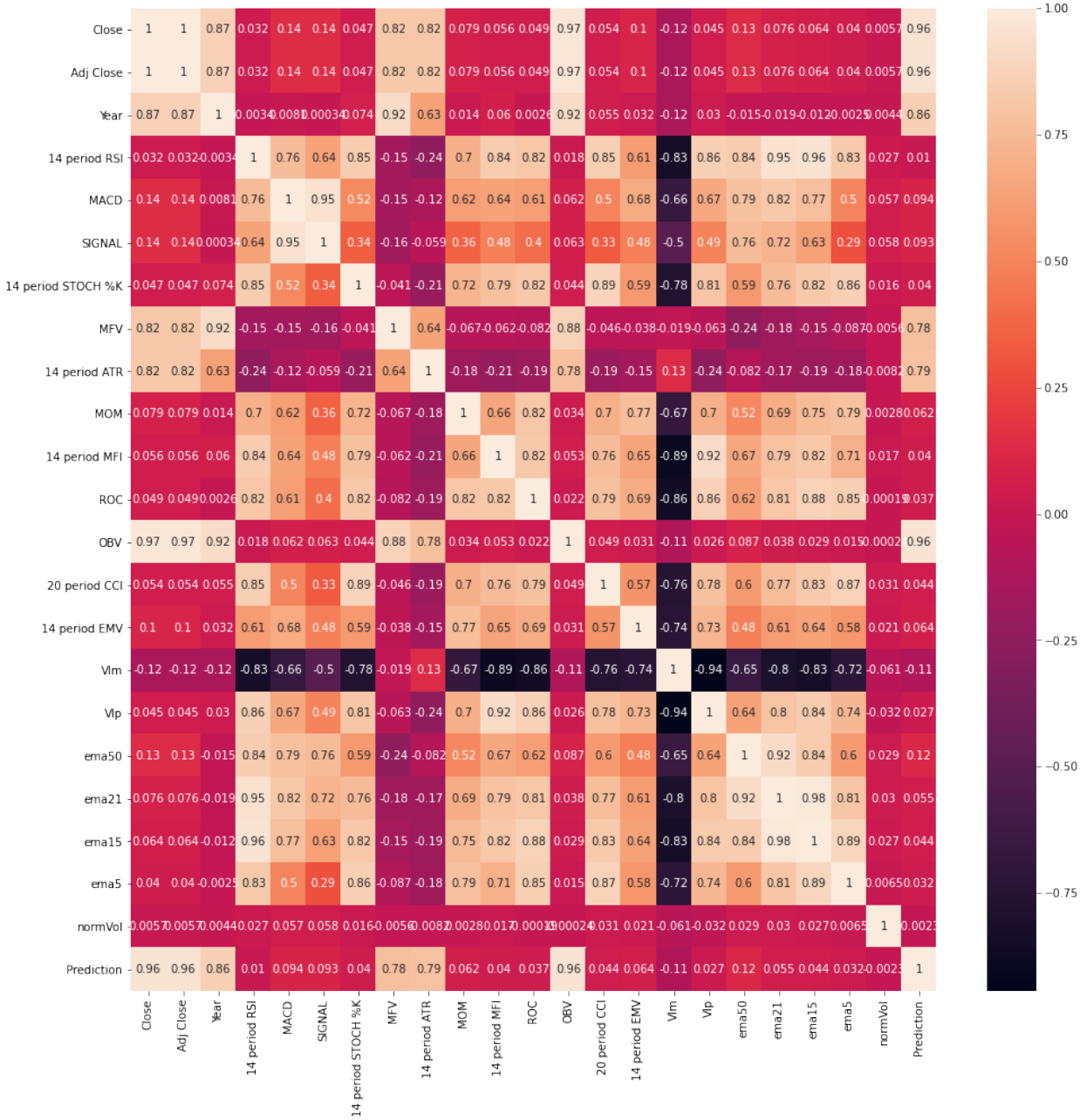


Correlation matrix before addition of latent variables:



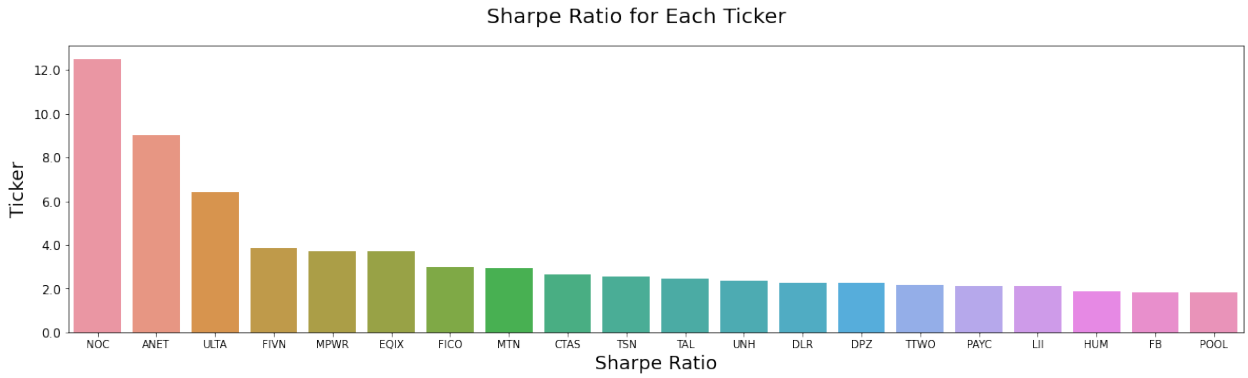
We noticed that the attributes open, high, low, close and adj close were highly correlated. Thus we dropped open, high, low attributes.

Correlation matrix once the latent variables were added.

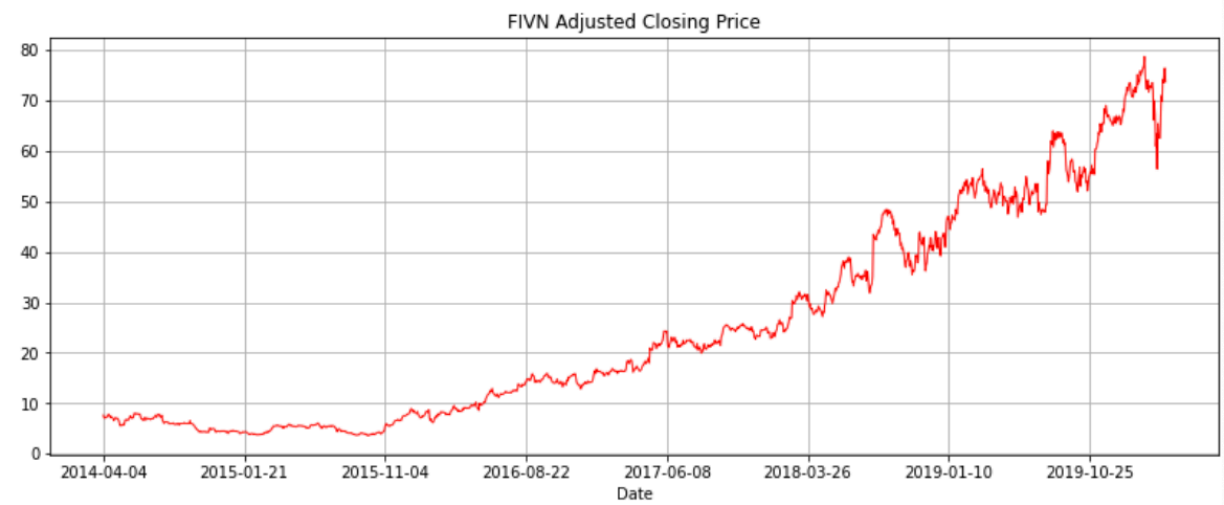


Using the golden cluster of stocks, show the market trends, Using this market trends highlight the low, medium and high value stocks.

Using the golden cluster, we picked out 8 stocks with highest sharpe score and performed our analysis.

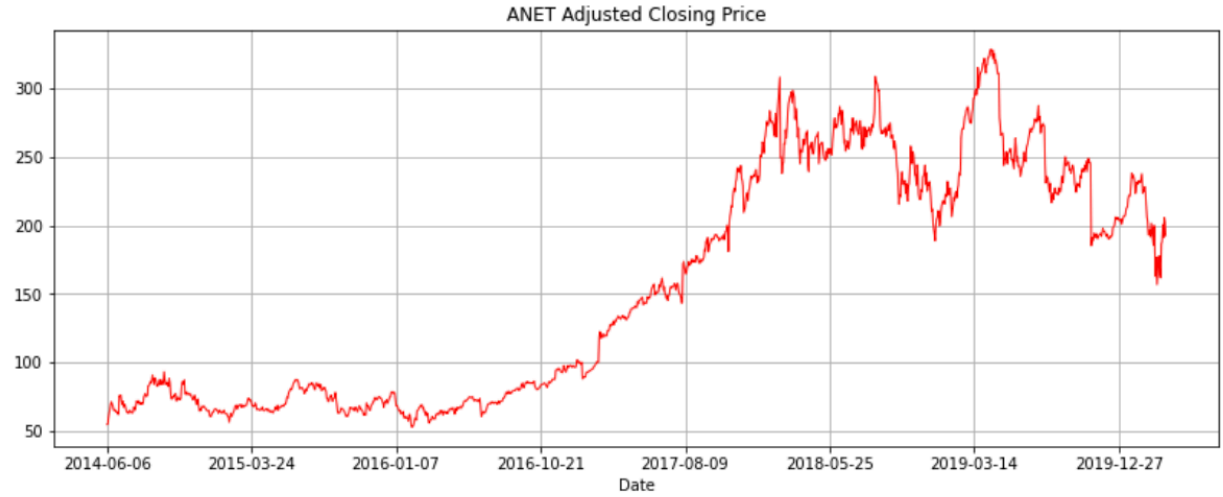


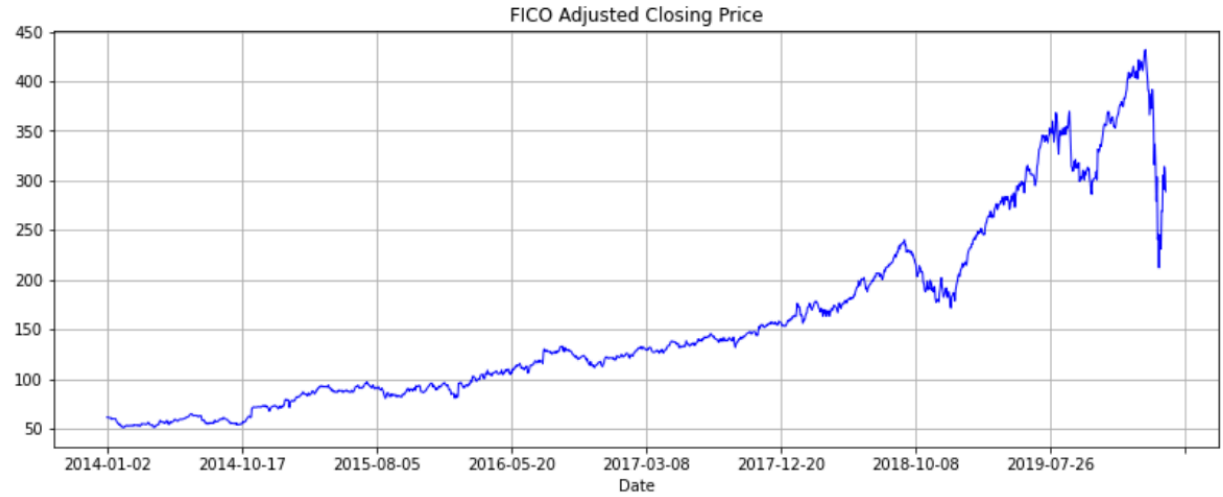
High



As we can see from the above graph the trend seems to be uptrend for a period of 5 years. Thus these stocks can be considered for long term investment which will return consistent high values.

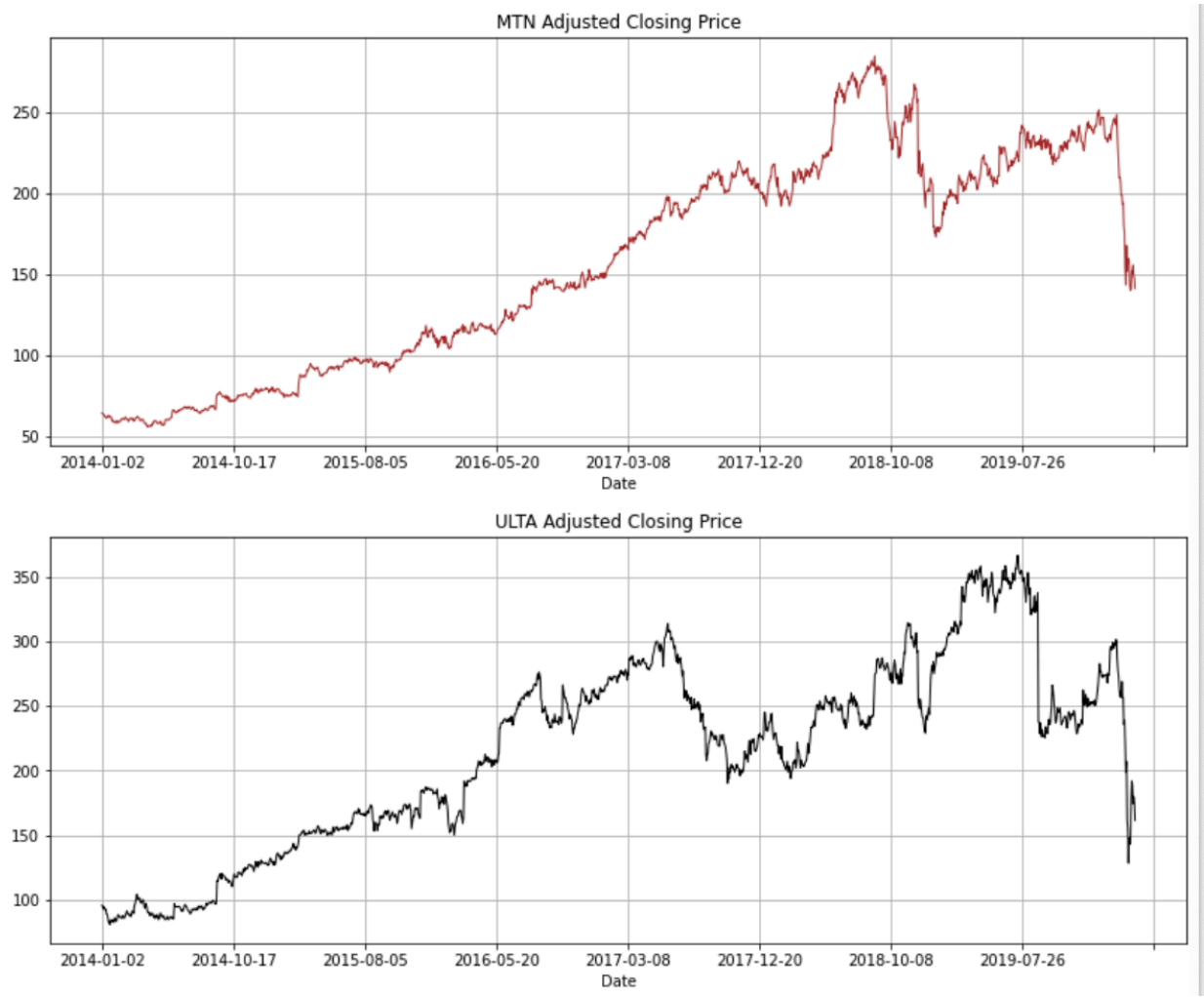
Medium:





As we can see from the above graph the trend seems to be sideways trend for last 2 years. The support and resistance price are within a band; thus, we can say these are medium valued stocks and may provide return over investment, but expectation cannot be too high.

Low:

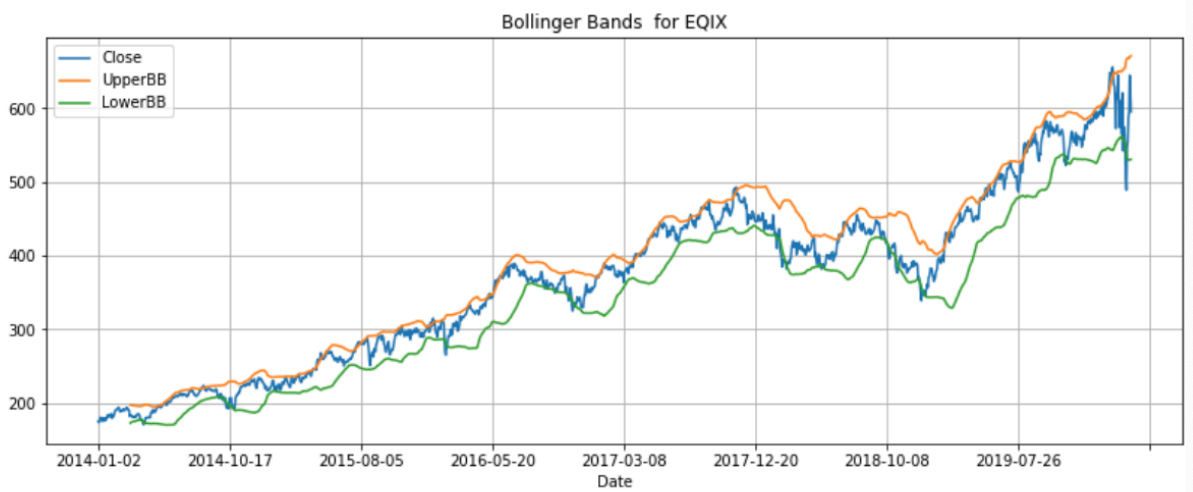


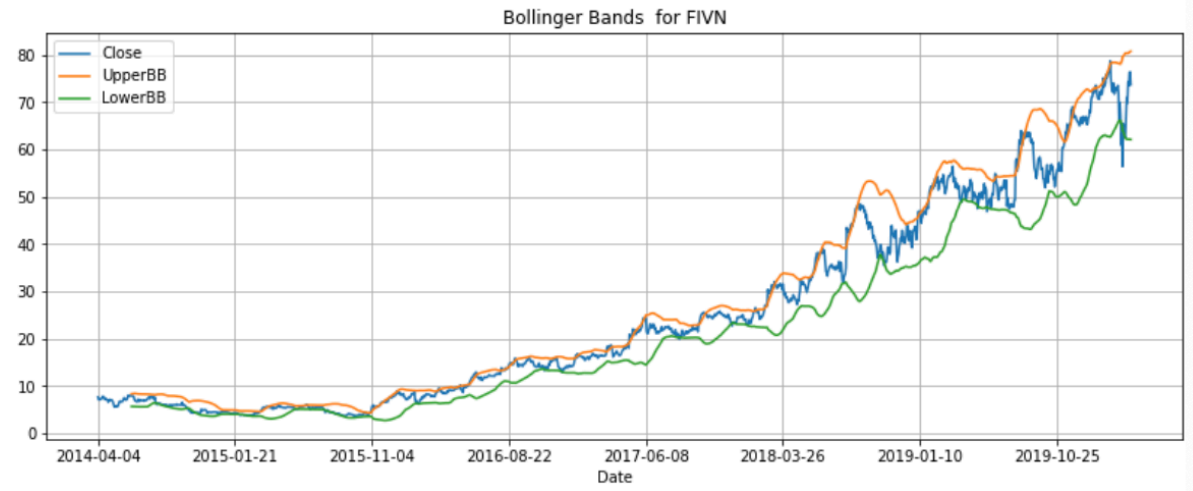
As we can see from the above graph we can see that there has been high volatility for past 3 years due to which these stocks have broken the support barrier multiple times and does not provide enough confidence for long term investment.

Based on the data derive the following:

1. **Most popular stocks that investors buy**
2. **Average investor interested stocks**
3. **Less investor interested stocks**

Using sharpe ratio we can recommend the following.
Most popular stocks: NOC, ANET, ULTA, FIVN, MPWR, EQIX
Average investor interested stocks: FICO, MTN, CTAS, TSN, TAL
Less investor interested stocks: UNH, DLR, DPZ, TTWO





5. Scrape the earnings-related information / latent variables [choose 2 out of 5 factors/latent variables]

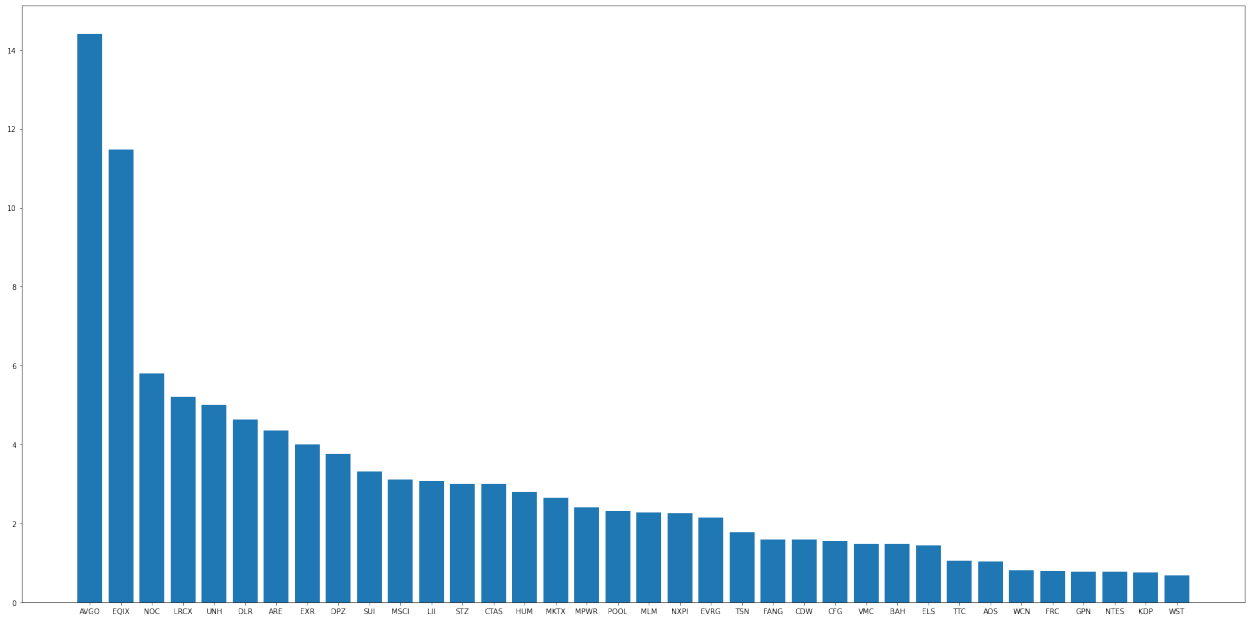
. How close to earnings did stock value change

a. Is this a stock to hold long term or a shorter-term stock investment?

    1. Will the value go steadily higher if you hold on and ride the ups and down?
    2. Or is it a stock that constantly goes up and down and you should buy on the lows and sell on the highs rather than hold long term?

b. Which other companies does this company work with? Show 2 of the other companies and how their stocks vary over time

c. Which are the highest dividend yield stocks in the golden cluster? . Many stock pay a "dividend" a return percentage based on how much of their stock you have



From the above we can see that the highest yielding stocks are as follows:-

1. AVGO - **14.40**
2. EQIX - **11.48**
3. NOC - **5.80**
4. LRCX - **5.20**
5. UNH - **5.00**
6. DLR - **4.64**
7. ARE - **4.36**
8. EXR - **4.00**

AVGO, LRCX, MPWR, NXPI, CWD are the companies that are somehow related due to the fact they work in same technology sector. We can get this from the data mentioned below.

6. Pick a stock from the golden cluster . If you buy it today how much will the price change (profit or loss, increase or decrease) after 2 years of holding on to it ? Discuss your answer if you cannot compute it directly.

We can take the yearly return of a particular stock and using its historic data we can perform the forecasting of the stock price. For our analysis we have used regression models to perform this forecasting. The details on the regression model can be found in the collab file.

---

1. **business case and value-- what hypotheses are you trying to prove? see above questions**
    1. **data narrative: tell a story to guide an investor or a retiree or just someone interested in buying...**

We are trying to identify the high yielding stocks in the market which can serve as long term investment. We are also trying to identify the high dividend paying stocks so that we can have a mix and match of the two above.

To assist user in making decision of buy, sell or hold a stock we have created a classification model with high confidence score.

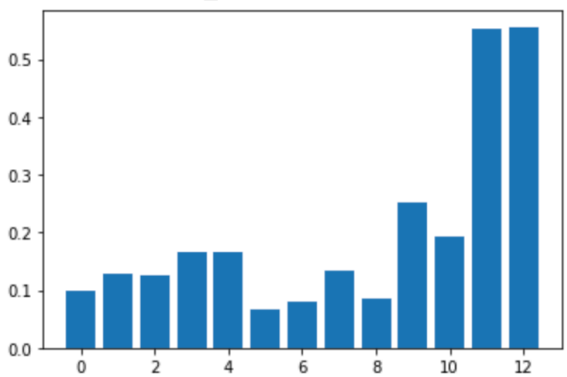
Also, we have created a forecasting model to predict the stock price for future days.

2. **visualizations, of data prep using first data enrichment (add dataset to base data set)**
    1. **feature importance; gini score**

Data set details have been mentioned above, following are the data enrichment process performed.

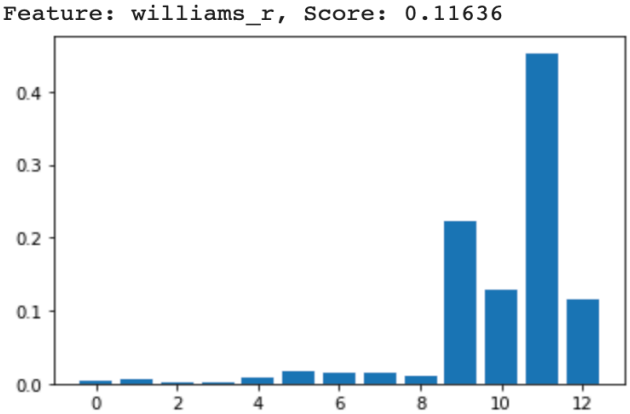
SelectKBest Method:

```
Feature Open: 0.09821461762389894
Feature High: 0.1278054672113058
Feature Low: 0.12479437453103892
Feature Close: 0.1658298094988917
Feature Adj Close: 0.16649143346683504
Feature Volume: 0.06710456952946031
Feature bb_bbm: 0.07942955400902285
Feature bb_bbh: 0.13298329946030818
Feature bb_bbl: 0.08585779531477833
Feature bb_bbhi: 0.25266472381605776
Feature bb_bbli: 0.19232859115628176
Feature rsi: 0.5517256261896075
Feature williams_r: 0.5558026868244808
```

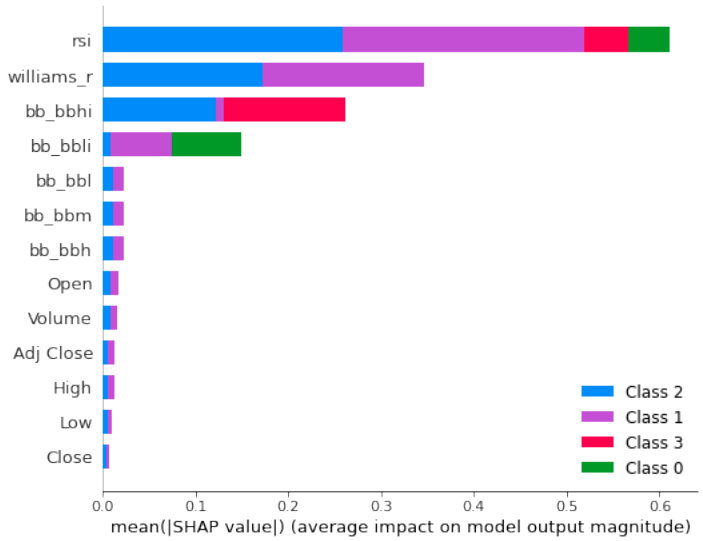


Decision Tree Classifier:

```
Feature: Open, Score: 0.00470
Feature: High, Score: 0.00673
Feature: Low, Score: 0.00090
Feature: Close, Score: 0.00131
Feature: Adj Close, Score: 0.00770
Feature: Volume, Score: 0.01781
Feature: bb_bbm, Score: 0.01407
Feature: bb_bbh, Score: 0.01410
Feature: bb_bbl, Score: 0.01151
Feature: bb_bbhi, Score: 0.22316
Feature: bb_bbli, Score: 0.12864
Feature: rsi, Score: 0.45300
Feature: williams_r, Score: 0.11636
```



SHAP value:



3. Data Prep
    1. feature transformation;
        1. transform features, add new features to dataset via amalgamations (see below) , compare results with original
        2. data distribution
    2. include **dim reduction via**: PCA

We have performed the feature engineering to clean the data after the data merge and filtered the data by first and last day of the year to calculate features like, yearly returns, avg yearly returns, over all variances. The details can be found in **Data Manipulation / Processing** section of Collab.

4. Implement 3 amalgamations:
    1. first data set
    2. second data enrichment -- get an amalgamation; each enrichment enables you to implement / use more algorithms as needed.
    3. Third data enrichment -- scrape data from a source and amalgamate

We have used the following data sources:

1) For initial investigation and analysis, we will be using the data from [Kaggle stock Dataset]. This Dataset has around 5800+ stock symbols.

2) We pulled the stock cap details from Nasdaq stock screener [https://www.nasdaq.com/market-activity/stocks/screener].

3) We explored Yahoo finance API for historical stock data and subsided with Kaggle after comparing the output are the same.

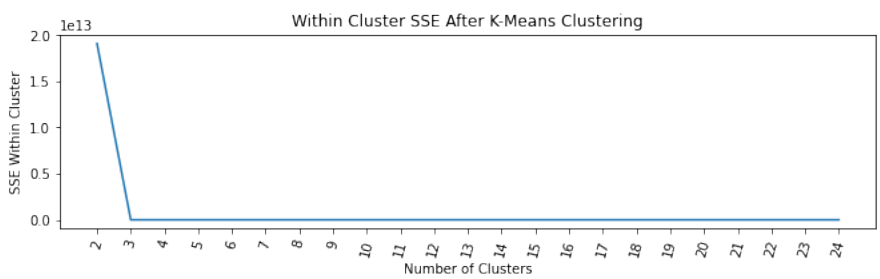
4) We web scrapped yahoo financial site using BeautifulSoup to get dividend and earning details of the stocks

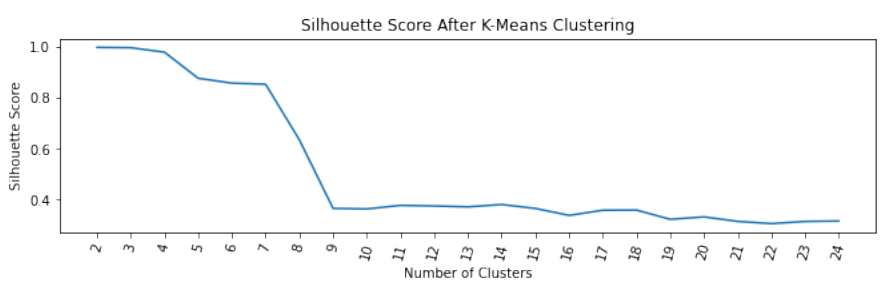
All the above datasets can be found in the following drive location:
https://drive.google.com/drive/u/1/folders/1SXxkjKv2zRg7ojx7XCqGY6Mwvi1jOlXN

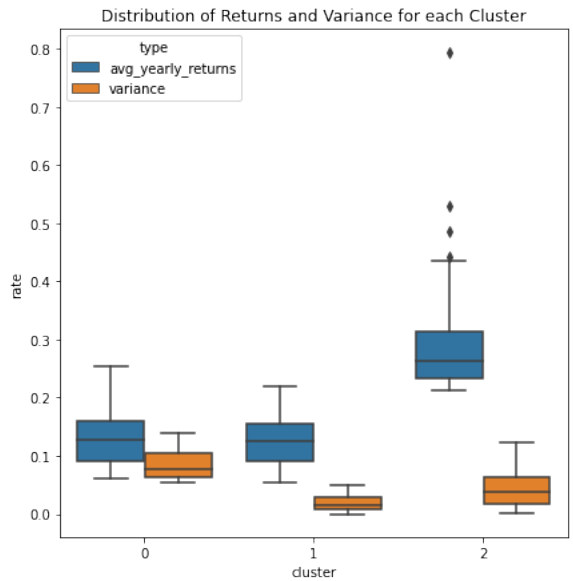
The data wrangling and amalgamation were performed to get the final dataset on which we performed our analysis.

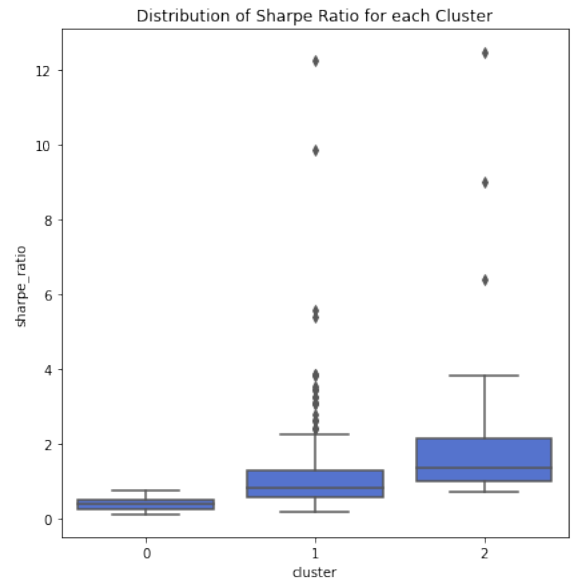
5. implement ml algorithms to build models and refine your data narrative
    1. Define a Golden cluster and use Fractal Clustering to find it based on the business case you formulate
    2. train and Apply algorithms : you can use the muller loop
        1. **classify**: Muller, at least 5 algos
        2. **Regression**: Muller, at least 7 algos
    3. Compare relevant tasks in the same table.
        1. Write a data narrative to interpret results of each algorithm

- Fractal Clustering was achieved using KMeans and Silhouette scores for generating the Golden Cluster









We had performed 3 iterations but decided with the second iteration cluster as our golden cluster.

- Classification algorithms: Using this algorithm we are able to classify if the stock is a buy, sell or hold.

Classification performed without latent variables:

Model	Accuracy	F1 Score	Precision	Recall
Linear SVM	93.10344827586206	0.9599688473520249	0.9612155388471177	0.9589635507066699
RBF SVM	91.0344827586207	0.9001831501831501	0.9553615422517862	0.8589360986149978
Decision Tree	93.79310344827586	0.9637687969924812	0.9667904290429042	0.9617375402925862
Random Forest	91.72413793103448	0.9367912703906998	0.9557704402515723	0.9204295219203475
Neural Net	94.48275862068965	0.96797507788162	0.9692669172932331	0.9669290850483512
AdaBoost	92.41379310344827	0.9554438280166435	0.9605509020618557	0.9525632283659806
Naive Bayes	93.79310344827586	0.9641550954624349	0.9641550954624349	0.9641550954624349

Best performing model was MLPClassifier (Neural Net) with 94.4% accuracy

Classification performed with latent variables:

Model	Accuracy	F1 Score	Precision	Recall
Linear SVM	94.13793103448276	0.9664594664594665	0.9678991905813098	0.9653897442114195
RBF SVM	94.48275862068965	0.9683180428134557	0.9709956709956711	0.9666020123651351
Decision Tree	92.75862068965517	0.9583645911477869	0.9613833746898263	0.9565401866892956
Random Forest	93.44827586206897	0.9622304090874203	0.9663107281129013	0.9599648442235422
Neural Net	95.17241379310344	0.9724099099099098	0.9735321100917431	0.9715268517396048
AdaBoost	90.6896551724138	0.9463274234400184	0.9501373797926813	0.9442659716329251
Naive Bayes	94.13793103448276	0.9665349243748576	0.9673123856009762	0.9658898048248272

Best performing model was MLPClassifier (Neural Net) with 95.1% accuracy

With the induction of latent variables, we noticed that there was a slight increase in performance.

- Regression algorithms: using these models we are able to forecast / predict the stock price for future days.

Regression without latent variables:

	MODEL	R2_SCORE	MSE	MAE	TIME
0	Decision Tree Regressor	95.336211	344.057049	12.510127	0.011348
1	Random Forest	95.471560	334.072091	12.060435	0.043782
2	MLP Regressor	89.929573	742.915547	19.125206	0.728509
3	AdaBoost Regressor	94.618149	397.029934	14.248729	0.068027
4	Kernel Ridge	91.394216	634.865930	17.816332	0.107927
5	GradientBoostingRegressor	95.562819	327.339716	12.335880	0.310055
6	RandomForestRegressor	96.136038	285.052205	11.092476	0.643491
7	LinearRegression	91.725243	610.445416	17.272294	0.003845
8	LGBMRegressor	95.458362	335.045700	12.229442	0.181994

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=1, verbose=0, warm_start=False)
Regresssor with better R2 score= RandomForestRegressor, Score (test, accuracy) = 96.14
```

Without latent variable RandomForestRegressor has the best performance with 96.14% accuracy.

Regression with latent variables:

	MODEL	R2_SCORE	MSE	MAE	TIME
0	Decision Tree Regressor	9.748127e+01	1.940232e+02	8.804992	0.020332
1	Random Forest	9.809689e+01	1.466005e+02	7.957862	0.096373
2	MLP Regressor	-1.168491e+06	9.001891e+07	5931.443165	2.335185
3	AdaBoost Regressor	9.796854e+01	1.564877e+02	9.477848	0.357950
4	Kernel Ridge	9.357071e+01	4.952614e+02	15.847667	0.664339
5	GradientBoostingRegressor	9.930949e+01	5.319143e+01	5.010679	0.812878
6	RandomForestRegressor	9.951567e+01	3.730877e+01	3.513178	1.561080
7	LinearRegression	9.496179e+01	3.881034e+02	14.278542	0.007741
8	LGBMRegressor	9.957934e+01	3.240437e+01	3.556725	0.410550

```
LGBMRegressor(bagging_fraction=0.8, boosting_type='gbdt', class_weight=None,
              colsample_bytree=1.0, feature_fraction=0.8,
              importance_type='split', learning_rate=0.17, max_depth=-1,
              metric='rmse', min_child_samples=20, min_child_weight=0.001,
              min_split_gain=0.0, n_estimators=100, n_jobs=-1, num_leaves=1200,
              objective='regression', random_state=None, reg_alpha=0.0,
              reg_lambda=0.9, silent=True, subsample=1.0,
              subsample_for_bin=200000, subsample_freq=0)
Regresssor with better R2 score= LGBMRegressor, Score (test, accuracy) = 99.58
```

We notice that with latent variables the performance of the model was improved.
LGBMRegressor had the highest accuracy if 99.58%.
With these regression models we are able to predict stock price for future dates.

Pickle files were generated for the best performing model in both muller loops, the same can be found in the following drive location:

https://drive.google.com/drive/u/1/folders/1UGA6PsWKIE151i5_5xwbz2sI7WwwVhHa

Pickle file for regression without latent variable:

```
regressor=load('/content/drive/MyDrive/MLSpring-2021/MLSpring-2021/teams/musketeers/Midterm/PickleModel/regressor_WOLV_RandomForestRegressor')
regressor.predict(X_test)
```

```
array([ 67.2051  , 189.502101, 239.729401, 232.517499, ..., 170.039604, 133.725702, 246.5369  ,  74.071702])
```

Pickle file for regression with latent variable:

```
regressorlv=load('/content/drive/MyDrive/MLSpring-2021/MLSpring-2021/teams/musketeers/Midterm/PickleModel/regressor_WLV_LGBMRegressor'
regressorlv.predict(X_test)
```

```
array([246.575075, 242.599456, 242.315397, 206.189105, ..., 260.202791, 240.34868 , 209.774981, 271.100447])
```

Pickle file for classification without latent variable.

```
classifier= load('/content/drive/MyDrive/MLSpring-2021/MLSpring-2021/teams/musketeers/Midterm/PickleModel/classifier_WLV_Decision Tree')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
prediction = classifier.predict(X_test)
print("Probability ", prediction)
```

```
Probability  [2 2 2 
```

```
classifier= load('/content/drive/MyDrive/MLSpring-2021/MLSpring-2021/teams/musketeers/Midterm/PickleModel/classifier_WLV_Decision Tree')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
prediction = classifier.predict(X_test)
print("Probability ", prediction)
```

```
Probability  [2 2 2 2 ... 2 2 2 2]
```

6. Suggest Latent Variables or Latent Manifolds, add then to the features and see how prediction results change

We have generated the following latent variables

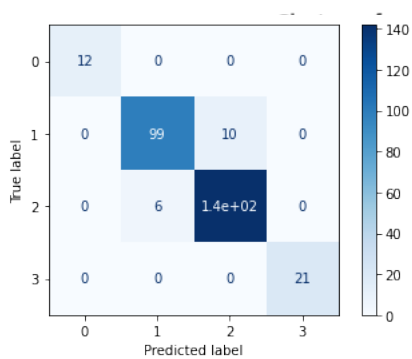
- Bollinger band
- Bollinger Indicator
- RSI
- williams_r momentum

We have performed both classification and regression with and without latent variables. As seen above with the addition of latent variables the performance of the models improved.

7. use appropriate metrics for measuring models and compare them in a table: regression metrics and/or classification metrics (confusion matrix, f1)

Used R2 score for evaluating regression models and confusion matrix with F1 score for classification models.

```
Classifier = Neural Net, Score (test, accuracy) = 94.48, Training time = 1.86 seconds
```



The detailed confusion matrix for all the classification models can be found in the collab.

Classification performed without latent variables:

Model	Accuracy	F1 Score	Precision	Recall
Linear SVM	93.10344827586206	0.9599688473520249	0.9612155388471177	0.9589635507066699
RBF SVM	91.0344827586207	0.9001831501831501	0.9553615422517862	0.8589360986149978
Decision Tree	93.79310344827586	0.9637687969924812	0.9667904290429042	0.9617375402925862
Random Forest	91.72413793103448	0.9367912703906998	0.9557704402515723	0.9204295219203475
Neural Net	94.48275862068965	0.96797507788162	0.9692669172932331	0.9669290850483512
AdaBoost	92.41379310344827	0.9554438280166435	0.9605509020618557	0.9525632283659806
Naive Bayes	93.79310344827586	0.9641550954624349	0.9641550954624349	0.9641550954624349

Best performing model was MLPClassifier (Neural Net) with 94.4% accuracy

Classification performed with latent variables:

Model	Accuracy	F1 Score	Precision	Recall
Linear SVM	94.13793103448276	0.9664594664594665	0.9678991905813098	0.9653897442114195
RBF SVM	94.48275862068965	0.9683180428134557	0.9709956709956711	0.9666020123651351
Decision Tree	92.75862068965517	0.9583645911477869	0.9613833746898263	0.9565401866892956
Random Forest	93.44827586206897	0.9622304090874203	0.9663107281129013	0.9599648442235422
Neural Net	95.17241379310344	0.9724099099099098	0.9735321100917431	0.9715268517396048
AdaBoost	90.6896551724138	0.9463274234400184	0.9501373797926813	0.9442659716329251
Naive Bayes	94.13793103448276	0.9665349243748576	0.9673123856009762	0.9658898048248272

Best performing model was MLPClassifier (Neural Net) with 95.1% accuracy

With the induction of latent variables, we noticed that there was a slight increase in performance.

- Regression algorithms: using these models we are able to forecast / predict the stock price for future days.

Regression without latent variables:

	MODEL	R2_SCORE	MSE	MAE	TIME
0	Decision Tree Regressor	95.336211	344.057049	12.510127	0.011348
1	Random Forest	95.471560	334.072091	12.060435	0.043782
2	MLP Regressor	89.929573	742.915547	19.125206	0.728509
3	AdaBoost Regressor	94.618149	397.029934	14.248729	0.068027
4	Kernel Ridge	91.394216	634.865930	17.816332	0.107927
5	GradientBoostingRegressor	95.562819	327.339716	12.335880	0.310055
6	RandomForestRegressor	96.136038	285.052205	11.092476	0.643491
7	LinearRegression	91.725243	610.445416	17.272294	0.003845
8	LGBMRegressor	95.458362	335.045700	12.229442	0.181994

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=1, verbose=0, warm_start=False)
Regresssor with better R2 score= RandomForestRegressor, Score (test, accuracy) = 96.14
```

Without latent variable RandomForestRegressor has the best performance with 96.14% accuracy.

Regression with latent variables:

	MODEL	R2_SCORE	MSE	MAE	TIME
0	Decision Tree Regressor	9.748127e+01	1.940232e+02	8.804992	0.020332
1	Random Forest	9.809689e+01	1.466005e+02	7.957862	0.096373
2	MLP Regressor	-1.168491e+06	9.001891e+07	5931.443165	2.335185
3	AdaBoost Regressor	9.796854e+01	1.564877e+02	9.477848	0.357950
4	Kernel Ridge	9.357071e+01	4.952614e+02	15.847667	0.664339
5	GradientBoostingRegressor	9.930949e+01	5.319143e+01	5.010679	0.812878
6	RandomForestRegressor	9.951567e+01	3.730877e+01	3.513178	1.561080
7	LinearRegression	9.496179e+01	3.881034e+02	14.278542	0.007741
8	LGBMRegressor	9.957934e+01	3.240437e+01	3.556725	0.410550

```
LGBMRegressor(bagging_fraction=0.8, boosting_type='gbdt', class_weight=None,
              colsample_bytree=1.0, feature_fraction=0.8,
              importance_type='split', learning_rate=0.17, max_depth=-1,
              metric='rmse', min_child_samples=20, min_child_weight=0.001,
              min_split_gain=0.0, n_estimators=100, n_jobs=-1, num_leaves=1200,
              objective='regression', random_state=None, reg_alpha=0.0,
              reg_lambda=0.9, silent=True, subsample=1.0,
              subsample_for_bin=200000, subsample_freq=0)
Regresssor with better R2 score= LGBMRegressor, Score (test, accuracy) = 99.58
```

We notice that with latent variables the performance of the model was improved. LGBMRegressor had the highest accuracy if 99.58%.

With these regression models we are able to predict stock price for future dates.

Executed the regression on **AutoGluon** and saw that it also gave high accuracy of 98.70%

```
Evaluation: root_mean_squared_error on test data: 9.538491323144136
Evaluations on test data:
{
    "root_mean_squared_error": 9.538491323144136,
    "mean_absolute_error": 6.0364668747840815,
    "explained_variance_score": 0.9870646303880267,
    "r2_score": 0.9870646276174408,
    "pearson_correlation": 0.9935874917660213,
    "mean_squared_error": 90.98281672169598,
    "median_absolute_error": 3.5916366577148153
}
Predictions:
 Date
2014-06-06     64.150978
2014-06-09     64.679672
2014-06-10     64.898544
2014-06-11     65.736801
2014-06-12     66.800644
                  ...
2020-02-12    198.943756
2020-02-13    201.926743
2020-02-14    210.411469
2020-02-18    207.764008
2020-02-19    198.253586
Name: Prediction, Length: 1436, dtype: float32
```