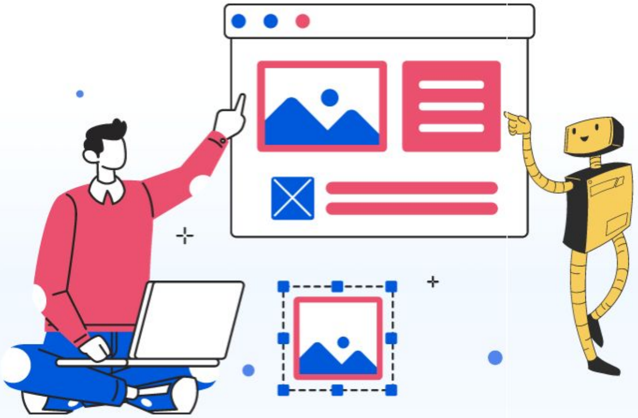
J.A.R.F.I.S.

(Just A Random Fine Intelligent System)

Contributors: Somya Panda, Andy Liang and Daryl Redmond

Contents

- Project Objective(s)
- Model Summaries
- Deep Learning Neural Network Approach
- Times-Series Approach
- Decision Tree & Random Sampling
- Postmortem/Next Steps



Project Objective(s)

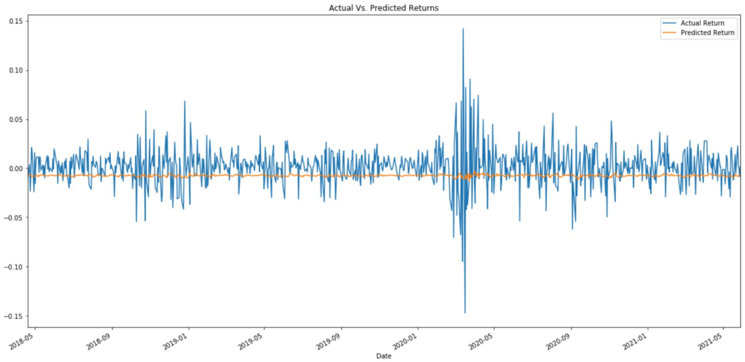
- Determine which model yields the best predictive result when analyzing historical stock data

Model Summaries

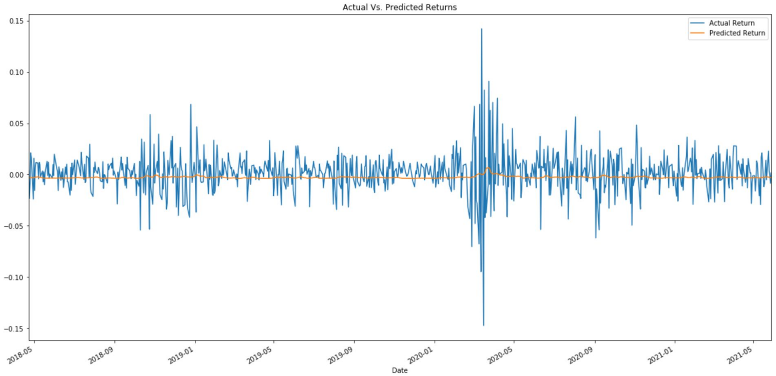
- Neural Network
  - A LSTM RNN model to predict entry and exit points that might generate profitable trades
  - Sequential model with four layers
- Time-Series
  - Univariate time series modelling using ARIMA to forecast closing stock price.
  - Multivariate time series modelling with correlated assets and sentiment scores as dependent variables using ARIMA.
- Decision Tree & Random Sampling
  - Create a decision tree model to determine entry & exit point of the selected public equity
  - Determine the precision through multiple random sampling model

Neural Network Approach

Models have difficulty predicting stock returns



Historical returns to predict future returns

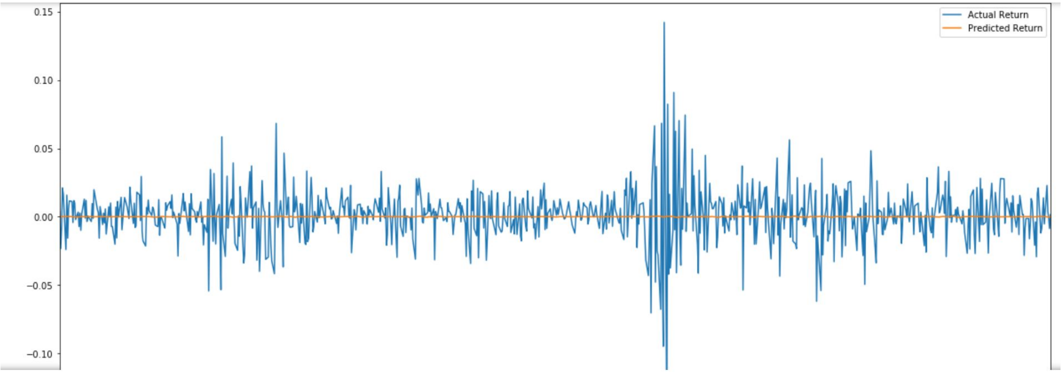


Standard deviation to predict future returns

Data Cleanup and Model Training

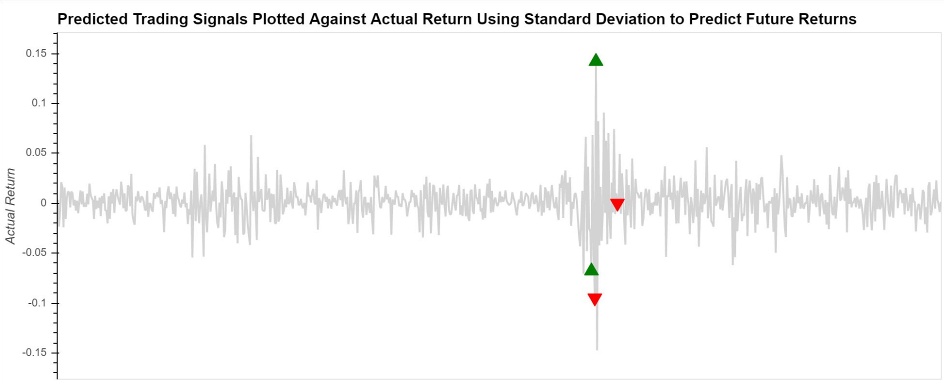
- Significant time was spent trying to generate results that generated trade recommendations when trying to predict future returns using historical returns
- A number of window time frames were tested
- Different train/test splits were tried
- Different optimizers and loss functions were test as well
- Finally, it was determined that a different approach should be taken
- Rather than predict returns, predict prices then calculate returns

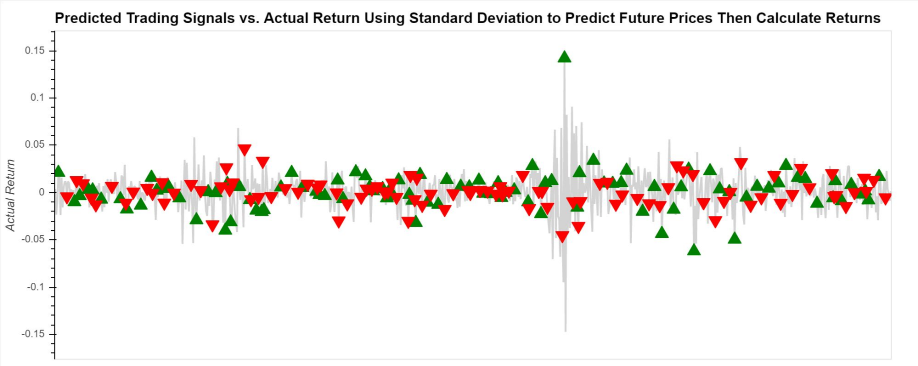
Standard deviations to predict future prices the calculate returns



Model Evaluation

- A practical approach for model evaluation was our focus - does the model generate trade recommendations with a second dimension of profitability



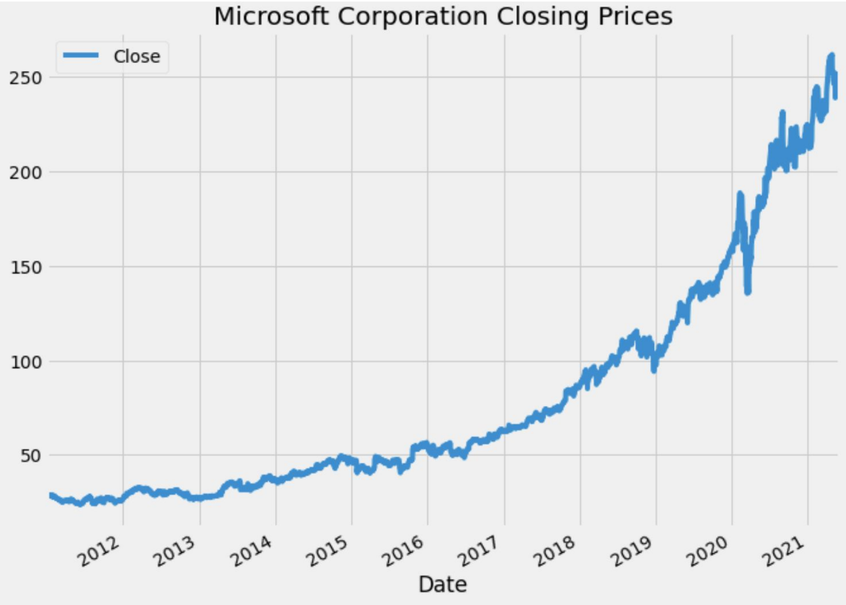


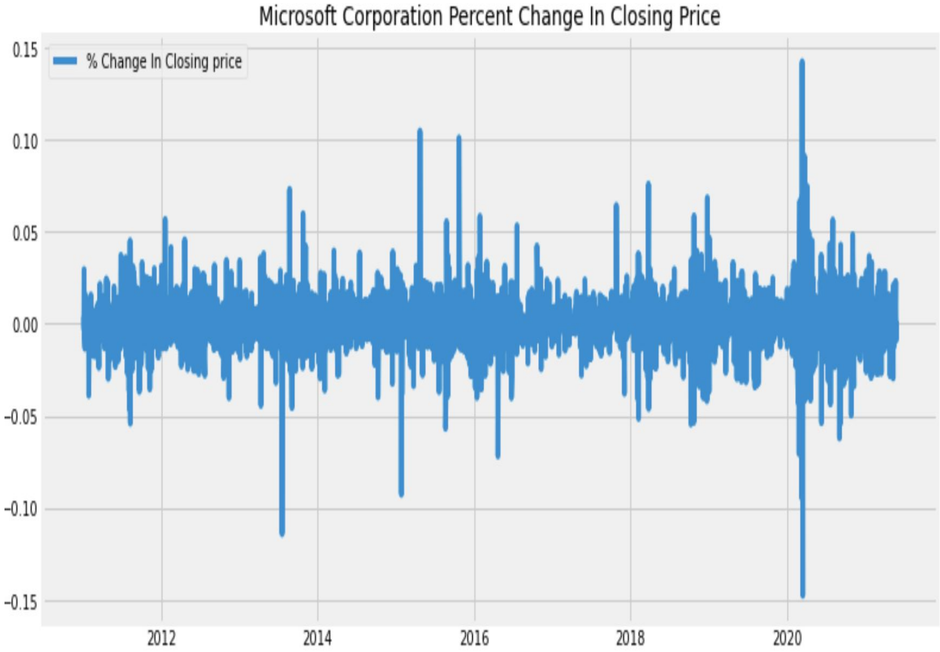
- Trading signals:
  - If predicted return is greater than zero then generate buy signal
  - If predicted return is less than zero then generate sell signal
- Outcome - predicted trading signals overlaid actual returns:
  - Predicted buy signals purchased 100 shares on that day at the actual closing price of the stock
  - Predicted sell signals sold those 100 shares on that day at the actual closing price of the stock
  - Between April 2018 and May 2021 the model generated over 100 entry and exit pairings
  - For a cumulative profit over $12k

Time-Series Analysis Approach

Historical Stock Price

- Pulled 10 years historical data of Microsoft from Yahoo Finance (https://finance.yahoo.com/)





Time Series Modeling Using ARIMA

Auto**R**egressive **I**ntegrated **M**oving **A**verage

In this project, we performed

1. Univariate Time Series Modeling
2. Univariate Time Series Modeling With Rolling Forecast
3. Multivariate Time Series Modeling With Correlated Assets
4. Multivariate Time Series Modeling With Sentiment Scores

Univariate Time Series Modeling

- ADF (Augmented Dickey-Fuller) Test for stationarity check
- Model parameters (p,d,q) (1,1,1) - ACF & PACF graphs
- Train and evaluate model
- Performance
    - MSE = 0.073
    - RMSE = 0.270
    - MAPE = 0.042

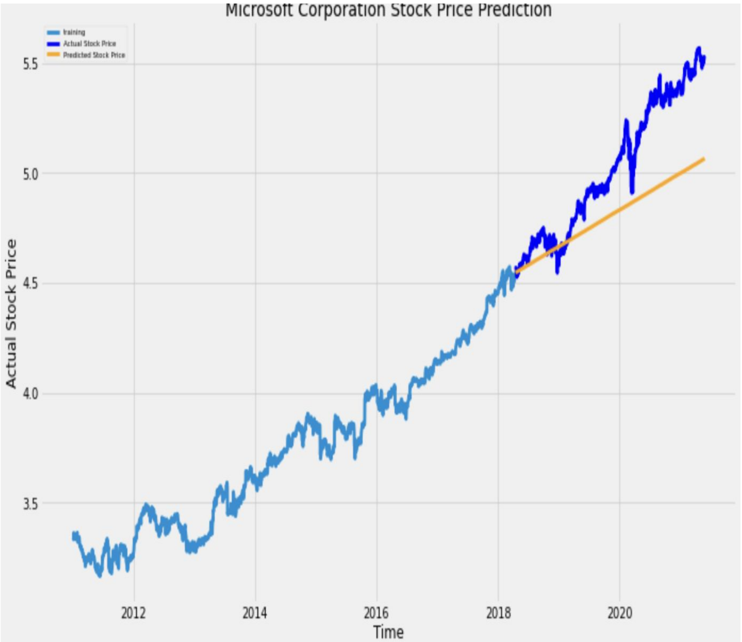
ARIMA Model Results

Dep. Variable:	D.Close	**No. Observations:**	1832
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	5179.910
Method:	css-mle	**S.D. of innovations**	0.014
Date:	Fri, 11 Jun 2021	**AIC**	-10351.819
Time:	09:05:18	**BIC**	-10329.767
Sample:	1	**HQIC**	-10343.686

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	0.0007	0.000	2.544	0.011	0.000	0.001
ar.L1.D.Close	0.8285	0.085	9.778	0.000	0.662	0.995
ma.L1.D.Close	-0.8672	0.075	-11.542	0.000	-1.014	-0.720

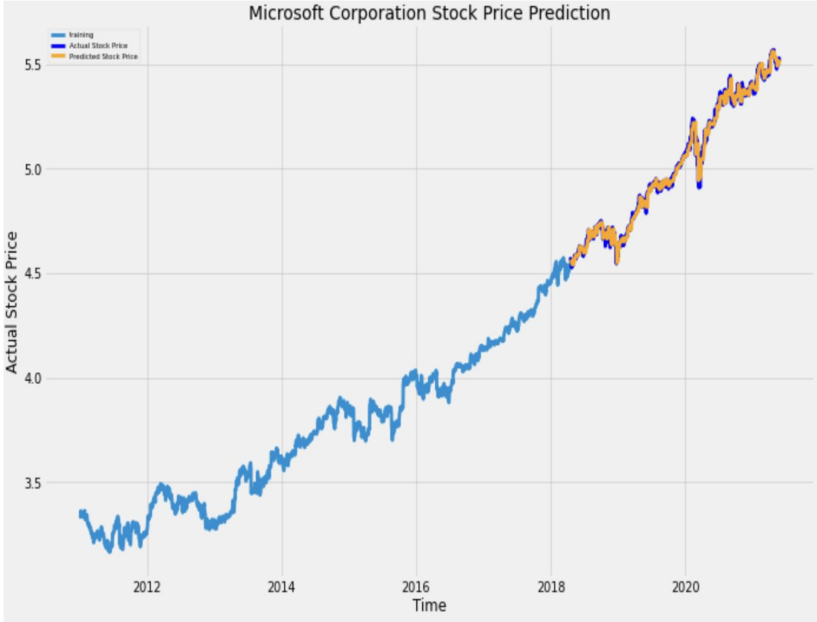
Roots

	Real	**Imaginary**	**Modulus**	**Frequency**
AR.1	1.2071	+0.0000j	1.2071	0.0000
MA.1	1.1532	+0.0000j	1.1532	0.0000



Rolling Forecast Using Auto ARIMA

- Selects the best model parameters using a grid search
- Retrain model with new data points. Here new data points are taken from test data.
- Performance
  - MSE = 0.000
  - RMSE = 0.028
  - MAPE = 0.004



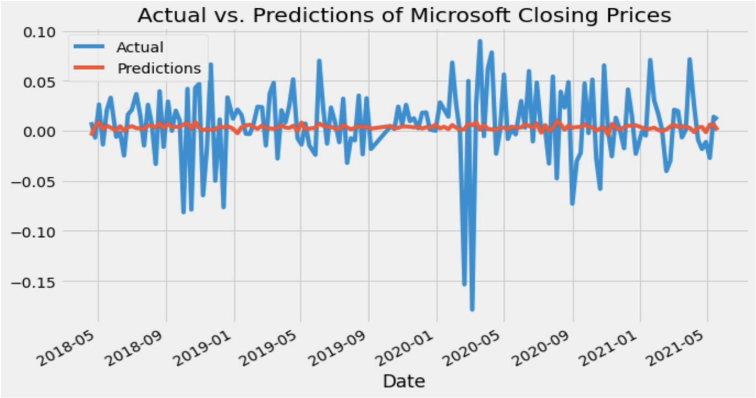
Multivariate Time Series Modeling With Correlated Assets

- Independent variables - related stocks, indices and currency exchange

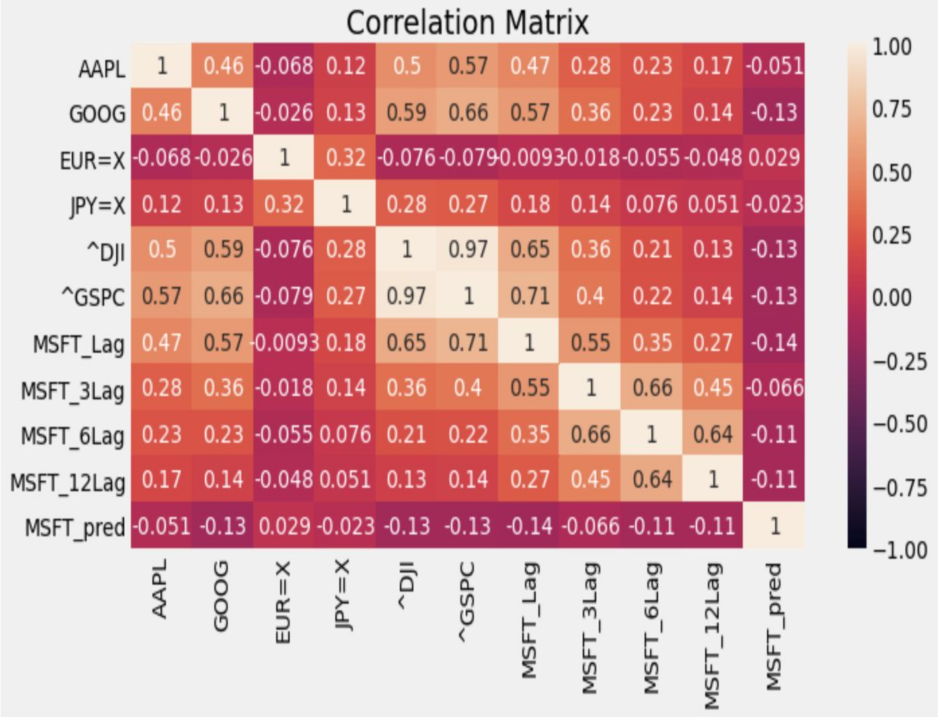
- Evaluate combination of p, d and q values to find the best parameters - (2,0,1)

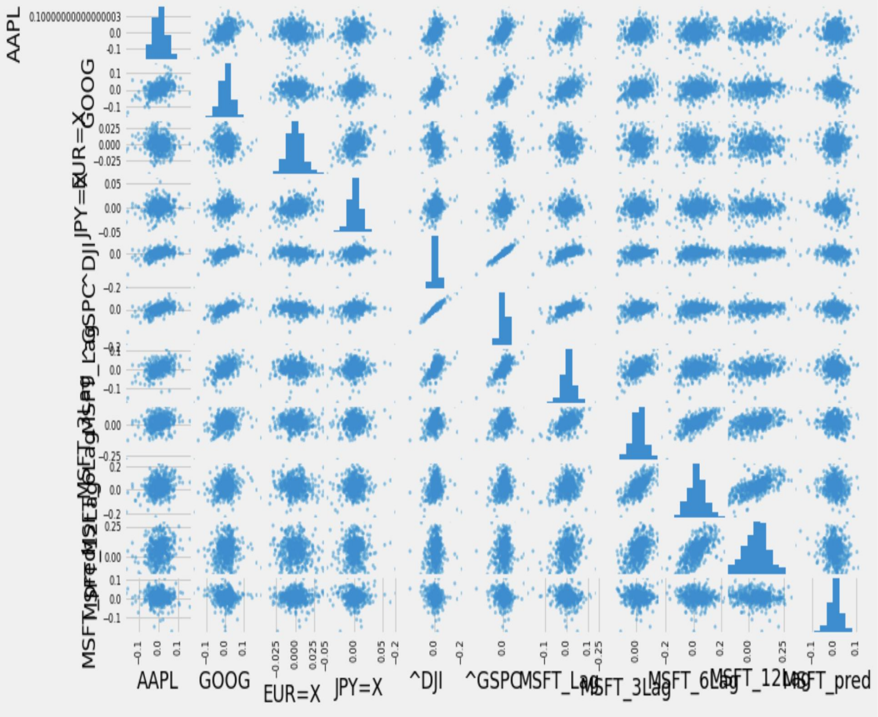
```
model = ARIMA(endog=Y_train,
exog=X_train_ARIMA,
order=(2,0,1))
```

- Performance
  - MSE = 0.001
  - RMSE = 0.037



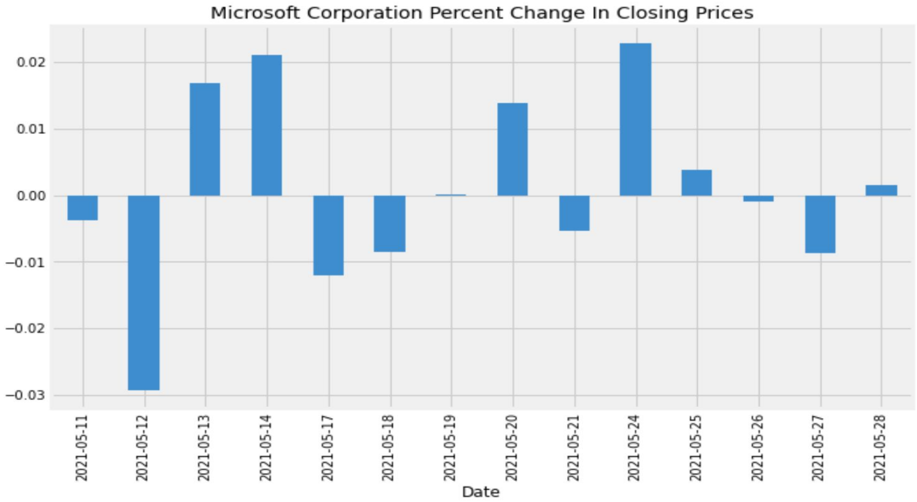
Correlation Matrix And Scatter Plot

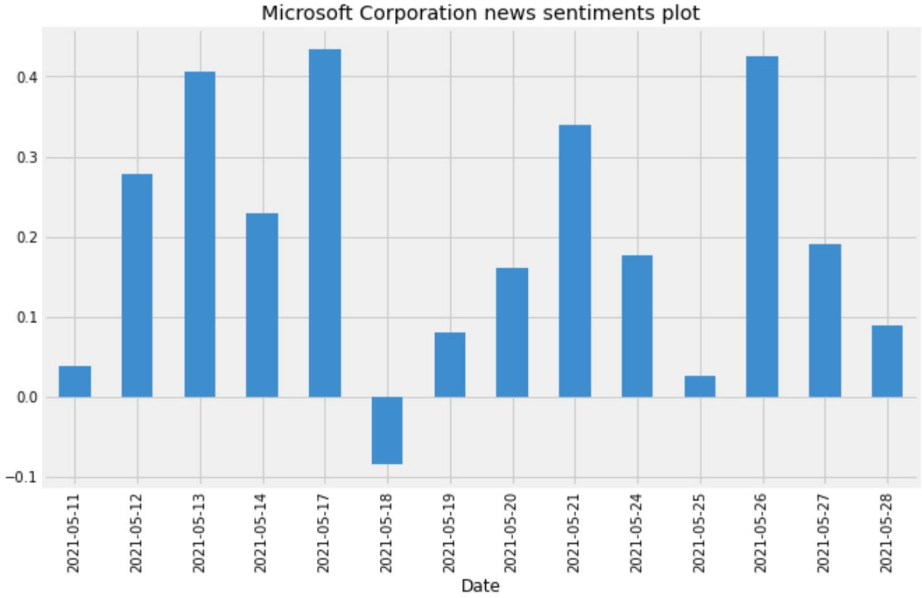




Multivariate Time Series Modeling With Sentiment Scores

- Calculate percent change of the closing prices
- Pulled latest articles of Microsoft using NewsAPI (https://newsapi.org/)
- Compare current sentiment score with next day percent change closing price





Prediction Of Stock Price Using Sentiment Scores

- Compound score is the dependent variable
- Evaluate combination of p, d and q values to find the best parameters - (1,0,0)

```
model_ARIMA =
ARIMA(endog=Y_train,
exog=X_train,
order=(1,0,0))
```

- Performance
  - MSE=0.000
  - RMSE=0.011

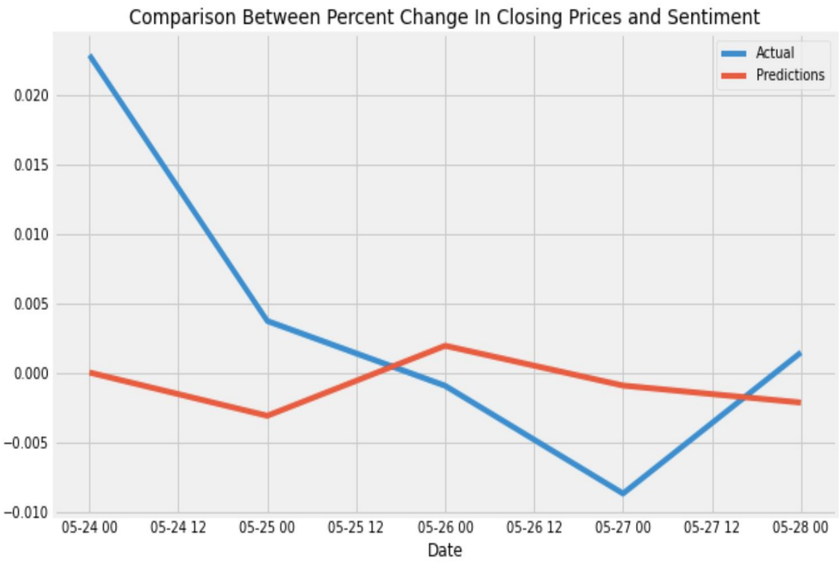
ARMA Model Results

Dep. Variable:	Pct_change	**No. Observations:**	9
Model:	ARMA(1, 0)	**Log Likelihood**	25.124
Method:	css-mle	**S.D. of innovations**	0.015
Date:	Fri, 11 Jun 2021	**AIC**	-42.249
Time:	14:42:06	**BIC**	-41.460
Sample:	05-11-2021	**HQIC**	-43.951
	- 05-21-2021		

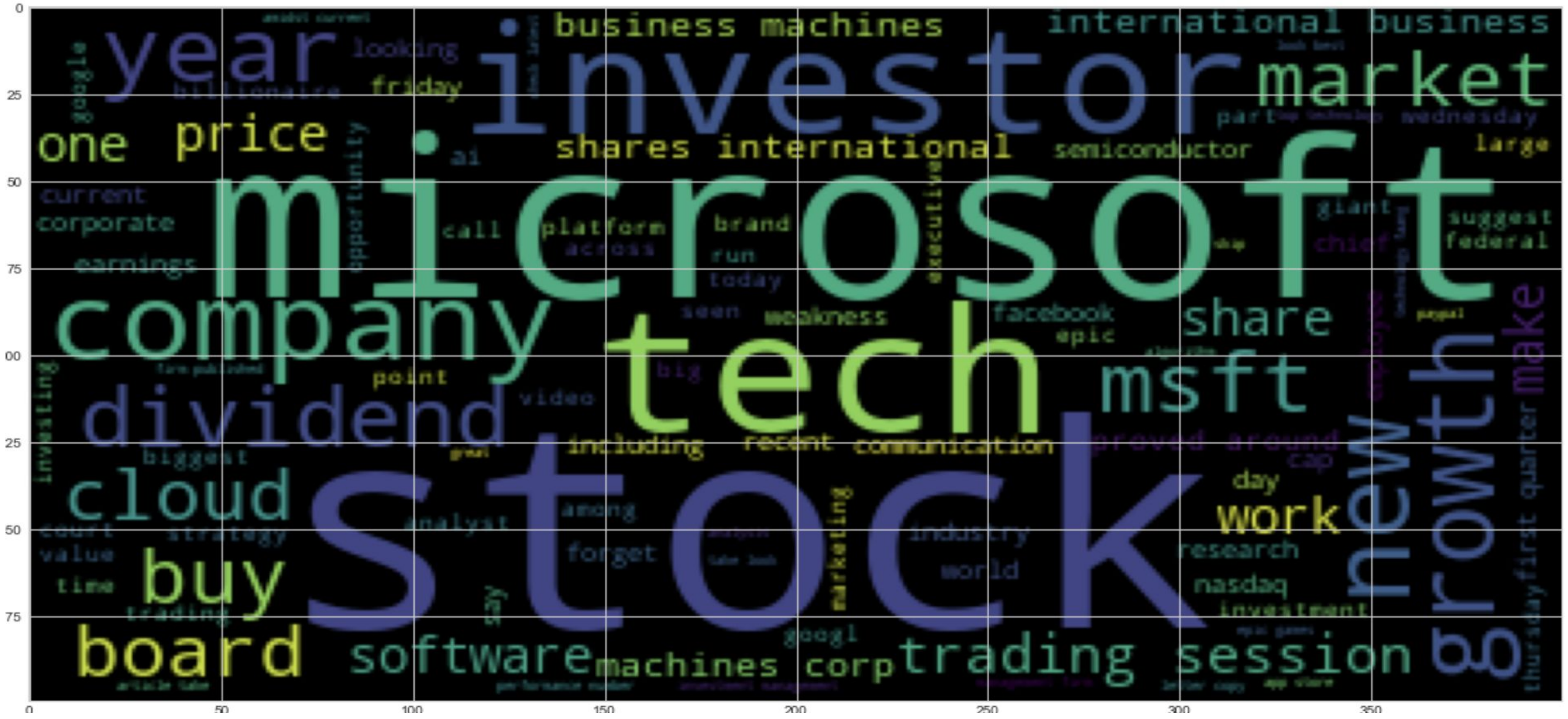
	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.0032	0.008	-0.382	0.703	-0.020	0.013
Compound	0.0120	0.035	0.346	0.729	-0.056	0.080
ar.L1.Pct_change	-0.1772	0.363	-0.488	0.625	-0.888	0.534

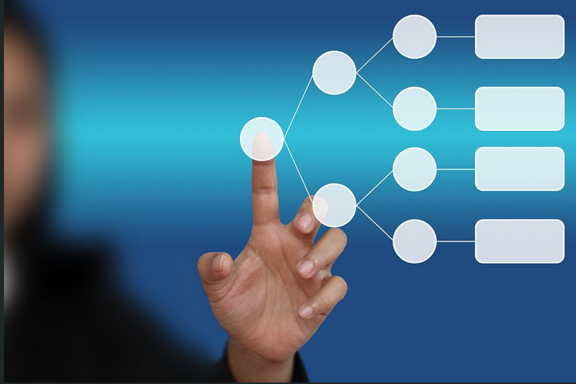
Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-5.6431	+0.0000j	5.6431	0.5000



Word Cloud


microsoft
stock
investor
tech
company
dividend
year
market
msft
cloud
buy
board
growth
new
software
trading session
machines corp
business machines
international business
shares international
one
price
share
work
make
semiconductor
looking
friday
billionaire
google
current
corporate
earnings
opportunity
call
platform
across
brand
run
today
executive
giant
suggest
federal
chief
large
wednesday
part
seen
weakness
facebook
epic
point
video
big
investing
biggest
including
recent
communication
proved around
cap
day
among
analyst
strategy
court
value
forget
industry
research
world
time
trading
nasdaq
investment
marketing
googl
thursday first quarter
employee
ai
0
25
50
75
100
125
150
175
200
250
300
350



Decision Tree & Random Sampling Approach

Preparation & Cleanup

- Utilize yfinance to pull in historical price
- Remove columns that are unnecessary to the model
- Created 2 new columns that will help determine and predict entry and exit point
  - Daily Return
  - Rolling Volatility
- Split test and training variables and scaled the X variables
- Create model using DecisionTreeClassifier

```
ticker = 'MSFT'
start_date = '2011-01-01'
end_date = '2021-05-31'

stock_df = yf.download(ticker, start=start_date, end=end_date)
stock_df['Daily Return'] = stock_df['Close'].pct_change().dropna()
stock_df['Adj Daily Return'] = stock_df['Close'].pct_change().dropna().shift()
#stock_df['Daily Rolling Volatility'] = stock_df['Daily Return'].rolling(window=10).std()
stock_df['Adj Daily Rolling Volatility'] = stock_df['Daily Return'].rolling(window=10).std().shift()
stock_df['Decision'] = np.where(stock_df['Adj Daily Return'] > 0, 'Entry' , 'Exit')
stock_df.dropna(inplace=True)
stock_df.drop(['Close', 'High', 'Low', 'Adj Close', 'Volume'], axis=1, inplace=True)
stock_df.to_csv('Export Files\decision df.csv', index=False)
stock_df.tail(20)
```

```
X = stock_df.copy()
X.drop(['Decision', 'Daily Return', 'Adj Daily Return'], axis=1, inplace=True)
X.dropna(inplace=True)
X.tail(20)
```

```
y = stock_df['Decision'].values.reshape(-1,1)
y[:20]
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=78)

scaler = StandardScaler()
X_scaler = scaler.fit(X_train)
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)
```

```python
model = tree.DecisionTreeClassifier()
model = model.fit(X_train_scaled, y_train)
```

```python
predictions = model.predict(X_test_scaled)
```

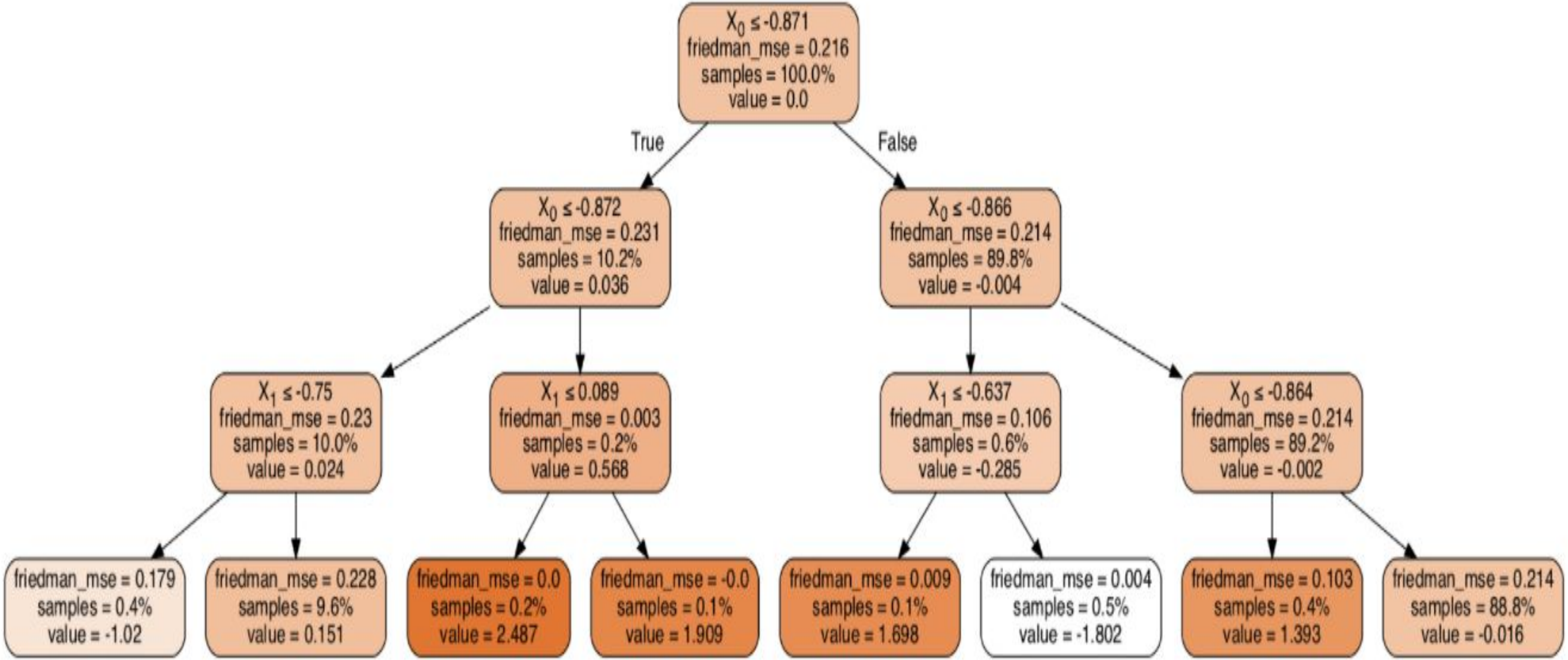
Gradient Boosting Classifier

- Determine model’s different learning rate by utilizing Gradient Boosting Classifier
- Set features up to 2

```
learning_rates = [0.05, 0.1, 0.25, 0.5, 0.75, 1]
for learning_rate in learning_rates:
    model = GradientBoostingClassifier(
        n_estimators=1000,
        learning_rate=learning_rate,
        max_features=2,
        max_depth=3,
        random_state=78)
    model.fit(X_train_scaled,y_train.ravel())
    print("Learning rate: ", learning_rate)

    # Score the model
    print("Accuracy score (training): {0:.5f}".format(
        model.score(
            X_train_scaled,
            y_train.ravel())))
    print("Accuracy score (validation): {0:.5f}".format(
        model.score(
            X_test_scaled,
            y_test.ravel())))
    print()
```

```
Learning rate:  0.05
Accuracy score (training): 0.87945
Accuracy score (validation): 0.51596

Learning rate:  0.1
Accuracy score (training): 0.96164
Accuracy score (validation): 0.52235

Learning rate:  0.25
Accuracy score (training): 1.00000
Accuracy score (validation): 0.51469

Learning rate:  0.5
Accuracy score (training): 1.00000
Accuracy score (validation): 0.50702

Learning rate:  0.75
Accuracy score (training): 1.00000
Accuracy score (validation): 0.53257

Learning rate:  1
Accuracy score (training): 1.00000
Accuracy score (validation): 0.50192
```

Decision Tree



$X_0 \leq -0.871$
friedman_mse = 0.216
samples = 100.0%
value = 0.0
True
False
$X_0 \leq -0.872$
friedman_mse = 0.231
samples = 10.2%
value = 0.036
$X_0 \leq -0.866$
friedman_mse = 0.214
samples = 89.8%
value = -0.004
$X_1 \leq -0.75$
friedman_mse = 0.23
samples = 10.0%
value = 0.024
$X_1 \leq 0.089$
friedman_mse = 0.003
samples = 0.2%
value = 0.568
$X_1 \leq -0.637$
friedman_mse = 0.106
samples = 0.6%
value = -0.285
$X_0 \leq -0.864$
friedman_mse = 0.214
samples = 89.2%
value = -0.002
friedman_mse = 0.179
samples = 0.4%
value = -1.02
friedman_mse = 0.228
samples = 9.6%
value = 0.151
friedman_mse = 0.0
samples = 0.2%
value = 2.487
friedman_mse = -0.0
samples = 0.1%
value = 1.909
friedman_mse = 0.009
samples = 0.1%
value = 1.698
friedman_mse = 0.004
samples = 0.5%
value = -1.802
friedman_mse = 0.103
samples = 0.4%
value = 1.393
friedman_mse = 0.214
samples = 88.8%
value = -0.016

Confusion Matrix

	Predicted Entry	Predicted Exit
Actual Entry	213	187
Actual Exit	203	180

Accuracy Score: 0.50192

Classification Report

	precision	recall	f1-score	support
Entry	0.51	0.53	0.52	400
Exit	0.49	0.47	0.48	383
accuracy			0.50	783
macro avg	0.50	0.50	0.50	783
weighted avg	0.50	0.50	0.50	783

Random Sampling Results

Confusion Matrix

	Predicted Entry	Predicted Exit
Actual Entry	213	187
Actual Exit	202	181

Balance Accuracy Score: 0.50254

Classification Report (Imbalanced)

	pre	rec	spe	f1	geo	iba	sup
Entry	0.51	0.53	0.47	0.52	0.50	0.25	400
Exit	0.49	0.47	0.53	0.48	0.50	0.25	383
avg / total	0.50	0.50	0.50	0.50	0.50	0.25	783

Random Oversampling

Confusion Matrix

	Predicted Entry	Predicted Exit
Actual Entry	163	237
Actual Exit	126	257

Balance Accuracy Score: 0.53926

Classification Report (Imbalanced)

	pre	rec	spe	f1	geo	iba	sup
Entry	0.56	0.41	0.67	0.47	0.52	0.27	400
Exit	0.52	0.67	0.41	0.59	0.52	0.28	383
avg / total	0.54	0.54	0.54	0.53	0.52	0.27	783

Random Undersampling

Cluster Centroid

Confusion Matrix

	Predicted Entry	Predicted Exit
Actual Entry	140	260
Actual Exit	112	271

Balance Accuracy Score: 0.52879

Classification Report (Imbalanced)

	pre	rec	spe	f1	geo	iba	sup
Entry	0.56	0.35	0.71	0.43	0.50	0.24	400
Exit	0.51	0.71	0.35	0.59	0.50	0.26	383
avg / total	0.53	0.52	0.53	0.51	0.50	0.25	783

Balanced Random Forest

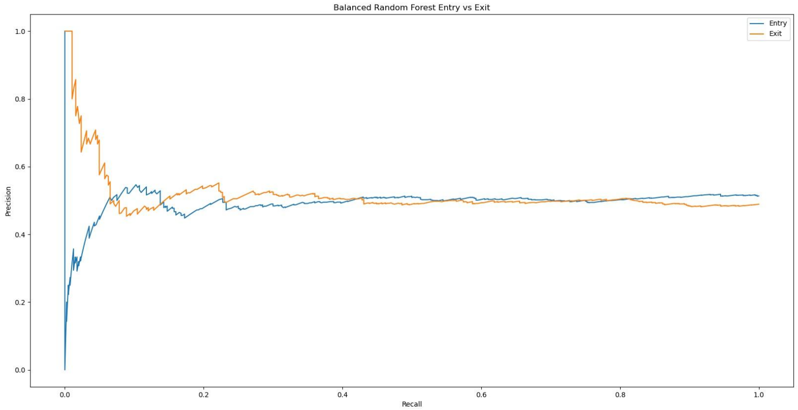
Confusion Matrix

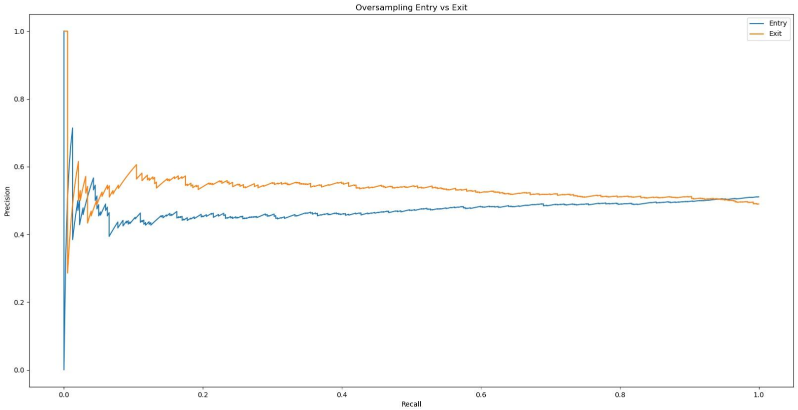
	Predicted Entry	Predicted Exit
Actual Entry	200	200
Actual Exit	193	190

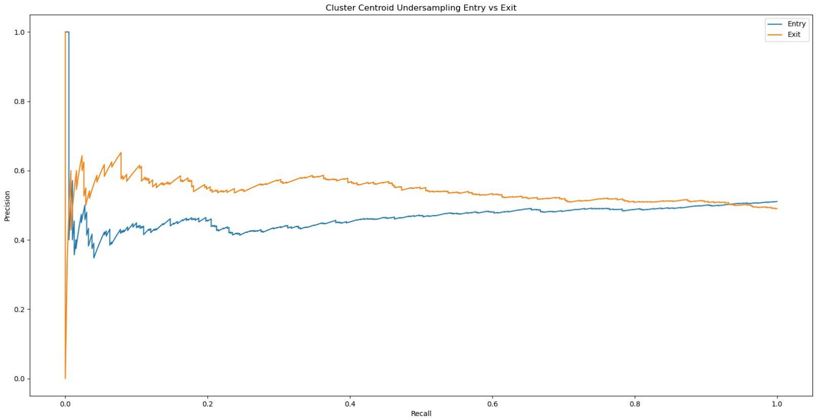
Balance Accuracy Score: 0.49804

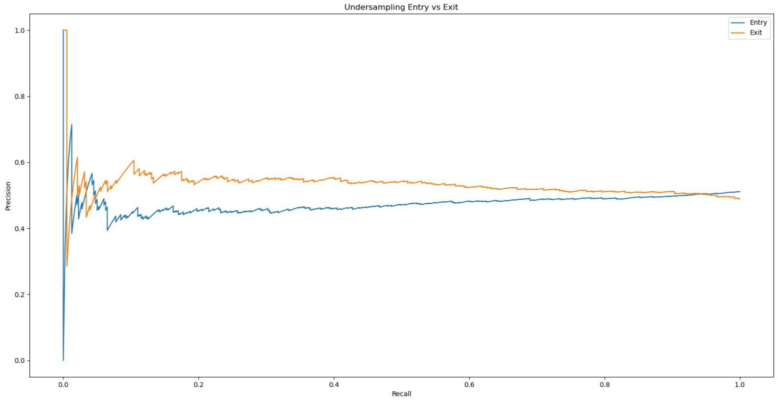
Classification Report (Imbalanced)

	pre	rec	spe	f1	geo	iba	sup
Entry	0.51	0.50	0.50	0.50	0.50	0.25	400
Exit	0.49	0.50	0.50	0.49	0.50	0.25	383
avg / total	0.50	0.50	0.50	0.50	0.50	0.25	783


Balanced Random Forest Entry vs Exit
Entry
Exit
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0


Oversampling Entry vs Exit
Entry
Exit
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0


Cluster Centroid Undersampling Entry vs Exit
Entry
Exit
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0


Undersampling Entry vs Exit
Entry
Exit
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0

Q & A

Postmortem/Next Steps

- Test the model with different equities and equities from different sectors
- Use different time frames to evaluate the model
- Incorporate different variables to enhance the models in determining entry/exit strategy
- Insert more data point for sentiment analysis (API restriction to past 30 days)