STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

FA 692 – Natural Language Processing for Financial Applications

Spring 2023

Predicting Market Returns using Sentiment Analysis

Final Project

Sandeep Ranjan

Overview

In this project we incorporate *Sentiment Analysis* in predicting the returns of SPY and compare how this model performs in relation to other Machine Learning models without using any Natural Language Processing techniques.

We extract Bloomberg twitter feeds to gauge the market sentiments, quantify these sentiments (in terms of positive/negative/neutral polarities) and then incorporate them in various Machine learning models to predict the returns.

Data Collection

For the purpose of this project, we retrieved adjusted closing prices of SPY from ***yahoo finance*** for the period January 1st, 2010 – March 15th 2023

Bloomberg Twitter Feeds were collected using python *vaderSentiment* API from January 2010 – March 2023:

Time	Tweet	Date	Sentiment	Positive	Negative	Neutral
2023-03-14 19:40:29-04:00	One Japanese fintech firm is making it compuls...	2023-03-14	0.0000	0.000	0.000	1.000
2023-03-14 19:40:29-04:00	An unlikely startup guru has emerged in Japan,...	2023-03-14	0.0000	0.000	0.000	1.000
2023-03-14 19:35:41-04:00	Some US cities are late in making financial di...	2023-03-14	0.1280	0.097	0.079	0.824
2023-03-14 19:31:07-04:00	The shipping industry is looking to rethink ev...	2023-03-14	-0.8462	0.000	0.368	0.632
2023-03-14 19:25:09-04:00	A biotech wants to cut fashion waste by using ...	2023-03-14	-0.5994	0.000	0.290	0.710
...	...	...	...	...	...	...
2010-01-05 13:14:16-05:00	Bloomberg News: U.S. Retail Sales Boosted by G...	2010-01-05	0.3612	0.217	0.000	0.783
2010-01-05 12:39:05-05:00	Bloomberg News: Google May Introduce Its Own A...	2010-01-05	0.0000	0.000	0.000	1.000
2010-01-05 12:00:45-05:00	Bloomberg News: Manhattan Apartment Prices Fal...	2010-01-05	-0.3182	0.000	0.161	0.839
2010-01-05 11:26:25-05:00	Bloomberg News: Apple Said to Plan Tablet PC I...	2010-01-05	0.0000	0.000	0.000	1.000
2010-01-05 10:43:45-05:00	Bloomberg Markets Hedge Fund Ranking: Tepper T...	2010-01-05	0.4588	0.267	0.140	0.593

800003 rows × 6 columns

The tweets were then aggregated for each day for each polarity and their average was calculated:

	avg_positive	avg_negative	avg_neutral
2010-01-04	0.000000	0.000000	0.000000
2010-01-05	0.094077	0.094231	0.811692
2010-01-06	0.065000	0.115500	0.819500
2010-01-07	0.000000	0.191000	0.809000
2010-01-08	0.092125	0.188750	0.719125
...	...	...	...
2023-03-08	0.076004	0.056796	0.867198
2023-03-09	0.070778	0.070409	0.858809
2023-03-10	0.063741	0.077536	0.858731
2023-03-13	0.069274	0.075441	0.855300
2023-03-14	0.067350	0.078338	0.854321

3321 rows × 3 columns

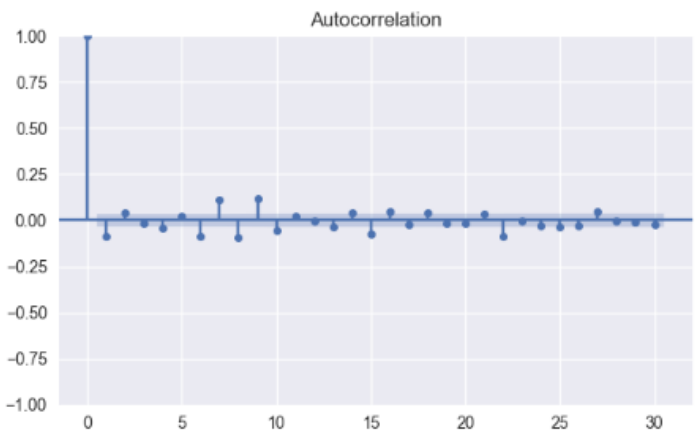
Establish Baseline for the Returns

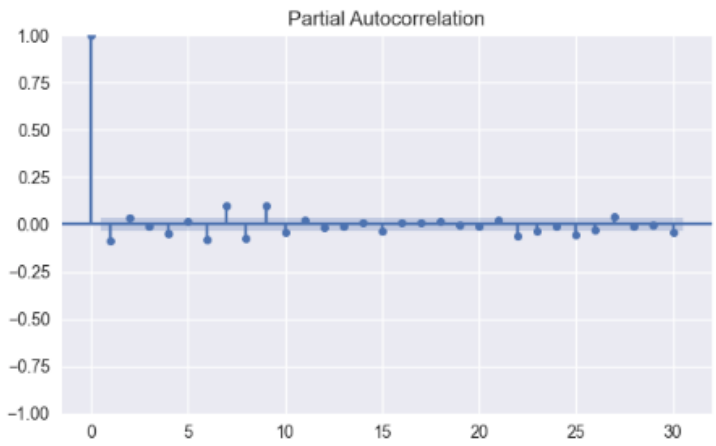
For comparison purposes, we used ***ARIMA*** model to forecast SPY Returns

ARIMA (Auto-Regressive Integrated Moving Average) is a way of modelling time series data for forecasting (i.e., for predicting future points in the series), in such a way that:

- a pattern of growth/decline in the data is accounted for (hence the term “auto regressive”)
- the rate of change of the growth/decline in the data is accounted for (hence the term “integrated”)
- noise between consecutive time points is accounted for (hence the term “moving average”)

The parameters used in ARIMA were calculated using the ACF and PACF functions when applied on the SPY adjusted Closing prices.



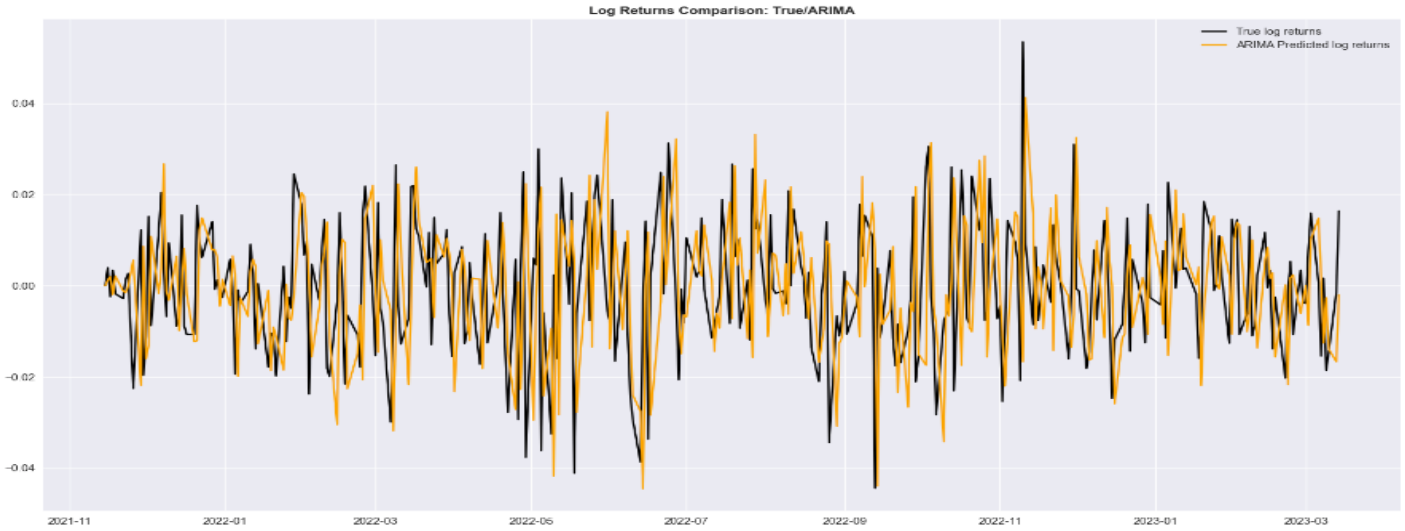


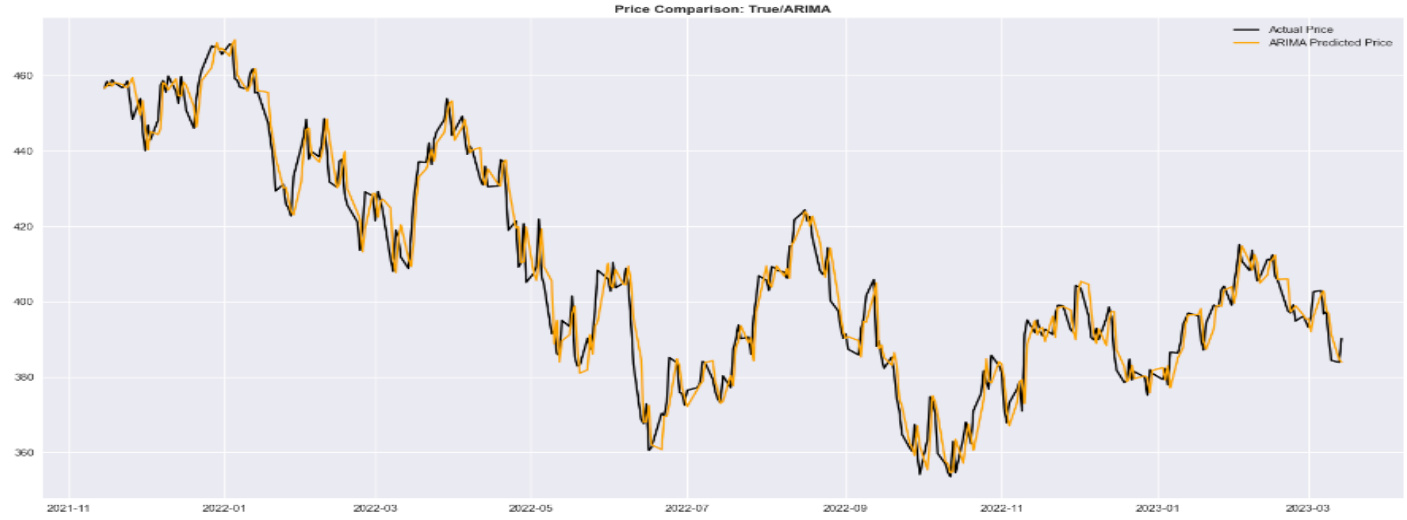
ADF test was used to ensure that the time series that is being fed into ARIMA model is stationary and that it's p-value is < 0.05.

```
# ADF test to check if price series is now stationary

result2 = adfuller(diff1)
print(f'ADF Statistic: {result2[0]}')
print(f'p-value: {result2[1]}')
for key, value in result2[4].items():
    print('Critial Values:')
    print(f'   {key}, {value}')
```

```
ADF Statistic: -12.109374157235292
p-value: 1.9352545560074517e-22
Critial Values:
   1%, -3.4323391822085685
Critial Values:
   5%, -2.862418902926822
Critial Values:
   10%, -2.5672378582976876
```





RMSE for test data using ARIMA Model: 0.019689

Machine Learning Models

Multiple Machine learning models were used with different sets of features to predict the returns. Root Mean Squared was calculated for each of these:

Machine learning models	RMSE Train	RMSE Validation	RMSE Test
Linear Regression (2 lagged returns)	0.010786	0.008835	0.014485
Linear Regression (2 lagged sentiment scores)	0.010909	0.008841	0.014271
Linear Regression (2 lagged returns/2 lagged sentiment scores)	0.010778	0.008821	0.014490
Feed Forward Neural Network (2 lagged returns)	0.010838	0.008820	0.014327
Feed Forward Neural Network (2 lagged returns/2 lagged sentiment scores)	0.011245	0.009526	0.014456
Feed Forward Neural Network (2 lagged sentiment scores)	0.011724	0.010089	0.014616
Recurrent Neural Network (2 lagged returns)	0.010797	0.008855	0.014572
Recurrent Neural Network (2 lagged returns/2 lagged sentiment scores)	0.010972	0.009387	0.015133
Recurrent Neural Network (2 lagged sentiment scores)	0.011128	0.008949	0.014556
LSTM (2 lagged returns)	0.010918	0.008851	0.014269
LSTM (2 lagged returns/2 lagged sentiment scores)	0.010921	0.008877	0.014250
LSTM (2 lagged sentiment scores)	0.010926	0.008896	0.014243
Random Forest (2 lagged returns)	0.004421	0.009554	0.015779
Random Forest (2 lagged returns/2 lagged sentiment scores)	0.004209	0.009005	0.014733
Random Forest (2 lagged sentiment scores)	0.004181	0.009161	0.014356

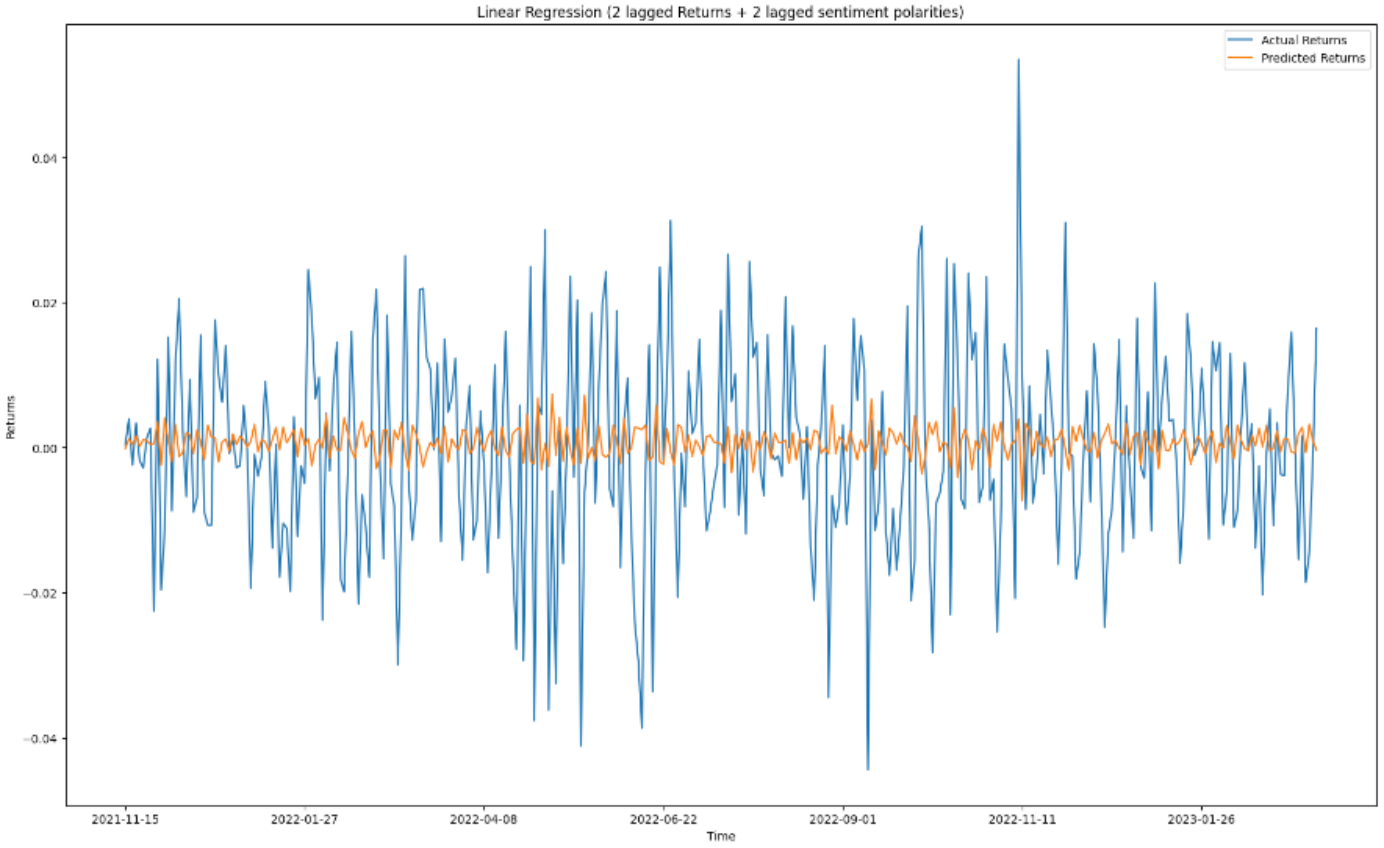
Comparing the RMSE for validation set, the following 2 models seem to have identical RMSE values, although feed forward NN seem to have performed marginally better.

- Linear Regression model using 2 lagged returns along with 2 lagged sentiment scores (0.008821)
- Feed Forward Neural Network using 2 lagged returns (0.008820)

Both the models were again trained using 90% data, the RMSE was calculated on the 10% test data:

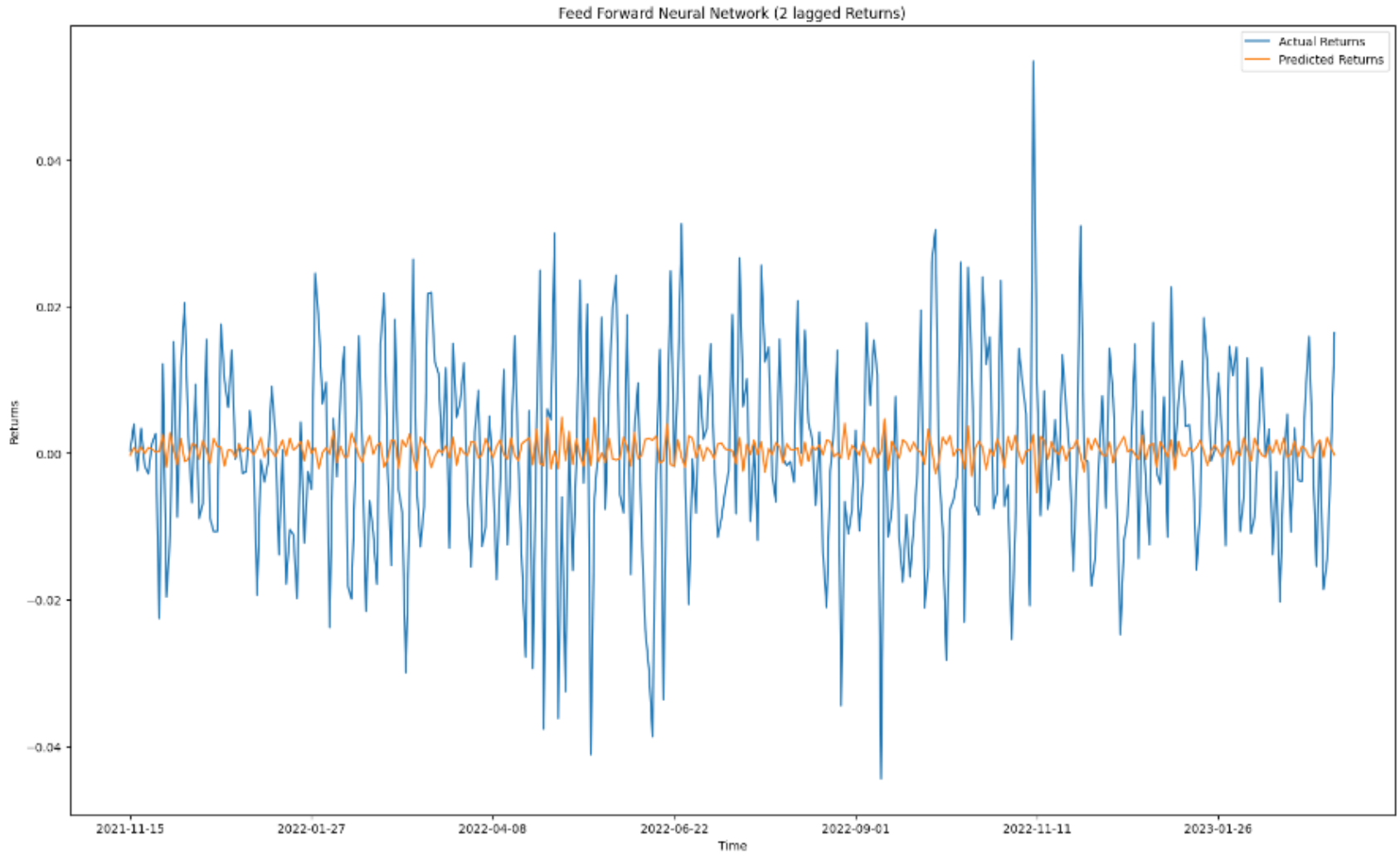
Linear Regression model (2 lagged returns + 2 lagged sentiment scores):

RMSE on training data = 0.010578
RMSE on test data = 0.014491



Feed Forward Neural Network using 2 lagged returns:

RMSE on training data = 0.010611
RMSE on test data = 0.014379



Comparing the RMSE values of Linear Regression and Feed Forward Neural Network model, it appears that sentiment analysis didn't have much of an impact on the overall SPY returns. To confirm this, we calculated the correlation of SPY returns with each of sentiment polarity scores.

```
In [226]: # Calculate Correlation between log returns and sentiment scores

          from scipy.stats import pearsonr

          ## Correlation between Predicted returns and Positive sentiment score
          pearsonr(yhat.tolist(),data[int(len(data) * 0.9):]['avg_positive'].tolist())

Out[226]: PearsonRResult(statistic=-0.023191785745490 87, pvalue=0.6732617286239236)

In [227]: ## Correlation between Predicted returns and Negative sentiment score
          pearsonr(yhat.tolist(),data[int(len(data) * 0.9):]['avg_negative'].tolist())

Out[227]: PearsonRResult(statistic=0.10229619488434011, pvalue=0.06223675618379024)

In [228]: ## Correlation between Predicted returns and Neutral sentiment score
          pearsonr(yhat.tolist(),data[int(len(data) * 0.9):]['avg_neutral'].tolist())

Out[228]: PearsonRResult(statistic=-0.07099874603325583, pvalue=0.19623070714701532)
```

Clearly the p-value was > 0.05, hence we can confirm that including the sentiments as features in the models didn't add much value to the overall SPY returns.

Another point to note was that both these Machine learning models performed better than the ARIMA model (RMSE: 0.019689)

Next Step:

In this project, the sentiments were gathered only from Twitter. There could be multiple other sources of collecting investor sentiments. However, getting all this information is extremely costly in terms of time and resources. As a next step, we could explore all other possible sources of these sentiments and incorporate them in the model.