Estimate British Pound Volatility using NLP and LSTM Neural Network

During the Brexit period

Cristobal Gonzalez
Deniz Kural

NYU FRE 7871: Special Topics
News Analytics and Machine Learning

Master of Science in Financial Engineering
(M. Sc.)

October 20, 2020

Finance and Risk Engineering
Tandon School of Engineering
New York University

Professor Dr. Andrew O. Arnold

Abstract

Natural Language Processing has been used extensively to quantify market sentiment in order to estimate financial markets' behaviors. In this report, NLP was used to estimate the Great Britain Pound volatility along with the Recurrent Neural Network model, LSTM. Using data from Twitter, the model created two indicators, the first measured the raw amount of data related to this currency, and the second generated a sentiment analysis about it. From the last years we could see that the Brexit has been a strong factor on the British Pound fluctuations, therefore our sentiment analysis will focus only on reading and processing Brexit information with the purpose of being more accurate in our sentiment score. Then, the LSTM model joined these indicators with the historical volatility to predict future volatility values. In order to isolate the GBP movements the model compared it to a basket of currencies using daily information. The selected model consisted in one LSTM Layers with 50 neurons plus a Dense Layer, using the sentiment analysis created with the Valence Aware Dictionary and Sentiment Reasoner (VADER). The model generated an accurate volatility tracker and a good accuracy metric predicting the volatility direction of the next day (60.3% on the testing data).

Students Information

Our team for this project consist of two MSFE's students whose information could be seen below:

Deniz Kural
Net id: dk3703
Email: dk3703@nyu.edu
Available for live presentation.

Cristobal Gonzalez Corssen
Net id: cg3516
Email: cg3516@nyu.edu
Available for live presentation.

Our team worked together at each sub task, the two members feature gave us dynamism and agility to develop the project with ease. In addition, we both came from different backgrounds, which allowed us to complement and help each other in every difficulty that one of the two faced during the assignment. Deniz has strong knowledge in Python, along with an extensive experience in data cleaning and preparation that was extremely helpful in the first stage of the project. On the other hand, Cristobal has strong background in statistical models and investment strategies, which allowed us to choose and develop an algorithm very fast, leaving us time to focus on calibrating and achieving accurate results.

Contents

1 Introduction

In today's world, almost all of the news could be found in platforms such as Twitter, Reddit etc. These are some open sources websites or apps, where everyone can share their opinions and information, but it has been reached that official media channels are sharing their news and products throughout these platforms. The main advantage is that the light and informal format of the "post" allows deliver information extremely fast and concentric in one single place. It is getting popular for finance people to use these platforms to extract the information in these websites with the usage of Natural Language Processing (NLP). In this project, we propose to come up with an algorithm that will predict the volatility movements of the British Pound currency (GBP) by analyzing the effects of Brexit trade talks using NLP.

Natural language processing is used widely for estimating stock prices nowadays. This happened due to the shortcomings when the stock performance is analyzed just with its past data. Thus, comments and emotional states of people and companies have taken into account with NLP. In this project, we wanted to analyze not just a stock performance but also, an important event's effect on the forex market. More specifically, we wanted to observe the Brexit trade talks effect on GBP against a basket of currencies by using Twitter as our main source to get data from by using NLP. Our motivation will be to estimate GBP's volatility with a considerable accuracy by using the news (Tweets) related to the trade talks.

We are planning to achieve this goal by dividing our project to subparts such as web scraping for getting Tweets from Twitter, NLP modeling, generating the sentiment score for individual tweets and prediction of GBP's volatility. At web scraping and data cleaning stage, removal of special characters and most common but not useful words ('the', 'and', etc.) will be done. Moreover, VADER that is a lexicon embedding technique could be an option to use it with the LSTM Recurrent Neural Network model so as to get a more accurate result in the sentiment analysis stage of the project. At the end of the project, we will test our result by observing the accuracy of our results compared with the Forex market and future work could be considered to achieve a better model.

2 Problem Statement

The prediction of financial asset prices is one of the most performed tasks in the financial field and at the same time one of the most complex and hard to estimate. Most of the time the accuracy is extremely low and some models that have been successful do not have great prosperity. However, if we work with returns, moreover, with the second moment of the returns, i.e. the volatility, empirical evidence has shown that it has higher accuracy in its prediction than prices. From historical data we can easily identify clusters in volatility, periods where the prices fluctuate large amounts, and other periods when those prices are calm and oscillate much less. For that reason, in this work we are trying to estimate the volatility of a foreign exchange rate, considering the historical prices and news related to the country.

Specifically we will work with the British Pound and our hypothesis consists that Brexit has been and will be one of the most important factors in determining local currency fluctuations. We believe that analyzing the news related to this process of leaving the UK to the European Union, we could forecast the volatility of the FX. Therefore, the problem consists in a little more than three years of data, to calibrate a sentiment analysis and predict markets fluctuations. The Brexit originates in mid 2016, so we will consider from the beginning of that year until end of 2019.

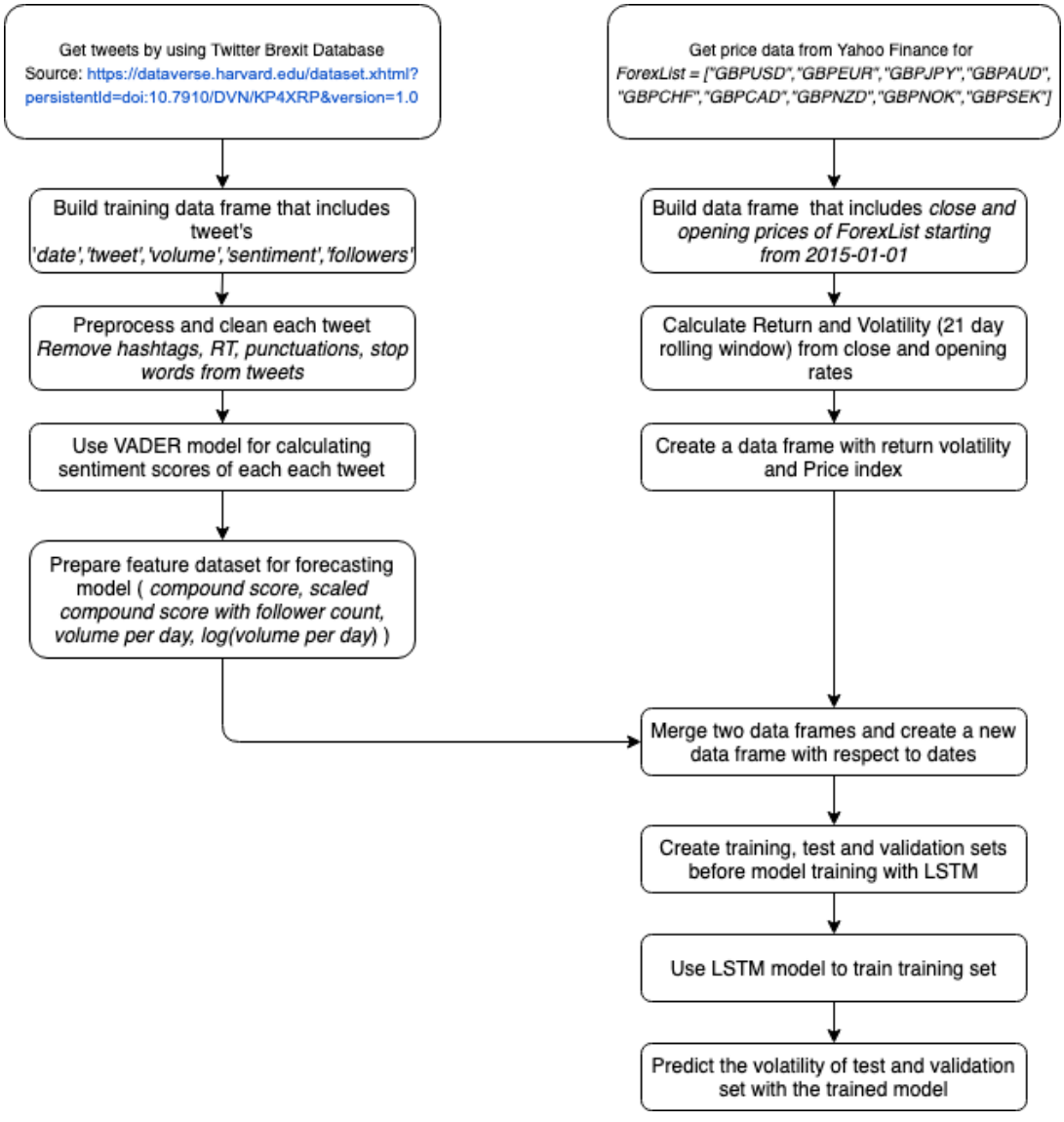

Figure 2.1: Flow Chart for the Project

3 Data

In this section we will discuss what type of data is needed for the project, how we extracted that information, and finally the methodology of processing and storing. As we mentioned before, the main source for our analysis consisted in the platform Twitter, where there are available Python libraries to connect to the its API. On the other hand, Yahoo finance was extremely useful to download daily prices for the foreign exchanges used in this project. The following subsections will explain in more details the steps mentioned.

3.1 Prices

3.1.1 Description

The purpose of the project is to forecast the GBP's volatility, so the most important series are the prices of those foreign exchange needed. Until now, we have not explained how we will construct the volatility of one single currency, considering that in the Forex market are only available pairs of currencies. The purpose was to isolate the movements of this particular country, so the best way to do it is to compare the value of that currency against many other currencies. The details about that process will be in the next Processing subsection, but the idea is to get in general if the Pound is more cheap or expensive for the rest of the world.
Since England is a developed country, its currency tends to be more stable than some emerging market countries against for example the US dollar. For that reason, we thought that it will be better if we compare the Great Britain Pound against the G10 currencies, which are considered the more stable and traded currencies in the world. This list is composed by:

- United States dollar (USD)
- Euro (EUR)
- Pound sterling (GBP)
- Japanese yen (JPY)

- Australian dollar (AUD)
- New Zealand dollar (NZD)
- Canadian dollar (CAD)
- Swiss franc (CHF))
- Norwegian krone (NOK)
- Swedish krona (SEK)

In the following subsection we will describe how we downloaded these pairs of currencies and how we isolated the British Pound.

3.1.2 Extraction

From Yahoo Finance we could download plenty securities' daily prices. The foreign exchange market is not absent in this platform, se we were able to download daily close prices for the nine pairs of currencies between the GBP and the rest of the G10 currencies. It was very important to focus in which of the two currency the pair is settle, i.e. the direct or indirect quotation. Since we wanted to create a basket of currencies, it is easier if we download the same quotation for the pairs, making the joining process only a matter of average. That said, we downloaded from January 1st, 2015, to December 31st, 2019 of every currency expressed in terms of GBP. Those quotes are the following: GBPUSD=X, GBPEUR=X, GBPJPY=X, GBPAUD=X, GBPNZD=X, GBPCAD=X, GBPCHF=X, GBPNOK=X, and GBPSEK=X.

Date	GBPUSD	GBPEUR	GBPJPY	GBPAUD	GBPCHF	GBPCAD	GBPNZD	GBPNOK	GBPSEK
2015-01-01	1.558094	1.28720	186.429993	1.90534	1.54832	1.80897	1.99661	11.61900	12.1208
2015-01-02	1.557972	1.28850	186.764999	1.90510	1.54910	1.80980	1.99930	11.63140	12.1365
2015-01-05	1.528491	1.27940	184.078003	1.89430	1.53720	1.80220	1.99471	11.63100	12.1530
2015-01-06	1.525832	1.27800	182.205002	1.88370	1.53529	1.79350	1.98115	11.64600	12.0774
2015-01-07	1.513798	1.27446	179.625000	1.87300	1.53070	1.79150	1.95408	11.72000	11.9970

Figure 3.1: Foreign Exchanges Rates of G10 currencies against GBP.

3.1.3 Processing and Storage

The process of creating the GBP index was very straightforward, we tried to be as simple as possible, so we made an equal weighted basket among the currencies. The goal was to create an index that were equivalent to have a portfolio of different currencies. Imagine you have Pounds and wanted to invest diversified in Forex market, so you just buy different currencies assigning the same amount of Pounds to each of those assets. To get that portfolio we only need to generate the returns of each forex rate, multiply by 1/9, and add them all up. We used the log returns, so after adding all the currencies, the process of creating the new index consisted in a rolling summation and a multiplication each value by the first price, in this case we used base 100. This time series will be used later in the forecast model, so we storage this data in a csv, where the Python code could later upload.

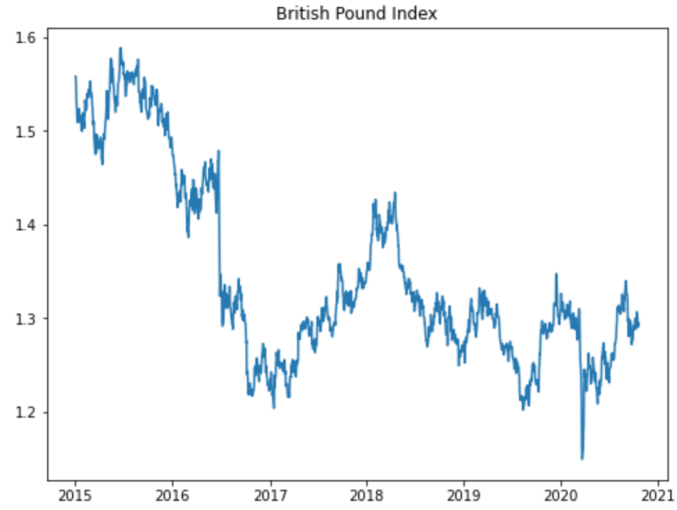

Figure 3.2: British Pound Index performance

3.2 Tweets

3.2.1 Description

A prediction of volatility using only prices data is probably the first homework assignment of every financial engineer student. In this project we wanted to go way

beyond that, so we had to add a not so trivial source of information, that maybe everyone is able to see it, but only a few could use it. All data in words belongs in this category; although many data scientist used for their jobs, there are still a large portion of the market that cannot extract that information efficiently. As we already mentioned, Twitter is one of the most used platforms to share ideas, opinions, and news about any topic, and in finance is very used to follow live news about the markets.

3.2.2 Extraction

Problem

As it was mentioned, the Twitter API was used to extract the Tweets. However, we faced a lot of limitation from Twitter in the data extraction process. The search API is REST API which allows users to get tweets with specific queries; one of the main limitation was the number of tweets retrieved. In the library that we used, Tweepy, the users are allowed to obtain 900 tweets per 15 minute, which is a very slow rate, considering that we need a large database to train the Machine Learning model. Tweepy requires API key authentication from the user that can be obtained through Twitter Developer Platform. Here are the limitations that Twitter reported in their developer website.

Endpoint	*Requests per 15-minute window*	
	Per app	**Per user**
Tweets lookup	300	900
Users lookup	300	900
Recent search *	450	180

Figure 3.3: Taken from https://developer.twitter.com/en/docs/twitter-api/rate-limits

Just doing a simple math, if we wanted a database of one million tweets, we would need more than 11 days of successful continues downloading. That is creating three apps in our user account and downloading in every app simultaneously. The training set needs a lot data to give accurate results and for the success of the model. Thus, Brexit Dataset from Harvard Dataverse will be used for the training part of the project [1]. To describe this dataset, it contains the public tweets about Brexit, posted between

Jan 2016 and September 2019. It also contains additional attributes including political stance classification, sentiment analysis, and automated account (bot) scores. In total, the number of lines in the tweet list is 50,897,534. Each line corresponds to a unique tweet containing Brexit keyword. Tweet attributes in these files are separated with the '~' character. It has 4 attributes:

- ID of tweet: Long number
- ID of user: Long number
- Sentiment of tweet: Positive, negative or neutral
- Stance of tweet: remain, leave or other

As it can be seen, tweet as a text is not in the attributes of the training set. This is due to the fact that Twitter does not allow storing tweets on a personal device, so they are fetched through a process in Python. Therefore, considering the above limitations, it would take us 589 days of continues downloading to request all the tweets using the tweets' IDs. At the end, the tweet corpus will consist of tweet_id, tweet, sentiment, and volume of tweets per day. The tweets that are published on the same day will be merged together.

Solution

First of all, the Brexit Dataset from Harvard Dataverse had already a sentiment score, but since we do not have access to that process we will be limited to work within the database and never test our model using more recent data. For that reason, we came up with another sentiment analysis which will be explained in the next chapter.

One important feature of the Harvard Dataset was that the chronological order of the tweets. Considering that fact, we needed to download at least a few tweets per day, so we would be able to construct the time series. Since the dates are not visible in the dataset unless you submit a query with the ID, we created a list with 10,000 elements uniformly distributed from the beginning to the end of the dataset. Assuming that each day had the same amount of tweets, we should expect to receive approximately 9 tweets per day.

The problems continued, after almost 3 hours of downloading tweets, we realized that only around 60% of the tweets were deleted or damaged, so the ratio decreased

even more. Since we needed much more than 9 tweets per day, we created 3 more Twitter's users, which required approval, and 3 apps per each account. That way we were able to download 14,400 tweets in one hour, that was equivalent to around 5,000 effective tweets per hour. With a continues working of 56 hours, we could download 280,000 tweets from the Harvard Dataset, getting a little more than 200 tweets per day, which was our threshold to establish a good sentiment score.

3.2.3 Processing and Storage

The dataset of tweet texts include words, symbols, hashtags, URLs, emoticons and also references. As symbols, hashtags, references and URLs will not give useful information for sentiment analysis, they need to be ignored, in other words, removed from the tweet. So, the dataset is left off with words and emoticons. Moreover, each word consists of upper and lowercase characters and there could be some punctuation in the tweet which are not important for sentiment analysis. Thus, there is a need for operation to remove punctuation, getting rid of lower and uppercase character differences and also remove misspellings. This will be done by preprocessing the dataset which will be explained in the next chapter.

4 Natural Language Processing

4.1 Preprocessing

The sentiment analysis mostly cares about the polarity of the analyzed item. In our case, we are analyzing tweets where words and emoticons are showing polarity of the expression. However, tweets consist of images, videos, URLs, usernames, emojis in which it can not show any sign of polarity for tweets. Seeing tweets with these properties have become common unfortunately. Thus, there is a need for cleaning this noisy dataset. Tweet data has to be normalized or modified so as to get a clean dataset that Natural Language Processing tools can work nicely to get accurate predictions. To do that, various pre-processing methods have to be applied which can be seen below [2], [3]:

- Normalize all characters to be lower-case across all our data.
- URLs and usernames are removed from the tweet.
- The number sign (i.e. #) is removed from every hashtag.
- Replace 2 or more dots (.) and 2 or more spaces with a single space.
- Duplicate characters are rid o from their respective word in the tweet.
- Remove stop words such as “at, and, the, a/an, etc. “

Lastly, the tweets are tokenized, that is, each tweet is broken into their words as a vector to use it as in that format for the future processing of the algorithm. An example for preprocessed tweets could be seen below in the next image.

23	2016-01-01 00:00:00	le brexit au centre des vœux de cameron pour 2016	0	neutral	3848
24	2016-01-01 00:00:00	police intercept migrant train tuberculosis sc...	0	negative	5113
25	2016-01-01 00:00:00	im soooo sorry thought someone else happy new ...	0	positive	2243
26	2016-01-01 00:00:00	promise take us corrupt eu please voteleaveeu ...	0	negative	582
27	2016-01-05 00:00:00	probe launched claims migrants charging nhs ca...	0	positive	3191
28	2016-01-05 00:00:00	likely brexit vote coming soon enough	0	neutral	311
29	2016-01-05 00:00:00	eu leaders punish uk brexit threat economic chaos	0	negative	98

Figure 4.1: Example for Preprocessing Tweets Texts

4.1.1 Handling Usernames and URLs

It is really common for Twitter users to use URLs for referencing and usernames as they are reaching someone or some community. However, these features are not useful for sentiment analysis. Therefore, we have replaced all the URLs in tweets with the word URL and replace all @USER with AT_USER [4].

Replacement in Python can be made by Python's Regular Expressions (RegEx) library. This library allows parsing strings and modifying them in an efficient way without having to explicitly iterate through the characters comprising the particular string.After changing these parameters, we have added those to the 'stopword' dictionary which will be explained in the 'Handling Stopwords' part of the report. At the end, those will be removed before sentiment analysis as they do not affect the polarity of tweets.

4.1.2 Handling Hashtags

Hashtags are getting popular each day in social media and generally used for emphasizing to a specific point. Thus, it can be said that they are important for understanding polarity and topic of Twitter. So, we need to remove just the hashtag from the word that is attached to it rather than all words. Basically, when the hashtag is found in a word, first character of that word is (#) removed (ex: # Brexit to Brexit). This can be done by Regex Library.

4.1.3 Handling Stopwords

Stopwords are the English words which do not add much meaning to a sentence. They can safely be ignored from our algorithm without sacrificing the meaning of the Tweet. For example, the words like the, he, have, a, an, etc. We would not want these words to take up space in our database, or take up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. This could be done by usage of the Natural Processing Toolkit that Python provides. Stop words can be filtered out by a few lines of code with this library and the algorithm for polarization can work better.

4.1.4 Handling Emoticons

Emoticons are commonly used before the invention of emojis but they are still being used frequently. Thus, it would be a good idea to use them as a sign of polarity. For instance, ":) , :), :D, :O " could be a sign for positivity or positive move in markets. Also, " :(, :((, :'(" could be a sign of negativity. Some kind of regex transformation also could be done for these but as VADER that is used for sentiment analysis in our project handle emoticons, regex library nor any other library is used for modification of emoticons.

4.2 Feature Extraction and Sentiment Analysis

4.2.1 VADER Sentiment Model

Our dataset will consist of dates and tweets that are merged together corresponding to that day. There is a need for scoring the merged tweets to see how positive and negative the sentiment is in order to train the model. In our application, we would benefit from being able to determine not just the binary polarity (positive versus negative), but also the strength of the sentiment expressed in text. Due to this reason, VADER will be used. VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. Another reason why VADER is good for our project is it does not require any training data but is constructed from a generalizable, valence-based, human-curated gold standard sentiment lexicon. So, we have gained a great amount of

time by using a ready to use training set in VADER. More details about implementation and model of VADER can be seen below at Figure 4.2 in detail. Moreover, complete details and performance comparison of VADER could be read in the [5].

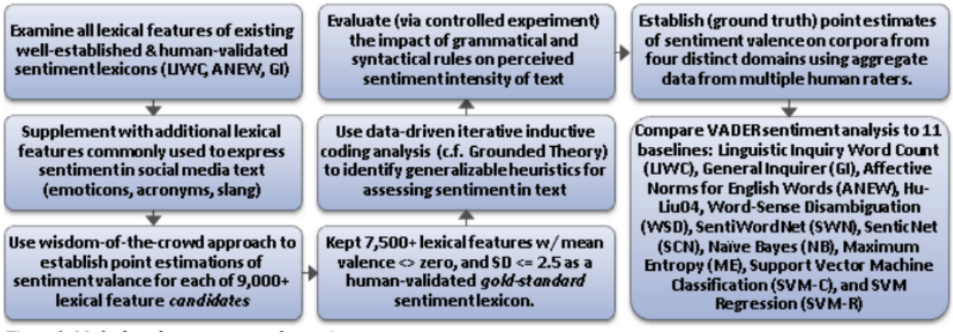

Figure 4.2: Methods and process approach overview [5]

As a result of VADER analysis, we will get output showing the polarity of the sentiment. To clarify, VADER returns a dictionary of scores in each of four categories:

- Negative
- Neutral
- Positive
- Compound (computed by normalizing the scores above)

We are mostly interested in compound score as it indicates the general result of polarity and we are planning to train our model and use compound value in our forecasting section. The Compound score is a metric that calculates the sum of all the lexicon ratings which have been normalized between -1(most extreme negative) and +1 (most extreme positive).

- positive sentiment : (compound score >= 0.05)
- neutral sentiment : (compound score > -0.05) and (compound score < 0.05)

- negative sentiment : (compound score <= -0.05)

An example output from VADER can be observed below which is obtained from the code of the project.

- *Vader score: ('neg': 0.0, 'neu': 0.872, 'pos': 0.128, 'compound': 0.2263)*
- *Vader score: ('neg': 0.0, 'neu': 0.769, 'pos': 0.231, 'compound': 0.4019)*
- *Vader score: ('neg': 0.0, 'neu': 0.732, 'pos': 0.268, 'compound': 0.5106)*

Moreover, we are using another metric for our model that is compound score scaled with follower's count for each user corresponding to tweet. By this way, we are giving more importance to tweets that are coming from users who are more respectable or more accepted from the community. Results for both metrics will be discussed in the results section of this report.

4.2.2 "Volume of Tweets per Day" Feature

Another feature that is added to preprocessed data is volume of tweets per day. Volume of tweets per day is crucial since we are especially interested in volatility of the prices. Basically, we are using volume as another feature for our forecasting model. Volume is calculated by adding the tweets that are sent each day and as the volume for some days can be quite high, log of volumes have been also tested as a feature. The effects of these two features will be discussed in the results section.

At the end of preprocessing of tweets, the obtained preprocessed data set could be observed in the following image.

	neg	neu	pos	compound	compound_multiplied_scaled	volume
2016-01-01	0.141111	0.558778	0.300000	0.243633	0.006302	9.0
2016-01-04	0.141111	0.558778	0.300000	0.243633	0.006302	9.0
2016-01-05	0.107700	0.640600	0.251700	0.245635	0.049407	20.0
2016-01-06	0.107700	0.640600	0.251700	0.245635	0.049407	20.0
2016-01-07	0.107700	0.640600	0.251700	0.245635	0.049407	20.0
2016-01-08	0.158161	0.669161	0.172710	0.027365	0.016389	31.0
2016-01-11	0.141150	0.629100	0.229700	0.146015	0.029081	20.0
2016-01-12	0.141150	0.629100	0.229700	0.146015	0.029081	20.0

Figure 4.3: Example for Preprocessing Tweets

An important point that should be considered from Figure 4.3 is that there are some missing dates. Those missing dates correspond to the weekend dates and those dates are removed as trading days do not include weekend days. For that reason, tweets texts from weekends are added to friday's tweet texts so that friday's sentiment score will be obtained from friday, saturday and sunday's tweets.

5 Volatility Prediction

In this section, we will forecast the rolling window volatility of one month one day ahead. As we mentioned in the previous chapters the model implemented will be the Long Short-Term Memory model that belongs to the Recurrent Neural Network type of models, because it has a time component feature which gives the model a good understanding of the evolution of the features along time. The main advantage of the LSTM is that keeps in its memory important information that happened several steps ago, rather than overweight the most frequent information as most of the Recurrent Neural Network models do.

Also in this chapter we will see some features engineering, what type of innovations did work, what others did not, along with the final training data set and selected neural network model.

5.1 Data

Before any calculation or transformation the data from the finance part and from the tweets side were both divided in training and testing data. The proportion was 90% for the training and 10% for the testing.

5.1.1 Financial

From the financial data we have 3 time series, the GBP prices, its returns, and the rolling of 21 business days standard deviation. From the last one, we will extract also the target variable or the Y. In the following table is shown this financial data in the format of Pandas DataFrame:

Date	Vol_Index	Price_Index	Returns_Index
2015-01-30	0.007165	99.898087	0.000013
2015-02-02	0.007164	99.830014	-0.000682
2015-02-03	0.007109	99.307926	-0.005243
2015-02-04	0.007065	99.534349	0.002277
2015-02-05	0.007072	100.198768	0.006653

Figure 5.1: Data received from the financial part.

5.1.2 Twitter

On the other hand, from the data that is coming from Twitter and the NLP process, we have the 3 main scores: Positive, Neutral, and Negative; the compound score which is obtained only using the last 3 scores in the same day; the volume, which is the number of tweets used that day; and finally the compound multiplied and scaled score. It is important to highlight that the last score is using the sklearn.preprocessing.StandardScaler method to scale the factor, this method records the values obtained in the training data and applied them to the test data, avoiding the look ahead biased.

On the training dataset, the results of these features and calculation are shown in the following table:

date	neg	neu	pos	compound	compound_multiplied_scaled	volume
2016-01-01	0.141111	0.558778	0.300000	0.243633	0.006302	9
2016-01-05	0.107700	0.640600	0.251700	0.245635	0.049407	20
2016-01-08	0.158161	0.669161	0.172710	0.027365	0.016389	31
2016-01-11	0.141150	0.629100	0.229700	0.146015	0.029081	20
2016-01-14	0.154590	0.663077	0.182308	-0.003656	-0.068832	39

Figure 5.2: Data received from the Twitter part.

5.2 Transformations

5.2.1 Features Engineering

From this point of the prediction process until the results, it was more like an iterative process, rather than a step by step. The features engineering applications were created along with the training process, sometimes adding more information to the neural network, and other times passing only a few features.

That said, some of the new features created in this process were for example, adding a long term performance of the index. This value consisted in the return of the past 3 months, which it was longer than the lookback period of the Recurrent Neural Network, so maybe it adds some value in the prediction process. We incorporated this feature because investors are always seeing the past returns or the performance in the past months to see when is the right time to invest, so giving that information to the model could be a good improvement.

5.2.2 Final Features Selection

After several iterations, changing the model structure and the features established, we realized that for this small training data, the more information we give to the model, the worse the prediction is. Our analysis about this is based in the idea that the best naive forecast of the volatility tomorrow is the volatility of today. First because we are using a rolling window and the value is changing because of two different over 20 equal returns.

For example, if we give to the model all the features, the results are much worse than the naive approach, and even worse than a perfect prediction. In the following chart there is this prediction where the level was right, but the magnitude or the volatility of this volatility was much lower than the historical values.

Figure 5.3: Not good performance using all features.

Therefore, the final selection was just using the volatility index, which is just a lag of order one from our target variable, and the compound score from the tweets data. This simple combination ended up being the most accurate and stable, the model recognize the high importance of the past value and since it was better than just volatility, we conclude that it is using the news information accurately.

5.3 Training

We have not talked yet about the model chosen for this forecast, because the LSTM is just the type of the Neural Network, but the are still things to determine in order to have a proper prediction model. First, we have to decide how many layers in the NN model, so first we started with one LSTM model and a Dense layer which corresponds to the regression. Then we tested the model with two sequential LSTM layers. After many iterations with different set of features, we realized that a model with one layer has systematically better performance than other models, regardless the amount of neurons.

Finally, we ended up using two layers: one LSTM of 50 neurons, and a Dense layer. Given that, the features flow as follow: the initial X data has 10 days lookback and 2 features, this means that for one day the model is having 20 values to fit. After the LSTM layer, the data has no past periods but now it has 50 new features. Finally, the dense layer that makes the regression and returns just one value.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_25 (LSTM)               (None, 50)                10600
_________________________________________________________________
dense_25 (Dense)             (None, 1)                 51
=================================================================
Total params: 10,651
Trainable params: 10,651
Non-trainable params: 0
_________________________________________________________________
```

Figure 5.4: Machine Learning Model Structure.

In the previous plot we can see how many parameters the entire model has to fit. In this case the model had to fit 10,651 parameters, which is a huge number for only 849 days of training data.

To test each model and iteration, we have to check first if the model is learning through each epoch, for that reason we make the following graph to see how the result from the error function is changing over iteration.

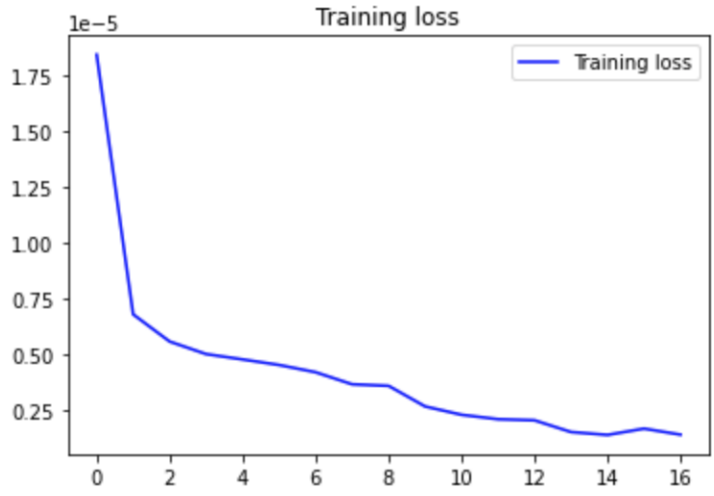

Figure 5.5: Training Loss chart.

We can clearly see how this total error value is decreasing significantly after each epoch, which tells us that the model is working correctly.

However, with the mean squared error is hard to see if the model is predicting good or not. 0.25 * 1e-5 could be a very bad performance for the model or it could be a very good fit. Therefore, the best way to see if the model is performing correctly is again with a historical tracking performance:

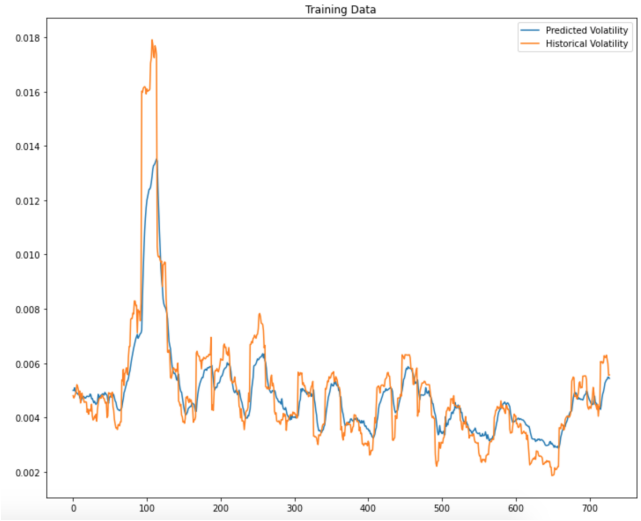

Figure 5.6: Volatility prediction on the training dataset.

This time is easy to conclude that the model is getting the right values for volatility. Nevertheless, we have to compare it with the naive approach which tells us that the best future value for volatility is the present value of it. We will see those results in the following section.

5.4 Results

One of the most important things in machine learning is to be careful with the over fitting. In this model we were using a lot of parameters, which can perfectly fit the entire curve. However, the cross validation process in the calibration helped us to prevent this type of problems, but we still need to check the results in the out of sample data.

For that reason, we will show now the performance in the validation data and also in the test data. The last one, it was the first time that it was used, in the selection

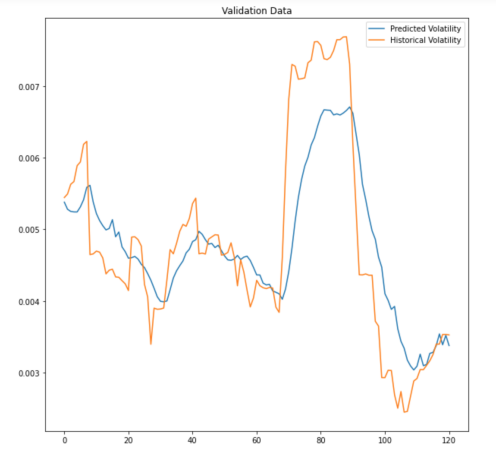

Figure 5.7: Validation Data.

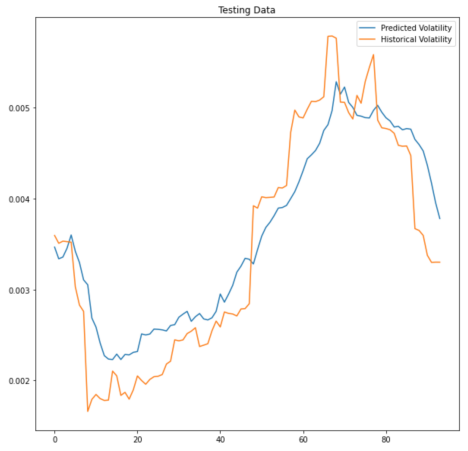

Figure 5.8: Testing Data.

process we did not touch this data, so the purpose is to determine now if the model worked after all.

The performance in both dataset are very similar, which is good, meaning that our model is general enough to keep performing the same. In addition, in both graph we believe that the model did a great job forecasting the volatility.

We mentioned the naive approach, which only repeat the last value. In the last plot, it will be the same line shifted to the right, but we wanted to have a metric which tells us if we beat the naive method or not.

In our search, we thought that maybe a classification metric could be a better, so we can transform our regression problem into a classification. That said, we tried to measure if the model at least predict the direction of the change in volatility correctly. In other words, if the volatility of the tomorrow will be greater or smaller than the volatility of today. Then, we can compare both model using the accuracy metric.

Of course, the comparison has to be in the testing sample. The results for both strategy are very good, but our model performed a little better, with 60.3% of accuracy and the naive method 59.1%.

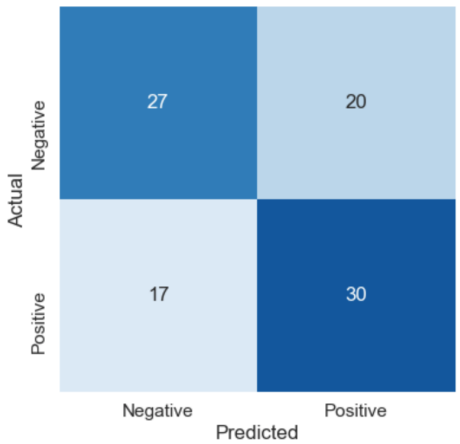

Figure 5.9: Test Data with Selected Model.

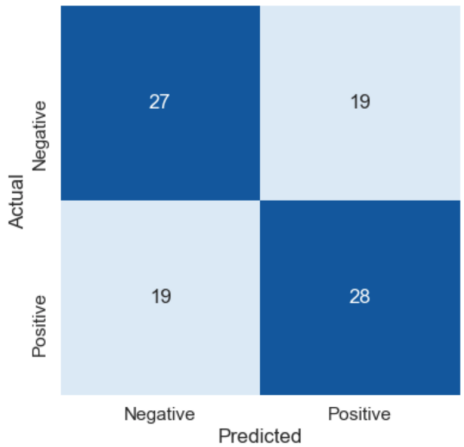

Figure 5.10: Test Data with Naive approach.

6 Conclusions & Outlook

This project performs a complete prediction process, using alternative and not conventional data, and encompassing complex calibration methods to determine a future value of our instrument. In particular, we focused on predicting volatility, which we can clearly observe that it has a rough and cluster behavior over the time. These features makes this parameter much more interesting to forecast that for instance, returns. For that reason, we chose a financial asset and predicted its volatility using a Long Short-Term Memory model. To make this process a little different from previous research we added to this forecast an additional and non trivial set of information, the Tweets. This data in words is extremely valuable, it keeps a record of the market sentiment, people share their ideas and high authorities of the world post their opinions. Most of the traders and investors keep track of these tweets frequently, but we believe that this process can be automate and incorporated in a prediction model.

Tweets about a company could reflect some changes in the price, but companies have a intrinsic value that supports them. On the other hand, the Forex market does not have those fundamental, the countries does not have a balance sheet, so most of the movements are purely speculation, since we know that economic indicator have a huge lag from the real life. Finally, we wanted to find a topic, which people talk and could be directly related with a country, otherwise, tweets about a country could be any tweet, making it difficult to track. For that reason, we wanted to analyze the Great Britain Pound, with its long Brexit process, so the tweets information could be found more easily, and for sure it is a topic that is directly correlated with the currency's performance.

To achieve this, we used a Harvard dataset about Brexit that has 50 million tweets about Brexit. Moreover, Yahoo Finance has been used as a data source for price information. It is thought that daily sentiments and volume change effects could be observed more significantly in volatility of GBP indexes rather than price index. Thus, volatility of GBP forex indexes have been calculated by 21 day rolling windows.

In our results, the one layer LSTM with 50 neurons and a Dense layer worked great. The Neural Network model could replicate the volatility behavior efficiently

and it beat the naive model of forecasting the next value with the present volatility. This model was one of the simplest models that we tried, and also the features used were a very small proportion of all the information available in our database. One good conclusion about this model, was that we believed in the beginning of this project that the volume or amount of tweets about the topic would be useful in the prediction accuracy. However, this time, the volume feature always made the results worse. One possible explanation could be that the database was manipulated, and maybe the amount of tweets where not representative to the amount of tweets available for those days. Another possibility could be that our tweets selection method from the database distorted the actual amount of tweets for a specific date, having later a non representative value. Finally, and our favorite explanation is that the problem is with the discrete number that counts, maybe if we would created a continues scaled indicator, perhaps the ML model would performed better with it.

Feature sets were needed to be preprocessed as the original tweet texts include hashtags, images, videos, URLs, usernames, emojis, stop words which can not show any sign of polarity for tweets. Hence, these are removed from tweet text and then VADER analysis has been implemented to get compound scores. It has been observed that VADER is such a great tool for social media analysis and time series prediction because it allows to get intensity scores rather than polarity classification such as negative and positive. Also, it goes through all tweets by considering emoticons and it has a ready to use training set which is a generalizable, valence-based, human-curated gold standard sentiment lexicon.

Our model is not perfect and it is open to improvements in the future. Firstly, although twitter dataset is really large, it could be increased even more as time series prediction requires large amounts of data. One thing that could be improved is that VADER is a tool that could understand some of the punctuation positivity or negativity such as “OMG!”. So, there could be some change in the preprocessing part for differentiating between unnecessary and necessary punctuations.

7 Work Distribution

As a project team, we have tried to distribute work evenly among the each team member and also, work has been distributed according to the strengths of each team member. Each team member was responsible for literature review in order to come up with solutions to the project. After the literature review process, possible methods are discussed and then, the work has been distributed for the coding part of the project. The responsibilities of each team member could be seen below:

Cristobal Gonzalez: Preparation of Price & Tweet dataset, Volatility Prediction with LSTM Implementation.

Deniz Kural: Preparation of Tweet Dataset, Preprocessing of Tweets, Feature Extraction and Sentiment Analysis.

Bibliography

[1] Calisir, Emre; Brambilla, Marco, 2020, "Twitter dataset about Brexit", https://doi.org/10.7910/DVN/KP4XRP, Harvard Dataverse, V1, UNF:6:5wutKhAU/2JSByEfCbI2Tg== [fileUNF]

[2] Patel, Nirali (2016). Analysing Stock Market Movements Using Twitter Sentiment Analysis. https://github.com/jaiminsanghvi/Analysing-Stock-Market-Movements-Using-Twitter-Sentiment-Analysis

[3] Al-Masri, Anas. (2019, Feb 13). Creating The Twitter Sentiment Analysis Program in Python with Naive Bayes Classification. Retrieved from https://towardsdatascience.com/creating-the-twitter-sentiment-analysis-program-in-python-with-naive-bayes-classification-672e5589a7ed?gi=d366a0a2bcd6

[4] Fatir,Abdul (2017, November 12). Twitter Sentiment Analysis. Retrieved from https://github.com/abdulfatir/twitter-sentiment-analysis

[5] Hutto, C.J. and Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.