Predicting Volatility in Equity Markets Using Macroeconomic News

1. Introduction

On June 26, 2015, months of debt negotiation between the Greek government, headed by Prime Minister Alexis Tsipras, and its creditors, including the IMF and fellow Eurozone countries, broke off abruptly. Tsipras announced a snap referendum regarding the terms of the pending bailout. By the following morning, S&P500 was down significantly. As political and economic uncertainty grew in Europe, investors across the world were moving their funds away from risky assets that could be negatively affected by a Greek default, and towards safer assets. Days later, when the Greek situation had 'resolved', the process reversed as equities rallied.

These market movements are not uncommon; just weeks later, crisis erupted in the Chinese equity markets, and asset values were responding to the uncertainty this provoked. In this paper, we investigate how macroeconomic sentiment immediately impacts the volatility in liquid markets, by measuring market volatility through the VIX (volatility index of the S&P 500). Using data pulled from Twitter, we are researching how breaking economic news affects the markets, and subsequently how to predict which news stories can increase volatility. We will consider a tweet 'significant', in that the news presented in the tweet contributes to volatility in the market, if within 30 minutes of the tweet being tweeted, the volatility of the asset increases by one-fifth of a standard deviation.

By employing three machine learning techniques for classification, we aim to predict the volatility-inducing power of a single tweet and/or news story. Using a training set, we will identify key words and their associated probability of increasing volatility in the markets using Naive Bayes, SVM and logistic regression. The input of our algorithm will be the word count of each bucket of tweets in a dictionary we created. We then use the three prediction methods to output a predicted increase in the VIX, which will be a binary variable (does it increase by a fifth of a standard deviation or not). In order to create a viable trading strategy, we aim to predict moves with above 51% accuracy.

2. Past Work

There is a lot of research on predicting market price movements by conducting technical analysis using historical and time-series data [5]. A large portion of the studies fall into developing techniques to analyze markets trends from texts such as financial market news [5] or social media posts [7]. Some tactics include tokenizing each article, and matching them with stop word lists [2], using a subset of article terms as features [4], and using tagging of named entities to group nouns into predefined categories [6]. The latter two techniques are more frequently used in Question Answering (QA) systems, where themes have been predetermined [6]. However, to analyze the sentiments within markets without knowing the objectives, the lexicon-based approach is a more robust technique, although it is subject to the effect of words around a single word, and trends can sometimes be hard to detected by using single-word weighting [2].

Once texts have been dissected into data sets, there are several machine learning approaches to make predictions in price movements in various financial products. One is Support Vector Regression, which performs linear regression in the high dimension feature space to predict stock prices [3]. Logistic Regression and Neural Networks are also plausible in analyzing stocks trends [1]. However, all these methods are sensitive to bias and noise, and the study performances tend to improve as more noises are removed from original data sets. In addition, the majority of past research focus on price prediction of specific stocks, and very few of them focus on the volatility of the overall markets. We have taken inspiration from various article and the techniques used, such as SVM and logistic regression, but have elected to view this is a classification problem since we are not focusing on single-name stocks but rather market volatility.

3. Dataset and Features

We have pulled macroeconomic news from Twitter; news sources will tweet a headline and the link to the accompanying news article, which allows for us to access the key topics frequently and in a rather condensed format. In addition, hedge funds, banks, and analysts will frequently tweet either articles or short views on the market. We have selected 70 accounts that we deemed relevant, and tweeted with a significant frequency. The Twitter API allows for the most recent 3200 tweets per account to be exported from the website, which provides at least a month of tweets (and hence, news) per account. Because we are investigating the immediate market reaction, and therefore using intraday data, this is more than sufficient. This amounted to more than 200,000 tweets. Bloomberg retains intraday data, at 30 minute intervals, for 6 months, which we have also pulled. We bucketed tweets into thirty minute intervals, so that instead of regarding an individual tweet as a single data point, we are looking at clusters of tweets. The aim of this strategy is to reduce the noise in the model and reduce the amounts of false classification. For instance, if CNN breaking news is tweeting about Kanye West having a baby at the same time that oil prices begin to plummet, both would be associated with moves in volatility. The Kanye West tweet has no impact on moves in the S&P 500, but if treated as its own data point, the model might believe that 'Kanye' was a key word. We consider only tweets between 9 am and 4pm to align with the market intra-day data.

We also constructed our own dictionary, based on our universe of tweets, rather than using a pre-implemented dictionary. Because of the hashtag feature on Twitter, we suspect that there are key words that will affect market movements that are not real words (such as #Grexit) and are commonly used to unify a particular topic. Similarly, analysts often will use the stock ticker in a tweet when discussing an update about a single-name stock (such as $AAPL). By constructing our own dictionary, we ensure that these trending topics are included, which can perhaps be even more significant than actual english words. We cleaned the tweets by eliminating all characters that were not letters, and eliminated URLs. The dictionary that we used contains just over 10,000 words, which serves as our features for the algorithms.

After the processing of data, we reduced our 200,000 tweets to 988 data points. Each data point is a vector the length of the dictionary, corresponding to the frequency of each token (feature) in that block of tweets. Because we are using time-series data, we cannot use cross validation techniques, so we use 70% of the data (692 points) for training and 296 points for testing.

Figure 1 shows a plot of the VIX since 2008 (daily data). While we used intra-day data going back 2 months, this illustrates how events, such as the Greek financial crisis, the Lehman Brothers default, the devaluation of the yuan, and other macro events effect the volatility.

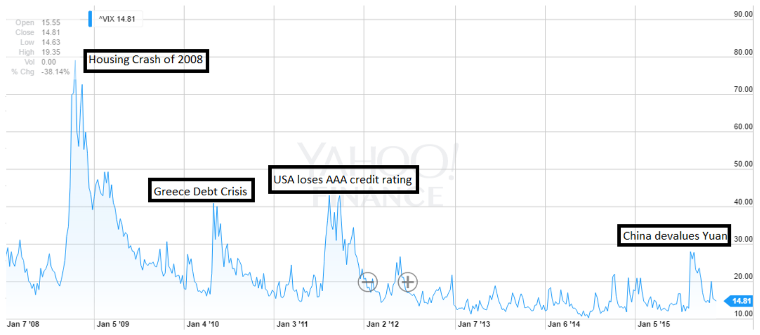


FIGURE 1.

4. METHODS

4.1. **Naive Bayes.** The first classification model we consider is the Multi-factor Naive Bayes. The key assumption of this model is the conditional independence of the features. This amounts to the following assumption: for any features x_i and x_j, given that $y = 1$, i.e. that the VIX has gone up substantially, then the knowledge of x_i has no effect on our beliefs about x_j. Mathematically, we say:

$$\mathbb{P}(x_i, x_j | y = 1) = \mathbb{P}(x_i | y = 1, x_j)$$

This is a strong assumption to make about our Twitter data set. The Naive Bayes is used as a preliminary classification, but we explore two other methods that relax this assumption.

We want to fit the following parameters: $\phi_{k|y=1} = \mathbb{P}(x_k|y=1)$, $\phi_{k|y=0} = \mathbb{P}(x_k|y=0)$ and $\phi_y = \mathbb{P}(y)$, for all k features. In order to fit this model, we maximize the log-likelihood function to find the estimates:

$$L(\phi_y, \phi_{k|y=0}, \phi_{k|y=1}) = \prod_{i=1}^{m} \mathbb{P}(x^{(i)}, y^{(i)})$$

We additionally apply Laplace smoothing to avoid issues when we arrive at a word that is unknown in our dictionary. We have the following estimates:

$$\phi_{k|y=1} = \frac{\sum_{i=1}^{m} \mathbf{1}(y=1)n_{k,i} + 1}{\sum_{i=1}^{m} \mathbf{1}(y=1)n_i + N}, \quad \phi_{k|y=0} = \frac{\sum_{i=1}^{m} \mathbf{1}(y=0)n_{k,i} + 1}{\sum_{i=1}^{m} \mathbf{1}(y=0)n_i + N}, \quad \phi_y = \frac{\sum_{i=1}^{m} \mathbf{1}(y=1)}{m}$$

where m is the number if training points, $n_{k,i}$ is the frequency of the k^{th} token in the i^{th} training point, n_i is the total number of tokens in the i^{th} training set, and N is the number of tokens.

Having fit these parameters, we make a prediction on a new example with given features x by calculating the ratio

$$\frac{\mathbb{P}(y=1|x)}{\mathbb{P}(y=0|x)}$$

4.2. **Support Vector Machines (SVM).** Subsequently, we employed the support vector machine method, which performs linear regression in the high dimension feature to maximize the functional margin. Given a training set $S = \{(x^1, y^1), \cdots, (x^n, y^n)\}$, for a linear regression model with the linear function$f(x) = \langle w, x \rangle + b$,we want to find the weight vector $\mathbf{w}$ that solves the following problem

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{2}||w|| \\ \text{subject to} \quad & y^{(i)}(\langle w, x^{(i)} \rangle + b) \geq 1, i = 1, \cdots, m \end{aligned}$$

where m is the number of training sample. We can solve the dual problem of the above optimization as

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j K(x^{(i)}, x^{(j)}) \\ \text{subject to} \quad & \alpha_i \geq 0, i = 1, \cdots, n \\ & \sum_{i=1}^{m} \alpha_i y_i = 0, i = 1, \cdots, n \end{aligned}$$

where $K(x^{(i)}, x^{(j)})$ is a kernel function, which we have chosen to be Gaussian, and m is the number of support vector.

For our project, we use the *LIBLINEAR* package to perform the SVM. To use the built-in functions in the *LIBLINEAR*, we constructed our training sets and test sets into sparse matrices of size $(a \times b)$, where a = # of tweets and b = # of tokens. Each row of the matrix represents a tweet; the j^{th} column of the i^{th} row represents the number of times the j^{th} token appeared in tweet i. This is a sparse matrix, and therefore we only record non-zero values along with their index numbers. VIX going up is indicated as class +1, and VIX not trending as class -1.

4.3. **PCA and Logistic Regression.** Finally, we performed logistic regression. Logistic regression makes the least amount of assumptions on the dataset, and since we cannot be sure that the assumption of conditional independence holds true in the Naive Bayes algorithm, logistic regression makes sense. Logistic regression is a model that measures the relationship between a categorical binary dependent variable, which we have taken to be whether or not the VIX has increased by a certain amount, and the independent variables (the features of our dictionary, from the tweets). It estimates the probabilities of the categorical variable using the logistic function.

Our hypothesis has the form $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$, the sigmoid function. We assume that $\mathbb{P}(y=1|x,\theta) = h_\theta(x)$. Letting n be the number of features in our dictionary, $\theta \in \mathbb{R}^{nx1}$ is the vector that we are trying to fit in from out training sample, and $x \in \mathbb{R}^{nx1}$, such that $x_0 = 1$ and x_i is the count of the i^{th} word in the data point.

Using log-likelihood methods, we should be able to fit the parameters θ. However, we have 10,000 features and only 988 data points, which makes this impossible. Therefore, we need to employ dimension-reduction techniques to make this a feasible strategy.

Principal component analysis (PCA) is a procedure that reduces the dimension of a set of observations (tweets, for us) by using orthogonal transformations to convert correlated covariates (the features) into a set of linearly uncorrelated variables, know as principal components. The first principal component accounts for as much variance in the data set as possible, and as we increase the number of principal components, we increase the amount of variance accounted for. Each principal component is an eigenvector of the covariance matrix of the data set, guaranteeing the orthogonality of the components. Effectively, we can use a number of principal components significantly less than the number of features. We began with 10,000 features, perform principal component analysis, and selected the first 100 principal components to use, reducing our dimension by a factor of 100.

By multiplying the first 100 loadings (a 10,000 by 100 matrix) by our matrix of covariates, we reduce create a new design matrix of size 100 x 988. Using this new design matrix, we perform a logistic regression using the binary VIX as our response. The function 'glmfit' in Matlab inputs the data and outputs the model with fitted parameter θ. The 'predict' function in Matlab, inputs the model created from the training set and tests the model using the 30% testing data by making predictions of the probability of a VIX increase.

5. Results

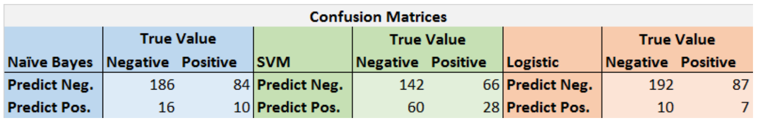
We want to compare the three classification models to determine which one best fits our data. Therefore, we use hold-out cross validation with 70% of our data for training, and the remaining 30% for testing, and compare the estimated generalization error/accuracy of each model. Since the impact of certain words on the VIX changes in time, it does not make sense to train with data that occurs after the testing data. For this reason, we do not use k-fold cross validation, as we need our testing data to be the most recent 30% of the tweets.

The primary metrics for each model that we are interested in are accuracy, precision, and recall. Accuracy is the proportion of correctly classified data points to total number of data points. Precision or positive predictive value is the number of true positives over the total number of positives predicted, or the probability that a positively predicted data point is actually positive. Recall is the proportion of positive data points that are correctly classified. In other words, we have the following formulas:

$$Acc = (TP + TN)/M, \quad Prec = TP/(TP + FP), \quad Recall = TP(TP + FN),$$

where TP stands for true positive, TN true negative, FP false positive, FN false negative, and M number of data points.

We first look at the confusion matrix to get an idea of how each model is performing.



Confusion Matrices

	True Value			True Value			True Value	
Naïve Bayes	Negative	Positive	SVM	Negative	Positive	Logistic	Negative	Positive
Predict Neg.	186	84	Predict Neg.	142	66	Predict Neg.	192	87
Predict Pos.	16	10	Predict Pos.	60	28	Predict Pos.	10	7

Figure 2.

We see from figure 2 that both Naive Bayes and Logistic Regression do a good job predicting negative data points, but do not predict very accurately the positive data points. The SVM algorithm predicts more positive data points than Naive Bayes and Logistic Regression, but also has a lot more fale positives, lowering its precision. These observations are made more precise in figure 3, which compares each of the models main metrics to each other.

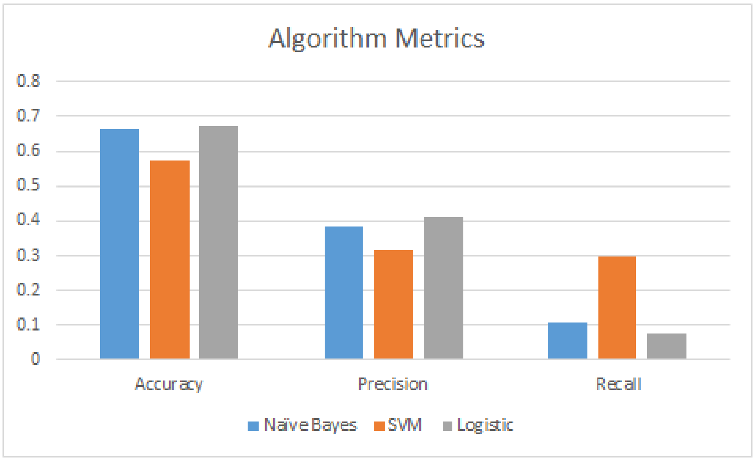


Figure 3.

From figure 3, we see that Logistic Regression has the best accuracy and precision at 67% and 41% respectively, with Naive Bayes trailing with 66% and 38% for accuracy and precision. Although SVM does much better than both Naive Bayes and Logistic Regression in recall, we still consider SVM to be the worst performing model for our purposes. This is because our trading strategy will only enter into a position in the market if we predict a positive result from the data. Thus, the most important metric for us is precision, since we can see it as the probability that the position we enter into will be profitable or not. Therefore, since Logistic Regression has the highes accuracy and precision, we come to the conclusion that it is the best model for our problem.

The SVM algorithm performed much worse than the other two algorithms. This was caused by the noise in our data, and small number of data points. Most of the twitter accounts we used had many tweets that had no impact on the markets at all, but the algorithm still included them in that analysis. This along with the small number of data points led to the SVM algorithm choosing a decision boundary that was not very accurate. The reason that Logistic Regression outperformed the Naive Bayes algorithm is that the Naive Bayes assumption of the conditional independence of the features definitely does not hold in our case. Also, due to the size of the dictionary, it is possible that both SVM and Naive Bayes were overfit. To mitigate this effect, we created our dictionary using a subset of the tweets, before performing our analysis. Due to the fact that PCA was performed before doing the Logistic Regression, it depended on fewer features, lowering the chance that we overfit the model.

6. Conclusion

Using three supervised learning techniques, we have developed a methodology for predicting volatility movements, with an accuracy between 57-67%. The Logistic Regression model outperformed both Naive Bayes and SVM due to less assumptions being made, and a lower chance that the model was overfit. We believe that with more concentrated tweets, and eliminating tweets that are not macroeconomic headlines, or headlines unrelated to the S&P500 movements, the accuracy and precision of the models would greatly increase. Also, by modifying and shorteningthe dictionary to include exclusively marketrelated words and tokens, we would lower our chance of overfitting, and see an increase in accuracy. With more time, we would look into different ways to obtain our data to make it cleaner, such as pulling headlines from Bloomberg. We would also look into tweaks of our models, to address some of the problems they were having, such as using a different kernel for SVM, or adding regularization to account for outliers.

References

[1] T. Ding, V. Fang, and D. Zuo, *Stock market prediction based on time series data and market sentiment*, 2013.

[2] T. L. Im, P. W. San, C. K. On, R. Alfred, and P. Anthony, *Impact of financial news headline and content to market sentiment*, International Journal of Machine Learning and Computing, 4 (2014), p. 237.

[3] P. Meesad and R. I. Rasel, *Predicting stock market price using support vector regression*, in Informatics, Electronics & Vision (ICIEV), 2013 International Conference on, IEEE, 2013, pp. 1–6.

[4] D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu, *Performance issues and error analysis in an open-domain question answering system*, ACM Transactions on Information Systems (TOIS), 21 (2003), pp. 133–154.

[5] R. P. Schumaker and H. Chen, *Textual analysis of stock market prediction using breaking financial news: The azfin text system*, ACM Transactions on Information Systems (TOIS), 27 (2009), p. 12.

[6] S. Sekine and C. Nobata, *Definition, dictionaries and tagger for extended named entity hierarchy.*, in LREC, 2004, pp. 1977–1980.

[7] M. S. A. Wolfram, *Modelling the stock market using twitter*, School of Informatics, (2010), p. 74.