2022 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr) | 978-1-6654-4234-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CIFEr52523.2022.9776196

Applying Sentiment Analysis, Topic Modeling, and XGBoost to Classify Implied Volatility

Farshid Balaneji
Faculty of Business and Economics
University of Basel
Basel, Switzerland
farshid.balaneji@unibas.ch

Dietmar Maringer
Faculty of Business and Economics
University of Basel
Basel, Switzerland
dietmar.maringer@unibas.ch

***Abstract*—*Implied volatility* is an important indicator that shows the market participants' expectations about the future fluctuations in the options market. This paper evaluates the question of whether the combination of topics and sentiment scores extracted from mainstream financial news could improve forecasting the directional changes of the expected implied volatility index in the next month (iv30call). We select six stocks from the Dow Jones list of companies and acquire over 190,000 news published between January 2019 and September 2019. By building text processing and topic modeling pipelines, we can examine (i) the role of daily mean and medium of sentiment scores; and (ii) the influence of topic models on the classification metrics. The results demonstrate that adding a topic model has a positive effect on the model's accuracy, which reaches higher accuracy in classifying the iv30call of the next business day in five out of six companies. The outcome suggests that applying the mean of the daily sentiment scores improves the models' accuracy compared to the daily median for the selected assets.**

***Index Terms*—sentiment analysis, implied volatility, topic models, financial news, XGBoost**

I. Introduction

A. Motivation

Information availability for stakeholders is one pillar of the Efficient Market Hypothesis (EMH) [1]. According to the semi-strong form of EMH, all public information such as past stock price movements and financial news is already reflected in the price of financial assets. Thus fundamental and technical analyses should not help predict a stock's future price change. This model attracted a significant amount of attention since its presentation and was challenged in the 1980s when some literature in time series analyses had witnessed excessive volatility of stock prices relative to what would be predicted by the EMH. As many inconsistencies and deviations from normal market conditions were noted, the research area of behavioral finance was accelerated in the 1990s [2].

In recent years, an abundance of public news and messages in social networks on one side and advances in text analysis and machine learning methods on the other side shed more light on the impact of news and social media on market movements. One major obstacle to investigating financial news and its role in the markets is the unstructured form of text data, which has posed numerous challenges to analysts.

Information in the news is certainly not random sequences of characters, and in any case, it is not simply possible to normalize text like numerical data. The word order and meaning of them cannot be captured either. The challenge here is to map text data to quantifiable measures that could easily be combined with financial time series like market return or volatility. Sentiment category and score is one of such measures that is widely used in financial applications. The relationship between sentiments from financial newswire [3], social media [4], and market time series is well investigated.

Most of the former studies boil down information in financial texts to a sentiment score and disregard valuable content about associated entities mentioned in the document. For example, consider the following headline from an online article on USA TODAY [5] that was published on May 25, 2018,

> "Samsung ordered to pay Apple nearly $539 million in damages in a longstanding patent dispute."

Following the sentiment analysis approach followed by Kordonis, Symeonidis, and Arampatzis [6] who applied a popular Natural Language ToolKit (NLTK) [7], this statement is categorized as neutral. Nevertheless, according to Yahoo Finance, Samsung Electronics Co Ltd (005930. KS) lost 6.07% of its value five days after the news announcement. However, Apples' stock (AAPL) slid down at the beginning the closing price was 1% higher at the end of the same period. Finding the contribution of this specific news on the loss and gain of stakeholders is not a straightforward task.

This example shows that sentiment score, per se, does not provide a complete picture of news articles. There is a gap in the literature to investigate the combined impact of sentiment and discussed financial news topics in the implied volatility of equity options.

In this paper, we aim at filling the gap, and the main research question is: does combining sentiment of news with topics assist us to explain changes in options implied volatility? The answer to this question is worthwhile with practical implications. Firstly, it will help to decouple the role of sentiments from topics in a text. Additionally, the answer paves the way to find out what content makes news momentous and will support investors in making decisions about their portfolio structure.

The main contribution of this study is to suggest a framework for forecasting implied volatility in the light of text mining techniques with compared results. We accomplish this objective by considering the following items collectively,

- comparing the results of the widely used VADER and TextBlob sentiment analysis methods,
- exploring the joint role of topic modeling and sentiment analysis in classifying the next day's implied volatility, and
- investigating the role of mean and median daily sentiments on the model's accuracy.

This framework that includes comparisons between widely used open-source libraries could be adapted to other classification and regression applications in finance reliant on text data.

The remainder of this paper is structured as follows to address the outlined research gap concerning the role of sentiment analysis and information extraction in the volatility of financial markets. First, related work on the application of NLP in finance is reviewed. Section II articulates sentiment analysis, topic modeling, and classification algorithms applied in the framework. Section III introduces financial time series data and financial news sources. Section IV details how the introduced algorithms fit into the pipelines to generate features from input data; and Section V reports the final results from the classification algorithm with some ideas for future works.

B. Literature Review

A tremendous amount of studies has investigated the extent to which textual data in finance correlate with stock market prices [8], [9].

In pioneering research, Tetlock [10] considers the Wall Street Journal's "Abreast of the Market" column over the 16 years and creates a measure of media pessimism using the General Inquirer (GI) categories from the Harvard psychological dictionary. He finds a two-way relationship between the media pessimism factor and market returns. The results testify that high media pessimism predicts downward pressure on market prices, and low market returns lead to high media pessimism.

Loughran and McDonald [9] argue that the GI dictionary classifies as many negative words such as tax, costs, expenses that are neutral in a financial/economic field, thus creating their dictionary that includes 353 positive and 2337 negative words.

Financial literature also adapts novel algorithms with advances in deep learning and its wide applications in Natural Language Processing (NLP). In [11] authors present a deep learning framework that train Convolutional Neural Networks on annotated financial headlines and financial tweets as the training dataset.

The sentiment of financial texts proved to be an influential factor in predicting stock returns and volatility; nevertheless, it is not the only measure applied in financial prediction. Our study belongs to a group of literature that applies a text mining approach to create features from financial documents to investigate a relationship with stock movements.

In an attempt to combine sentiment score with topics [12], authors study social media messages and create a topic-sentiment feature, which describes the sentiments of the specific topics of the company (product, service, dividend), and utilize this feature for prediction of stock price movement. The study shows that the topic-sentiment feature outperforms models that only consider sentiments or topics. However, their sentiment relies on the annotation of users who post messages on the message board, a method that cannot be adapted to financial news.

In a comprehensive study, Audrino, Sigrist, and Ballinari [13] aggregate different types of sentiment scores for financial news from Ravenpack News Analytics. The study makes use of volumes of search engines from Google Trends and information consumption data from Wikipedia as a proxy for attention measures of stock companies. The authors observe that (i) sentiment data is less informative for future volatility when the companies have either small market capitalization or high shares of institutional investors, and (ii) the attention measure has more impact on future volatility than the sentiment scores.

II. Underlying Methodology

A. Sentiment Analysis

One way to define sentiment is to express it as the underlying feeling, attitude, evaluation, or emotion associated with an opinion and represent it as a triple (y, o, i), where y, o and i are the type, the orientation, and the intensity of the sentiment, respectively [16]. Accordingly, sentiment analysis studies people's opinions, sentiments, evaluations, attitudes, moods, and emotions by applying computation methods.

There are two general approaches for measuring sentiment in text.

1) Lexical methodology: this set of methods is constructed upon predefined lists of lexicons that are labeled with a score from $\{-1, 0, 1\}$ for showing its sentiment as negative, neutral, or positive. Then, the sentiment of the text is a category based on the prevalence of negative vs. positive words in the corpus. This approach ignores the order and co-occurrences of words in the text, and in general, does not consider the contextual characteristics specific to the text [17].
2) Machine Learning (ML) techniques: to capture the complexities of natural language, ML-based methodologies are applied to classify the sentiment of a given set of texts. A crucial step in sentiment classification is training an ML predictive model based on large labeled text containing a mapping between textual utterances and sentiment ratings assigned by humans [17], that require a great deal of manual effort.

The lexicon-based and machine learning-based sentiment measurement can be applied at a different level of granularity, such as document-level and sentence-level. We employ two open-source sentiment analysis methodologies to quantify the sentiment of documents.

1) VADER: The first method to extract the the sentiment of news data is Valence Aware Dictionary for sEntiment Reasoning (VADER), a rule-based and lexicon-based framework that accounts for a word's context within the sentence. It assigns a sentiment score to a sentence by aggregating sentiments of words within the sentence.

VADER sentiment performed better or at least equally well when compared against seven sentiment analysis lexicons [18]. According to the documentation on its GitHub page[1], the VADER lexicon was created manually by annotating more than 9,000 token features on the scale from "extremely negative" to "extremely positive." The VADER sentiment analyzer can handle negations, as well as acronyms, slang, and punctuation. For each document, it calculates the compound score that is derived by summing the sentiment scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 (most extreme negative) and $+1$ (most extreme positive).

2) TextBlob: The second sentiment analysis method is TextBlob [19]. It applies the sentiment lexicon and the *pattern.en* sentiment analysis engine [20], that leverages WordNet[2], an electronic lexical database, to score sentiment according to the English adjectives used in the text. When TextBlob runs sentiment analysis on text, it returns a tuple of the form (polarity, subjectivity), where polarity is afloat within the range $[-1, 1]$, which is the value used as the sentiment of news documents.

B. Topic Modeling

A topic model is an unsupervised machine learning model to determine abstract "topics" in the corpus. Topic modeling belongs to the family of probabilistic models and is a text-mining approach that aims to discover the hidden semantic structure in a body of text [21].

The basic idea of topic models (e.g., [22], [23]) is that documents are combinations of topics, where a topic is a probability distribution over words that is individually interpretable and picks out a coherent group of correlated terms.

Intuitively, given that a document like financial news is about Microsoft, we expect to find some words, e.g., Windows, Azure, from the news more frequently. A document typically comprises multiple topics in different proportions. For example, if an article reports only 10% about Microsoft and the rest about Apple, there would probably be about nine times more Apple-related words than Microsoft-associated ones.

This study employs the widely used Latent Dirichlet Allocation (LDA) algorithm from [22]. LDA assumes each document is generated from randomized mixtures of hidden topics, regarded as probability distributions over words. To represent it mathematically, we adapt the following notation from [24]:

- The topics are $\beta_{1:K}$, where each β_k is a distribution over the vocabulary.
- The topic proportions for the dth document are θ_d, where $\theta_{d,k}$ is the topic proportion for topic k in document d.
- The topic assignments for the dth document are z_d, where $z_{d,n}$ is the topic assignment for the nth word in document d.
- The observed words for document d are w_d, where $w_{d,n}$ is the nth word in document d, which is an element from the fixed vocabulary.

The LDA algorithm comprises of (i) the observed variables which are the words of the documents; (ii) the latent variables which are the topic structure; and (iii) the generative process that corresponds to the following joint distribution of the hidden and observed variables,

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\beta_i) \cdot \prod_{d=1}^{D} p(\theta_d) \cdot \left(\prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n}) \right). \quad (1)$$

There are some dependencies in this distribution that define the LDA, like,

- the topic assignment $z_{d,n}$ depends on the per-document topic proportions θ_d, and
- the observed word $w_{d,n}$ depends on the topic assignment $z_{d,n}$ and all of the topics $\beta_{1:K}$.

To infer the hidden topic structure from documents, one needs to compute the conditional distribution of the hidden variables given the documents [24], which is called *posterior* and as follows,

$$p(\beta_{1:k}, \theta_{1:D}, z_{1:D}, |w_{1:D}) = \frac{p(\beta_{1:k}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})}. \quad (2)$$

The numerator is the joint distribution of all the random variables in (1), and the denominator is the probability of seeing the observed corpus under any topic model and is known as the *evidence*. Topic modeling algorithms aim to approximate (2).

One of the popular implementations of plain LDA in Python is Gensim [25], which provides an option to tune the main parameter of LDA, the number of topics in each document. We use the *coherence score* to select the optimal number of topics on each collection of daily news.

A set of statements or facts is said to be coherent if they support each other [26]. Topic coherence measures score a single topic by measuring the degree of semantic similarity between high scoring words in the topic. The UMass measure [27] is one of the coherence measures designed for LDA, which computes the coherence of a topic as the sum of pairwise distributional similarity. UMass is employed in this study to find the optimum number of topics in each news article and defines the score on document co-occurrence:

$$score(v_i, v_j, \epsilon) = \frac{\log D(v_i, v_j) + \epsilon}{D(v_j)}. \quad (3)$$

where $D(x, y)$ counts the number of documents containing words x and y, and $D(x)$ counts the number of documents containing x. Significantly, the UMass metric computes these counts over the original corpus used to train the topic models, rather than an external corpus.

[1]https://github.com/cjhutto/vaderSentiment

[2]https://mitpress.mit.edu/books/wordnet

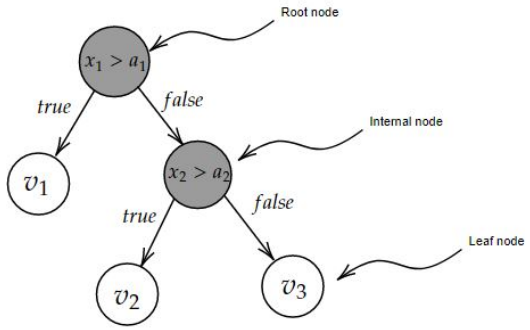

Fig. 1. Illustration of the node split in the CART model.

C. Classification

This study aims to learn the direction of the $iv30call$ in the next business day and tackles the problem by applying the Extreme Gradient Boosting (XGBoost) algorithm [28]. XGBoost is the family of ensemble techniques that construct a strong learner after running weak learners, i.e., models that perform better than random guessing. XGBoost uses a tree ensemble model, a set of classification, and regression trees (CARTs). Each CART comprises a root node, a layer of leaf nodes, and a group of internal nodes between them. At the beginning of the algorithm, all the data are in the root node and based on binary conditions on each internal node, data proceed to the next level, and the tree structure grows. At the end of the algorithm, the leaf nodes denote the final classes (Fig. 1).

Because one tree might not yield good results, multiple CARTs can be applied together, and the final forecast is the sum of individual CART's scores.

The way the XGBoost works is as follow: if we have a dataset $\mathcal{D}$ that has m features and n observations $\mathcal{D} = \{(X_i, y_i)|X_i \in \mathbb{R}^m, y_i \in \mathbb{R}, i = 1, 2, \ldots, n\}$. Let f_k be the kth independent tree, then XGBoost trees uses K additive functions to predict the output,

$$\hat{y_i} = \sum_{k=1}^{K} f_k(X_i), \;\; f_k \in \mathcal{F}, \tag{4}$$

where $\hat{y_i}$ represents the predicted value of the sample i, and $\mathcal{F}$ is the space of regression trees (also known as CART); for more details, see [28].

To solve (4), XGBoost finds the best set of functions by minimizing the regularized objective function $\mathcal{M}$:

$$\mathcal{M} = \sum_i g(\hat{y_i}, y_i) + \sum_k \Lambda(f_k), \tag{5}$$

where g can be any convex differentiable loss function that measures the difference between the predicted and actual value for a given training sample. $\Lambda(f_k)$ represents the complexity of tree f_k and is defined in the the XGBoost algorithm [28] as,

$$\Lambda(f_k) = \gamma\ell + \tfrac{1}{2}\lambda||\omega||^2, \tag{6}$$

where ℓ is the number of leaves of tree f_k and ω is the leaf weights (i.e. the predicted values stored at the leaf nodes). Here γ is the minimum loss reduction required for splitting a new leaf and λ is the coefficient for the regularization term and both are configurable by the user. The term $\gamma\ell$ adds a constant penalty for each increased tree leaf and $\lambda||\omega||^2$ penalises high weights. Thus, including $\Lambda(f_k)$ in the objective function 5, forces the model to optimize for a less complex tree that in the same time minimises $g(\hat{y_i}, y_i)$ therefore reduces the risk of overfitting.

III. Data Sources

This study uses two primary data sources for financial news and options time series to extract the implied volatility.

A. Financial News Data

We select leading corporations from information technology, financial services, and the energy sectors listed in the Dow Jones Industrial Average (DJIA).

The first step revolves around getting news data of the selected companies from Event Registry[3], a leading news intelligence platform that scrapes news and press releases from different publishers.

It is possible to query news based on different criteria such as mentioned keywords, publication date, publisher. One needs to consider different parameters of the query statement to get the desired results from the Event Registry. Firstly, the first option is to query for news of an entity based on a keyword or a *concept*. The *concept* is an annotation that can be assigned to an article, a story, or an event. It could represent entities (e.g., people, locations, organizations) or non-entities (e.g., table, personal computer). Event Registry employs Wikipedia's URLs to identify unique concepts. All concepts have a unique ID called Uniform Resource Identifier (URI). Using the concept URI in the query instead of a keyword is particularly helpful when multiple instances point to the same keyword in other languages or, like Apple, in English. For instance, the concept URI for company Apple is *https://en.wikipedia.org/wiki/Apple_Inc*, while for the apple fruit is *https://en.wikipedia.org/wiki/Apple*.

Table I lists the company name and the last part of their concept URIs. The rightmost column indicates the number of acquired news for each company between January 1, 2017, and September 1, 2019. Dow Jones list changes over time, and companies in this study were part of the index between April 6, 2020, and August 31, 2020, as Exxon Mobil was dropped from the average from September 2020.

The second vital query parameter is the list of news publishers and stock investment research sites. As an input to query news, we selected 28 major financial and stock news publishers, based on popularity and availability, including Reuters, The Wall Street Journal, Bloomberg, Seeking Alpha.

Apart from concept and source, the other query parameters are *lang* i.e., language, which is English; *dateStart* and *dateEnd*

[3]https://eventregistry.org/

TABLE I
THE LIST OF COMPANIES FROM DOW JONES SUJECT TO OUR STUDY (THE FIRST PART OF CONCEPT URIS FOR ALL COMPANIES IS HTTPS://EN.WIKIPEDIA.ORG/WIKI).

Company Name	Symbol	Industry	Concept URI	Number of News
Goldman Sachs	GS	Financial services	Goldman_Sachs	40457
JPMorgan Chase	JPM	Financial services	JPMorgan_Chase	38249
Microsoft Corporation	MSFT	Information technology	Microsoft	29157
Apple Inc.	AAPL	Information technology	Apple_Inc.	68042
Chevron Corporation	CVX	Energy: Oil and gas	Chevron_Corp.	8682
Exxon Mobil Corporation	XOM	Energy: Oil and gas	Exxon_Mobil_Corp.	11237

which are the date range in which a news is published; and *dataType* that in the query set to 'news'. The date range of collected data starts on January 1, 2017 and ends on September 1, 2019.

B. Financial Market Data

Historical Options Prices[4] is an online data provider that collects the end-of-day historical option prices for all U.S. Equity options including stocks, indexes, and exchange-traded funds (ETFs) from 2002. We obtained the historical stock prices and options statistics from Historical Options Prices, providing three tables for each stock symbol.

- *Options*: comprises of fields like the underlying price (*underlyingprice*), the expiration date (*expiration*), the strike price(*strike*), the option type (*type*), bid and ask (*bis, ask*), open interests (*openinterest*) (*calloi, putoi*), the implied volatility (*iv*), and Greeks (*delta, gamma, theta, vega*).
- *Optionstats*: includes the implied volatility index of options in a one-month expiration date (*iv30call, iv30put, iv30mean*) that suggests what volatility is expected to be in the ensuing 30 days. It also provides daily traded volumes (*callvol, putvol, totalvol*).
- *Stockquotes*: contains daily data for stocks (*open, high, low, close, volume, adjustedclose*).

IV. FRAMEWORK: FROM DATA TO CLASSIFICATIONS

Fig. 2 depicts the decision-making system that commences querying daily news from Event Registry and concludes with classifying the iv30call. In this section, we explain each part of the framework.

A. Date Aggregation

Event Registry provides an Application Programming Interface (API) in Python[5] that allows acquiring documents in JavaScript Object Notation (JSON) format. Event Registry provides date and time information when a piece of news is published and parsed. In some cases, publication time is unavailable or cannot be interpreted by the data provider; therefore, it is replaced with the time news parsed and available on the Event Registry. If a document is published after the New York Stock Exchange (NYSE) closing time, or over the weekends or Bank Holidays, the pipeline aggregates it with the news in the subsequent business day. For example, all news published after Friday 16:00 ET and before Monday 16:00 ET will be assigned to the Monday bracket, unless Monday is a Bank Holiday. In that scenario, the news will be aggregated with published articles until Tuesday 16:00 ET, and so on.

[4]For more details on the available data catalog, refer to https://www.historicaloptiondata.com/.

[5]https://github.com/EventRegistry/event-registry-python

B. Text Cleaning

News documents belong to the unstructured data category and require rigorous cleaning and processing pipelines before feeding them into an algorithm. To clean the news body, we apply an open-source library called *textacy* [14] that provides functionality for cleaning and normalizing raw text.

First, we normalize

- the bullet point symbols in a text to just the basic standard character form (ASCII),
- the Unicode characters in text into canonical forms,
- words in a text that have been split across lines by a hyphen,
- repeating characters in text like "--" by truncating their number of consecutive repetitions to 1,
- single and double quotation marks in text to just the basic ASCII equivalents,

and then replace

- currency symbols, hashtags, emails, punctuations, URLs, and numbers in text with white space,
- contiguous zero-width spaces with an empty string and strip any leading/trailing white space.

The above pipeline is the same for sentiment analysis and topic modeling tasks.

C. VADER and TextBlob

The next step is to lemmatize the words, i.e., group words with the same root or lemma but with different inflections or derivatives of meaning to be analyzed as one item. The goal here is to remove inflectional suffixes and prefixes to bring out the word's dictionary form. The lemmatized text is tokenized, both by using Spacy [15] which is a widely applied open-source library for NLP tasks in Python.

The preprocessing concludes with removing stopwords, common words which would appear to be of little value in measuring the sentiment of a document. SpaCy comes with a pre-defined list of stop words in the English language, and we manually remove some words from the SpaCy list that could be important in the sentiment analysis task like "but", "not". Although, in the case of topic models, the list from SpaCy is utilized without any changes. These steps are illustrated in the middlebox in Fig. 2. The first 900 characters of each news go through the VADER and TextBlob algorithms that output sentiment. The reason for not applying the sentiment algorithms on the full documents is that the first paragraphs contain the critical message of the article, and the rest could obscure the sentiment of the news.

Companies could have more than one news article in a day; therefore, we require an aggregation method to boil sentiment scores down one number per business day per company. In this regard, both the mean and median of daily sentiments for

companies are computed and considered the main features of the sentiment analysis step.

D. LDA Algorithm

In the case of topic modeling, we train the LDA model on the corpus of news for each company and tune the number of topics by using the coherence measure. The number of topics starts from two, increases by three, and stops at 18. For each number of topics, the pipeline trains the LDA model, computes the coherence measure, and finally selects the model with the maximum coherence score. LDA outputs two sets of probability distributions for each news article: (i) the set of distributions of topics for each document; and (ii) the set of distributions of words for each topic. For example, a piece of news about Apple could include the company's quarterly report, its market share in operating systems, and new products. Each of these topics will be assigned a probability with aggregated values of 1. These probability distributions are the extracted features from the topic modeling step.

E. Feature Engineering

Feature engineering is an essential part of machine learning workflows. In this phase, the goal is to utilize domain knowledge to select and transform the most relevant factors and improve the performance of the classification algorithm. The main focus goes to features from sentiment analysis and topic modeling algorithms. Therefore, for each day, we consider the following features:

1) topic modeling results: mean of topic distributions,
2) sentiment scores: VADER and TextBlob for the first 900 characters;
3) options data: *iv30call, iv30put, iv30mean, callvol, putvol, totalvol, calloi, putoi*; and
4) equity data: *open, high, low, close, volume, adjustedclose, return*.

For parameters from items 2) and 3) in the above list on date t with var_t, the values of $var_t - var_{t-1}$ and $var_{t-1} - var_{t-2}$ are also computed and added to the list of features in this step. Each company's features amount to 36 features from sentiment and market data and a varying number of features due to the different number of topics.

F. Classification With XGBoost

In this step, the assumption is that at the time of running the model at date t, the options and equity data, sentiment scores, and topic distributions of news at date t is known to us, and the goal is to understand the directional change of iv30call in the next day. When the iv30call increases in the next business day, the dependent variable is +1, and when iv30call decreases, it is -1. Thus, the problem becomes a binary classification problem. The full date range of data for each company starts on January 1st, 2017, and ends on September 1st, 2019. The first 80% of observations are used for training the model, and the last 20% is reserved for validating the model.

Due to the high number of hyperparameters of the XGBoost framework, tuning and model calibration play an essential role

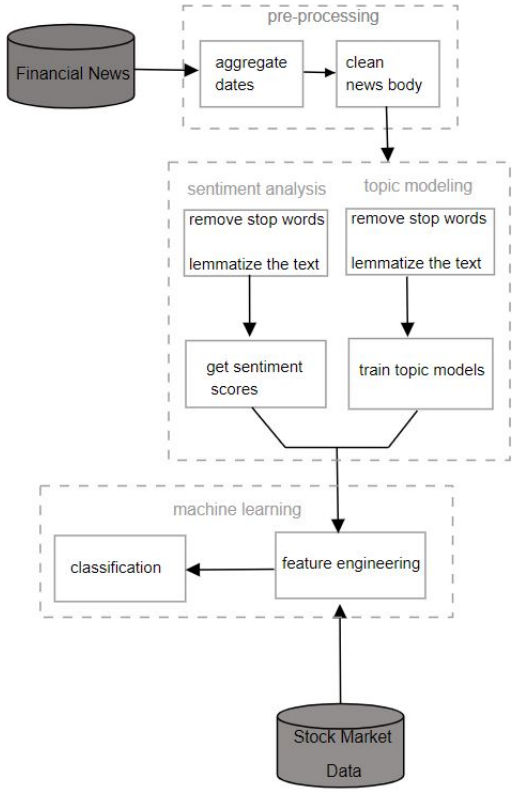

Fig. 2. Illustration of the data pipelines in classifying implied volatility using news and market data.

in the model selection process. We define a range of values for the following parameters and then train and validate the model for each set of parameters, calculate the accuracy of the models and select the model with the highest accuracy.

- eta: step size shrinkage used in the update to prevent overfitting.
- gamma: minimum loss reduction required to partition a leaf node of the tree further.
- lambda: L2 regularization term on weights. Increasing this value will make the model more conservative.
- alpha: L1 regularization term on weights. Increasing this value will make the model more conservative.
- maximum depth: Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit.

We do an exhaustive search by running 27,649 combinations of the hyperparameters mentioned earlier for the mean and another time for the median of daily news sentiments. To consider the impact of topic models, we include the topics distributions in the feature space and train the model. We exclude the topic models in another experiment and only focus on sentiment and features based on market data.

V. Results and Discussion

Table II demonstrates the results of experiments with XGBoost for six companies. Each cell in top section for mean

TABLE II
TABLE OF ACCURACY AND AUC SCORES FOR XGBOOST CLASSIFIER.

	Accuracy			
	With topics		Without topics	
Company symbol	Mean	Median	Mean	Median
XOM	**0.642**	**0.642**	0.619	0.627
AAPL	0.664	**0.672**	0.619	0.634
JPM	**0.709**	0.694	0.672	0.664
GS	**0.716**	0.709	0.687	0.672
MSFT	0.657	0.687	**0.694**	0.687
CVX	0.657	**0.687**	**0.687**	0.664

	AUC			
	With topics		Without topics	
	Mean	Median	Mean	Median
XOM	0.566	**0.592**	0.544	0.541
AAPL	0.657	**0.695**	0.572	0.617
JPM	**0.729**	0.655	0.560	0.623
GS	0.703	**0.723**	0.601	0.695
MSFT	0.644	**0.747**	0.687	0.521
CVX	0.618	**0.640**	0.563	0.525

	A priori for y	
	Training data	Validation data
XOM	0.528	0.537
AAPL	0.541	0.500
JPM	0.556	0.567
GS	0.526	0.522
MSFT	0.509	0.537
CVX	0.529	0.559

and median represents the maximum accuracy of out of samples data after tuning the hyperparameters. Accuracy is the rate of correct classifications divided by the total number of predictions made,

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{7}$$

where, TP (true-positives) are correctly classified up movements; TN (true-negatives) are correctly classified down movements; FP (false-positives) are down movements incorrectly classified as up movements; FN (false-negatives) are up movements incorrectly classified as down movements.

Another popular metric for evaluating the classification algorithms is the ROC (Receiver Operating Characteristic) curve, which plots the true-positive rate as a function of the false-positive rate. The true-positive rate measures the ability of a classifier to find all the positive instances (increase in iv30call). It is also called *sensitivity* and is defined as

$$\text{sensitivity} = \frac{TP}{TP + FN}. \tag{8}$$

The false-positive rate is defined as

$$1 - \text{specificity} = 1 - \frac{TN}{TN + FP} = \frac{FP}{TN + FP}. \tag{9}$$

Plotting the false-positive rate on the x-axis and the true-positive rate on the y-axis yields a curve similar to the ones depicted in Fig. 3. The area under the receiver operating characteristic curve (AUC) represents the ability of the classifier to produce a higher score for an actual positive than an actual negative. A perfect model would score an AUC of 100%, while a random classification would score 50%. Fig. 3 portrays the AUC of the trained model (mean daily sentiment with topics) for AAPL that amounts to 65.7%. The AUC score of mean and median with and without topics are represented in the middle

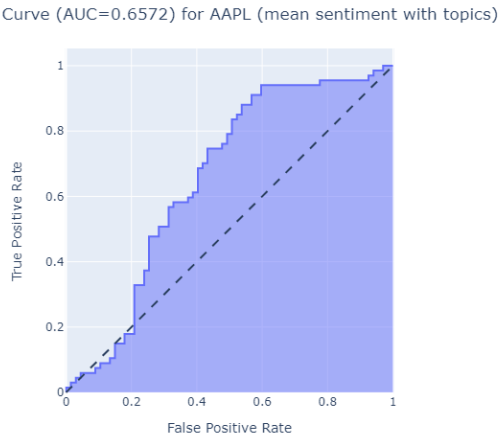

Fig. 3. ROC curve and AUC value for iv30call classification of AAPL.

section of Table II. The results suggest models inclusive of topic distributions as the features have a higher AUC measure than models exclusive of topics.

The column *"A priori for y"* in Table II shows in how many observations from the training and validation data the implied volatility increased (i.e., belong to the positive class). It acts as a benchmark for the goodness of the results. The model accuracy is higher than the a priori of training data between 10%–15%. The results demonstrate that the mean of daily sentiments leads to better accuracy than the median. This outcome was not expected, as using the mean operator, negative and positive sentiments cancel out each other, making a diffuse aggregated sentiment.

The XGBoost framework provides an *important features* table based on different criteria and ranks the input features in order of importance and accordingly takes decisions while classifying the data. Fig. 4 reports the feature importance scores based on *Gain* value that shows the improvement in accuracy brought by a feature to the branches it is on. When compared to another feature, a higher value of this metric implies it is more important for generating a prediction. According to the diagram, the daily closing price has the highest contribution in classifying the direction of the change of iv30call for Apple options. The following important feature is the sentiment from the VADER method for the first paragraphs, and the *topic_0* is also among the ten most defining features, that agrees with the results from Table II that also validates higher accuracy for the model includes the topics in the case of AAPL.

VI. CONCLUSION AND OUTLOOK

In this study, we applied a novel framework that combines the sentiment scores with topic models to classify the direction of changes in tomorrow's iv30call based on today's news. The results indicate that adding topics could improve the model performance, although models that included the topics do not consistently outperform models without topics. Of course, studying more companies would be one way of verifying the impact of topic models, which is left to future research.

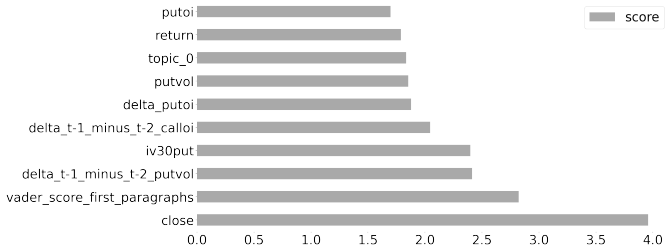

Fig. 4. Score of the top 10 features on classifying the change in next day's iv30call for AAPL.

One of the shortcomings of this study is the minimal number of corporations. Companies like Apple may generate much more news than smaller companies as they get more coverage than smaller companies. Therefore, it could be the case that some companies do not have any news for many days in a row, and in such scenarios, the framework of this study will not be helpful.

Another point to consider is that the method uses the end of daily news and volatility to discover the iv30call of the next day; as such following day positions had already been closed. The results of our analysis may not be used independently for making financial decisions but could be combined for short-term volatility forecasting models.

This study showed how going beyond sentiment analysis could improve the accuracy of classification problems in the finance context. One area that remained untouched is considering the role of named entities and concepts in iv30call. Using a trained sentiment analysis model on labeled financial news is another venue that could improve the model. Another research question is how considering the quarterlies of daily sentiment instead of mean could impact the classification results.

ACKNOWLEDGMENT

The authors are grateful to Irena Spasic and Stefan Feuerriegel for valuable discussions and inputs.

REFERENCES

[1] B. G. Malkiel and E. F. Fama, "Efficient capital markets: A review of theory and empirical work," The Journal of Finance, vol. 25, no. 2, pp. 383–417, May 1970.

[2] R. J. Shiller, "From efficient markets theory to behavioral finance," The Journal of Economic Perspectives, vol. 17, no. 1, pp. 83–104, March 2003.

[3] M. W. Uhl, "Reuters sentiment and stock returns," Journal of Behavioral Finance, vol. 15, no. 4, pp. 287–298, Oct. 2014.

[4] M. Nofer and O. Hinz, "Using Twitter to predict the stock market," Business & Information Systems Engineering, vol. 57, no. 4, pp. 229–242, Aug. 2015.

[5] E. C. Baig, "Samsung ordered to pay Apple nearly $539 million in damages in longstanding patent dispute," USA TODAY, para. 1, May 24, 2018.[Online]. Available: https://eu.usatoday.com/story/tech/2018/05/24/samsung-must-pay-apple-539-million-copying-iphone-patents/623908002/. [Accessed Oct. 20, 2021].

[6] J. Kordonis, S. Symeonidis, and A. Arampatzis, "Stock price forecasting via sentiment analysis on Twitter," in Proc. 20th Pan-Hellenic Conf. on Informatics, Article No. 36, Patras, Greece, ACM, November 10 - 12, 2016., p. 3.

[7] S. Bird, E. Klein, and E. Loper, Natural language processing with Python: Analyzing text with the natural language toolkit. California, O'Reilly Media, Inc., 2009.

[8] M. Hagenau, M. Liebmann, and D. Neumann, "Automated news reading: Stock price prediction based on financial news using context-capturing features," Decision Support Systems, vol. 55, no. 3, pp. 685–697, June 2013.

[9] T. Loughran and B. McDonald, "When is a liability not a liability? textual analysis, dictionaries, and 10-ks," The Journal of Finance, vol. 66, no. 1, pp. 35–65, Feb. 2011.

[10] P. C. Tetlock, "Giving content to investor sentiment: The role of media in the stock market," The Journal of Finance, vol. 62, no. 3, pp. 1139–1168, May 2007.

[11] H. Jangid, S. Singhal, R. R. Shah, and R. Zimmermann, "Aspect-based financial sentiment analysis using deep learning," Companion Proc. The Web Conf. 2018, pp. 1961–1966.

[12] T. H. Nguyen, K. Shirai, and J. Velcin, "Sentiment analysis on social media for stock movement prediction," Expert Systems with Applications, vol. 42, no. 24, pp. 9603–9611, Dec. 2015.

[13] F. Audrino, F. Sigrist, and D. Ballinari, "The impact of sentiment and attention measures on stock market volatility," Int. Journal of Forecasting, vol. 36, no. 2, pp. 334–357, Apr. 2020.

[14] B. DeWilde (2021), "Textacy: NLP, before and after spaCy," (Version 0.11) [Source code].https://github.com/chartbeat-labs/textacy.

[15] M. Honnibal and I. Montani, "SpaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," to appear, July 2017.

[16] E. Cambria, D. Das, S. Bandyopadhyay, and A. Feraco, A practical guide to sentiment analysis. Switzerland, Springer, 2017.

[17] A. H. Shapiro, M. Sudhof, and D. J. Wilson, "Measuring news sentiment," Journal of Econometrics, Nov. 2020.

[18] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in Proc. of the Int. AAAI Conf. on Web and Social Media, Michigan, vol. 8, no. 1, 2014, pp. 216-225.

[19] S. Loria (2021), "TextBlob: Simplified text processing," (Version 0.17) [Source code]. https://github.com/sloria/textblob

[20] S. Sohangir, N. Petty, and D. Wang, "Financial sentiment lexicon analysis," in 2018 IEEE 12th Int. Conf. on semantic computing (ICSC). CA, IEEE, pp. 286–289.

[21] K. Batmanghelich, A. Saeedi, K. Narasimhan, and S. Gershman, "Nonparametric spherical topic modeling with word embeddings," in Proc. Conf. Association for Computational Linguistics. Meeting, pp. 537-572, Aug. 2016.

[22] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," the Journal of Machine Learning Research, vol. 3, pp. 993–1022, Mar. 2003.

[23] T. L. Griffiths and M. Steyvers, "Finding scientific topics," Proc. of the National Academy of Sciences, vol. 101, no. suppl 1, pp. 5228–5235, Apr. 2004.

[24] D. M. Blei, "Probabilistic topic models," Communications of the ACM, vol. 55, no. 4, pp. 77–84, Apr. 2012.

[25] R. Rehurek and P. Sojk (2021),"Gensim – Topic modelling in Python," (Version 4.1) [Source code]. https://github.com/RaRe-Technologies/gensim

[26] M. Roeder, A. Both, and A. Hinneburg, "Exploring the space of topic coherence measures," in Proc. eighth ACM Int. Conf. on Web search and data mining, 2015, pp. 399–408.

[27] D. Mimno, H. Wallach, E. Talley, M. Leenders, and A. McCallum, "Optimizing semantic coherence in topic models," in Proc. 2011 Conf. on empirical methods in natural language processing, Edingurg, 2011, pp. 262–272.

[28] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system", Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining, 2016.