Predicting Share Price Volatility Using Yearly Financial Reports

Robert Taddeo
Courant Insititute
New York University
New York, NY 10003
rt1727@nyu.edu

Ryan Schwarz
Courant Insititute
New York University
New York, NY 10003
rjs724@nyu.edu

William Humphrey
Tandon School of Engineering
New York University
New York, NY 10003
whh293@nyu.edu

Abstract

We seek to use publicly traded companies' yearly documents known as 10-Ks to predict a real-world continuous risk quantity associated with the text's meaning. We use a web scraper to read in the documents from the SEC website, and store information about the companies past and current volatility in relation to the document's filing date. Following that, we parse the text and calculate the TF-IDF score for each word in every text. Finally, we use Random Forest Regression and Decision Tree Regression models to predict a value for volatility for the period of 1 year following the documents filing date, which we compare with the actual volatility for the same time period.

1 Introduction

We are trying to solve a text problem. If we are given text, can we predict the volatility associated with that specific text. More specifically, can we use a company's yearly financial report, known as a 10-K document, to predict a measure of financial risk of investing with that specific company, known as volatility.

The topic of volatility is a very important one to anyone who currently invests their money in publicly traded companies or is interested in investing in certain companies. If investors were to have an accurate representation of future volatility of a stock, it will help their decision on whether to invest or not tremendously. Financial reports are required by the government, and are a staple of the financial world. They have a vast amount of information about each publicly traded company and their worth. It is worth it to note that these reports are costly to produce.

The quirks of this problem are enticing from the perspective of Natural Language Processing (NLP). This is due to the fact that there is nothing complicated fundamentally with volatility. As, often NLP problems are based off human interpretation of a text. When dealing with volatility, this is a statistic that is not interpreted very differently from person to person. Hence, using a statistic as our result makes the analysis much more efficient and easy to process.

Also, the data we are reading in was able to be accessed without much difficulty. This is because all of the data was readily available on the SEC website, since these reports are required by law for any publicly traded company. All that was required by us, the researchers, was a way to read in the data in an efficient and useful manner.

In this paper, we will show you that predicting the volatility based off of text is at times challenging, and at times relatively easy. But in end the goal is the same, output the volatility of a company's stock based off of their yearly financial reports in an accurate fashion.

2 What is Volatility?

Volatility, in finance, is used to measure **risk**. This metric is measured as the degree of variation of a trading price over time. It represents how much a stock's prices jump around the average price. The higher the volatility of a stock, the less predictable the share prices is, which in turn makes it a riskier investment. In the securities markets, volatility is often associated with big swings in either direction. For example, when the stock market rises and falls more than one percent over a sustained period of time, it is called a "volatile" market. An asset's volatility is a key factor when pricing options contracts.

An important note is that volatility is different than return value. We are not predicting what the return value of a stock is, what we are predicting is how much the price of a stock can stray from its average price.

As discussed in the Introduction section, there are many reasons as to why investors care about the volatility of a stock price. One reason is that

$$\sigma_T = \sigma\sqrt{T}$$

σ_T = volatility over a time horizon

σ = standard deviation of returns

T = number of periods in a time horizon

Figure 1: Equation for calculating volatility.

when a stock is volatile, it can give the investor an opportunity to buy low and sell high, due to the unpredictable nature of the stock. Another reason is that when saving for retirement, higher volatility can lead to an unpredictable final portfolio value. Understanding and using volatility is important for building any successful portfolio.

3 How is Volatility Calculated?

Given a filing date t for document d, we calculate two sets of volatility from the range (t- 1 year, t) and (t , t + 1 Year). Volatility is calculated by taking the standard deviation of share price over a given time period, and then multiplying the standard deviation by the square root of number of trading days. This can be seen in the photo above. We use the yfinance python package to grab share prices of a company with a stock ticker, which is provided for us in our dataset. To limit outliers, we reduce our dataset to companies listed on the New York Stock Exchange, which helps remove extreme volatiles of Over the Counter markets which generally trade at a much riskier rate. After removing all non NYSE companies, we find the following statistics of volatility for 17,190 documents:

Volatility (+1 Year):	
Mean:	14.79
Min:	0.0
Max:	17,798.57
Volatility (-1 Year):	
Mean:	17.104321
Min:	0.0
Max:	65,875.41

4 Dataset

The United States government created an agency called the SEC (Securities and Exchange Commission) after the stock market crash of the 1920s to protect the national banking system and to protect investors. Part of their efforts to protect investors is requiring all publicly traded companies to release yearly reports known as "10-Ks". These reports often have information about the history of the company and how its organized, it also has financial information. All of these reports are readily available on the SEC's website [1].

Year	Words	Documents	Words/Doc
1994	34.1M	273	125,263
1995	29M	254	114,189
1996	29.8M	282	105,824
1997	41.7M	372	112,321
1998	46.1M	384	120,228
1999	54.5M	422	129,341
2000	46.8M	413	113,523
2001	25.5M	435	58,829
2002	31.7M	491	64,675
2003	52.4M	713	73,582
2004	56.5M	739	76,525
2005	54.2M	667	81,374
2006	74.9M	810	92,533
2007	88.5M	842	105,110
2008	54M	531	101,726
2009	95.8M	886	108,172
2010	102.7M	909	113,059
2011	109.1M	947	115,294
2012	99.4M	999	99,559
2013	59.8M	650	92,118
2014	60.8M	677	89,942
2015	112.4M	1,231	91,318
2016	50.6M	571	88,679
2017	51.8M	579	89,583
2018	111.4M	1,243	89,701
2019	66.5M	870	76,513
Total	1.64B	17,190	95,404

Table 1: Dimensions of the dataset used for this paper. These number reflect the totals after the filtering, tokenization, and cleaning of the text.

4.1 10-K Dataset

In our original dataset, we are provided with the SEC link to a 10-K document, along with the Filing Date, and Company Symbol. In order to actually grab text from a 10-K document, we had to build a web scraper that provided a url, grabbed and cleaned the document text. We use the requests python package to send an HTTP request to the SEC website, and then used the BeautifulSoup python package to parse the raw HTML from the site. Sending 17,190 requests to the SEC website is

[1] https://www.sec.gov/edgar.shtml

a long and lengthy process, so to speed up the parsing, we essentially multi-threaded our algorithm by running several python kernels on different sections of the dataset. An issue we ran into during the first iteration of our scraping was that many of the documents were being stored on the stack, and at the end of the scraping job we would write all of them at once to a CSV file. This led to several problems as the scraping job progressed, with each document taking up more space in RAM and leading to a slower and slower algorithm, so much to the point where after 400 documents were loaded the algorithm would essentially slow to a stop. Instead, our second iteration of the scraping job wrote to a CSV file directly, so no documents were ever stored on the stack. This made a huge difference in speed and allowed us to use different cleaning methods for our text without taking hours to run. Once we had all of the text stored in separate CSV files, we could then extract relevant information that we would eventually use in our model.

5 Document Cleaning

The documents were stored in their original HTML format, which included many formatting characters that are not relevant to the document itself. We removed all non number and word characters using a regex pattern ([^^A-Za-z 123456789]) which left us with only textual information. We then needed to find the relevant information useful for predicting volatility, which is mainly found in Item 7A of the document. Item 7A, "Quantitative and Qualitative Disclosures about Market Risk" requires information about the company's exposure to market risk, such as interest rate risk, foreign currency exchange risk, commodity price risk or equity price risk. The company may discuss how it manages its market risk exposures. Extracting Item 7A proved to be a challenge, as many of the documents are unique in their formatting, and some documents don't include Item 7A entirely. In order to generalize the problem, we lower all characters in the document, split the document using the string "item 7a". That returned us a list of the text split wherever the term "Item 7A" occurs. We check the length of the items in the list, and if there is more than 1 item in the list, we take the last item returned in the list, as the first is generally the mention of Item 7A in the table of contents. Still, after splitting the document, we were left with the last occurrence of Item 7A to the end of the document. In order to grab what was

Word	Frequency
company	503,962
financial	493,456
income	441,732
net	411,775
million	402,057
stock	363,502
value	341,491
assets	340,809
cash	311,381
interest	308,419
consolidated	281,463
tax	258,384
fair	243,811
ended	241,123
year	235,140
plan	234,390
total	221,196
rate	221,003
form	213,712
agreement	202,301
common	187,035
loss	184,322
related	177,648
report	175,651
incorporated	168,286
reference	166,663
may	164,996
exhibit	163,692
accounting	158,487
share	157,510
based	156,417
equity	154,465
credit	150,384
debt	146,062
certain	145,445
compensation	143,038
period	139,538
per	138,596
operating	137,886
amount	134,343
current	133,889
cost	132,887
balance	132,126
expense	132,079
market	131,518
date	126,713
deferred	124,218
earnings	119,667
benefit	118,640
us	115,832

Table 2: Frequency of the top 50 most common words from all the documents.

id	loss	market	interest	risk
0	0.0	0.0005912428631	0.0	0.000594029573
1	0.03953469954301	0.04768044036589	0.08663809638972	0.071857759464921
2	0.06166166194751	0.06692980360041	0.0900854030022	0.059773568349130
3	0.007971416920796	0.0016023083245736	0.0	0.006439441990176
4	0.028089662157778	0.013345588926533	0.03233293459321	0.010829934877164
5	0.015770001845536	0.009056789307062	0.01755381964193	0.00454973837319
6	0.0	0.02712675439893	0.0657211592935	0.0272546111529
7	0.014118886592873	0.034055892064735	0.278466796135	0.06843281596035
8	0.023482777996997	0.05664235260941	0.3087672157045	0.08536398821194
9	0.017570141547248	0.01141016075582	0.025669325673351	0.006550823062923

Table 3: A snippet of our TF-IDF matrix. The id column represents the document id's, while the score is the value for each word in that specific document.

relevant strictly to Item 7A, we then split our sub document by the string "Item 8", which is generally the next section in the document. This returns another list which included Item 7A and the remainder of the document. We take the first item in the list and use that to train our model. After extraction of all documents, we were left with 10,633 cleaned documents.

6 TF-IDF Matrix

TF-IDF (term frequency-inverse document frequency) is a measure that calculates how valuable a word is for a document based off a collection of documents. It is calculated by multiplying how many times a word appears in a specific document times the inverse document frequency of the word across a collection of documents. The inverse document frequency measures how common or rare a word is over an entire collection of documents. The higher the TF-IDF score, the more relevant that word is in that particular document. Most scores returned were 0 as they do not appear in the document at all.

The way TF-IDF works is that it reads in a word and how many times it appears in a document, over a collection of documents. It increases proportionally to the number of times a given word appears in a single document, but then that is offset by the amount of documents that this specific word appears in. What this means is that words that are appearing in many documents in large frequencies, such as "then" or "when", score lower than a word like "remote" that appears very often in one document but not others. This is because the word "remote" is more valuable to a document than the words "then" or "when". This was very a important metric for our research as it allowed us to locate which words are the most valuable over our entire collection of documents.

We generated a row within the TF-IDF matrix for each 10-K document in our dataset. These values will be used within our models to increase the overall performance when compared to just using the historical volatility.

7 Regression Models

Our dataset was then split into training and testing subsets. The train_test_split function from the Scikit-Learn library works by randomly sampling the dataset and creating subsets of the assigned percentages. The training set consisted of 80% of the data and the testing set 20% of the data. Once the data was separated into training and testing subsets, we trained a Decision Tree Regressor and a Random Forest Regressor to predict realized volatility based on the input variables of historical volatility and the generated TF-IDF matrix.

The Decision Tree Regressor works by utilizing multiple algorithms to decide to split a singular node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. The output is decided by continuing to split and select nodes until the subsequent nodes are similar enough to the previous that they have no significant impact within the results of the model. The specific advantages of Decision Tree Regressions as opposed to other types of regression models include not having to normalize data, not having to scale data, and not having to treat null or missing values.

The Random Forest Regressor works by creating many trees on the subset of the data and com-

bines the output of all the trees. In this way it reduces the problem of overfitting and also reduces the variance and therefore should improve the overall performance. Random Forest Regressions are also usually robust to outliers, less impacted by noise compared to other regression techniques, and works well with non-linear parameters. The disadvantages were that Random Forest Regressions are much more complex and have longer run times when compared to Decision Trees and other types of regressions.

8 Results

All of the regression models were evaluated using the same metrics of Root Mean Squared Error, Mean Absolute Error, Mean Squared Error, and R^2 score. These metrics help to evaluate the different models in a comparative way. The best method will be decided based on a combination of all of these metrics while ensuring that the model is not overfit.

For our baseline prediction using the Decision Tree Regressor excluding the generated TF-IDF matrix, the Mean Squared Error value is 0.000917, the Mean Absolute Error value is 0.000827, the Root Mean Squared Error is 0.030280, and the R^2 score is 0.278801. For our baseline prediction using the Random Forest Regressor, excluding the generated TF-IDF matrix, the Mean Squared Error value is 5513.164667, the Mean Absolute Error value is 3.26833, the Root Mean Squared Error is 74.25069, and the R^2 score is 0.283617.

For hypothesis model using the Decision Tree Regressor including the generated TF-IDF matrix, the Mean-Squared Error value is 0.000499, the Mean-Absolute Error value is 0.001825, the Root-Mean-Squared Error is 0.023326, and the R^2 score is 0.717715. For hypothesis model using the Random Forest Regressor including the generated TF-IDF matrix the Mean-Squared Error value is 6028.243936, the Mean-Absolute Error value is 4.223658, the Root-Mean-Squared Error is 77.641767, and the R^2 score is 0.573014.

Based on these results it is fair to say that introducing the generated TF-IDF matrix increases model accuracy when predicting realized volatility for a given year. The best method for predicting based on our research is using a Random Forest Regression which had the most reliable error values and the best R^2 score of 0.717715. This score should range from -1 to 1 and represents how accurately the model describes the data.

Predicted Values	Actual Values
2.55394353564958	12.31342821202810
1.5984019297864700	3.65953822746081
26.531740280902600	11.840177889128800
2.1684893338371700	0.9251869990024490
0.7173032065686530	1.9699744104661900
2.4655550365222700	9.770599929128840
0.7187985113758110	0.8820430846547410
2.5212887365735600	4.896323096917860
15.149230133785700	11.496660089472200
2.896004686219650	4.019413629431790
0.2489417858636920	1.0260718713169400
0.6410585151338260	3.618021670463510
1.699795120955510	1.5393358751058600
3.7930470917349400	3.5085967893808500
3.6581149606665500	0.9830039224572980
6.730976423263710	1.714977694241910
16.927488577514300	7.134471963941150
8.199141705542700	12.060496929554500
0.3795551975662580	0.4868745235501240
12.2443187358198	3.3407031556997100
3.868939015885980	4.449535973957870
4.746008050108490	3.6412800756152100
4.4779433314631300	3.7138339421413
5.663258655660710	4.340673994218840
8.107442312074120	5.448204873924780
10.096624762225400	4.442204721876060
2.579295660527830	2.4862420179635700
2.9594257022167500	7.579378868126430
2.54502950571784	2.95384883827994
5.910044563641050	1.9955235110047800
12.2443187358198	3.288206394438670
2.0144078367615400	2.6445553351458200
1.9431463771488100	1.2233699638808800
4.370019049111480	1.0947255353101800
0.49849231813044	4.798756565109090
1.937965688909320	5.3699063570219500
4.1696499265174800	3.7119433778181500
4.065166641713430	4.819522745001780
6.285915303544720	3.7884551650752500
9.842230302333590	10.247546875880900
11.980172411039400	13.352510549699100
0.5436687420759850	0.4209254990918770
2.5524433658790100	3.4970762145443700
3.5207471418244200	2.2460507847356300
5.263581666803450	3.058545785906610
1.1157434688444900	2.5743617692017700
1039.3535761788700	1246.125751004040
1.1243332197661600	2.2229529263244600
3.167877513412740	1.32332757182423
1.549500546923100	1.5163416559745700

Table 4: Predicted vs Actual values from our model that was run on our test set of documents.

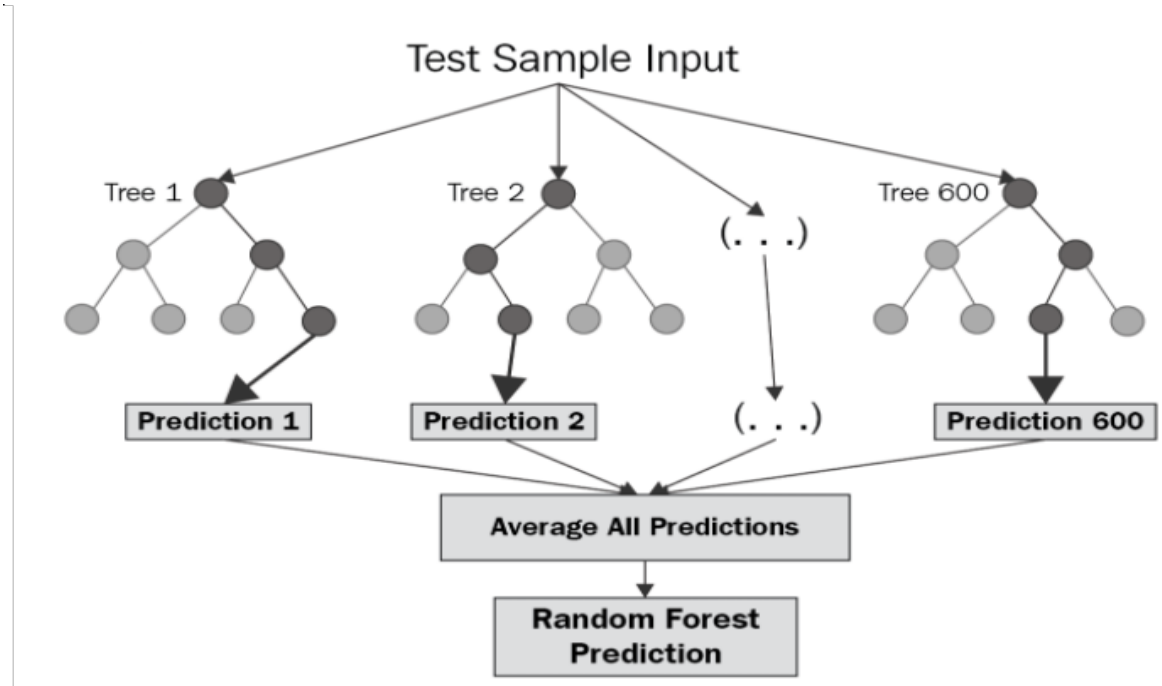

Figure 2: Visual representation of a Random Forest Regression.

9 Conclusion

Overall, our results show that there is a significant advantage to using parsed 10-K documents as a way to predict volatility. In both the Random Forest and Decision Tree models, our TF-IDF Matrix paired with past volatility achieves over double the R^2 score of the baseline, with the Decision Tree Regression performing the best. Overall, these predictions can play a huge role for large investment funds in financial markets, and in the future we hope to continue working towards achieving a higher score. The methods and approaches used throughout our research are also applicable to different time periods such as weekly, monthly, quarterly, and bi-annually, and would be interesting to see if the same results appear over different periods.

In a next iteration of our project, we hope to better our model score by improving our parsing of the documents themselves. We hope to test the incorporation of other sections of the document into the TF-IDF matrix, as well as using POS Tag weighting to further improve accuracy. Eventually, we seek to use our predicted volatility's to define a low-risk trading strategy in which knowing volatility plays a huge role. A great example of this would be Options, as the market price of an option is based on a predicted volatility, however this predicted volatility is calculated using only past volatility, similar to our baseline model. If we can predict a more accurate volatility than the one used to price an option, we could potentially find undervalued or over-valued trades. In conclusion, our efforts to predict volatility using Natural Language Processing prove both possible and valuable for investors.

Word	Value
historical_vol	0.6694409951150030
billing	0.23032943889812700
extraordinary	0.06059714917566680
such	0.012690586725233500
impact	0.008018039314873820
act	0.006307657606769120
available	0.00363629027555721
into	0.0036013041418793200
subject	0.0011644139506874000
secretary	0.0011454369444970800
at	0.0005572127814714560
liquid	0.0004870790076183710
same	0.00028867742481116400
aggregate	0.0002656633006230410
any	0.00025625992698074400

Table 5: Most important features of Decision Tree Regressor Model

References

Scikit-learn:Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Predicting Risk from Financial Reports with Regression, Leven *et al*.

Overby, Brian. *The Options Playbook: Featuring 40 Strategies for Bulls, Bears, Rookies, All-Stars and Everyone in Between.* Ally Invest/Ally Financial Inc., 2018.

Option-Price, Yang, Yuhao, Qsctech-Sange, PyPI, 2020.

Bakshi, Chaya. "Random Forest Regression." *Git Connected*, 2020, levelup.gitconnected.com/random-forest-regression-209c0f354c84.

"What Is TF-IDF?" *MonkeyLearn Blog*, 10 May 2019, monkeylearn.com/blog/what-is-tf-idf/.

"How to Read a 10-K." *SEC Emblem*, 1 July 2011, www.sec.gov/fast-answers/answersreada10khtm.html.

Kuepper, Justin. "Volatility." *Investopedia*, Investopedia, 28 Aug. 2020, www.investopedia.com/terms/v/volatility.asp.