The Effect of the News on the Stock Market

Abstract

Understanding how the stock market works is an essential skill in life. Investing properly can lead to security and stability in financial life, but it is hard to find a way to do it consistently. This brought up the question of automating this skill. This is what this paper is about. Using a CNN() and Hugging Face's Transformers() model, we created a sentiment analysis model that can provide both the certainty of the sentiment and the sentiment itself. The higher the certainty, the more likely it is that the stock will increase in the near future. The CNN and Transformers models both yielded exceptional results, with the Transformers model beating out the CNN model by a mere 1%. This means that the Transformers model was more accurate in predicting the sentiment of sentences than the CNN model was, but only by a little bit. The dataset used consisted of three columns: title, stock, and date. The 'date' column will not be used, but the title and stock columns will be used to train the model and display graphs. The purpose of this machine is not to control a buyer's account and only make purchases that will be guaranteed successes in the short run; it is to provide the buyer with exterior information unwittingly to the buyer so they can make a more educated decision.

Intro:

I remember opening my investing account one day and looking at my stocks, noticing that the majority of them were declining pretty severely. Ever since the Russia and Ukraine war, this had been the case. I saw many news headlines talking about the market, and how it was affected from this war. It was then I realized, why isn't there a way for us to predict these changes in the market given sufficient news headlines? This is what I set out to solve and build. I first found a dataset that had a diverse number of stocks; a diverse sentence structure with news headlines; and a long enough time period. I will be using this dataset to train a sentiment analysis model that I will then use to predict the market fluctuations. A sentiment analysis model can analyze sentences and predict whether the sentence has a positive connotation or a negative connotation.

Along with the sentiment, a confidence score (or a certainty rating i.e. 0.9 for 90 percent confidence) will be given. This allows the user to understand how certain the model is about a positive or negative sentiment. This means that the more certain the model is, the more likely it will be that the stock will rise in the near future. Now, only using one headline to predict the change of an entire company's worth is an inadvisable idea. It is *better* to use multiple headlines so the model can be sure about the fluctuations in the stock. For example, if the model uses 2000 headlines to predict the change in a stock, it is much more likely that the change will happen because over 2000 headlines, it says so. Realizing this, I was able to remove the values in the dataset which have too few values to contribute anything to the training of the model and the predictions.

Background:

Before writing the code necessary, I already knew how to find accuracy and how to create a viable dataset that can be used. An intermediate to advanced understanding of python if needed to code in machine learning especially because machine learning can be accessed in multiple languages and presenting the results requires more programming knowledge than just python. Also understanding what accuracy is and how it is applicable to this question is crucial because it is the best way of gauging how well a model is performing. The final piece of information one needs to code a machine learning model is being able to lay out code so that it is readable and understandable to the audience.

This report will cover the dataset used in the next section, but it is crucial to understand one thing about it. There was no true sentiment given in the dataset. In other words, after building a model, there was nothing to compare the accuracy to. So, finding an outside sentiment analysis model became prevalent in this study. After running an outside model with the data being used, there was finally something that was able to be compared to. Now, building a new model is the next step. Before the final model, Tfldf Vectorizer and Word2Vec models were used and provided worse results than CountVectorizer at 84 and 83 percent accuracies respectively. Other classifier models used before the final Logistic Regression model were Ridge Classifier, Random Forest Classifier, Decision Tree Classifier, and Support Vector Classifier. All of these provided worse and

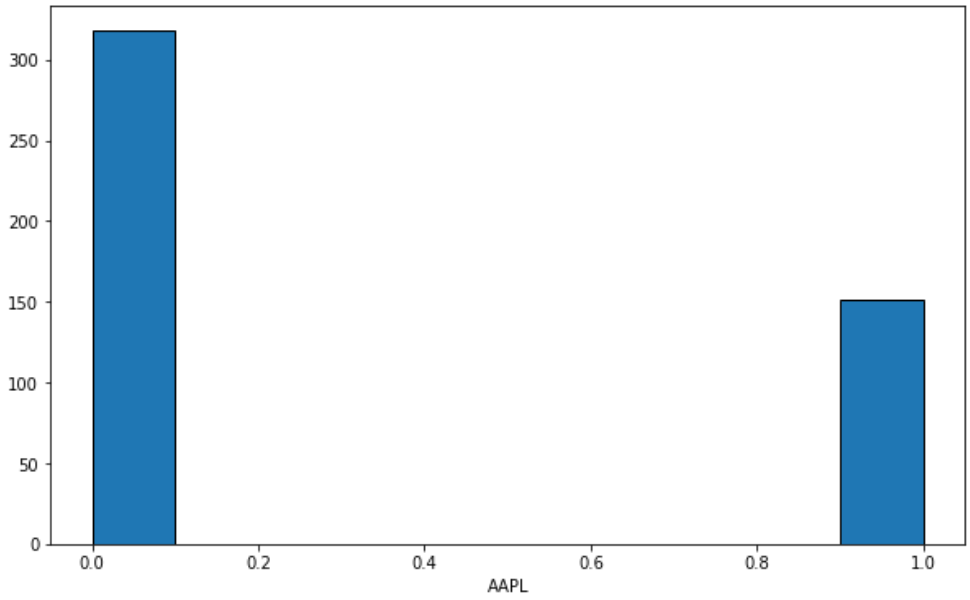
worse accuracies as they trained, going down to the worst accuracy of 72 percent.

Dataset:

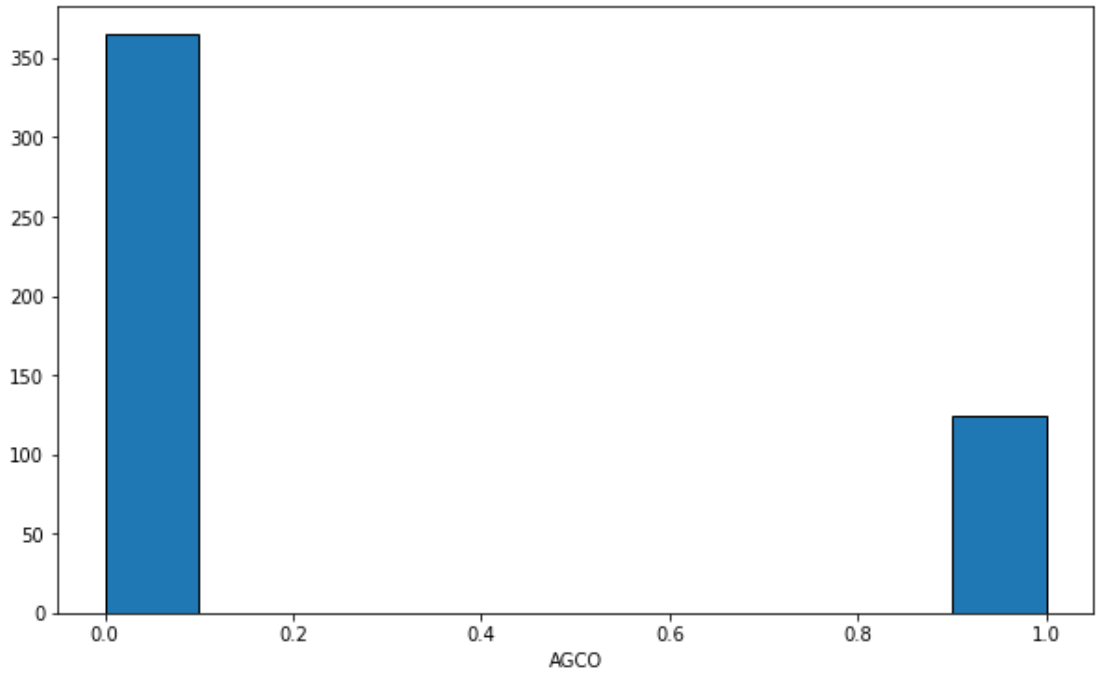
The dataset used in this project consists of three columns: date, title, and stock. The title and stock columns will be the two focused on the most in this model. There are 101 different stocks that are documented in this dataset. Some stocks were not used in this analysis because they had less than 100 headlines which did not contribute to the model's development. Any stock with more than 100 headlines was used to train and test the model. The model was trained on 80 percent of the data and 20 percent were tested on. This allowed for sufficient training while also being able to test on enough data. About 10% of the training data became validation data which serves a purpose equally vital as training and testing data. The validation data allows the model to test on headlines that will not be seen to the public and provide results to the programmer. This allows the programmer to change the necessary code as seen fir by them. Moreover, the validation data is actually used everywhere in machine learning around the world because the results are not disclosed to the public, so programmers are allowed to alter the code freely without worrying about backlash.

The graphs below were created via iterating through a dictionary. Each key in the dictionary was a stock name while each key was a list of all the sentiments from the exterior model. Creating the values as done by first iterating through a list of all the stock titles from the dataset and then indexing the list of the exterior model's sentiment at the point where the stock name changed. This allowed for these graphs to be built with the proper scale, and the programmer does not have to choose a certain scale. This will also allow the model to print the graphs of our sentiment analysis because keys will be the same and all that will change is the value. From this, we can compare and contrast the accuracies of the two models. Something to notice is the bias toward negative reviews. This is understandable because negative titles gain more clicks, meaning more profit. So, this is something that news companies do ofte. They will create more negative headlines because it generates more clicks than a positive headline.

Apple’s headlines from exterior model:

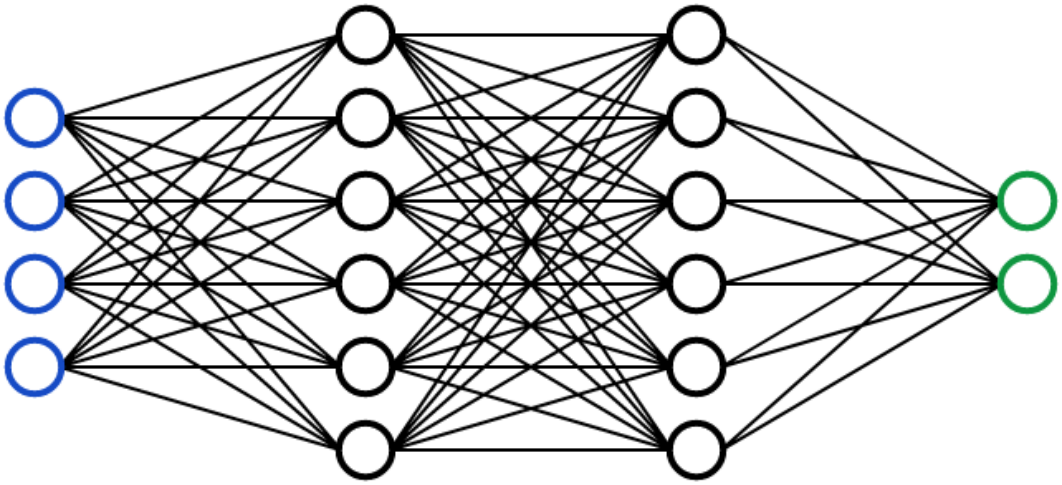


AGCO’s headlines from exterior model:



Methodology:

In order to solve this problem, training a CNN (Convolutional Neural Network) model will be the first step.



This model is a very basic description of a neural network. Given some input (blue circles), the computer will process the input (black circles called hidden layers) and distribute the data accordingly until the output is given (green circles). The input can be an image or numbers, and the output will be a number. These hidden layers have weights to them which allow the computer to understand what is most important when extracting data4. For example, in this dataset, text will be provided to the neural network to predict the sentiment of, and the most important part of that would be the number of positive and negative words in each sentence. Therefore, the number of positive and negative words will be weighted more than other nodes because it is more important.

It is pertinent, however, to understand that machines understand numbers, not words. Thus, we first need to change our data into numbers so the computer will be able to understand it. To do this, the word to vector function CountVectorizer() will be used; it yielded the best results out of all the vectorizers tried, and it did so most consistently. This dataset requires a classification model, and the model used was CNN(). This yielded the best results whereas the SVC model yielded *insert*% accuracy. Recall that an outside model was used to compare results to. This model was the pipeline model ‘sentiment-analysis’ from BERT’s Classifier. Then, after lemmatizing the words, it was run through the classifier model and appended to a list that could be compared with at the end.

In order to create the graphs shown above, the dataset had to be indexed so that a new list was created for each new stock ticker. Since the dataset is

organized alphabetically by stock ticker, there will be two variables: one for the start of the index and one for the end of the index. This would have to be changed and added for each new stock ticker. Then, a new dictionary would be created where each key is a new stock ticker and the value is a list of the classifier's results indexed properly. After that, all that was left was to iterate over each key in the dictionary and create a histogram that properly depicted the number of positive and negative results for each stock ticker. This also allowed me to remove some unwanted stock tickers which only had a few headlines and would not train or change my model in an effective way. After some manual cleaning up, the dataset was ready to be used and trained with the CNN model.

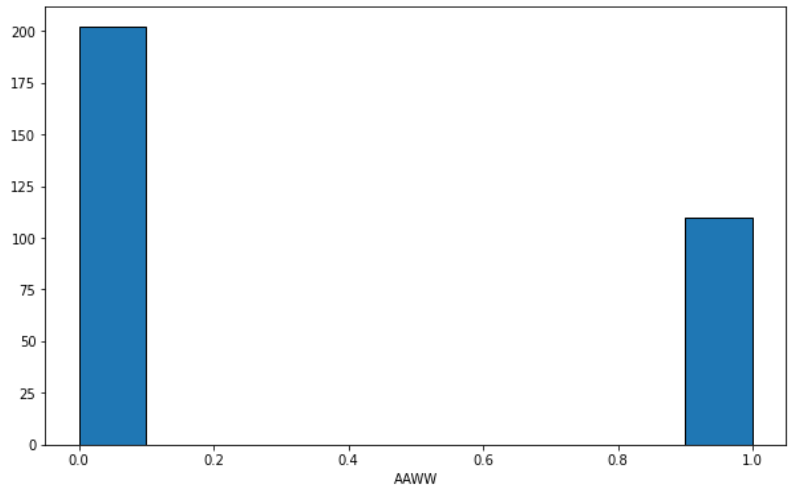
Then, we tried using a Transformers() model which actually provided much better results than previously thought. There were many ways the Transformers model was better. The data was trained quicker, using validation data as a new way of checking, and the data did not train for as long as it did with the CNN model.Due to this decrease in time taken to train, the entire dataset could actually be used. Previously, the dataset was shrunken since it took nearly 5 hours just to run the code once. So, the dataset was taken down to around 50, 000 values which allowed it to train, test, and evaluate accuracy faster. The next reason why the Transformers() model was used was because it is far more accurate than the CNN model; the CNN() model averaged around a 91.2% accuracy, and the transformers model averaged around *INSERT* accuracy. The Transformers model is a pretrained model that I train on my own data so it is more validated towards the kind of headlines that it will be tested on.

Results and Discussion:

As stated previously, the Transformers() model was able to beat out the CNN() model by a very thin margin. This means that for this particular sentiment analysis case, the Transformers() model is more reliable as a source of sentiment than the CNN() model. It is important to understand that this is not true for all sentiment analysis cases, as both CNN() models and Transformer() models are great in their own ways. There are benefits and downfalls to both: one of the largest being that the Transformers() model is pretrained on some data which allows it to be more accurate sometimes, however, this also leaves room for a higher chance of overfitting which will result in even worse results. Overfitting, in simple terms, basically means that the model is basing its predictions off every little detail in the training data. This will not allow it to

perform well on the testing data because the data will not have the exact same features.

This is the result from the Transformers() model for the AAWW stock:



A large source of error in this model can come from two false positives or two false negatives. This means that when comparing my model to the exterior model, both of the models could have predicted the sentiment of a headline incorrectly, thus making it seem like both of them had predicted correctly. Now, this is very rare since both models were trained and tested and validated well and properly, however, this margin for error still exists. This is important to take into account if using this model to predict a stock. Going back on a previous topic, this further solidifies why we must use multiple headlines to predict the sentiment because if just one or two are used, the margin for error is exponentially larger. Using many values will allow the rest of the sentiment to negate the double false negative/positive results.

This is not a bad thing, however. It shows how there is always room for improvement and that it is okay for things not to be perfect. We could make this model more perfect by having a person manually go through and submit their sentiment analysis for every sentence, and then we would have a much more reliable quantity to compare with, as opposed to using another artificial intelligence model that might not be correct either. This would take a very long time, though, since the entire dataset has over a million headlines.

The CNN and Transformers() models thoroughly outperformed the models listed in the background section by over 10 percent. This shows the power of these two types of models, but remember that this does not mean the models

listed in the background section are bad. In fact, it means that those models are better suited for a different type of classification purpose (note that the difference between classification and regression questions are that classification questions have no numerical answer and regression questions have numerical answers.) Many machine learning questions hardly have one correct answer, and most have multiple ways to go about them.

Conclusion:

So, we now know the ins and outs of this machine. How it uses an exterior model to compare against to test accuracy. How it uses a Transformers() model to train, validate, and test its data. How the dataset has over a million quantities that was shortened to have faster training and testing times. The main thing to understand from this model is not its results, that is easy to understand from any paper about its model. It is that certain machine learning techniques are used for certain projects. Another thing that was seen through this project was human nature, and how negative titles generate more views, which in turn generates more profit. This takeaway will help provide context to the negativity of news outlets and sources. Finally, try to use more than one way when tackling a problem because it could work out better than another.

Acknowledgements:
Thank you to my family, my teachers, and my mentor for this project: Eric Bradford.

References:

Bentrevett, “Bentrevett/Pytorch-sentiment-analysis: Tutorials on getting started with pytorch and torchtext for sentiment analysis.,” *GitHub*. [Online]. Available: https://github.com/bentrevett/pytorch-sentiment-analysis. [Accessed: 26-Aug-2022].

Letthedataconfess, “Sentiment-analysis/build_your_first_sentiment_analysis_project_from_scratch.ipynb at main · letthedataconfess/sentiment-analysis,” *GitHub*. [Online]. Available: https://github.com/letthedataconfess/Sentiment-Analysis/blob/main/Build_your_First_Sentiment_Analysis_Project_from_Scratch.ipynb. [Accessed: 26-Aug-2022].