Estimating The News Impact On Financial Market

* **Team members —** Zheng Wei Lim | Ye Jin Yu | Yee Dang Tay
* [Link to Google Doc](https://docs.google.com/document/d/1wTRE1zm75n_OGIUkifLPLZNjDqnY-ttF/edit?usp=sharing&ouid=116093492831692731415&rtpof=true&sd=true).
* [Link to Github repo](https://github.com/lioneltayyd/MADS-M2-estimating-news-impact-on-financial-market).
* [Link to download all the cached and processed datasets](https://drive.google.com/drive/u/4/folders/1Scnd3fYHG_zd1wk6petvF-9U_QWosrLm).

# 1. Context & Motivation

Employees working in the financial sector are facing an immense amount of information everyday compared to other sectors. The industry grapples with an overwhelming amount of unstructured data, such as media articles, research reports, and company filings. Efficiency has become a need in today’s society as employers in the financial sector are fighting to get first-hand information and stay up-to-date with the current state of the world. Hence, we seek to close this gap by examining the extent to which financial news articles affect the volatility of the SPDR S&P 500 ETF (SPY). This is one of the largest ETF and a proxy for the 500 listed big-cap companies in the United States. This can be extended to include other media, regions, and markets, so as to determine the overall extent of media influence on the broader markets. Although this is our ultimate objective, we will only focus on smaller parts of the scope for this project such as the following:

**Project goals / Motivation \*\*:**

1. **Extract latent topic / theme:** Extract latent topics to obtain the overall theme for each article. The topic model will be used to categorize news articles and output a feature (the topic or theme) for downstream regression tasks to estimate the SPY movement or volatility given the occurrence of that theme.
2. **Extract news sentiment:** Extract headline sentiment and the influence of specific tokens on the sentiment when interpreting the model. The outcome will be used as a feature to estimate the movement of SPY ETF or volatility.
3. **Estimate the feature influence on SPY movement:** We will use extracted themes / topics, sentiment, and other features to estimate their influence on SPY movement when interpreting the model. We are aware of the complexity behind the financial market research and consider this a smaller-scale project and a means to explore various tools related to NLP. We already expect a weak performance for the model.

\*\* Despite the goals mentioned above, our underlying intention is to explore various tools, concepts, good practices, and approaches related to ML and NLP we have learnt so far throughout the MADS program. We do not expect any spectacular insights from this project since this is about the financial market. However, we can leverage this framework to further extend our analysis scope and depth.

# 2. Unsupervised learning

## 2.1 Data Source

1. “CNN Dataset of News Articles from 2011 – 2022” provides a collection of text articles. We will use columns like date\_published, headline, description, category, and section, out of 11 features. Will be used to model topics and extract features for supervised tasks alongside the dataset 3.

* **Size:** 205MB±
* **Row:** 38,000± rows / news articles
* **Location:** [Kaggle](https://www.kaggle.com/datasets/hadasu92/cnn-articles-after-basic-cleaning)
* **Format:** CSV
* **Access method:** Download

## 2.2 Topic — Feature Engineering & Representation

We first filtered the news and kept only the “Business & Finance” category. Hence, we left with 850± of news. Then we explored 2 feature engineering approaches after ensuring that the dataset contains no empty value:

1. **Using NER (Wikifier):** We use [Wikifier API](https://wikifier.org/info.html#wikification) to extract the entities for each article. We can think of each entity as a sub-theme for an article. We will then transform the list of entities or tokens for each article into BOW using “tfidfVectorizer” with a custom “analyzer” function and use the output as features for the topic model. The reason for using Wikifier is because Wikifier is able to normalize the word variation such as “Federal Reserve”, “Fed”, “Fed. Reserve”, etc into a single word “Federal Reserve”. It also resolves synonyms partially such as “employment”, “job”, etc and labels them as “employment”. Furthermore, we want to derive topics from meaningful tokens only, hence, we use this approach to extract entities. It also partially saves us from having to process stopwords, punctuation, and utilize ngram or phrases. However, the downside to this is the response and computation speed since we are making API requests and Wikipedia is a huge corpus.   
     
   **Source code summary:** The functions can be retrieved from “source/module/processor\_topic.py” named under “wikifier\_extract” and “raw\_token\_input” (analyzer). We only need to access the API and retrieve the postags and entities in JSON output. We only use the output called “title” from the extracted entities. We also removed some empty output (10 in total) due to exceptions such as exceeding the character limit (25,000 max). The function requires an API key to access the site. To avoid rerunning the code, we cached the dataset in CSV to make it reproducible without user signing up for an API key.
2. **Using embeddings:** We use Gensim “FastText CBOW” to create embeddings. The advantage of using this is that we can compute the coherence score for the topic model. The reason for using FastText compared to “GloVe” and “CBOW” is because FastText is less computationally expensive and able to handle rare and unseen words since it uses character ngram (example apple = app, ppl, ple).   
     
   **Source code summary:** The steps can be found in “source/notebooks/explore\_embeddings.ipynb”. We developed a custom corpus related to the financial domain from CNN news prior to building the embeddings.

In the end, we decided to go with the 1st approach using the Wikifier because embeddings do not yield a satisfying result when we evaluate the similarity between various words. Furthermore, the corpus size from CNN news might also affect the quality of the embeddings since we only have a small dataset after filtering.

## 2.3 Topic — Model & Evaluation

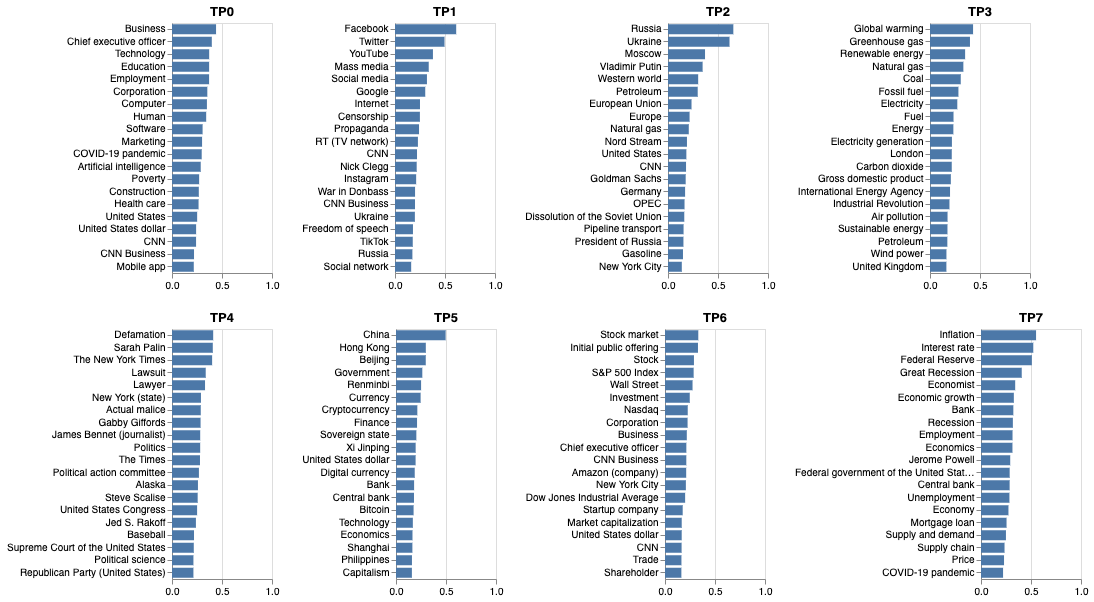
#### 2.3.1 Non-negative Matrix Factorization (NMF)

We pick NMF because it is easier to interpret compared to LSI and PCA since it only outputs the positive sign. Having to analyze the positive and negative sign using LSI and PCA is less intuitive to us when evaluating the topic cluster. We also assign “Nonnegative Double Singular Value Decomposition (NNSVD)” for the “init” parameter since we expect spase factors for the feature input.

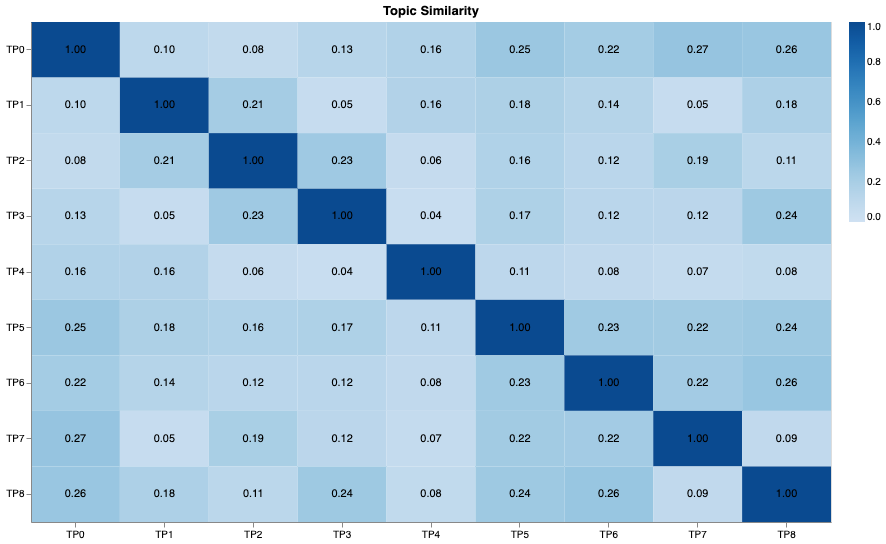
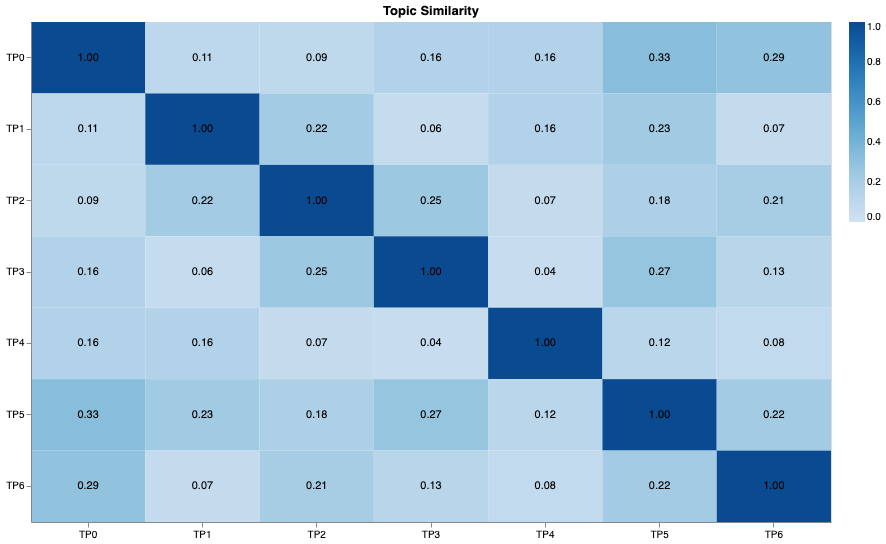
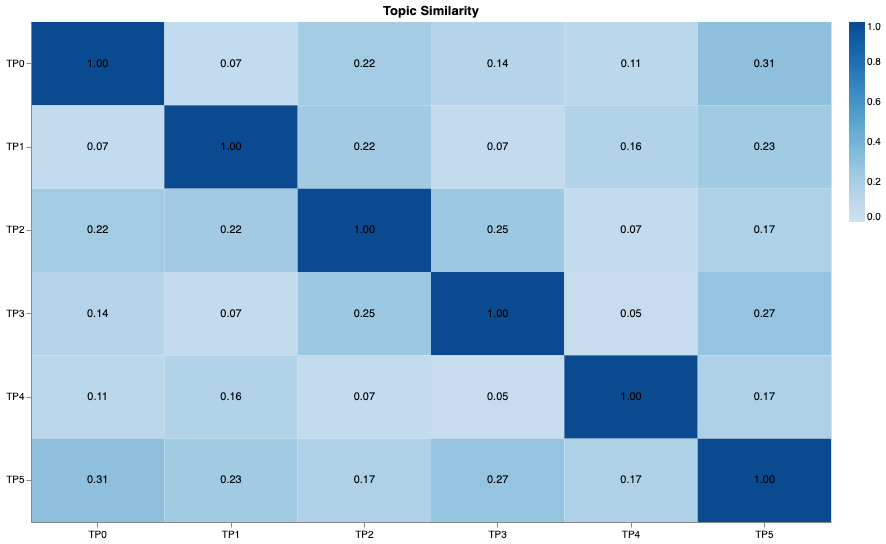
For the evaluation part, we manually evaluate the cluster quality of the topic model instead of relying on coherence or similarity score. The reason is because we do not have good embeddings to compute a reliable similarity score between the tokens due to the corpus size limitation (must be related to financial domain). Furthermore, the meaning and relationship of each token under each topic is more important to us than the score itself. We can do this manually because we have a small number of topics to work with although it is not scalable.

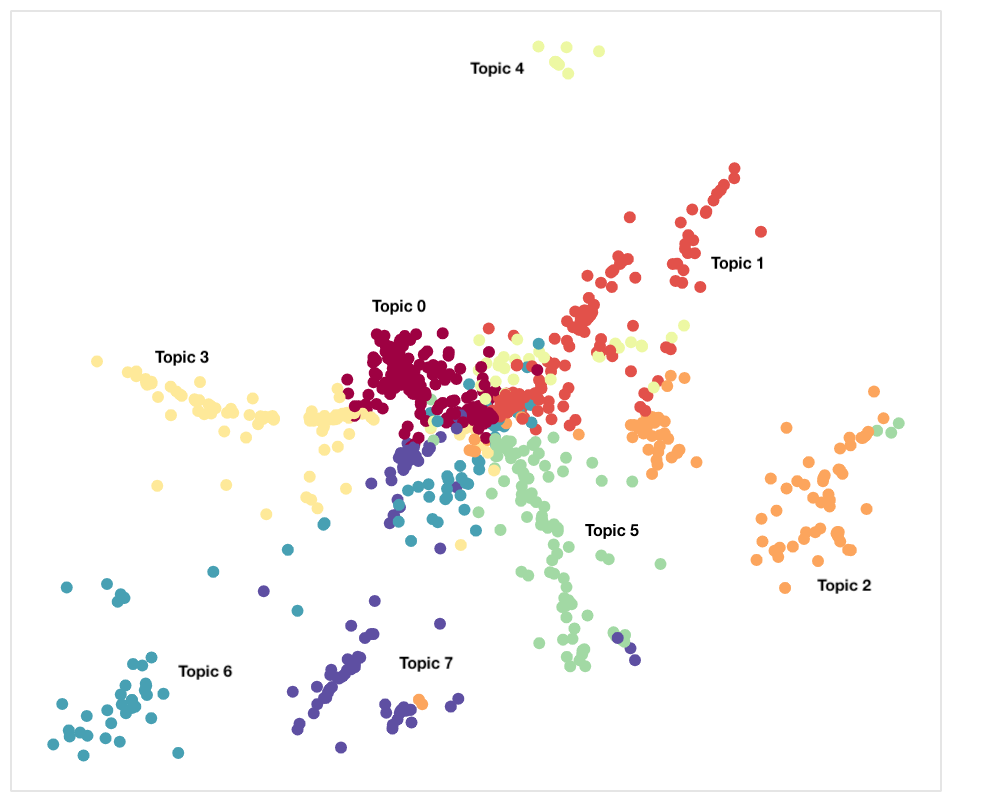
After some exploration and evaluation, we decided to go with 8 topics. This is a very subjective judgment from us after observing the outcome from the following bar chart (token weights for each topic). We do not claim this to be the best cluster, but we think we have achieved a fairly good cluster after seeing how the topics are well defined by most of the top 20 tokens (figure 2.1). We could increase the number of topics should we desire to capture the lower level of topics, however, due to the size limit of the dataset (850±), we stop exploring larger numbers.

This is how we label the topics in ascending order — Economic & Sector Performance, Social media & Geopolitics, Crude Oil & Natural Gas, Green & Sustainable Energy, Government & Politics, Asia Market, Equity Market, Monetary Policy & Economic.

  
**Figure 2.1 —** Bar chart showing the entity / token weight for each topic. Derived from NMF.

We also use cosine similarity to measure how separated each topic is. The figure 2.2 below is the outcome of the result (from 6 to 9 topics). The similarity between some topics increases slightly when the number goes below 8. Whereas when the topic goes above 8, more topics with a 20%± similarity appear.

  
**Figure 2.2 —** Heatmap showing the cosine similarity between topics from 6 to 9 topics. Derived from NMF.



We also visualize the cluster distribution to observe the cluster separation using UMAP. The preference for UMAP over T-SNE is due to its computation speed. Although UMAP and T-SNE are better at capturing the local density, UMAP is able to preserve more of the global structure than T-SNE using the configuration like “n\_neighbors” and “min\_dist”. The reason why topic 0 is similar to topic 3, 5, 6, 7 to some extent is because many of the data points are clustered on the center whereas for topic 2 and 4, they are far apart. Likewise for topic 7, it is closer to topic 0, 5, 6. The cluster distribution is derived from the count vector from “TF-IDF”.

**Figure 2.3 —** Chart showing topic cluster distribution.

**Source code summary:** The steps can be found in “source/notebooks/build\_topic\_modelling.ipynb”.

## 2.4 Topic — Limitations

* **Limitations:** We did not use PMI or LMI to measure how well separated the tokens are (Not topics. Another proxy for measuring the cluster quality) because we are not working with a natural sentence (we use extracted entities as features only). The extracted entities themselves do not depend on the token before and after. We can use token frequency and custom formulas to measure how frequently each token appears on a specific topic and less on others. However, NMF has captured that for us (token weights from NMF component, H matrix).   
    
  Although entity extraction is a useful feature extraction method for topic modeling, it doesn’t have some limitations when it comes to non-human evaluation approaches should we not be able to manually evaluate the topic quality due to scalability issues. Nonetheless, we can still use the embeddings (developed from large domain specific corpus if any) to transform the entities into dense representations to compute the coherence score.   
    
  Also, the Wikifier API might be slow to extract entities. We might need a custom NER that works similarly to the Wikidier to speed up the extraction process.
* **Improvement:** We did not do any co-referencing on the news articles as we ran into an issue installing “NeuralCoref”. Hence, the frequency may be underestimated slightly as Wikifier isn’t able to capture the entities for pronouns though they refer to the same word.

## 2.5 Discussion / Learnings

We are happy to see the outcome of the topic model we trained using the extracted entities since this approach has not been explored by people to the best of our knowledge. Typically, the analysts or engineers will use the vectorizer to transform the text directly instead of extracting the important and meaningful entities as features using Wikifier for topic modeling. This approach also partially addresses some issues that we commonly encounter such as synonyms, word variation, phrases, as well as removal of stopwords, punctuation, and others. It also helps explain and define the topics for us since the extracted entities are more informative. Nonetheless, as mentioned, Wikifier is slow to process. Also, it is not easy to build a similar NER like Wikifier.

There are more sophisticated methods for topic modeling also which we did not explore due to time constraints. For instance, two recent state-of-the-art deep learning, [DeepDPM](https://github.com/BGU-CS-VIL/DeepDPM) and BERTopic, have shown significant performance compared to traditional approaches, especially for DeepDPM which can identify the number of clusters if it is unknown. However, this will entail trade-offs of a higher learning curve, computational complexity, and high monetary costs for cloud-based deep learning platforms.

# 3. Supervised learning

## 3.1 Data Sources

1. “Sentiment Analysis for Financial News” provides unstructured data for sentiment analysis. Columns include headline and sentiment. The sentiment is labeled as either positive, neutral, or negative. The outcome will be used as a feature alongside the dataset 3.

* **Size:** 672KB±
* **Row:** 4837±
* **Location:** [Kaggle](https://www.kaggle.com/datasets/ankurzing/sentiment-analysis-for-financial-news)
* **Format:** CSV
* **Access method:** Download

1. “YahooFinance'' provides financial market data. We will use only SPY for this project. We will use columns such as “open”, "close”, and “volume”. The label will be the close-to-close return in t-score (need to pre-process on the “close” variable). We will also eliminate the negative sign since we are interested in measuring the SPY movement and not direction.

* **Period:** 1999-12-01 to 2022-03-21
* **Size:** 190KB±
* **Row:** Left with 750± dates (filtered non-business news category and weekends).
* **Location:** [Github](https://github.com/ranaroussi/yfinance)
* **Format:** Pandas object
* **Access method:** YFinance API

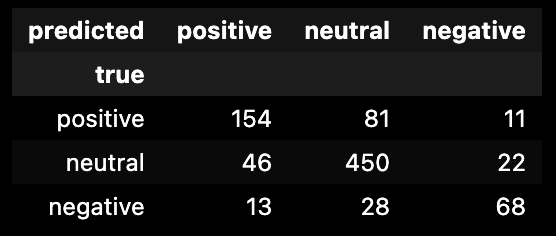
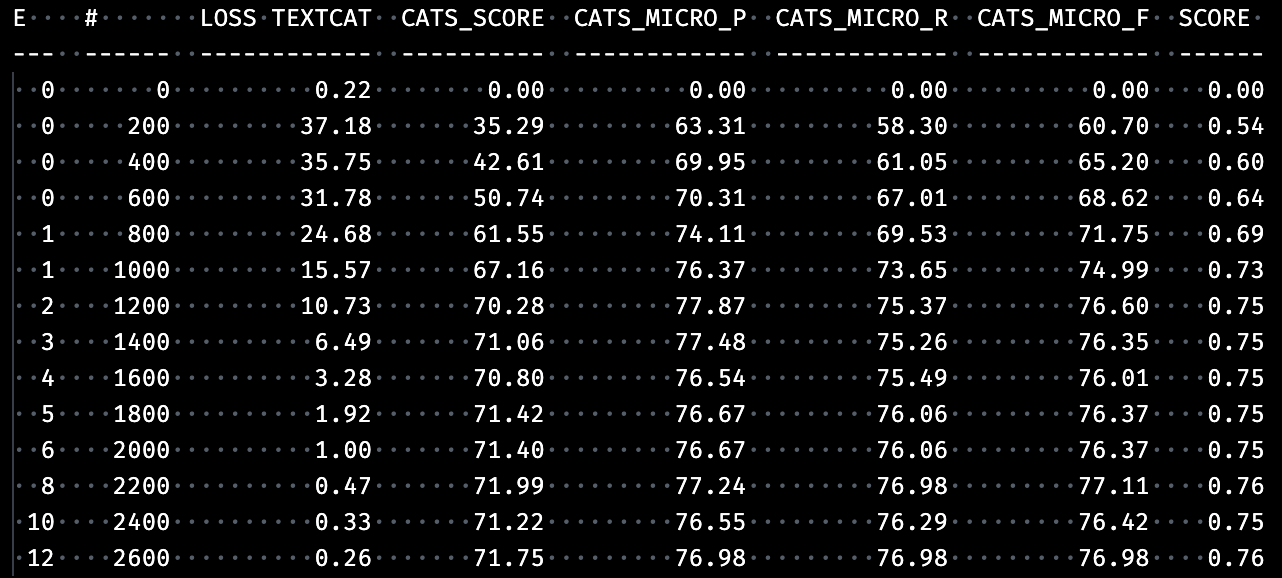
## 3.2 Sentiment — Feature Engineer, Model, Evaluation, Explainer

We use the [SpaCy configuration system](https://spacy.io/usage/training#config) to assign the processing step and choice of model. We are interested in exploring SpaCy using the config approach because it allows us to build multiple custom models such as NER, POS-tagger, text classifier, etc very quickly and use them for analysis or feature input without writing lengthy codes. Moreover, it is able to integrate other [open source tools](https://spacy.io/universe) such as “NeuralCoref”, “Coreferee”, etc into the SpaCy pipeline easily. We can also utilize transfer learning using the pre-trained models in SpaCy conveniently.

For the SpaCy pipeline, we use the default SpaCy tokenizer to tokenize headlines. Then SpaCy will pass the tokens to the text categorizer pipeline to transform the tokens into both BOW and embeddings. We are using “​​[spacy.TextCatEnsemble.v2](https://spacy.io/api/architectures#TextCatEnsemble)”, an ensemble model to train the text categorizer or classifier using both BOW and embeddings. The key parameter that we are attentive to is the ngram, which we assign as 4, since it affects the performance by 3-4%±. The ngram is determined by the average length of the phrases and surrounding context (about 2 to 4).

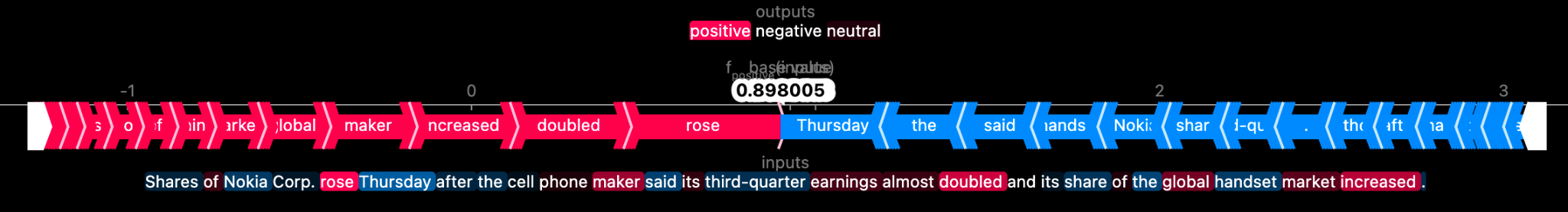
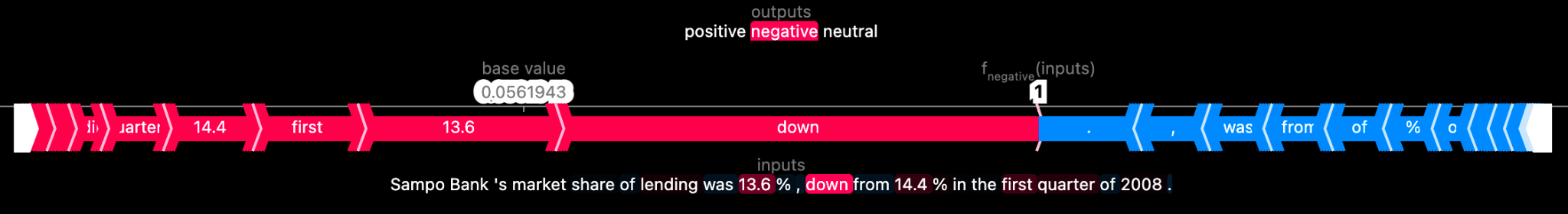
To avoid overfitting, we assigned early stopping for the “patience” parameter. Additionally, we include the L2 norm and gradient clipping under the “training.optimizer” section.

We obtained around 76% with 1.5± of standard deviation for micro precision, recall, and F-Score (figure 3.1) after retraining the model for multiple rounds separately (not referring to epoch or batch). Looking at figure 3.2, most of the mis-classifications are due to the model failing to discriminate the positive/negative sentiment from the neutral. We explored other Transformers models like Roberta too, however we decided to stick with the simplest model with faster training time.

  
**Figure 3.1 —** Score for “TextCatEnsemble.v2” model (single round). **Figure 3.2 —** Confusion matrix for “TextCatEnsemble.v2” model. 

We also utilize the “SHapley Additive exPlanation” (SHAP) library to increase machine learning model transparency which uses a game theory approach in explaining the model output. SHAP interprets how tokens influence the model prediction for each headline. SHAP takes very long to run the permutation explainer on the full dataset. Plus, it is computationally expensive to compute the average weight for each unique token (thousands of them). Hence, we do not have the full picture of the weight for the top N tokens.

The following figure 2.3 are 2 examples we showed to explain how we analyze the token weights for each class. Red indicates positive weight, whereas blue is negative. The impact is indicated by the color intensity. Generally, when the classifier predicts a negative sentiment, tokens like “down”, “dropped”, “fell”, “cut”, “decrease”, etc tend to show higher weights. Whereas for positive sentiment, tokens like “increase”, “double”, “rose”, “growth”, “positive”, tend to have higher weights. However, we also found irrelevant or vague tokens such as “widened”, “little”, “third-quarter”, etc showed up with higher weights for the negative class. And “global”, “able”, “percent”, “performance”, etc for the positive class. As for neutral sentiment, we did not observe the token weight. If neither of the highly weighted tokens related to the positive or negative class appear, then the prediction should be neutral.

**  
Figure 3.3 —** SHAP for evaluating the weight / contribution of each text for specific class. Red = positive weight. Blue = negative weight.

**Source code summary:** The processing steps, choice of model, and parameters can be found in “source/config\_spacy/config\_tp.cfg” and “source/notebooks/build\_spacy\_sentiment.ipynb”.

## 3.3 Sentiment — Failure Analysis, Challenges, Limitation

* **Failure Analysis & Challenge:** We assume the reason for the weak performance of the model is because of the stopwords and punctuations. The config approach does not automatically filter the stopwords and punctuation. There is no parameter we can conveniently configure to add these filters to the pipeline. We still need to write a custom tokenization method and filter to remove stopwords, punctuation, and others. This adds to the learning curve since we will have to understand the SpaCy system from the documentation to solve this problem.
* **Limitation:** SpaCy utilizes neural network architecture to develop the models for various tasks like NER, pos-tagger, text classifier, etc. Hence, to take full advantage of SpaCy potential, one may need a GPU to train the neural network although we still can use CPU to train specific choice of models.   
    
  The system within SpaCy is very complex for starters also although we can use the config approach to build custom models. Hence, it relies heavily on prior knowledge related to NLP as well as neural networks architecture and parameters to ease the learning curve. Also, the config approach doesn’t allow us to perform hyperparameter tuning using the Bayesian optimization approach for neural networks.   
    
  Also, we had a limited time to figure out whether we can leverage the output from SpaCy NER and pos-tagger as features for the text categorizer or classifier.

## 3.4 SPY Movement — Feature Engineering & Representation

We will need to create the label, rolling t-score, to measure the SPY movement or volatility using the daily closing price from SPY. The rolling t-score is derived from the rolling window of one-year mean and standard deviation. We also remove the negative sign for the t-score since we are interested in measuring the volatility or movement and not the direction. Afterwards, we join the data from SPY with the CNN news articles by dates. We also have to remove rows that fall on weekends since the market is closed. Once the label is created and data is filtered, we will create the following features:

1. **T-score of the past 1 to 3 days:** We assume that there’s an autocorrelation between the current day t-score and the measure from 1 to 3 days ago.
2. **Extract latent topics / themes:** Extract the topics or themes from the topic model we had built previously. Then, we will transform them into one-hot encoding and drop the 1 category to avoid the dummy trap.
3. **Extract sentiment label:** Extract the sentiment label from the SpaCy model we had developed. We will also transform them into one-hot encoding and drop the 1 category to avoid the dummy trap.

We do not measure the correlation between the independent and dependent variables ahead of model training because we desire to explore the multiverse analysis to compare the performance variation between various combinations of features and approaches.

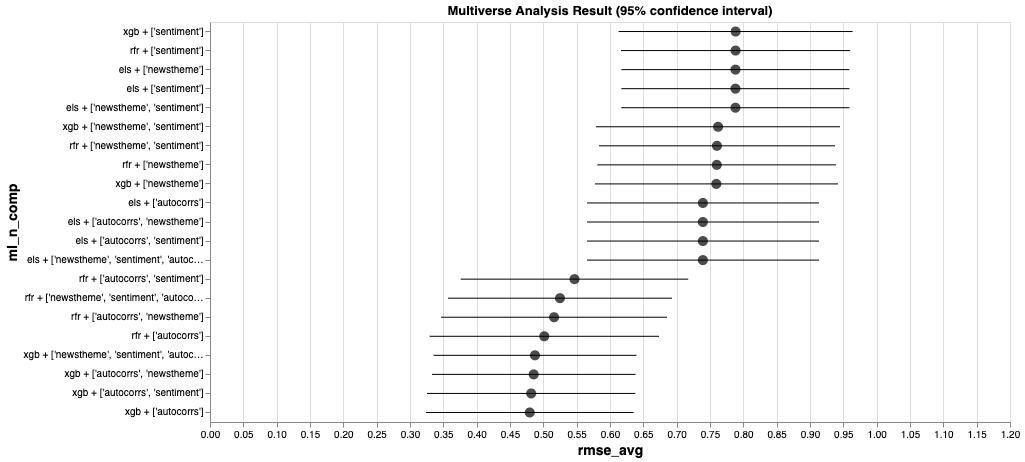
## 3.5 SPY Movement — Models, Evaluation, Explainer

We are interested in estimating the price movement of SPY ETF. Hence, this is a regression problem. The metric we use to measure the error is RMSE.

We use ElasticNet as an additional base model on top of the dummy regressor to compare the performance with the ensemble or boosting model. If the performance difference between the linear model and ensemble / boosting is not significantly huge, we might opt for a simpler linear model. The reason we choose ElasticNet over other regression models like Ridge and Lasso is because we can configure the L1 or L2 regularization parameter or assign the L1-L2 ratio.

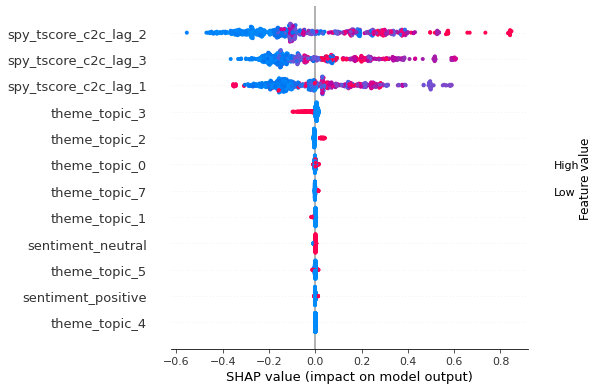
We also explore Random Forest and Xgboost. Random Forest addresses the variance (the performance deviation when predicting the validation set) more effectively than ElasticNet while reducing the bias. Whereas Xgboost is more effective at reducing the bias though it also addresses the variance by limiting the depth, pruning, or using the ensemble approach. These 3 models will help us to evaluate the tradeoffs between simplicity, variance, and bias when we compare each performance. Hence the choices.

For the optimization step, we utilize the Bayesian Optimization approach using “[Optuna](https://optuna.readthedocs.io/en/stable/)” for Random Forest and Xgboost since they have a wider optimization range for different parameters. Whereas for ElasticNet, we do not apply any optimization since the only parameter that we care about is the regularization method or L1-L2 ratio. Additionally, Optuna enables an early stopping during the optimization process. If the performance no longer increases beyond the threshold for multiple rounds, it will automatically stop the iteration. We only focus on optimizing the “max\_depth”, “n\_estimators”, and “min\_child\_weight” for both Random Forest and Xgboost to prevent the model from overfitting while improving its performance.

After setting up the training framework, we then perform a multiverse analysis to compare the performance and variance between various combinations of approaches, features, and models (figure 3.4). The baseline for the RMSE is 0.79± indicated by the red line (derived from the dummy regressor). Looking at the figure, the best performing models turn out to be Xgboost and Random Forest (the significant difference between the both is small). Whereas the most influential feature is t-score (of the past 1 to 3 days). This is unsurprising to us since market price movement generally has some degree of autocorrelation. Whereas for topic and sentiment features, they do not have significant influence on the model performance when we compare the difference between the mean and errorbar visually. We are expecting this since the topic feature is not well developed and the performance of the sentiment model is weak, not to mention the complex dynamic of the financial market.   
**Figure 3.4 —** Multiverse analysis. Y axis = model + components. Baseline = 0.79±. 

Diving into the RMSE, although the error is between 0.48 to 0.55 for the last 8 rows (figure 3.4), we do not think this is a good prediction result. The dependent variable, t-score, that we desire to estimate ranges between 0 to 3.0++. With an error of 0.5, that will equate to around half of the standard deviation (this is RMSE, not the exact t-score, so just an approximation). That means we will be overestimating or underestimating the volatility of SPY by approximately half of standard deviation most of the time. If we want to better measure the influence of news topics and sentiment as well as the autocorrelated feature on SPY, we will want the RMSE to be lower than 0.25 at least. The figure for 0.25 is a naive threshold that we use based on how the general traders categorize the volatility level. Generally, for SPY, a t-score of 0.25 indicates low volatility, 0.5 to 1.0 as medium, and 1.0 and above as high. If 0.25 is counted as low volatility, that means the price change is small or less significant. Hence the risk is less concerning.

After selecting the model and approach (Xgboost + “newstheme”, “sentiment”, “autocorrs”), we utilize SHAP to explain the model for us to further evaluate the influence in detail. Looking at figure 3.5, other than t-score, features like topic 2 (Crude Oil & Natural Gas) and 3 (Green & Sustainable Energy) seem to have a significantly small influence on the market volatility. However, if we were to observe figure 3.4 again, we will realize that the effect may be due to random chance since the difference or effect size is too small to consider a significant feature.

  
**Figure 3.5 —** SHAP Beeswarm. SHAP value = weight. Feature value = importance. Blue = low, Red = high.

Overall, we do not find topics or sentiment to be an influential factor for the SPY movement although we do observe some tiny influence for topic 2 and 3. However, that might be due to random chance. We are not interested in analyzing the autocorrelated feature (t-score) since the objective of our project here is to estimate the impact of the news topics and sentiment on SPY movement. Nonetheless, we do uncover the potential of text related features using various tools we mentioned in this report should we have access to proprietary or useful news or text related datasets.

**Source code summary:** The steps can be found in “source/notebooks/build\_mktmv\_regression.ipynb”.

## 3.6 Failure Analysis, Challenges, Limitation

1. **Failure Analysis & Challenge:** Financial market is a complex problem. Hence, we are not expecting a good result since topics and sentiments alone will not significantly influence the model prediction. However, we realize that the performance of the upstream task (topic and sentiment models) do affect the downstream task significantly. Financial market is already a difficult problem. If the upstream tasks do not yield satisfactory results or are not well developed, it will further aggravate the problem and bad performance of the downstream task.
2. **Limitation:** We have a limited row of news and topics to work with, hence we will not be able to properly evaluate the influence of the news topic and sentiment on the SPY movement. Also, the topic definition requires time to properly label them. Furthermore, the dataset is too small to obtain a wide range of topics (refer to unsupervised learning section). Nonetheless, we can reuse or extend this framework for similar tasks in the future.

## 3.7 Discussion / Learnings

After exploring the multiverse analysis, we think that it is a useful approach when comparing the different combinations of approaches, features, and models. We are able to measure the significant difference between each set of approaches or observe them visually if we desire to know whether a specific approach does contribute to the performance to a large degree and not due to random chance. Furthermore, instead of purely observing the feature weight, we can refer to the multiverse analysis to check whether there’s any large effect size or significant difference to draw a more holistic conclusion.

Although multiverse analysis is a useful approach. It does come with a significant downside. Since this approach utilizes the recursive approach to run various combinations of approaches, features, and models while simultaneously optimizing the model with SearchCV, it definitely consumes a significant amount of time to complete the process if the number of combinations is large. Nonetheless, we can still run a selective few combinations that might be insightful and useful if we already have some level of domain knowledge for the project. Alternatively, we can filter out the less impactful features using various metrics like pearson correlation, chi-squared, Mutual Information, Anova, Cramer, Phi, or others by measuring the correlation or the difference between the two distributions prior to multiverse analysis.

# 4. Moving Forward

This project focuses mostly on NLP related to financial news. Looking forward, we seek to extend this framework to include other features such as seasons, economic events, earnings results, etc. We also seek to learn deeper about SpaCy to be able to include custom functions, stopwords & punctuation removal, and others using SpaCy registry to see if we could improve the sentiment model further. Also, we will explore “Coreferee” to perform co-referencing first prior to entity extraction via Wikifier API.

Our analysis highlights that relatively manageable machine learning processes and analyses (in comparison to, for instance, cutting-edge high-frequency trading AI bots) have the capability to provide actionable insights, cut through the noise, and generate value for unsophisticated retail investors. If the project is to be extended in an enterprise environment, future areas of research may opt for closed sources of information such as Nasdaq DataLink, Bloomberg, WRDS, and CapitalIQ as well as greater computational resources available to institutional investors.

# 5. Ethical Considerations

There were no foreseeable ethical concerns relating to the datasets as the datasets used are publicly accessible data such as news articles and fund prices. Nonetheless, the ethics of deploying any ML tasks was top of mind throughout conceptualization and development. Throughout the project, we sought to minimize or mitigate harm in the model by adhering to ethical data science principles although we were not able to fully achieve them.

**Transparency & Accountability:** It is critical that the model is implemented with trust and transparency. We strived to version the datasets and models using DVC Studio although we abandoned it near the end due to time constraints and workload. One of our members also is not familiar with Git, hence we were unable to work on these effectively. However, we do utilize SHAP to explain ensemble and boosting models like Random Forest and XGBoost to ensure that we understand why the model makes a certain prediction. We were also interested in developing a framework for feature store and feature drift monitoring, but we gave up due to the limited timeframe available.

**Reproducibility:** We consider this an important matter and sought to make it reproducible to ensure good accountability and apple-to-apple feedback from team members. However, each of us have different levels of technical prowess, hence we were not able to make it reproducible entirely. Also, certain methods, especially neural networks and bayesian optimisation, that we used were not perfectly reproducible (the outcome is approximately similar although not exactly the same) though we have assigned a random seed. Due to a lack of version control of past models, a full comparison may not be possible.

# 6. Statement of Work

Team Member: Zheng Wei Lim (zhelim)

* Explore feature engineering methods for topic model and SPY movement.
* Develop visuals for model evaluation and interpretation for the topic model.
* Track model performance and hyperparameters exploration for the topic model.
* Report writing.

Team Member: Ye Jin Yu (yejinyu)

* Setting up the GPU cluster and neural network architecture for the sentiment model.
* Explore configuration and hyperparameters for the sentiment model.
* Develop modules for feature manipulation for the sentiment model.
* Develop visuals for cluster evaluation and interpretation for the sentiment model.

Team Member: Yee Dang Tay (liotayyd)

* Setting up repository, documentation, workflow, and project framework.
* Explore feature engineering methods for all supervised and unsupervised models.
* Develop modules for feature manipulation for all supervised and unsupervised models.
* Restructure code for readability and code maintenance.
* Report writing.