2023

Application of Sentiment Analysis in Stock Market Prediction

Jia Wei He
hexx1620@mylaurier.ca

Application of Sentiment Analysis in Stock Market Prediction

by

Jia Wei He

Thesis

Submitted to the Department of Mathematics

Faculty of Science

in partial fulfilment of the requirements for the

Master of Science in Mathematics

Wilfrid Laurier University

January 31, 2023

Acknowledgement

I would like to appreciate my co-supervisors, Drs. Roman Makarov and David Soave, for sharing their research experience with me. They also provide me with strong support for my research. I appreciate their guidance and encouragement during my master's study.

I also would like to appreciate Jake Tuero for providing the data set and excellent contribution for this thesis.

Many thanks to Dr. Zilin Wang for her support and participation in my presentation.

Abstract

This thesis aims to predict Apple's stock return using the sentiment of financial news headlines. There is a tremendous amount of financial news posts daily, and their headlines include crucial information. The financial news headlines may influence the readers and lead them to make certain trading decisions after reading the news. Therefore, financial news headlines could potentially impact the financial market. By understanding the connection between the headlines and the financial market, we can construct profitable trading strategies based on our analysis of daily headlines. Specifically, the sentiments in the headlines can result in positive, negative, or neutral impacts on the financial market. By analyzing the headline sentiments, we can learn the market trend. Before we examined the sentiments, we used natural language processing techniques to clean the texts and select essential features from the headlines. We also implemented the principal component analysis to reduce the dimension of the data set. Then, we constructed statistical models to explore how headline sentiments impact the financial market. Specifically, we focused on detecting how the headlines affect Apple's stock price. The headlines can impact the stock price positively, negatively, or neutrally. Logistic regression was proposed to model stock returns. We constructed one-stage, two-stage, and three-stage classification models. In addition, we used logistic, LASSO, and support vector machine (SVM) to construct regression models and analyze the connection between headline sentiments and stock price. Also, we introduced a new metric to assess the performance scores of multi-stage regression models so that we could select the best approach.

Furthermore, we constructed trading strategies consisting of vanilla call and put options based on the predictions made with the best models selected. We created nine trading strategies applied for the period from 2018-05-31 to 2019-05-31. The annual return for the best trading strategy is 169.265%. Even the worst trading strategy we constructed earned 58.39993% per year, while the annual return of the S&P 500 was 1.73% during this period. Therefore, our thesis successfully created multi-stage models that use financial news headlines to predict stock returns and applied predictions to construct trading strategies with high returns.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

A tremendous amount of financial news is posted daily. Financial news headlines usually contain the attitudes and objectives of the news, so the headlines may influence shareholders' trading decisions and consequently influence the stock market. The sentiment of texts represents the opinions or feelings expressed by text, which could be positive, neutral, or negative. Therefore, financial news headlines contain sentiments that could impact the stock market positively, neutrally, or negatively. For example, a headline positively impacts the stock price if the stock price increases unusually after the headline is posted. If we train an algorithm to detect the sentiments of headlines, we may be able to predict the financial market trend. Thus we introduce sentiment analysis and NLP techniques to identify the connection between financial news headlines and the stock market. Sentiment analysis aims to extract the sentiment strength from different textual sources to help make decisions [30]. This thesis aims to use headlines' sentiment to predict stock returns and construct trading strategies based on predictions.

1.2 Literature Review

1.2.1 Sentiment Analysis

Sentiment analysis can be applied to financial texts and help investors make decisions. Li et al. [19] analyzed the quantitative sentiment strength of textual news articles and found that a model that includes sentiment analysis outperformed a regular bag-of-word model. The bag-of-word model is a natural language processing technique that regards texts as a collection of words and uses the occurrence of words as features to do classification. They labeled the financial news articles as positive, neutral, or negative using the stock returns by setting two symmetric thresholds (upper and lower) for stock returns to classify financial texts into the three categories. For example, if the stock return was above the upper threshold after the financial headlines were posted, then the financial texts were labeled to be positive. Meanwhile, they used two dictionaries (Harvard psychological dictionary and Loughran–McDonald financial sentiment dictionary) to detect the financial texts' sentiment. In addition, they used the SVM (support vector machine) as a classification tool to train their model. They found that sentiment analysis improves prediction accuracy in validation and independent test data sets. The accuracy was measured by putting the diagonal values over the total sum of all elements in the confusion matrix. Therefore, sentiment analysis plays an essential role in financial text analysis. Meanwhile, Wang et al. [17] applied regression techniques to analyze the connection between sentiment words and financial risk. The volatility of stock return was used to measure the financial risk. Specifically, they used a financial dictionary to extract sentiment words from the texts to build a regression model and compared it with a regular bag-of-word model. They found that the model built with only sentiment words demonstrated comparable performance to the bag-of-word model, which further proved the importance of sentiment analysis in the process of the financial text. The models they had built suggested strong relationships between sentiment words and the risk of companies. Thus sentiment analysis is an important tool in predicting stock return because the risk of companies is valuable information in the stock market that benefits investors who make short trades. Meyer et al. [23] analyzed the daily financial news to detect how the financial news affects the stock market. They assumed that the financial

news headline sentiment reflected the sentiment of the financial news article. Also, they labeled the positive headline sentiment with +1 and the negative headline sentiment with −1, depending on the stock price. In other words, the headline was labeled with positive (+1) if the stock price was expected to rise after the headline was posted. They replaced all companies' names with the tag "CN" so that the frequency of the pattern was increased and yielded a more stable model. In addition, they used the SVM models as the classification tool to predict the stock price. They also defined some sentiment prediction methods that find some word combinations from the headlines as features such as Google-profit-beat, and they found this technique outperformed regular bag-of-word models. Therefore, a model that includes word combinations may have higher accuracy than regular bag-of-word models.

1.2.2 Logistic Regression

Logistic regression is widely used in many fields. Prabhat et al. [24] classified the sentiment using the Naive Bayes and logistic regression methods. The data (twitter reviews) were labeled with positive and negative sentiments. They estimated the performance of regression models by accuracy and precision from the confusion matrix. They found that logistic regression gave 10.1% more accurate results and was implemented five times faster for the same data size than Naive Bayes. Finally, the logistic regression had 76.767% accuracy and 73.575% precision, and logistic regression only took 3689 milliseconds to run the entire logistic regression model and finish the classification of sentiment. Therefore, logistic regression is a great sentiment classification tool. Meanwhile, Hasanli et al. [28] applied logistic regression, Naive Bayes, and SVM to analyze sentiments. They found that both logistic regression and SVM worked well when using bag-of-words as features. Before classification, they applied natural language process (NLP) techniques to clean the texts, including cleaning noise words, and converting words to lower case. Then they created the data set with the bag-of-word model and labeled the data as positive or negative. Lastly, logistic regression, Naive Bayes, and SVM were applied to classify the sentiment of texts (twitter data). They found the logistic regression and SVM both worked well and had

the same accuracy with bag-of-words features. Thus both logistic regression and SVM are excellent choices for analyzing the financial headline sentiment.

1.2.3 Principal Component Analysis

The principal component analysis (PCA) can reduce the data set's dimension and work as a feature selection tool by controlling the number of principal components. In this thesis, PCA is our choice to reduce the dimension of the bag-of-word model. Zainuddin et al. [25] combined PCA and SVM to construct a hybrid sentiment classification for Twitter data. They found that the hybrid method could increase the accuracy performance by 76.55% compared with normal SVM models. The SVM method was the only classification tool they applied to classify the sentiment, but they used principal component analysis (PCA), latent semantic analysis (LSA), and random projection (RP) as feature selection tools before classification. They compared the performance of PCA+SVM, LSA+SVM, and RP+SVM by confusion matrix, and the models built with PCA+SVM outperformed the others. Therefore, the combination of PCA and SVM could also be applied to classify financial headline sentiment. At the same time, we also combined PCA with a logistic regression model with or without penalization. In addition, Vinodhini et al. [16] applied PCA as a feature reduction method for sentiment analysis of product reviews because a large number of features makes many potential sentiment analysis applications infeasible. They used the accuracy in the confusion matrix and the receiver operating characteristic (ROC) curve to estimate the performance of SVM and Naive Bayes models with or without PCA. They found that both the SVM and Naive Bayes models worked better being combined with PCA. Specifically, the models with PCA had higher accuracy and more area under the curve in the ROC plot. Besides, the SVM models always outperformed Naive Bayes with or without PCA. Therefore, the combination of PCA and SVM could be a good hybrid model because it both worked well in Zainuddin's and Vinodhini's research.

1.2.4 Trading Strategies

Trading strategies can be made if we can predict the stock returns using the headlines' sentiment. Kazemian et al. [21] analyzed the sentiment of Reuters news documents. The news they used was labeled to be positive, neutral, or negative by two human annotators depending on the author's belief about the company. They designed trading strategies based on their predictions about news's sentiment. For example, they designed a momentum strategy that compared the current stock price and the stock price h days ago, where h was defined depending on the models. If the current stock price is higher, the strategy goes long for h days. If the current stock price is lower, the strategy goes short for h days. At the same time, they made trading strategies that compared the current stock prices and the stock price h days in the future and made trading decisions based on the comparison. At last, they made strategies that ignored the financial news and just went long on the S&P 500 in the holding period.

1.3 Contribution

The objective of this thesis is to construct profitable trading strategies by using the financial news headlines to predict stock returns. We listed our four main contributions below.

- We applied NLP techniques to process the headlines and extract features from processed headlines. We also calculated the sentiment scores for the headlines collected from Reuters. In addition, we used the Alpha Vantage API in Python to download the Apple stock indexes. Lastly, we created our data set with the occurrence of features, sentiment scores, and stock indexes.

- We constructed logistic regression models with the combinations of PCA+logistic, PCA+LASSO, and PCA+SVM. Also, we introduced an algorithm by defining particular matrices to assess the performance of regression models so that we could identify the best models by comparing their performances.

- We built one-stage, two-stage, and three-stage logistic regression models in Section in Section 3.3 to classify multiple sentiments of headlines based on the stock returns.

- We constructed option trading strategies and applied them on the period from 2018-05-31 to 2019-05-31 based on the predictions of regression models. Our best trading strategy had an annual return of 169.265%. The annual return of our worst trading strategy was 58.39993%. At the same time, the annual return of the S&P 500 was 1.73% during this period. Therefore, the annual returns of our trading strategies were much higher than the annual return of S&P 500.

We organize this thesis as follows. First, in Chapter 2, we propose some natural language processing (NLP) techniques to process financial news headlines, extract sentiment features, and generate the bag-of-word model. In Chapter 3, we construct the final data set with features extracted from headlines and financial data downloaded by Alpha Vantage API. Meanwhile, we introduce one-stage, two-stage, and three-stage logistic regression models to classify the headlines using categories of stock returns. In addition, we combine PCA with logistic, LASSO, and SVM models to predict stock returns. In Chapter 4, we construct trading strategies based on our classification models and verify if they allow for profitable trading. In Chapter 5, we summarize our results and outline the future work.

Chapter 2

Application of NLP Techniques to Textual Data

Natural language processing (NLP) is applied to understand given natural texts or speech and transform them into things we need [29]. We implement the NLP method to convert the qualitative texts to quantitative data so we can analyze the numeric data with regression models. One of the most popular NLP techniques is sentiment analysis which extracts the opinions or attitudes contained in the texts. In this thesis, we propose the NLTK (Natural Language Toolkit) package to help us extract the sentiment scores. The NLTK is a popular open-source package in Python that is widely used in the NLP field. The bag-of-word method is another popular NLP tool, which is widely used in financial text analysis. We apply the bag-of-word method to count the occurrence of words and create the data set. Some words in the headlines are extracted as features to build the regression models. Also, individual words in different word combinations may have dramatically different meanings, so we also count the occurrence of word combinations. In addition, the position of words in headlines may lead to different sentiments. The words at the headlines' start and end can contain different intensities. Thus we count the occurrence of words at the start and the end of headlines separately to extract features.

A few steps are required to clean headlines and transform them into standardized textual data before we extract the features from them. Firstly, we remove the noise words and

substitute all synonyms with one word. Secondly, we apply tokenization to split sentences into individual words to count the occurrence of words. In addition, lemmatization is also used to restore the base form of words so that the same words with different versions can be regarded as a single word. Lastly, we extract features from the cleaned and standardized texts for further analysis.

2.1 Clean the Text

Cleaning the text is a crucial step for feature extraction. The headlines we collected contain a tremendous amount of redundant and duplicated information that could lead to overlapping extracted features. We implement some NLP techniques to clean the financial news headlines thoroughly.

2.1.1 Remove Noise in Headlines

The objective of this section is to remove the content that is irrelevant to the sentiments of headlines, such as noise words and characters. The noise words could be stop words and meaningless words that are not relevant to the subject of the headlines. The noise characters could be delimiting characters and punctuation marks.

The stop words could be identified as short words but have the same likelihood of occurring in the text as words that are relevant to the subject [1]. There is no universal stop word list that is suitable for all situations. We define a list of stop words based on our purpose. It consists of words irrelevant to the sentiment or not impacting stock prices.
Stop Word List = [”a”, ”to”, ”in”, ”on”, ”update”, ”of”, ”for”, ”says”, ”say”, ”they”, ”as”, ”at”, ”with”, ”by”, ”and”, ”the”, ”is”, ”it’s”, ”it”, ”its”]

The words on the above list represent redundant information that doesn’t help the regression models predict the stock return, so all the words on the list are not included in the final data set for further analysis. The meaningless words are those words that are irrelevant to the subject of the headlines. For example, some headlines may include the topic or categories of headlines such as ”Morning News Call” and ”Deals of the Day”. Those meaningless words

are not connected to the actual content of the news, so we create a list to collect them and exclude all such words in the list from the headlines. We also remove any special character by excluding all the words with lengths smaller than two. The sentiments for those characters or words whose lengths smaller than two are not significant for our analysis. In conclusion, we preliminarily clean the data by excluding all the noise words. Next, we remove the noise characters to clean the headlines further.

Recall that the noise characters are delimiting characters and punctuation marks. The delimiting character is applied to separate words. They are not relevant to any content of headlines, and their sentiments are difficult to recognize, so we create a delimiting list and remove all characters on the list:

Delimiting List = ["(", ")", "[", "]", "<", ">", "{", "}", " ' ", " " ", " ‘ ", "'s", "‘s", "'S", "‘S", "/"]

Also, punctuation marks are not features we need to analyze the headlines' impact. The punctuation marks like commas and periods show up frequently. They could be extracted as features with high possibilities if we do not remove them, so we create a list of punctuation marks and remove all of them.

Punctuation List= [",", ";", ":", "-", ".", "!", "?"]

However, we still need to separate different sentences, so we use the "|" symbol to replace the semicolons and periods in the texts.

In conclusion, we remove the noisy words and characters to clean the text. However, it is insufficient to remove the noises in the headlines. We still need to keep pre-processing the headlines to get clean and standardized texts so that the extracted features are not overlapped.

2.1.2 Substitution

The substitution implementation aims to standardize the headlines to guarantee that no overlapped features are extracted for further analysis. Specifically, we substitute natural numbers, synonyms and homology words, and capital characters with universal tags.

First, we substitute some particular natural numbers with words so that all of those num-

bers are standardized to universal forms. For example, we transform all the types of millions to AMOUNT_MILLION, including 1000000,1,000,000, million, millions, Million, Millions. Meanwhile, different expressions of countries' names are also substituted to a universal form. For example, we substitute u.s., u.s, Us, US, and United States with United_States to ensure we do not repeatedly extract the countries' names.

Second, some synonyms and homology words are substituted with universal words to ensure that the extracted features are exclusive. In this step, the most imperative substitution is the synonyms and homology words for Apple products or shares because the extracted features are applied to build regression models that predict Apple's stock performance. We categorize the Apple's relevant words to Apple product, Apple settlement, Apple shares, and Apple shareholder. Apple product includes all the devices manufactured by Apple, so all the Apple products like the iPhone and MacBook are substituted with APPLE_PRODUCT. If the headline reveals that Apple will sue some company or be paid, we substitute those words with a positive Apple settlement. For example, if we find "must pay Apple" in the headlines, we substitute "must pay Apple" with AAPL_SETTLEMENT_GOOD. However, if Apple pays some institutions or some institutions sue Apple are found in the headlines, we replace them with a negative Apple settlement. For example, if we find "lawsuit against Apple" in the headlines, we replace "lawsuit against Apple" with AAPL_SETTLEMENT_BAD. In addition, we replace all the words that are relevant to Apple shares and Apple shareholders in the headlines with APPLE_SHARES and APPLE_SHAREHOLDER. The point of these implemented substitutions is to standardize different Apple-relevant features to the four standard Apple features. Although we aim to analyze Apple's stock price with the extracted features, headlines relevant to other companies may also impact Apple's stock price. We also propose substitution methods for the headlines that are relevant to Google, Boeing, Microsoft, McDonald, etc.

At last, we convert all capital characters to lower case so that the same words are not extracted twice as different words from the headlines. After removing noises and implementing substitutions, we preliminarily cleaned headlines. Next, we need to start processing the headlines to ensure the texts are feasible for feature extraction.

2.2 Process the Text

The financial news headlines are preliminarily cleaned and standardized in 2.1. However, we still need to process the headlines to extract features directly from the texts. The objective for this section is to split sentences into words and lemmatize all the words individually. The headlines are tokenized and then added to a word list. Then all the words are lemmatized to a base form. In addition, we include two other word combinations such as bigrams and words with positions in the word list.

2.2.1 Tokenization and Lemmatization

Tokenization is an imperative technique in NLP techniques that can split words into sub-word groups and remove punctuation and unnecessary tags [26]. We use the NLTK package in Python to tokenize the headlines into sub-words groups. Then we generate a word list by collecting the words from tokenization. Developing a word list makes it feasible for the final cleaning of the texts and makes it easier to extract the features. In conclusion, tokenization is the first step in generating the word list. We split all the headlines into individual words and put all of those words in the word list.

We implement lemmatization to find normalized forms of all words [5]. Again the NLTK package is used to lemmatize the word list. However, we find a problem during the process of lemmatization. Some particular words may have different meanings. For example, the word “issue” could have two different meanings depending on its part of speech. In this subsection, we tackle this problem by implementing lemmatization on each word’s most frequently used meaning. We process the word ”issue” as a noun and lemmatize with its meaning for a noun.

In summary, with tokenization and lemmatization, we construct a lemmatized word list. All the words on this list are potential features that could be applied to build regression models. Moreover, we add more potential features to this word list to improve the performance of our models as explained in the next section.

2.2.2 Words Bigrams and Positions

In Section 2.2.1, we generate a word list from the headlines. All the words on the list can be potentially extracted as features. Although we consider all individual words' impacts on the financial market, we are still dedicated to finding more features to emphasize the capability of the regression models. Sometimes the combination of words may contain stronger sentiment than the cumulative sentiment from the words that make the combinations. In other words, the word combination could be a more impactful factor that affects the financial market with its sentiment. Therefore, we define word bigrams as potential features so that word combinations can be added to regression models. First, we apply the NLTK package again and find all the word bigrams in the headlines. Figure 2.2.1 reveals the bigram groups we receive after applying NLTK to unprocessed headlines [7]. In addition, we

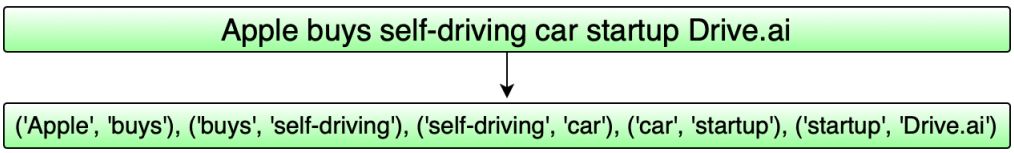

Figure 2.2.1: Bigram groups extracted from a sample headline

construct a bigram list to collect all the word combinations, and the bigrams in each group are connected with the symbol $\sim$ so that each bigram group can be regarded as a single feature. Meanwhile, the words' location can also emphasize the headlines' sentiment. Thus, we detect if the words are located in the first or the second half of the headline, then assign all the words in the headlines to their corresponding locations and add those words with positions to the word list. We connect all the words in the first and second half with *..start* and *..end*, respectively. Figure 2.2.2 shows how we detect the locations and generate words with positions with unprocessed headlines.

2.2.3 Feature Extraction

We preliminarily cleaned and standardized the headlines in the previous sections. The next step is to extract the crucial features from the headlines. Section 2.2.1 adds the words

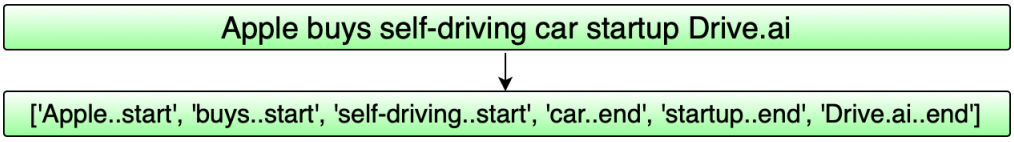

Figure 2.2.2: Examples of words with locations

in the headlines to a word list after tokenization and lemmatization. In Section 2.2.2, we generate bigram features and position features. Due to the different processing methods in this section, we add the bigram features to a new word list generated in Section 2.2.2. In contrast, the position features are added to the original word list generated in Section 2.2.1 so that we can extract the features conveniently. We are setting two word lists because we are collecting the features in order. The bigrams and words with positions may overlap if we add all the features in one list.

We extract the features by detecting the frequency of occurrences for the features. A feature can only be recognized as significant and added to the final list if it frequently occurs in the headlines in different timelines. Otherwise, we exclude the features because they cannot provide enough information for our regression models. Therefore, we set a threshold on the counts to extract the appropriate features. We set the threshold equal to five, meaning we only extract the features that appear in at least five headlines. More specifically, we detect the distribution of two lists; only the words that occur more than five times in the list can be extracted and added to the final feature list. In conclusion, the features in the final list consist of regular words, bigram groups, and words with positions that appear at least five times in the headlines.

2.3 Sentiment Analysis

Sentiment analysis is an NLP approach that extracts authors' opinions or feelings from texts. The extracted opinions could influence shareholder's behavior in a positive, neutral, or negative way, and thus the intensity of headlines affects the volatility of stock return. Therefore, analyzing sentiment intensities is extremely helpful in identifying how financial

news headlines impact the stock market. First, we used some Python packages to pre-process and standardize the headlines and transform the texts into their dictionary form, which had already been done in Section 2.1 and 2.2. Furthermore, we put the processed texts into a score function in Python and computed the quantitative intensities of the sentiment strength. In this section, we analyze the intensity of the headline sentiment to better understand the information contained in headlines.

2.3.1 Sentiment Intensities

Although we collect single words, bigrams, and words with positions as features for further analysis, we still want to find some variables that can present a comprehensive sentiment estimation for each headline. Therefore, we use a score function in Python to calculate the sentiment scores of each headline so that we can analyze the quantitative sentiment of texts. To obtain the sentiment score for a headline, we first calculate the sentiment scores for the words in the headline. It is more feasible to extract the sentiment scores for each word, aggregate the sentiment scores for individual words and then calculate the score for the whole headline.

We propose to use VADER in NLTK to calculate the sentiment score for headlines, where VADER can assign polarity and intensity values to the textual data [27]. However, before calculating the sentiment scores of the headlines, we need to apply some NLP techniques to pre-process the headlines. Figure 2.3.1 shows the steps that we need to take before the score of a headline is calculated.

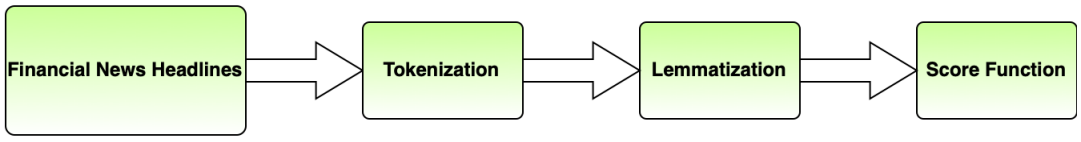

Figure 2.3.1: The steps to pre-process the financial news headlines before applying the score function

We first implement tokenization to split the headlines into individual words. Then we

lemmatize those words to transform different forms of words to their standard base form. After pre-processing the headlines, we use VADER to calculate the sentiment scores. VADER uses a lexical-sentiment dictionary that maps the linguistic features to the sentiment intensities. Next, each word in the lexicon is assigned scores developed by ten independent human raters. Finally, the scores for the headlines are calculated with this unique lexicon. The scores generated by VADER consist of positive, negative, neutral, and compound values. The positive, negative, and neutral scores are the proportion ratio of their total sentiment scores. Thus, their sum is one for each headline [18]. According to the formulas in VADER, the positive, negative, and neutral scores are limited from 0 to 1. The normalized compound score varies from -1 to 1. Figure 2.3.2 gives an example of how we pre-process the headlines and the scores we calculate using VADER. [18]. In conclusion, with VADER we

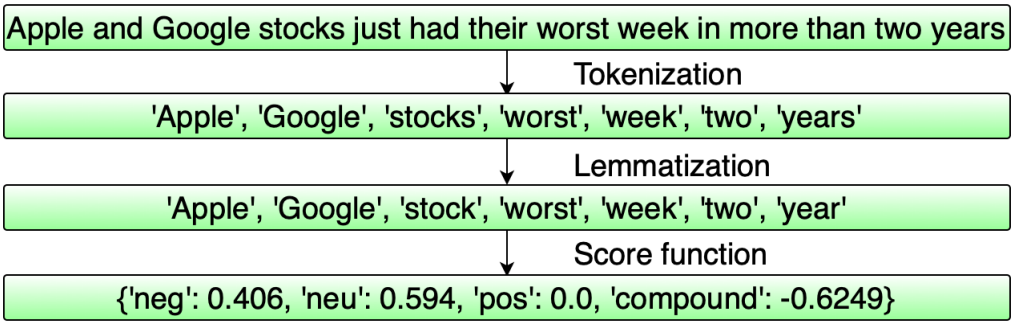

Figure 2.3.2: There are two steps used to process the texts, and the last line shows the output from VADER

can calculate the quantitative intensity of the sentiment for each financial news headline. Since there may have multiple headlines posted on some days, we will calculate the sum of each sentiment category and use the sum of positive, negative, neutral, and compound as four individual features for the regression models.

Chapter 3

Data Analysis

In Chapter 2, we implemented NLP techniques to process headlines and used the bag-of-word model to count the occurrence of selected features. In this Chapter, we describe the final numerical data set with the occurrence of features and financial data. In addition, we build regression models to predict the impact of headlines on stock returns.

We propose a new method in Section 3.3.5 to measure the performance of each regression model and compare the models' performance scores. Also, even though we cleaned headlines before generating the numerical data set, we still have a data set with dramatically large dimensions. Therefore, we apply the principal components analysis (PCA) to the final data set to reduce the number of features for model fitting. Then we use one-stage, two-stage, and three-stage regression models to predict the impact of headlines on stock returns. Also, we implement the k-folds cross-validation to guarantee the fairness of the model assessment. We use several statistical models such as the logistic, support vector machine (SVM), and LASSO to build regression models with principal components.

3.1 Final Data Set

Recall that we extracted significant features from the headlines in the previous chapter. In this subsection, we combine those features with financial data to generate the final data set. We collect the occurrence of the significant features and stock data on each trading date. There may be more than one headline posted on some trading days, so we detect the

occurrence of each feature in all the headlines posted on the same day. If any feature shows up in one of the headlines for a given trading date, then we append 1 to the column of this feature; otherwise, we append 0 to this column. Additionally, we introduce the polarity of sentiment and thresholds that can classify the headlines based on the stock return values.

3.1.1 Financial Data Selected

The financial data consists of the stock prices of Apple from 2012/01/01 to 2019/05/31, including close prices and dates. We calculate logreturn and zlogreturn for each day i as follows:

$$logreturn[i] = \ln \frac{Close[i]}{Close[i-1]} \tag{3.1.1}$$

$$zlogreturn[i] = \frac{logreturn[i] - \mu_r}{\sigma_r} \tag{3.1.2}$$

where $\mu_r = \frac{1}{n}\sum_{i=1}^{n} logreturn[i]$ and $\sigma_r = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(logreturn[i] - \mu_r)^2}$, and $Close[i]$ is the close price on day i. The value of n represents the total number of trading days in the data set which is 1502. The Alpha Vantage API allows us to download stock information directly in the Python code. Finally, with the financial data downloaded from Alpha Vantage, we generate our final data set consisting of features from headlines and Apple stock information.

3.1.2 Threshold of Impact

In this section, we investigate how financial headlines impact the stock market by tracking the return of Apple's stock. Specifically, the zlogreturn defined in (3.1.2) is used to classify the impact of a headline. The zlogreturn represents the trend on the stock price over two successive trading days. Given headlines and the target (impact on zlogreturn values), the classification for the sentiment polarity of headlines toward the target can be done by setting a threshold [20]. First, we find the empirical quantile function for zlogreturn as given in (3.1.3) and (3.1.4). Then we select an appropriate quantile of zlogreturn as the threshold. Since we use three or five categories to classify the stock returns, the impact thresholds vary

for different classification categories.

First, we set lower and upper thresholds for the three categories to classify the sentiment of a headline as negative, neutral, or positive. If the zlogreturn on some date is below the lower threshold, then the headlines posted on that date are regarded as containing negative sentiment. If the zlogreturn is above the upper threshold, the respective headlines are considered to contain positive sentiment, meaning those headlines bring positive impacts to the stock. The rest of the headlines are just regarded as neutral, which do not bring any significant impact on the stock price. Table 3.1.1 shows how the thresholds are applied to classify the sentiment into three categories. We use the quantile function to select the

zlogreturn		
zlogreturn < Lower Bound	Lower Bound < zlogreturn < Upper Bound	zlogreturn > Upper Bound
Negative Impact	**Neutral Impact**	**Positive Impact**

Table 3.1.1: Classification of three categories of stock returns

upper and lower thresholds, where the 90% or 95% percentile of zlogreturn is chosen to be the upper threshold, and the 5% or 10% percentile of zlogreturn is selected to be the lower threshold. We have four threshold combinations: 5% and 95%, 5% and 90%, 10% and 90%, and 10% and 95%, respectively. Later, we will train the regression models and select the best combination of the thresholds. We used the quantile function in R [3] to set the thresholds. The quantile number $Q(p)$ is calculated as follows. First, for a given value of p, we calculate the interpolation point k:

$$k = p \cdot (n - 1) + 1 \tag{3.1.3}$$

The interpolation point is used to separate the first 100 p% of data from the rest. Here, n is the total number of observations. Now, with the interpolation point k, the quantile number is given as follows:

$$Q(p) = (1 - p)Z_{\lfloor k \rfloor} + pZ_{\lceil k \rceil} \tag{3.1.4}$$

where the $\lfloor k \rfloor$ and $\lceil k \rceil$ are the floor and ceiling functions, respectively; Z values denote the sorted zlogreturn values from small to large, and Z_k represents the kth zlogreturn value after sorting.

The selection of thresholds for five classification categories is more complicated. We need four different thresholds to classify the sentiment completely. Since this thesis focuses on the two-stage logistic regression, we don't try different combinations of thresholds for five categories of sentiment and define the thresholds to be 5%, 30%, 70%, and 95% to classify the headlines as strong negative, negative, neutral, positive and strong positive, respectively. Table 3.1.2 illustrates how we organize the headlines into five categories. Based

zlogreturn				
zlogreturn < Q(0.05)	Q(0.05) < zlogreturn <Q(0.3)	Q(0.3) < zlogreturn <Q(0.7)	Q(0.7) < zlogreturn <Q(0.95)	zlogreturn > Q(0.95)
Strong Negative	**Negative**	**Neutral**	**Positive**	**Strong Positive**

Table 3.1.2: Classification with five categories of stock returns, $Q(p)$ denotes the empirical quantile function fo zlogreturn defined in (3.1.4).

on Table 3.1.2, the total number of observations classified as strong negative, negative, neutral, positive and strong positive, are 66, 325, 520, 325, and 66, respectively.

In conclusion, setting thresholds for zlogreturn is the last step before building regression models. With the thresholds, we classify the daily headlines into individual sentiment categories. The response variable that represents the sentiment is denoted by zvalue. We construct zvalue (it is a binary variable) to classify the headlines at different stages. For example, let us consider a two-stage regression model that classifies positive and non-positive at the first stage and then normal and negative at the second stage. At the first stage, having zlogreturn above the upper threshold on some days means that the headlines posted on this day contain positive sentiment; thus, zvalue on this day is 1. Headlines on other dates with zlogreturn below the upper threshold have zvalue equal to zero. At the second stage, we classify non-positive headlines (zvalue=0 at the first stage) into negative and neutral. For the headlines posted on the dates with zlogreturn below the lower threshold, zvalue equals 1, otherwise it equals 0. Generally, zvalue represents the occurrence of positive and negative sentiments at the first and second stage, respectively. Most of the work done in

this thesis is analysis for three categories, but we can do more analysis for five categories in the future. Although the five categories can give us more insights than the three categories, we are facing additional challenges with the five categories. For example, we need to tune additional parameters and the data we have may not be sufficient for training when using five return categories.

3.2 Cross-Validation

Before building the regression models, we still need a method to estimate the performance of regression models in an unbiased way. Therefore, the k-fold cross-validation is applied to estimate the performance of regression models and tune the models' parameters. The k-fold cross-validation splits the data set into k non-overlapping folds and assigns them to the training and testing data subsets. The nested cross-validation is applied when constructing LASSO and SVM models. Nested cross-validation is a common approach applied to select the optimized hyperparameter in the inner cross-validation, while the outer cross-validation computes the unbiased estimation of the model's performance [31].

3.2.1 Balanced Data Set

In Section 3.1.2, we introduced the threshold of impact, and our headlines were split into three or five categories. We randomly redistribute the training data by sample function [15] in R when doing the k-fold cross-validation. Therefore, the training and testing data set would be unbalanced if we apply the k-fold cross-validation to the data set without balancing it in advance. For example, in the training data, 30% of headlines may be positive impactful (zlogreturn>upper bound), but in test data, there may only have 15% of headlines that are positive impactful. The unbalanced organization of training and testing data sets may increase extra misclassification. Therefore, when we build the regression models for three return categories, the data set is split into three data subsets using the thresholds of impact. Thus we first create three collections consisting of only positive, neutral, and negative headlines, respectively. Next, the resampling methods are implemented on each data subset

so that the training and test sets are composed of samples selected proportionally from the three collections. Meanwhile, before building regression models with five return categories, we split the headlines into five data collections consisting of strong positive, positive, neutral, negative, and strong negative headlines, respectively. After that, the training and testing sets are sampled from the data subsets in a balanced way.

3.2.2 K-fold Cross-Validation

Cross-validation is proposed as a resampling method in this section to create an unbiased assessment of the out-of-sample performance of the regression models. Specifically, cross-validation splits the data into training data applied to train the regression models and testing data used to validate the model's performance [8]. The k-fold cross-validation is implemented by splitting the data into k non-overlapping folds so that one fold is picked up as the testing data set, and the remaining $k-1$ folds are used as the training data set. Since the data set is split into k fold, selecting the training and validation data set is repeated k times. Therefore, k-fold cross-validation gives a fair and unbiased evaluation when training the models.

Specifically, we implement 10-fold cross-validation, which means that 10% of the total data is used as testing data, and 90% of the total data is used as training data. Meanwhile, recall that we introduced the balanced sampling in Section 3.2.1. Thus the 10-fold cross-validation is implemented to each data subset, and we extract one fold from each data subset as the testing data set and use the remaining as the training data set. For example, when we create regression models with three categories, the original data set is split into three data subsets: positive, neutral, and negative. We split each subset into ten folds and select one fold from each collection as the validation set. The remaining nine folds in the data subsets form the training data set. We implement the 10-fold cross-validation to the training data from 2012-01-01 to 2018-05-31 to tune parameters for the regression models. Once the 10-fold cross-validation identifies the best model, we use all the training data to train the best model and then apply it to the validation data from 2018-06-01 to 2019-05-31. Additionally, when training the LASSO and SVM models, we want to use the optimized parameters

(introduced in Section 3.7.2 and 3.8.1) during the training process. Thus we apply 10-fold cross-validation on the training data that composed of 9 folds of the original data set in the period from 2012-01-01 to 2018-05-31 to tune the models' hyperparameters. Also, the 10-fold cross-validation generates ten confusion matrices during the training process. We collect the sum of each category in the confusion matrix to construct an aggregate confusion matrix for further analysis. For example, if we are using cross-validation to estimate the performance of a regression model that classifies positive and negative. The aggregate confusion matrix is calculated as described in Table 3.2.1. Specifically, in the aggregate

	Predict Positive	Predict Negative
Actual Positive	True Positive = $\sum_1^{10}$ true positive	False Negative = $\sum_1^{10}$ false negative
Actual Negative	False Positive = $\sum_1^{10}$ false positive	True Negative = $\sum_1^{10}$ true negative

Table 3.2.1: Example of the aggregate confusion matrix from the 10-fold cross-validation

confusion matrix, we calculate the sums of the true positive, false negative, false positive, and true negative, respectively, all collected from the 10-fold cross-validation.

3.3 Logistic Regression

Logistic regression is a powerful method used to analyze the impact of independent variables on binary outcomes by testing the unique contribution of each independent variable in the regression model [13]. Logistic regression is usually proposed to create regression models with two-class response variables. A logistic regression model returns the probabilities of success of one outcome. In this thesis, we propose logistic regression to analyze the impact of headlines, and it can return the probability that a headline is impactful. The logistic function is a smooth and monotonic function that maps the real value to limit bounds [2]. The logistic function in logistic regression returns the probability that a headline is impactful, which means its value is bounded between 0 and 1. The typical logistic function $f(x)$ is defined by

$$f(x) = \frac{1}{1 + e^{-Y}}, \tag{3.3.1}$$

where

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \cdots + \beta_k X_k \tag{3.3.2}$$

The intercept β_0 and other $\beta's$ are regression coefficients of independent variables. The Y value is a linear combination of independent variables, and k is the number of independent variables. $f(X)$ is the probability that the headline described by the vector X is impactful.

In conclusion, the logistic regression models are usually proposed to analyze two-class variables and return the probability of success for one event, so a probability threshold is required to classify the classes based on the probabilities generated by logistic regression models.

3.3.1 Probability Threshold

Logistic regression is proposed to analyze the data with binary variables. The binary variable is the zvlaue introduced in Section 3.1.2. Recall that the logistic function returns the probability of the event's impact. Therefore, we can only investigate the event's probability of impact instead of the classification between impactful and non-impactful. Thus a threshold is required to assist in the classification of binary variables. If the probability calculated by the logistic regression model is above this threshold, we classify this prediction to be 1, otherwise the prediction is 0. The threshold for probability is learned with the training data and selected by the measure of accuracy and precision. Specifically, the accuracy and precision are measured by sensitivity and specificity, two essential descriptions of the accuracy of a statistical model. The formulas of sensitivity and specificity are given below:

$$\text{sensitivity} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}} \tag{3.3.3}$$

$$\text{specificity} = \frac{\text{True negative}}{\text{True negative} + \text{False positive}} \tag{3.3.4}$$

We are looking for a threshold that maximizes the sum of sensitivity and specificity with the given training data.

The receiver operating characteristic (ROC) curve is introduced to find the best probability threshold that maximizes the regression model's sensitivity and specificity. The ROC curve is created by plotting sensitivity (true positive rate) on the y-axis against 1–specificity (false positive rate) on the x-axis to estimate the performance of the binary classifier [22]. Thus the points at the top left of the ROC curve are typically selected as the threshold. We further calculate the sum of sensitivity and specificity and choose the point that maximizes it. In addition, the ROC curve could be used to estimate the performance of the logistic model by investigating the points of the ROC curve and the area under the curve. The higher the area under the curves, the better performance the regression model presents to us. We use the pROC package [12] in R to extract the threshold that maximizes the sum of sensitivity and specificity. Figure 3.3.1 is a ROC curve created from the positive one-stage SVM model. In this model, we classify the positive and non-positive headline sentiments.

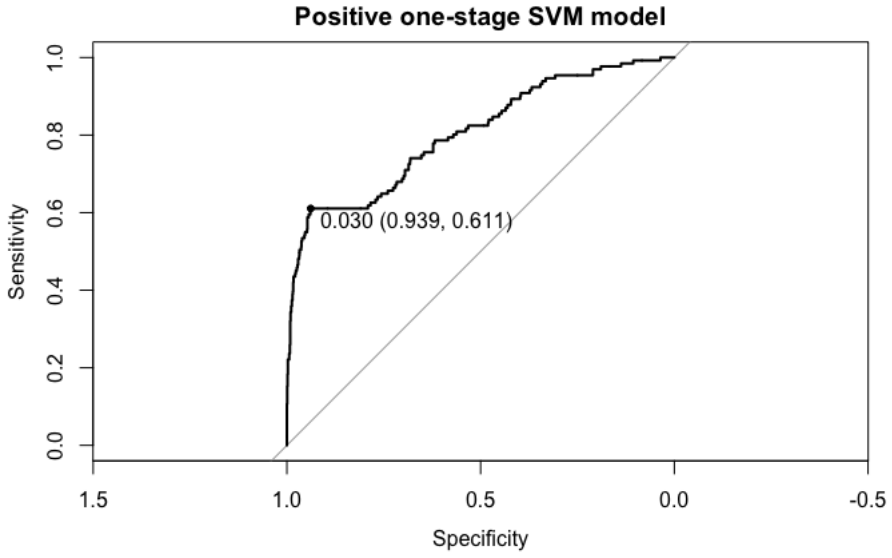

Figure 3.3.1: The ROC curve for the positive one-stage SVM mode, the value 0.03 is the selected probability threshold. The values 0.939 and 0.611 represent the specificity and sensitivity, respectively.

Figure 3.3.1 illustrates how we select the probability threshold. We point out the best threshold in the plot. Usually, the points in the top left are appropriate thresholds for the regression models, and the point we selected is the one that maximizes the sum of specificity and sensitivity.

In conclusion, the probability threshold is not a fixed number. Each regression model has a unique probability threshold. Specifically, the regression models developed in this thesis have different objectives to classify the three, four, or five categories in different scenarios. Therefore, each of the regression models has its unique thresholds for probabilities that maximize their sum of sensitivity and specificity.

3.3.2 One-Stage Regression Models

The one-stage logistic regression analyzes the data with binary dependent variables. However, the headlines we analyze contain at least three categories of sentiment, so we can't include all categories in one logistic regression model. Therefore, we design a multi-stage classification. However, we first construct one-stage models to classify between positive and non-positive or negative and non-negative. For example, suppose we implement the one-stage logistic regression to classify positive and non-positive sentiments. In this case, the logistic regression model returns the probability of a headline with positive impacts. If the probability is higher than the probability threshold, this headline assumes to have a positive impact on the stock market. Figure 3.3.2 illustrates how the logistic regression models use headlines as input and return the probability that the headlines are positively impactful.

Therefore, there are two one-stage models that analyze the positive or negative sentiments.

Positive one-stage model classifies positive and non-positive headlines. The upper threshold of impact could be the 90% or 95% quantile of zlogreturn, where the quantile numbers of zlogreturn can be calculated by Q(90%) or Q(95%) in (3.1.4).

Negative one-stage model classifies negative and non-negative headlines. The lower threshold of impact can be the 5% or 10% quantile of zlogreturn.

Since the one-stage logistic regression models can only classify one sentiment category at a time, we construct trading strategies with only one type of option based on one prediction. For example, given the prediction of the headline's positive impact, we can create trading strategies that consist of vanilla call options only. That is, with this approach, we can't

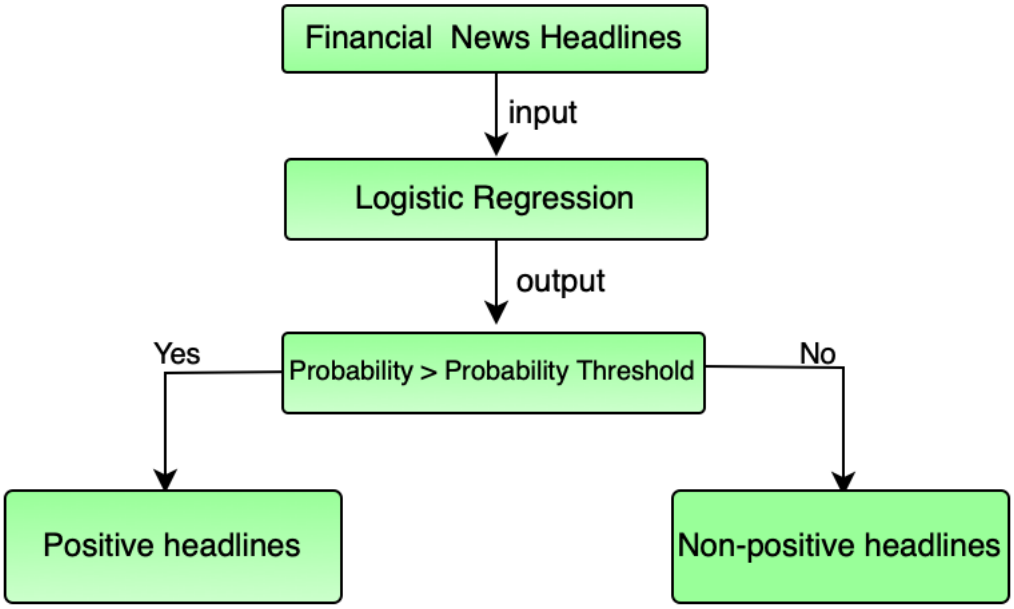

Figure 3.3.2: Example of a one-stage model that classifies positive sentiment

create more advanced trading strategies with a richer selection of options. Also, more detailed predictions are needed to make more aggressive trading strategies that increase returns and cash flow growth. Thus the two-stage and three-stage logistic regression models are introduced to give a more detailed classification of sentiment contained in the headlines.

3.3.3 Two-Stage Regression Models

We proposed the two-stage logistic regression models to analyze three sentiment categories for impact: positive, neutral, and negative.

Positive two-stage model classifies positive and non-positive at the first stage. At the second stage, we classify non-positive headlines as neutral and negative. The upper and lower thresholds for the stock return could be 5% and 95%, 5% and 90%, 10% and 90%, 5% and 95%, and percentages are quantiles of zlogreturn.

Neutral two-stage model classifies neutral and non-neutral at the first stage. At the second stage, we classify non-neutral headlines as positive and negative. The upper and

lower thresholds for the stock return could be 5% and 95%, 5% and 90%, 10% and 90%, 5% and 95%, and the percentages are quantiles of zlogreturn.

Negative two-stage model classifies negative and non-negative at the first stage. At the second stage, we classify non-negative headlines as neutral and positive. The upper and lower thresholds for the stock return could be 5% and 95%, 5% and 90%, 10% and 90%, 5% and 95%, and the percentages are quantiles of zlogreturn.

Figure 3.3.3 illustrates how the positive two-stage logistic regression model works.

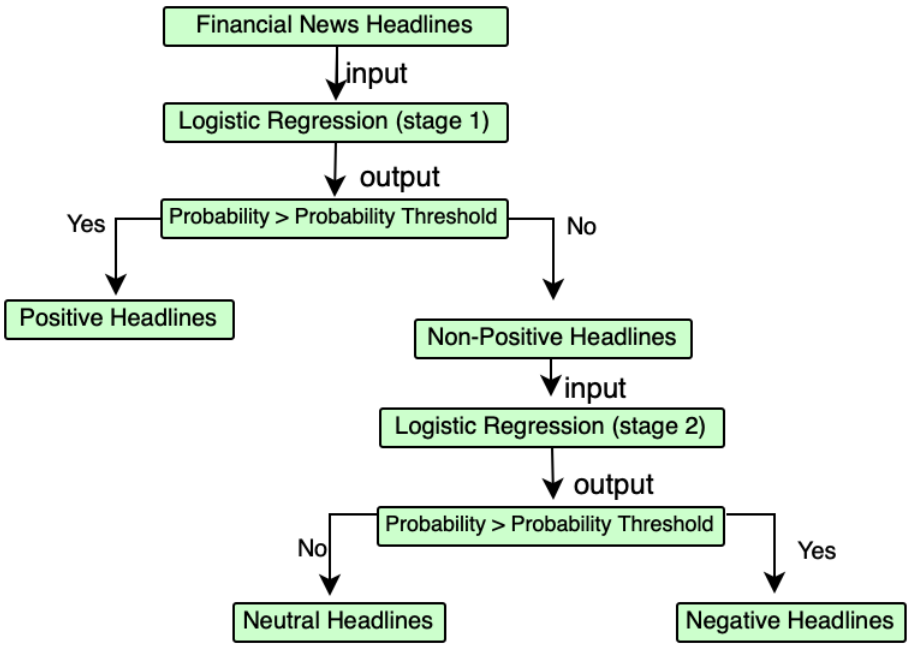

Figure 3.3.3: Example of a positive two-stage model that classifies positive, negative, and neutral sentiment.

The two-stage logistic regression model presents a more detailed classification than the one-stage model and provides more sentiment information. Therefore with two-stage logistic regression models, we can generate advanced trading strategies with more diversed combinations of options. For example, with positive two-stage models, we can create trading strategies consisting of call and put options, bringing us higher returns and cash flow growth than trading strategies consisting of only call or put options. However, more market in-

formation is required to create more aggressive trading strategies. For example, different intensities of sentiment cause different fluctuation levels in the financial market. Therefore, we can make an aggressive trading strategy by adding a larger number of options to the portfolio when we find strong positive or strong negative sentiments in the headlines. Thus, at the second stage of the logistic regression model, we can also classify positive and negative headlines as strong positive or non-strong positive and strong-negative or non-strong negative, respectively. In this case, we categorize our sentiments into four states, and we can extract more exhaustive information from headlines. Thus we can create strong positive and strong negative two-stage models.

Strong positive two-stage model classifies headlines into positive or non-positive at the first stage. At the second stage, we classify positive headlines into strong positive or non-strong positive and non-positive headlines into neutral or negative. Fixed thresholds of impact are 30%, 70%, and 95% for negative, positive, and strong positive, respectively.

Strong negative two-stage model classifies headlines into positive or non-positive at the first stage. At the second stage, we classify negative headlines into strong negative or non-strong negative and non-negative headlines into neutral or positive. Fixed thresholds of impact are 5%, 30%, and 70% for strong negative, negative, and positive, respectively.

However, with four sentiment states, we can only classify one strong sentiment at a time. Thus, we introduce three-stage models to include both strong positive and strong negative sentiments.

3.3.4 Three-Stage Regression Models

The objective of three-stage logistic regression models aims to classify more sentiment from the headlines. The strong positive and strong negative means the headlines strongly impact stock return positively or negatively. Thus three-stage logistic regression is proposed to tackle headlines with five sentiment categories. Meanwhile, to generate more aggressive trading strategies, we only implement one three-stage model that focuses on the headlines with strong sentiments.

Three-stage models classifies the headlines as strong or non-strong at the first stage. At the second stage, strong headlines are classified as strong positive and strong negative, and non-strong headlines are classified as neutral and non-neutral. At the third stage, non-neutral headlines are classified as positive and negative. Fixed thresholds of impact are 5%, 30%, 70%, and 95% for strong negative, negative, positive, and strong positive, respectively, where the percentages are arguments of the quantile function of zlogreturn.

In conclusion, we select the number of stages for logistic regression models based on our demand for the construction of trading strategies. We introduce a three-stage logistic regression model because we demand more information to generate a more aggressive trading strategy. Also, the three-stage logistic regression model is not necessarily more accurate and robust than the two-stage and one-stage models. Instead, the one-stage or two-stage models may give more accurate predictions than the three-stage models.

3.3.5 Performance of Models

We estimate the performance of models based on the prediction's accuracy. Since the objective of the sentiment analysis is to generate profitable trading strategies, the performance of the regression models depends on the accuracy of prediction. We propose performance scores to provide an overall assessment of the regression models. The score is calculated as a sum of the products of the corresponding entries of two square matrices: the assessment matrix $\mathbf{M}$ defined by us and the accuracy matrix $\mathbf{A}$ from the output table created based on the prediction of regression models. The formula for the performance score is given by (3.3.5).

$$\textbf{Performance Score} = \mathbf{M} \cdot \mathbf{A} = \sum_{i,j} m_{ij} a_{ij} \tag{3.3.5}$$

The matrix defined by us aims to estimate how the prediction impacts the generation of trading strategies. Meanwhile, the size of the matrices depends on the number of sentiment categories. For example, for the positive one-stage regression model, we receive a two-by-two output table, which is a classification between positive and non-positive headlines given by Table 3.3.1.

	Predict Positive	Predict Non-Positive
Actual Positive	$a\%$	$b\%$
Actual Non-Positive	$c\%$	$d\%$

Table 3.3.1: Prediction table for one-stage regression models

The percentages in Table 3.3.1 are calculated by dividing the elements in the aggregated confusion matrix (created from cross-validation) by their column sum. For example, when we are given an aggregate confusion matrix like Table 3.3.2. Then from confusion matrix

	Predict Positive	Predict Non-Positive
Actual Positive	True Positive	False Non-Positive
Actual Non-Positive	False Positive	Ture Non-Positive

Table 3.3.2: Aggregate confusion matrix for one-stage regression models

Table 3.3.2, we can calculate the a and c in prediction table as follows

$$a = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \cdot 100$$

and

$$c = \frac{\text{False Positive}}{\text{True Positive + False Positive}} \cdot 100$$

The True Positive + False Positive is the sum of the first column in the confusion matrix. All the prediction tables are calculated following this method. Then we set the matrices $\mathbf{M}$ and $\mathbf{A}$ as follows:

$$\mathbf{M} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Matrix $\mathbf{A}$ is the collected result from the prediction made by a positive one-stage regression model. In Table 3.3.1, the only element we are interested in is the number of correctly predicted positive headlines because it is the only factor that makes our trading strategy profitable. This element specifies the number of call options in the trading strategy based

on this one-stage model. Thus the performance score is given by

$$\textbf{Performance Score} = \mathbf{M} \cdot \mathbf{A} = 1 \cdot a + 0 \cdot b + 0 \cdot c + 0 \cdot d = a \tag{3.3.6}$$

Therefore, a in Table 3.3.1 is the final score we used to estimate the performance of all one-stage regression models.

However, for two-stage regression models, the matrix is more complicated since we have more sentiment categories to analyze and more information to generate trading strategies. For example, if we use a positive two-stage model, the output table is a three-by-three table. The matrix $\mathbf{M}$ and $\mathbf{A}$ for two-stage regression models that process three states of

	Predict Positive	Predict Neutral	Predict Negative
Actual Positive	$a\%$	$b\%$	$c\%$
Actual Neutral	$d\%$	$e\%$	$f\%$
Actual Negative	$g\%$	$h\%$	$i\%$

Table 3.3.3: Prediction table for positive two-stage model

sentiment are three-by-three matrices.

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

Thus the performance score for this example is

$$\textbf{Performance Score} = \mathbf{M} \cdot \mathbf{A} = \sum_{i,j=1}^{3} m_{ij} a_{ij} = a + e + i - c - g \tag{3.3.7}$$

For the two-stage regression models with three states of sentiment, the top left and bottom right entries of $\mathbf{A}$ are the most important elements in the output table because they indicate how accurate the prediction is. The top right and bottom left elements predict

headline sentiments to their opposite, which misleads us to make the opposite decisions when constructing trading strategies. For example, the top right element predicts negatively impactful headlines to be positively impactful. This prediction leads us to buy a call option when we are supposed to buy a put option, so we put -1 at those positions.

Recall that we introduced the two-stage regression models with four states of sentiment, which means the matrix is four-by-four with strong positive or strong negative two-stage models. The prediction table for the strong positive two-stage model is given by Table 3.3.4. The following $\mathbf{M}$ and $\mathbf{A}$ are designed to calculate the performance score of the strong

Actual\Predict	Strong Positive	Positive	Predict Neutral	Negative
Strong Positive	$a\%$	$b\%$	$c\%$	$d\%$
Positive	$e\%$	$f\%$	$g\%$	$h\%$
Neutral	$i\%$	$j\%$	$k\%$	$l\%$
Negative	$m\%$	$n\%$	$o\%$	$p\%$

Table 3.3.4: Prediction table for strong positive two-stage model

positive two-stage model.

$$\mathbf{M} = \begin{bmatrix} 1 & 1/2 & 0 & -1 \\ 1/2 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ -1 & -1 & 0 & 1 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

The matrix $\mathbf{M}$ is defined like this because the most important element is the diagonal values in the output table. They represent accurate predictions. Thus we put $1's$ in the diagonal positions. The $1/2$ in matrix $\mathbf{M}$ represents tolerable mispredictions because we can still make profits even if we predict the actual positive headlines to strong positive headlines or strong positive headlines to positive headlines. However, some predictions are intolerable because those predictions not only eliminate our chance of making profits and instead cause losses for the trading strategies. Some positive and actual strong positive headlines are predicted to be negative headlines. Those predictions mislead our trading strategies to make opposite decisions which causes extra losses. The actual negative headlines predicted to be positive or strong positive also mislead our predictions and may cause more losses due

to the aggressive trading strategies. Thus -1 is put in the positions that may cause losses for our trading strategies. Similarly, for two-stage strong negative models, the prediction table is given by Table 3.3.5. Thus with this prediction table, we define the $\mathbf{M}$ and $\mathbf{A}$ as

Actual\Predict	Positive	Predict Neutral	Negative	Strong Negative
Positive	$a\%$	$b\%$	$c\%$	$d\%$
Neutral	$e\%$	$f\%$	$g\%$	$h\%$
Negative	$i\%$	$j\%$	$k\%$	$l\%$
Strong Negative	$m\%$	$n\%$	$o\%$	$p\%$

Table 3.3.5: Prediction table for strong negative two-stage model

follows.

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 1/2 \\ -1 & 0 & 1/2 & 1 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

The parameters that are put in this matrix $\mathbf{M}$ for two-stage strong negative models have the same purpose as the two-stage strong positive models. Since our prediction tables are different for two-stage strong positive and strong negative models, we have two matrices with the same parameters but in different locations.

Three-stage regression classifies the headlines with five states of sentiment. The prediction table for the three-stage model is given by Table 3.3.6. Thus the matrices that estimate the

Actual\Predict	Strong Positive	Positive	Predict Neutral	Negative	Strong Negative
Strong Positive	$a\%$	$b\%$	$c\%$	$d\%$	$e\%$
Positive	$f\%$	$g\%$	$h\%$	$i\%$	$j\%$
Neutral	$k\%$	$l\%$	$m\%$	$n\%$	$o\%$
Negative	$p\%$	$q\%$	$r\%$	$s\%$	$t\%$
Strong Negative	$u\%$	$v\%$	$w\%$	$x\%$	$y\%$

Table 3.3.6: Prediction table for three-stage model

performance of the three-stage regression is a five-by-five matrix given by $\mathbf{M}$ and $\mathbf{A}$.

$$\mathbf{M} = \begin{bmatrix} 1 & 3/4 & -1/4 & -3/4 & -1 \\ 3/4 & 1 & -1/4 & -1/2 & -3/4 \\ -1/4 & -1/4 & 1 & -1/4 & -1/4 \\ -3/4 & -1/2 & -1/4 & 1 & 3/4 \\ -1 & -3/4 & -1/4 & 3/4 & 1 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} a & b & c & d & e \\ f & g & h & i & j \\ k & l & m & n & o \\ p & q & r & s & t \\ u & v & w & x & y \end{bmatrix}$$

We defined a more detailed matrix to estimate the comprehensive performance of the three-stage regression models. First, the diagonal value is still the most critical factor that impacts the profit of the trading strategies. The top right and bottom left corners are still the most negative factor that causes extra losses. Also, for strong positive or negative headlines but predicted to be positive or negative, we put 0.75 because we can still make profits from those incorrect predictions. We also put 0.75 there if positive or negative headlines are predicted to be strong positive or strong negative. Meanwhile, for headlines that are positive and negative but predicted to be strong negative and strong positive, we believe -0.75 is enough to process that information. In addition, for positive or negative headlines but predicted to be negative and positive, we put -0.5 because those predictions generate losses but do not create any extra losses. In addition, for the neutral headlines that are predicted to be strong positive, positive, negative, and strong negative, we place -0.25 with those predictions because that information could mislead us to buy options while those trading days are not profitable. Similarly, headlines that are strong positive, strong negative, positive and negative but predicted to be normal. Although that incorrect information generated no loss, the return and profit decreased by the misleading information generated from neutral predictions, so we put -0.25 at those positions in the matrix.

In conclusion, the model's performance is estimated based on the performance of the corresponding trading strategy. Generally, the higher the model's performance score, the more profits can be made by the trading strategy created with the model. Thus, the assessment matrix $\mathbf{M}$ aims to address the profitable and negative factors of the trading strategies. Also, more market information should be provided when the number of stages in the regression

model is increased. The objective of this matrix assessment method is to generate a comprehensive score for the performance of a regression model so that we can compare the performance of different models.

3.4 Feature Combinations in Two-Stage Models

In Chapter 2, we applied NLP techniques to extract features from financial news headlines. The extracted features are single words, words bigrams (word combinations), words with positions, and sentiment scores (positive, neutral, negative, compound). All of those features are used to build regression models. However, we are interested in exploring whether adding the features improves the models' performance. Therefore, we create five data sets to compare the feature combinations.

Basic data set consists of only single word features.

Basic+bigrams data set is built with single words and word bigrams.

Basic+position data set is built with single words and words with positions.

Basic+sentiment data set is built with single word and sentiment scores.

Full data set is equal to the final data introduced in Section 3.1 that consists of all the features.

We use the best two-stage models introduced in Section 3.9 to compare the performance of the models with different data sets. Table 3.9.2 states the best two-stage logistic, SVM, and LASSO models. We calculate the performance scores for the best two-stage models with different feature combinations. Figure 3.4.1 summarizes the performance scores of the best models with 5 data sets. The LASSO regression model is different from logistic and SVM models so that the addition of any feature to the basic data set decreases the performance score. However, we can still conclude that the LASSO model with the basic+sentiment data set outperforms other LASSO models except for the basic data set. The logistic model with a basic data set has the lowest performance score among all the models. The addition of bigrams increases the performance scores for both LASSO and logistic models. Also,

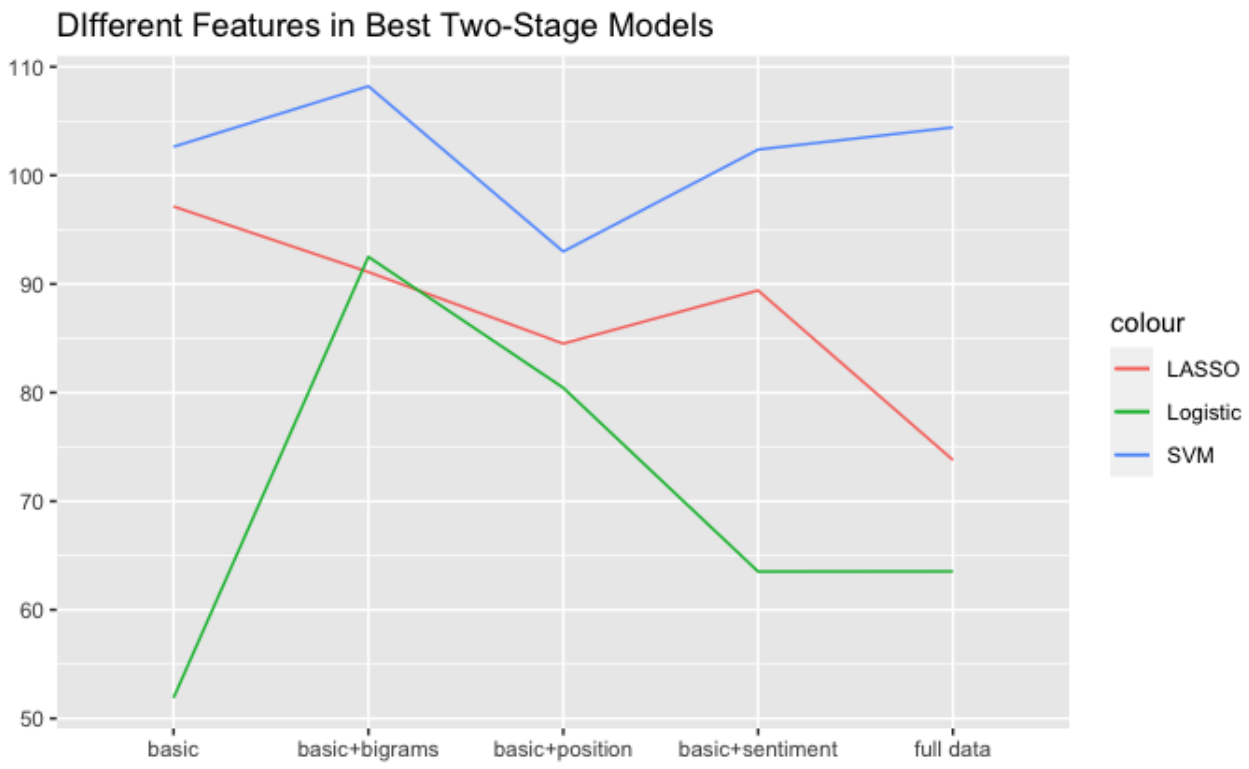

Figure 3.4.1: Performance scores of best two-stage models with different feature combinations

the addition of words with positions increases the performance scores of logistic models but decreases the performance score of SVM models. However, the logistic model with a basic+sentiment data set outperforms all the logistic models, but this data set doesn't work well with SVM. The full data set has a close performance score compared to the basic data set for SVM models.

Since the basic+bigrams and basic + sentiment data sets both have relatively high performance scores in Figure 3.4.1, especially for logistic models. Therefore, we create a data set that includes basic, bigrams, and sentiment features.

Basic+bigrams+sentiment data set is built with single words, bigrams, and sentiment scores.

In this case, we apply logistic, LASSO, and SVM regression models on this data set and use the highest and mean performance scores from Figure 3.4.1 to compare the performance of the regression models. The highest performance score is from the two-stage logistic

model with the basic+sentiment data set. Although basic+bigrams and basic+sentiment

Figure 3.4.2: The green line plot denotes the performance scores of best two-stage models with basic+bigrams+sentiment data set. The *highest score* and *mean score* are the highest and mean performance scores in Figure 3.4.1.

data set works well with high performance scores, the regression models applied to basic+bigrams+sentiment data are not working well as all of their performance scores are lower than the highest performance score in Figure 3.4.1. Moreover, the performance score of the LASSO model with basic+bigrams+sentiment data set is even lower than the mean performance score in Figure 3.4.1. However, the combination of basic+bigrams+sentiment data gives logistic model a relatively high performance score which means the addition of important features can improve the performance scores for logistic models.

In summary, the addition of words bigrams improves the performance of logistic and SVM models. Adding sentiment scores to the basic data lets the logistic model have a higher performance score than other logistic models. In addition, all the best models we used are trained with the full data set, but the full data set still does not work as well as others.

Thus a full data set that includes all features is not the best choice for two-stage models.

3.5 Principal Component Analysis

Principal component analysis (PCA) is an unsupervised technique that analyzes given data which consists of several correlated dependent variables [9]. Also, the PCA can work as a feature selection tool by controlling the number of principal components. The principal components are calculated by computing the eigenvectors of the covariance matrix for the original data. Then by controlling the number of principal components, we can control information extracted from the original data. For example, we can increase the number of principal components if more information is needed to build regression models and decrease the number of principal components if we want to remove redundant information. On the one hand, implementing the PCA can decrease the multicollinearity and help cluster the data. On the other hand, we can keep the information from the original data as we want by controlling the number of principal components.

In addition, scaling the original data set is imperative for implementing the PCA because the projection of an unscaled data set may bring skewness that misleads the classification. However, we did not scale our data set before doing PCA because of the following two reasons. First, the features extracted from headlines are all binary variables, meaning they are either 1 or 0. Second, the sentiment features are positive, negative, neutral, or compound that may have a larger scale than features extracted from headlines, but their maximum possible mean value is 3.33, which does not cause too much skewness during the projection. The compound score is the sum of normalized scores that varies from -1 to 1. Additionally, there is no need to scale the binary variables. We would like to keep the skewness from binary variables because they represent the features' occurrence in headlines. Overall, we believe that our data set does not need to be scaled before doing PCA, and we want to keep some skewness from the original data set.

3.5.1 Details in PCA

Since some machine learning algorithms (logistic, LASSO, SVM) will be proposed to analyze the data set after applying PCA, we need to project the training data set and test data set separately. The dimension of the original data set is 1502 x 1178. The number of columns (1178) corresponds to the number of features, and the number of rows (1502) means that there are 1502 trading days with posted headlines from 2012-01-01 to 2019-05-31. We set the training data to be the trading days in the first seven years and put them from rows 201 to rows 1502, so that the training data $\mathbf{X}$ has the dimension of 1302×1178, while the test data $\mathbf{Y}$ has the dimension of 200×1178. We first apply PCA to transform the training data $\mathbf{X}$ and apply the same transformation to the test data $\mathbf{Y}$.

The PCA aims to eliminate the multicollinearity during the projection of the data set. Thus we need to calculate the covariance matrix of our data set first and try to maximize the variance (diagonal value) as much as possible. Meanwhile, minimize the covariance of any two independent variables as much as possible. Therefore, the first step is computing the covariance matrix of training data $\mathbf{X}$, where $\mathbf{X}$ is listed below with a dimension of 1302 x 1178.

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,1178} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,1178} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{1302,1} & x_{1302,2} & x_{1302,3} & \dots & x_{1302,1178} \end{bmatrix} = \begin{bmatrix} X_1 & X_2 & X_3 & \dots & X_{1178} \end{bmatrix}$$

where X_i is a vector with length 1302 which represents the ith feature in the training data $\mathbf{X}$. The next step is to find the covariance matrix of the data $\mathbf{X}$, which means we need to compute the covariance of any pair of variables in $\mathbf{X}$. Note the covariance of the same variables is just equal to their variance.

$$\mathbf{C} = \begin{bmatrix} Var(X_1) & Cov(X_1, X_2) & Cov(X_1, X_3) & \dots & Cov(X_1, X_{1178}) \\ Cov(X_2, X_1) & Var(X_2) & Cov(X_2, X_3) & \dots & Cov(X_2, X_{1178}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Cov(X_{1177}, X_1) & Cov(X_{1177}, X_2) & Cov(X_{1177}, X_3) & \dots & Cov(X_{1177}, X_{1178}) \\ Cov(X_{1178}, X_1) & Cov(X_{1178}, X_2) & Cov(X_{1178}, X_3) & \dots & Var(X_{1178}) \end{bmatrix}$$

where $Var(X_i) = \frac{\sum_{j=1}^{j=1302} x_{j,i} - \bar{x_i}}{n-1}$, and $Cov(X_i, X_j) = \frac{\sum_{k=1}^{k=1302} (x_{k,i} - \bar{x_i})(x_{k,j} - \bar{x_j})}{n-1}$ for $i, j \in \{1, 2, \dots, 1178\}$, where the $\bar{x_i}$ and $\bar{y_j}$ are the mean values of X_i and X_j.

Now we calculate the eigenvalues and eigenvectors of the matrix $\mathbf{C}$ defined by

$$\mathbf{C}x = \lambda x \tag{3.5.1}$$

where λ is an eigenvalue, and x is the corresponding eigenvector. Thus we have 1178 eigenvectors. Let $\mathbf{V}$ represent the matrix of eigenvectors:

$$\mathbf{V} = \begin{bmatrix} V_1 & V_2 & V_3 & \dots & V_{1178} \end{bmatrix} = \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,3} & \dots & v_{1,1178} \\ v_{2,1} & v_{2,2} & v_{2,3} & \dots & v_{2,1178} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{1178,1} & v_{1178,2} & v_{1178,3} & \dots & v_{1178,1178} \end{bmatrix}$$

where the V_i represents the ith eigenvector. Now we can calculate the principal components with the eigenvectors.

$$PC_i = X \cdot V_i = \begin{bmatrix} X_1 & X_2 & X_3 & \dots & X_{1178} \end{bmatrix} \cdot \begin{bmatrix} v_{1,i} \\ v_{2,i} \\ v_{3,i} \\ \dots \\ v_{1178,i} \end{bmatrix}$$

which can be rewritten as follows:

$$PC_i = v_{1,i} \cdot X_1 + v_{2,i} \cdot X_2 + v_{3,i} \cdot X_3 + \cdots + v_{1178,i} \cdot X_{1178} \tag{3.5.2}$$

for $i \in \{1, 2, \ldots, 1178\}$.

In conclusion, we have calculated all the principal components, and those principal components bring us a new data set with the same dimension. Meanwhile, the new data set has less multicollinearity, and we can remove redundant information by reducing the number of selected principal components. Next, we will introduce how to choose the principal components.

3.5.2 Selection of Principal Components

The selection of the appropriate number of principal components is crucial in PCA. Recall that we want to decrease the covariance between variables and increase the variance of each variable when we project the data using PCA. Therefore, the variance of the new data set is an essential factor for selecting the principal components. We propose two methods of choosing the appropriate number of principal components for further analysis. First, we are selecting the principal components that could keep as much variance as possible, which means we are trying to keep more information from the original data set. Second, we select the first few principal components. Since most information in the original data set may be redundant, the first few principal components may be enough to build regression models.

Let us apply the first method that keeps as much information from the original data set as possible. Therefore, we select the principal components that can keep 80% to 95% of the total variance. We use the STATS package in R to project the data set and calculate the new principal components, as well as the cumulative variance of the principal components [15]. Table 3.5.1 shows the number of principal components we select.

According to Table 3.5.1, we select the first 183, 232, 302, and 418 principal components, which keep 80%, 85%, 90%, and 95% of the total variance of the data set, respectively. The

PCA	Selection 1	Selection 2	Selection 3	Selection 4
Principal Numbers	183	232	302	418
Proportion of Variance	0.1189%	0.087%	0.0578%	0.0308%
Cumulative Proportion	80.0741%	85.061%	90.0559%	95.0177%

Table 3.5.1: Selection of principal components by cumulative variance from 80% to 95%.

principal components are sorted by the variance explained by each principal component. The larger variance explained by the principal components, the smaller the number of the principal component. In addition, we are tuning the number of principal components to get the best number of principal components in building regression models.

Table 3.5.1 shows how we select the principal components by keeping as much variance as possible. Now we consider the other method that removes redundant information and only keeps the first few principal components to build the model. Since the first few principal components contain the highest variance among all the principal components during the calculation, we believe they contain the most important information, and the remaining principal components are redundant. Table 3.5.2 shows how we select the principal components.

PCA	Selection 1	Selection 2	Selection 3	Selection 4
Principal Numbers	2	3	4	5
Proportion of Variance	3.1349%	2.5677%	1.8956%	1.6484%
Cumulative Proportion	15.21204%	17.7798%	19.6753%	21.3238%

Table 3.5.2: Selection of the first five principal components

According to Table 3.5.2, we can investigate that the first five principal components only occupy 21.3238% of the total variance. Still, the first principal component has a much higher proportion of variance compared to the principal components from 183 to 418, which means that the first five principal components are more significant information than others since they dominate the direction of the data set after the projection.

In conclusion, we consider two methods to select the principal components depending on the variance of the new data set. The first one makes decisions based on the total cumulative variance of the principal components that aim to keep as much information as possible.

The second one aims to remove redundant information and only keeps the most valuable information, which means only a few variables with the highest variance are kept in the data set. During the construction of a regression model, the most valuable method will be selected and implemented in the validation data set. In this thesis, we focus more on the first method because we still want to keep as much information from the original data set as possible to have enough information to predict the headlines' impact on stock return.

3.6 Logistic Regression Model

The GLM (Generalized Linear Model) function [15] in R is proposed to generate logistic regression models. The generalized linear model applies a linear combination of independent variables to predict the dependent variable. Therefore, we create GLM models to perform logistic regression. The formula put in the generalized linear model is just all the principal components with cumulative variance from 80% to 95%.

3.6.1 One-Stage Logistic models

First, for the one-stage logistic regression, logistic models are proposed to classify the sentiment with two states. In this case, we are tuning the combinations of the threshold for impact and the number of principal components. Figure 3.6.1 illustrates the performance of different logistic regression models with different parameters.

Figure 3.6.1 illustrates that the logistic models that classify negative sentiment give the highest performance score. Especially the one-stage negative logistic model with 10% and 90% as the threshold of impact. Meanwhile, models work better with 10% and 90% as the threshold. This situation is because the models have few actual positive and negative headlines. Thus, increasing thresholds provide double positive and negative data for the logistic models to learn. The best model among all the one-stage logistic models is the negative model with the first 183 principal components, 10% and 90% quantile numbers as the threshold of impact. This negative logistic model has the highest performance score, which is 20.233463.

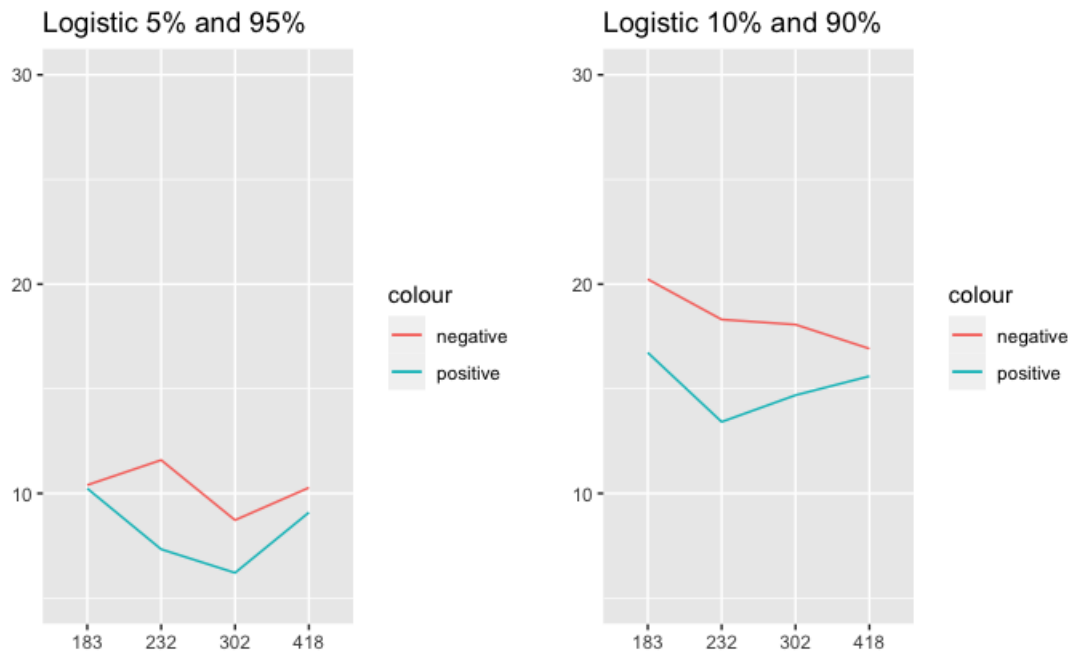

Figure 3.6.1: One-stage Logistic models with different thresholds of impact and numbers of principal components; the x-axis is the number of principal components that correspond to 80%, 85%, 90%, and 95% of the cumulative variance; the y-axis is the performance score for models. Positive and negative represent positive and negative one-stage logistic models.

3.6.2 Two-Stage Logistic models

Three sentiment categories are analyzed by two-stage logistic regression, and we analyze different sentiments in the first stage. Thus there are three different combinations of two-stage logistic regression (positive, negative, neutral). Also, in two-stage logistic regression, we are tuning the threshold of impact and the number of principal components.

Figure 3.6.2 illustrates the performance of logistic models in two-stage logistic regression. The overall trend for the number of principal components is decreasing, which means the lower the number of principal components, the better performance the model presents. Meanwhile, the best logistic model selected from two-stage logistic regression is the negative two-stage model, using 10% and 95% as the threshold of impact, and using the first 302 principal components as the features. This model gives the highest performance score among all the logistic models in two-stage logistic regression.

Also, with two-stage logistic models, we classified the sentiment into four states to include

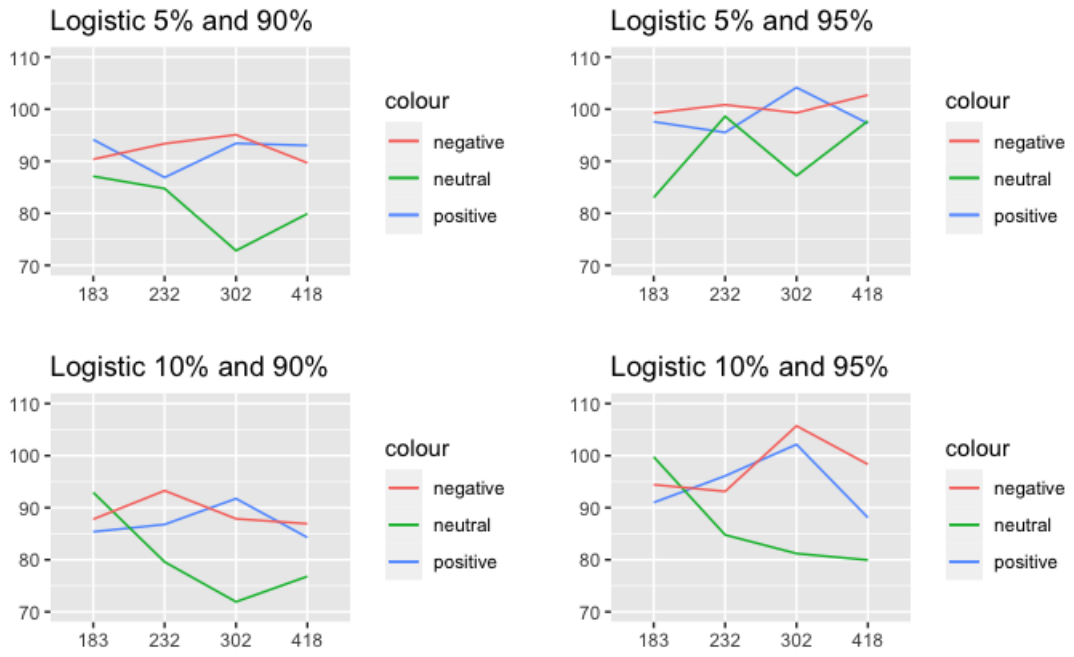

Figure 3.6.2: Two-stage Logistic models with different thresholds of impact and numbers of principal components; the x-axis is the number of principal components that correspond to 80%, 85%, 90%, and 95% of the cumulative variance; the y-axis is the performance score for models. Positive, negative, and neutral represent positive, negative, and neutral two-stage logistic models, respectively.

one strong sentiment in the two-stage logistic model. From Figure 3.6.3, the two-stage model that includes strong negative sentiment works better. Also, in the two-stage strong negative logistic model, the overall trend is decreasing, and the performance score decreases with the increase of the principal components. In the two-stage strong positive logistic model, the highest principal components have the lowest performance score, which means the principal components with 95% accumulate variance still contain redundant information. The two-stage strong negative model with the first 232 principal components is the best model.

3.6.3 Three-Stage Logistic Models

The three-stage logistic models are introduced to classify the strong negative, negative, neutral, positive, and strong positive sentiment with a fixed threshold of impact. Figure 3.6.4 is the plot that includes the performance score of three-stage logistic models with

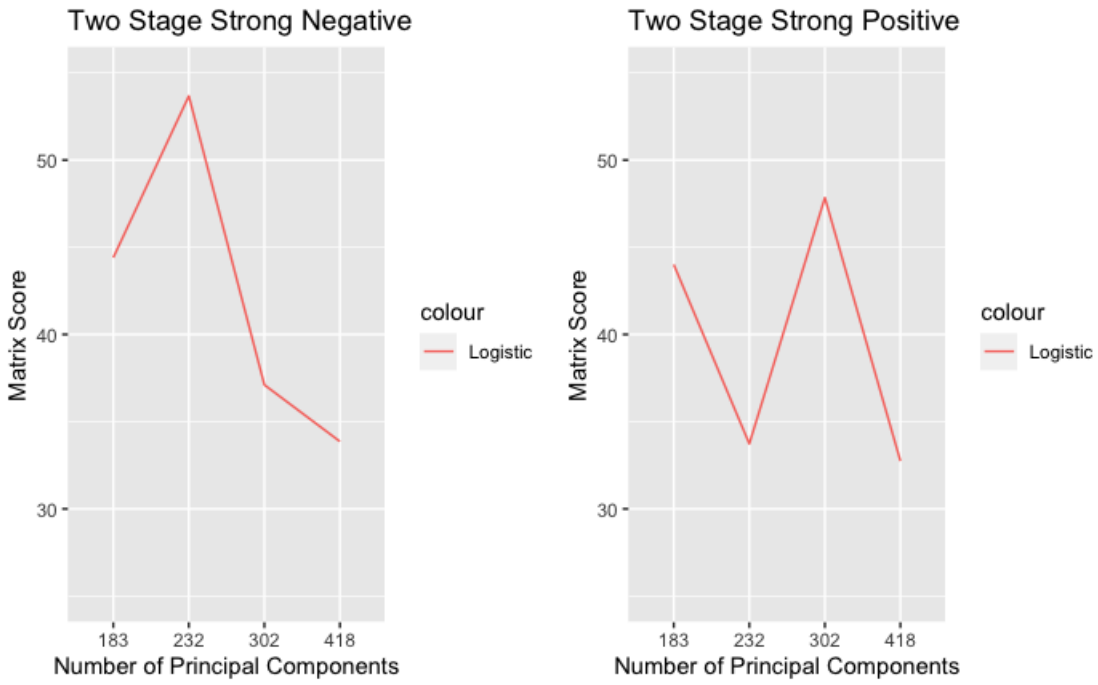

Figure 3.6.3: Two-stage logistic models with strong sentiment; the x-axis is the number of principal components that correspond to 80%, 85%, 90%, and 95% of the cumulative variance; the y-axis is the performance score for models

different principal components. First, the best three-stage logistic model is the model with the first 302 principal components. Also, we can investigate that the performance score of the models increased with the increase of principal components from 183 to 302. However, the model with the first 418 principal components returns the lowest performance score, thus again proving that principal components with 95% of the cumulative variance are unnecessary in the logistic models. The logistic model with the first 183 principal components has a low performance score which means the principal components with 80% of the variance does not include enough information to build a good three-stage logistic model.

3.6.4 Best Logistic Models

In this section, we summarize all the information from Section 3.6.1 to Section 3.6.3 to find the best logistic model in each approach. The neutral two-stage logistic model is the best two-stage model. Meanwhile, 302 is a frequently selected principal component number for

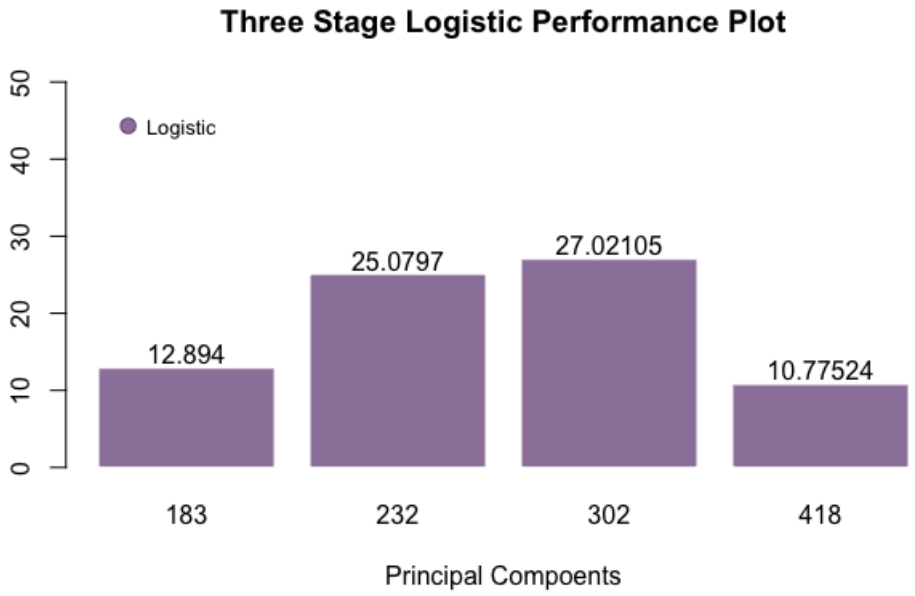

Figure 3.6.4: Three-stage logistic models

Best Logistic Models	Sentiment	Thresholds	PC's	Performance Score
One-stage	Negative	10%, 90%	183	20.233463
Two-stage	Neutral	10%, 95%	302	105.71553
Strong Two-stage	Negative	Fixed	232	53.66705
Three-stage	Strong	Fixed	302	27.02105

Table 3.6.1: Performance of logistic models. *PC's* is the number of principal components.

logistic models. In addition, the best models selected from logistic models usually have lower performance scores than those selected from LASSO models that are introduced in Section 3.7. This is because the LASSO models penalized the logistic models for decreasing multicollinearity, thus generating more efficient logistic regression models.

3.7 LASSO

In this thesis, we applied LASSO to penalize the logistic regression models. The LASSO is a penalized regression method that applies regularization to penalize the model for reducing the overfitting of the model by shrinking covariate effect coefficients. Here we present how

the LASSO penalizes the logistic regression models [10]:

$$\underset{\beta}{minimize}(-[\tfrac{1}{n}\sum_{i=1}^{n}(y_i \cdot (\beta_0 + x_i^\top \beta) - \log(1 + e^{\beta_0 + x_i^\top \beta})] + \lambda\|\beta\|_1) \quad (3.7.1)$$

where the y_i is the response value zvalue that represents if the headline is impactful or not, and x_i is the vector of predictors. The β_0 is the intercept and the other β is a vector of regression coefficients in the model; n is the number of the observations in the model, and $\|\beta\|_1$ is the sum of absolute vector values of β. The λ is the tuning parameter that controls the strength of the penalty. We proposed LASSO to process the headlines because we need a statistical tool to select a subset of features with less collinearity. Some penalized regressions can do this, such as RIDGE, Elastic net, and LASSO. However, we decide to use LASSO because LASSO can shrink the regression coefficients to zero, while other methods like RIDGE can only make the coefficients to be close to zero. Since the number of features in our data set is larger than the observations, we choose LASSO to select important features during the training process.

The most imperative factor in building LASSO regression models is penalized parameter that controls the scale of the penalization during the learning process. We are tuning the penalized parameter when training the LASSO regression models. In addition, we are also tuning the number of principal components and the threshold of impact for the LASSO regression models. The best combination of principal components and threshold of impact is selected after training. Then we can generate LASSO regression with the best combination of parameters.

3.7.1 Feature Selection

After principal component analysis, the data set still has hundreds of features. Thus we need a statistical tool to keep shrinking the data set. Penalized regressions like LASSO and RIDGE would be a great choice to compress the data. LASSO and RIDGE differ because LASSO directly deletes the redundant data. At the same time, the RIDGE keeps the redundant data but shrinks the weight of those data to be close to zero so that the

redundant information does not impact the learning of the regression models. The LASSO can shrink the coefficients of the model to exactly 0 because of its special L1 regularization while the RIDGE regression has another L2 regularization. Figure 3.5.1 illustrates how the L1 and L2 regularization works. The left part of the plot is L1 regularization in the LASSO

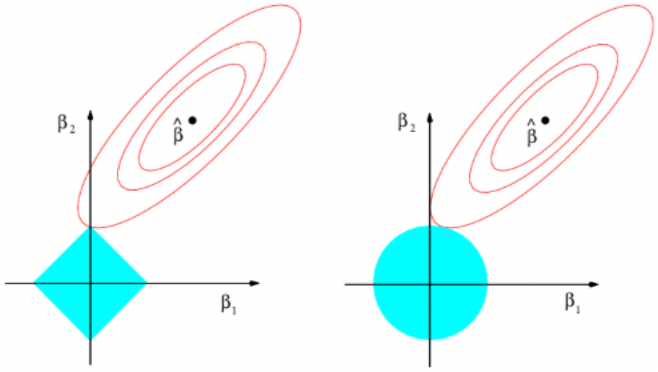

Figure 3.7.1: Example of L1 regularization and L2 regularization
[14, Fig 6.7]

regression, and the right part is L2 regularization in the RIDGE regression. The blue area illustrates how the coefficients constrained by the cost function and the L1 regularization in LASSO shrink the weight of independent variables to exactly zero. The cost function in LASSO hits the constraint area at the y-axis and shrinks the β_1 coefficient to exactly zero. The β_1 is shrunk to close to zero in the RIDGE regression. Since we have high-dimensional data with hundreds of principal components, the LASSO regression is a better choice than the RIDGE regression. Because the LASSO also works as a feature selection tool.

3.7.2 Tuning Parameters in LASSO

Recall that we introduced the different combinations of thresholds and the selection of principal components. Now, we are tuning those parameters with LASSO regression models to find the best combination of threshold and the best number of principal components. In addition, an appropriate penalized coefficient is imperative to build the LASSO regression models. The glmnet package [4] in R is proposed to build the LASSO regression models and tune the penalized coefficients λ. The penalized coefficient λ is tuned within the 10-folds cross-validation, and the final selected λ is the penalized coefficient that minimizes

the cross-validated errors, where the cross-validated errors are calculated by mean square error.

3.7.3 One-Stage LASSO Models

Recall that we introduced the one-stage, two-stage, and three-stage logistic regression. Thus at each stage, LASSO regression models are built to classify the sentiment. Meanwhile, the LASSO models generated with one-stage, two-stage, and three-stage are individual using different training data and testing data. Thus we are tuning the best parameters separately from the different scenarios. First, we introduce the LASSO regression models generated in the one-stage logistic regression.

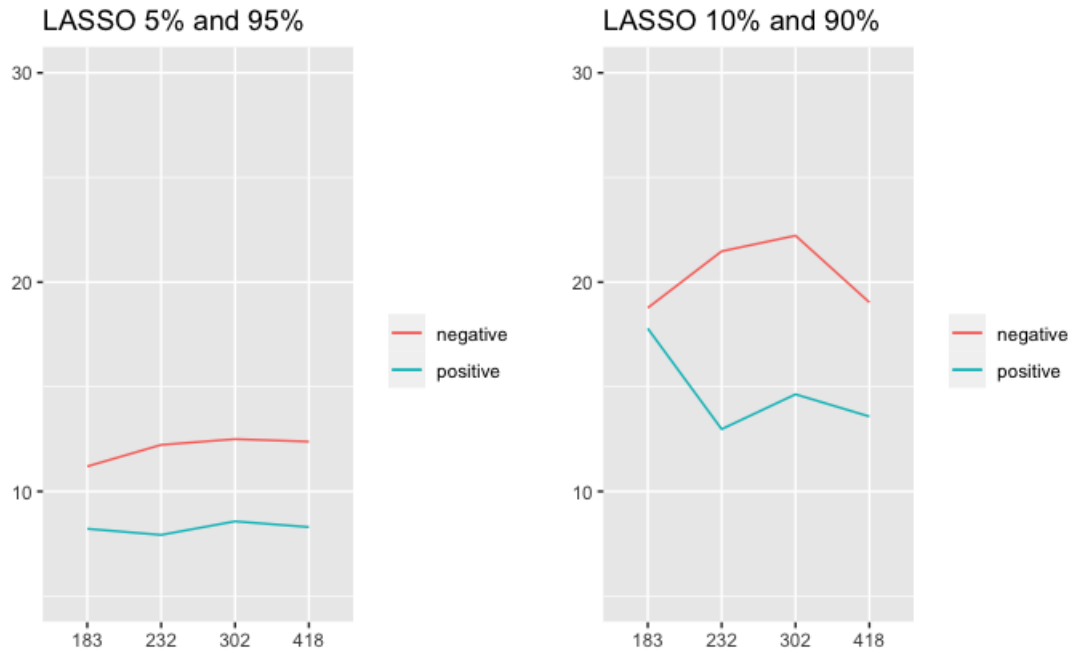

Figure 3.7.2: One-stage LASSO models with different thresholds of impact and numbers of principal components; the x-axis is the number of principal components that correspond to 80%, 85%, 90%, and 95% of the cumulative variance; the y-axis is the performance score for models. Positive and negative represent positive and negative one-stage LASSO models.

In Figure 3.7.2, the x-axis is the number of principal components, and the y-axis is the value of the performance score. The positive and negative legends represent the sentiment classified at the first stage. The percentage at the top represents the threshold of impact. Figure 3.7.2 illustrates that the model that classifies negative and non-negative hold a

higher performance score than the others, which means the LASSO models perform better in predicting negative headline sentiments. Also, the negative models using 10% and 90% quantile numbers as the threshold of impact give the highest performance score because more positive or negative headlines are used to train the regression models with this combination of thresholds. In summary, the best model selected is the negative model that used the first 302 principal components as predictors.

3.7.4 Two-Stage LASSO Models

The two-stage logistic regression analyzes the headlines with three states of sentiment. Thus we have three different two-stage models (positive, negative, neutral). Also, more combinations of thresholds are tried in the two-stage models. Figure 3.5.3 compares the performance between different LASSO regression models from two-stage logistic regression. From Figure 3.7.3, we can investigate that the principal components with 80% of the cumulative

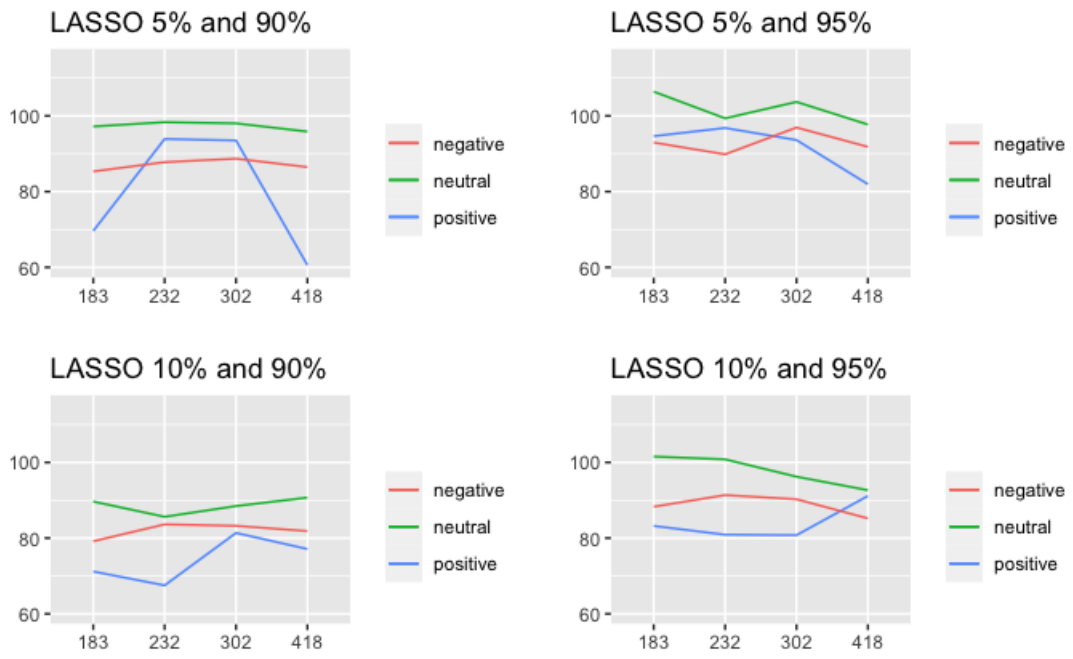

Figure 3.7.3: Two-stage LASSO models with different thresholds of impact and numbers of principal components; the x-axis is the number of principal components that correspond to 80%, 85%, 90%, and 95% of the cumulative variance; the y-axis is the performance score for models. Positive, negative, and neutral represent positive, negative, and neutral two-stage LASSO models, respectively.

variance have better performance than other principal components. In addition, the model used the combination of 5% and 95% quantile number as the threshold of impact has better performance than other combinations of the threshold of impact. The final model selected from training data is the neutral two-stage model, with 5% and 95% as the threshold, and uses the first 183 principal components as the features. This model gives the highest performance score among all the LASSO models in two-stage logistic regression.

The two-stage model also analyzed the strong sentiment. Figure 3.7.4 illustrates the performance scores for two-stage LASSO models that analyze the strong sentiments. In the

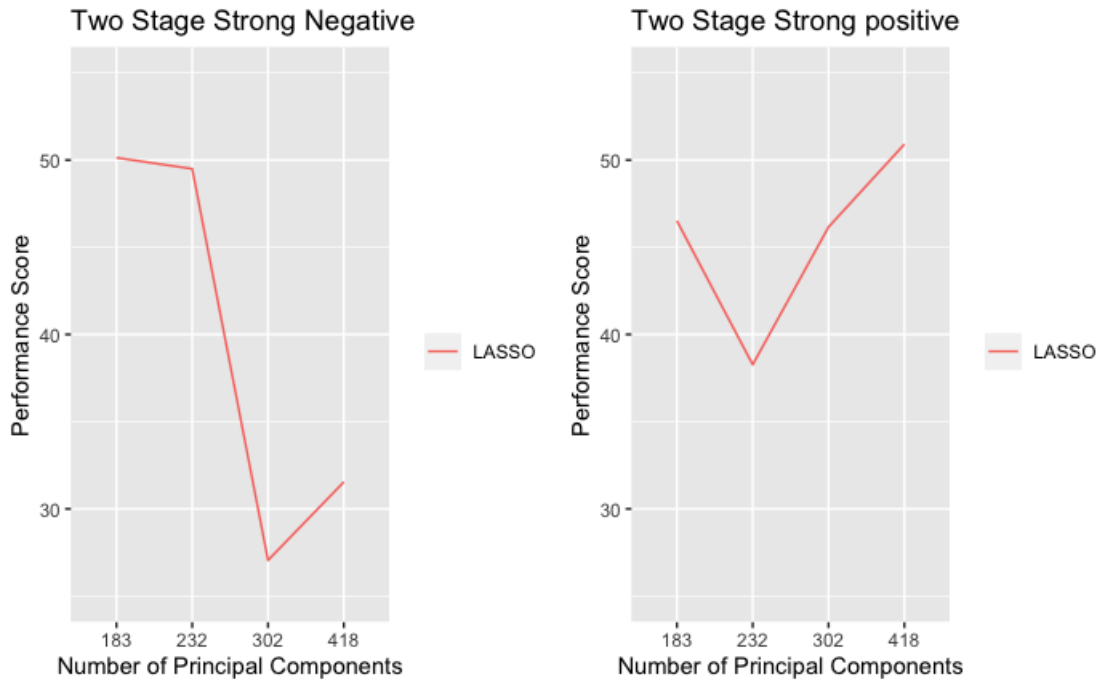

Figure 3.7.4: Strong positive and strong negative two-stage LASSO models; the x-axis is the number of principal components that correspond to 80%, 85%, 90%, and 95% of the cumulative variance; the y-axis is the performance score for models

two-stage strong models, we defined the 5%, 30%, 70%, and 95% as the threshold for strong negative, negative, positive, and strong positive sentiment states. The performance scores decrease as the increase of the number of principal components in the two-stage strong low models. However, the two-stage strong positive model with the first 418 principal components gives the highest performance score among all the two-stage strong models.

3.7.5 Three-stage LASSO Models

In the three-stage regression models, the impact threshold is no longer a tuning parameter because we fixed the impact of thresholds to be 5%, 30%, 70%, and 95%, which classifies the strong negative, negative, positive, and strong positive. Figure 3.7.5 decreases from

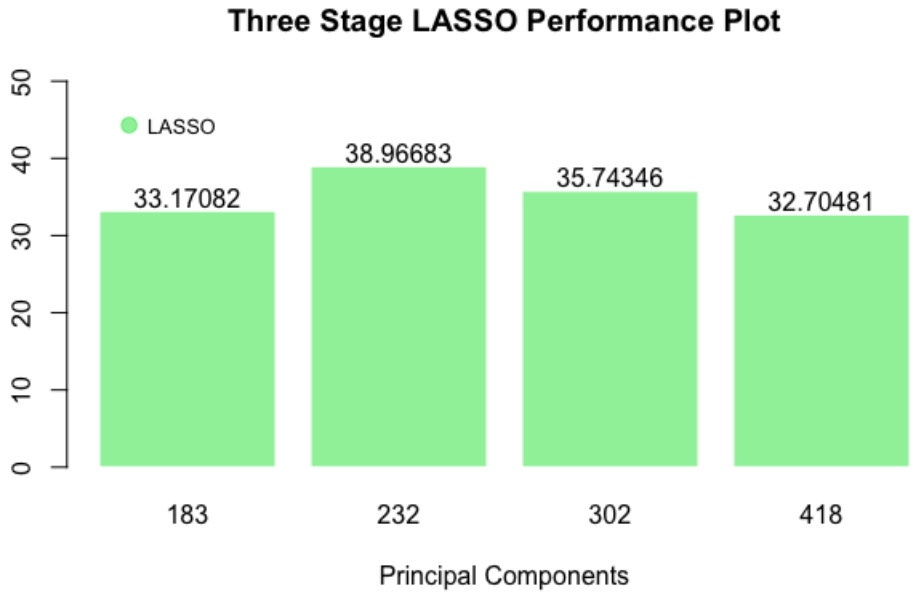

Figure 3.7.5: Three-stage LASSO models that analyze the strong positive and strong negative sentiment

principal components 232 to 418, and the best three-stage LASSO model selected is the model that used the first 232 principal components. We can investigate that the highest principal components have the lowest performance score, which means LASSO models still prefer fewer principal components. The first 232 and 302 also give the highest and second-highest performance scores. Thus the first 183 principal components may not be able to provide enough information to build LASSO models.

3.7.6 Best LASSO Models

We create one-stage, two-stage, and three-stage models with LASSO regression models. However, we received too much information with all possible tuning parameters. Thus we need to extract important information from each scenario and summarize the best models.

First, the sentiment in Table 3.7.1 means the sentiment classified at the first stage, while

Best LASSO Models	Sentiment	Thresholds	PC's	Performance Score
One-stage	Negative	10%, 90%	302	22.22222
Two-stage	Neutral	5%, 95%	183	106.40026
Strong Two-stage	Positive	Fixed	418	50.90337
Three-stage	Strong	Fixed	232	38.96683

Table 3.7.1: Performance of LASSO models. *PC's* is the number of principal components.

strong for the three-stage represents that it classifies strong and non-strong sentiment at the first stage. The threshold's percent numbers are the lower and upper bound set by the quantile number of stock returns. Fixed is put in the threshold for two-stage strong and three-stage because we used fixed thresholds for them. The table shows that the neutral two-stage LASSO model gives the best two-stage models. Also, 5% and 95% are the best thresholds for two-stage LASSO models. For strong two-stage and three-stage LASSO models, the best models use the first 418 and 232 principal components in the model, respectively. Thus the first 183 principal components did not conclude all the information required to build the best LASSO models that analyze strong sentiments.

3.7.7 Selected Variables in best LASSO models

Table 3.7.1 summarizes the best LASSO models. Recall that LASSO can also be used as a variables selection tool. Since all the variables in LASSO models are principal components, we also want to analyze the selected principal components. The one-stage, two-stage, strong two-stage, and three-stage approaches use one, two, three, and four LASSO models to classify the headlines' impact on stock return, respectively. Therefore, there are ten LASSO models in total, and we collect all the selected principal components together to get an aggregate result of selected variables. There are ten LASSO models in total so that the

Selected PC's						
5	PC 1	PC 4	PC 5	PC 88		
4	PC 15	PC 19	PC 34	PC 57	PC 138	PC 141

Table 3.7.2: The values in the first column denote how many times the principal components are selected; *PC* represents the principal components.

principal components can be selected ten times at maximum. However, no principal components are selected by all the LASSO models. We can only collect principal components that are chosen five times or less. Table 3.7.2 shows the principal components selected five times and four times because we only focus on the most frequently chosen variables. Principal component 1 is selected five times which is reasonable because it has the highest proportion of variance and is regarded as the most crucial principal component. Also, three of the most frequently selected variables are from the first five principal components, indicating the importance of the first five. Additionally, we can see that most of the selected principal components are from 1 to 100, which means a principal component that has a higher proportion of variance is easier to be selected by LASSO models. In summary, principal components 1 to 100 are frequently chosen by LASSO. A principal component with a higher variance proportion has a higher probability of being selected by LASSO models.

3.8 Support Vector Machine

Support vector machine (SVM) is a relatively new and outstanding statistical tool that learns the separating functions to do classifications [6]. The support vector machine aims to classify the different classes with a hyperplane. The hyperplane is the plane that maximizes the distance of points from different classes to the hyperplane. Since sometimes the data is not linear, the support vector machine can map the data to a higher dimension so that the hyperplane can better classify different classes.

3.8.1 Kernels in SVM

In the R studio, the e1071 package provides a kernel option to map the data. The kernel function maps the data from one dimension to another, and this mapping is usually from a lower dimension to a higher dimension. Since we focus on logistic regression, which is actually considered linear regression. Thus we select the linear kernel to map the data. In addition, we proposed the e1071 package in R [15] to create the SVM models, where the linear kernel means that we link the dependent variable with the linear combination

of independent variables. In addition, the cost parameter in the SVM models is tuned during the learning process, where the cost is the cost of the constraint of violation. Also, the cost can be regarded as the soft margin of misclassification, which means how much misclassification we could stand during the learning of the model. The default value of cost is 1 in the SVM model, and we tuned the costs using the list Cost = [0.001, 0.01, 0.1, 1, 10, 100], and usually, the cost = 0.01 gave the best performance of the model.

3.8.2 One-Stage SVM Models

First, the SVM is applied to construct one-stage logistic regression models. The tuning parameters in the one-stage logistic regression are combinations of threshold and number of principal components

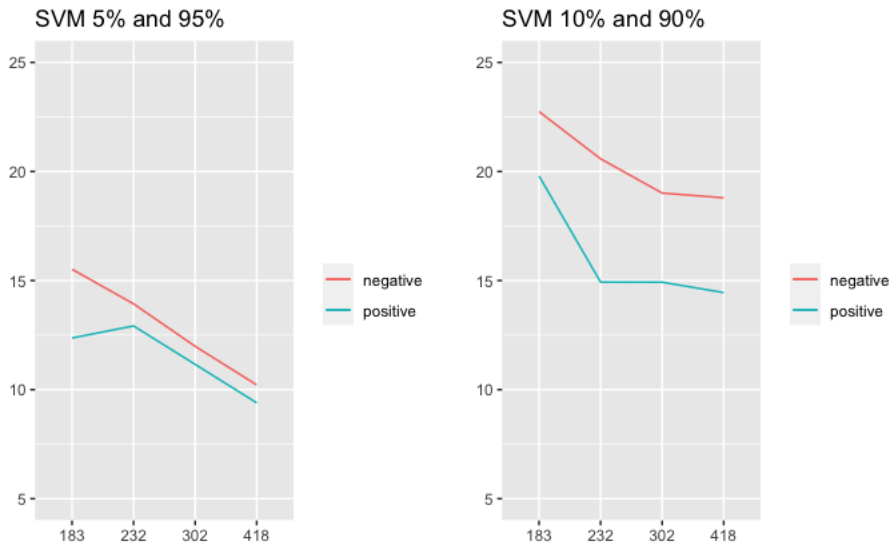

Figure 3.8.1: One-stage SVM models with different thresholds of impact and numbers of principal components; the x-axis is the number of principal components that correspond to 80%, 85%, 90%, and 95% of the cumulative variance; the y-axis is the performance score for models. Positive and negative represent positive and negative one-stage SVM models.

Figure 3.8.1 illustrates the performance scores of one-stage SVM models. The negative one-stage SVM models have higher performance scores than positive one-stage SVM models. In addition, the models that use 10% and 90% quantile numbers as the threshold of impact have higher performance scores, the same as the LASSO and logistic models. The best one-stage SVM model is the negative model that uses 10% and 90% quantile numbers as

thresholds and the first 183 principal components as features.

3.8.3 Two-Stage SVM Models

Also, we construct two-stage SVM regression models. Again the combinations of the threshold of impact and the number of principal components are tuned in the two-stage SVM regression models. Figure 3.8.2 illustrates the performance of the SVM models in the two-stage logistic regression.

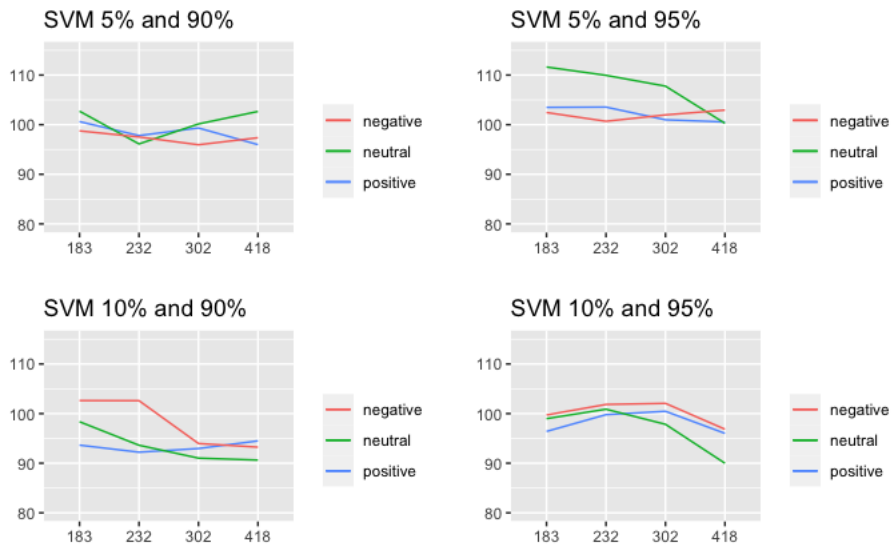

Figure 3.8.2: Two-stage SVM models with different thresholds of impact and numbers of principal components; the x-axis is the number of principal components that correspond to 80%, 85%, 90%, and 95% of the cumulative variance; the y-axis is the performance score for models. Positive, negative, and neutral represent positive, negative, and neutral two-stage SVM models, respectively.

First, again we can investigate a clear trend that the performance of the SVM models is decreasing with the increase of the principal components, which means there is still some redundant information in the data set. Also, the model with 5% and 95% as the impact threshold has higher performance scores, which means SVM models are better for classifying the data set with fewer positive or negative headlines. The best SVM model selected classifies the neutral two-stage model, using 5% and 95% as the threshold and applying the first 183 principal components as the features. The best SVM models in the two-stage logistic regression have a performance score of 111.6266, which is higher than the LASSO

and logistic in the two-stage logistic regression.

We also used the two-stage SVM models to classify the headlines into four sentiments, that is negative, positive, neutral, and one strong sentiment, which could be strong positive or strong negative. Figure 3.8.3 is the performance plot for two-stage SVM models with strong sentiments. The two plots that analyze strong negative and strong positive perform

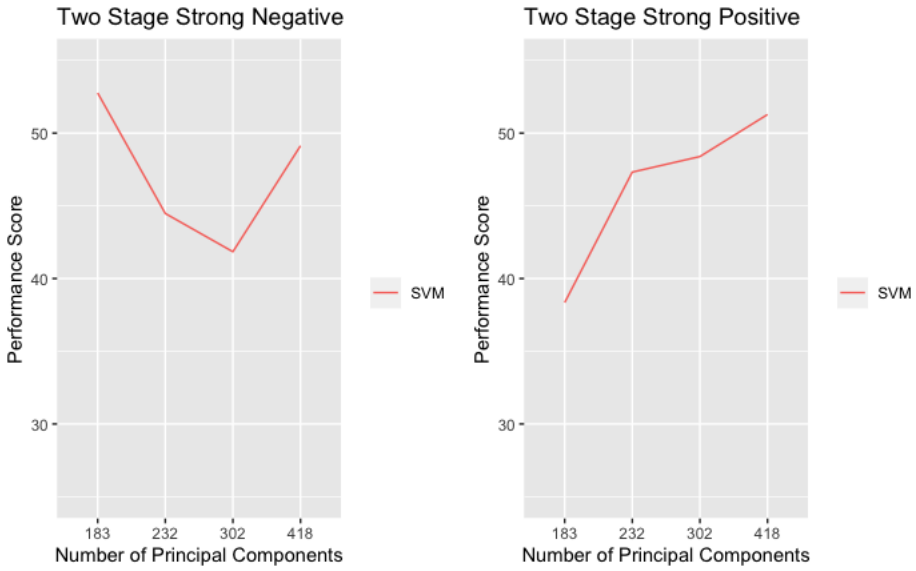

Figure 3.8.3: Two-stage SVM models with strong sentiments; the x-axis is the number of principal components that correspond to 80%, 85%, 90%, and 95% of the cumulative variance; the y-axis is the performance score for models

different patterns. The plot for two-stage strong negative SVM models is concave up, and the model with the first 302 principal components gives the lowest performance score. The plot for the two-stage strong positive is monotone increasing and achieved the highest at the principal components 418. The best model selected is the strong negative SVM model that used the first 183 principal components, and this model has a performance score of 52.76148.

3.8.4 Three-stage SVM Models

The three-stage SVM models also applied SVM models to classify the headlines at each stage with fixed thresholds of impact. The only tuning parameter in three-stage SVM models is the number of principal components. Figure 3.8.4 illustrates the performance

of three-stage SVM models. The best three-stage SVM model uses the first 232 principal

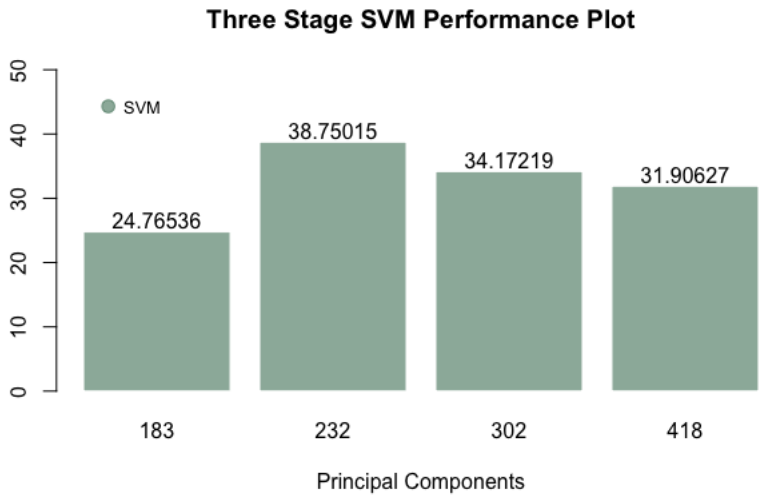

Figure 3.8.4: Three-stage SVM models

components and has a performance score of 38.75015. The performance score is monotone decreasing with the increase of principal components after 232. Also, the model with the first 183 has the lowest performance score. In addition, the principal components 232 and 302 have the highest and second highest performance scores, and this pattern is the same as the three-stage LASSO and logistic models. In conclusion, the three-stage models prefer to use the first 232 or 302 principal components. The first 183 and the first 418 principal components are insufficient to build strong three-stage SVM models.

3.8.5 Best SVM Models

In the previous sections, we tuned the parameters for the SVM models to select the best model so that we could generate a profitable trading strategy. Table 3.8.1 summarizes the information collected to select the best models. 183 is a frequently selected principal component number for SVM regression models. The only SVM model that does not use the first 183 principal components is the three-stage SVM model, which uses the first 232 principal components. Therefore, we can investigate that the SVM models prefer to use less information to classify the headlines than LASSO and logistic models. The two-stage SVM models still use 5% and 95% as the threshold of impact, which are the same with LASSO

Best SVM Models	Sentiment	Thresholds	PC's	Performance Score
One-stage	Negative	10%, 90%	183	22.74247
Two-stage	Neutral	5%, 95%	183	111.6266
Strong Two-stage	Negative	Fixed	183	52.76148
Three-stage	Strong	Fixed	232	38.75015

Table 3.8.1: Performance of best SVM Models. *PC's* is the number of principal components.

and logistic models. In addition, the performance of the best SVM models is close to the best LASSO models, and the best SVM and LASSO models have higher performance scores than the best GLM models.

3.9 Summary of Regression Models

In this Chapter, we proposed logistic, LASSO, and SVM models to classify the headline sentiments. Finally, we explored some findings from the regression models. First, the models that classify neutral vs non-neutral sentiments at the first stage usually outperform others in two-stage regression models. Second, one-stage models that predict negative and non-negative headlines have higher accuracy, which means it is easier for the models to detect how negatively impactful headlines influence the stock return. Now, we want to find the best one-stage, two-stage, and three-stage models.

3.9.1 One-Stage Models

We construct one-stage logistic regression logistic and LASSO models and one-stage SVM models. Table 3.9.1 lists the best one-stage models. From the one-stage table, we can

One-Stage Models	Sentiment	Thresholds	PC's	Performance Score
LASSO	Negative	10%, 90%	302	22.22222
Logistic	Negative	10%, 90%	183	20.233463
SVM	Negative	10%, 90%	183	22.74247

Table 3.9.1: Performance of the best one-stage LASSO, logistic, SVM models, *negative* represents classifying negative vs non-negative headline sentiments; *PC's* is the number of principal components. 183 PC's correspond to 80% of the cumulative variance and 302 PC's correspond to 90% of the cumulative variance

investigate that all the best models classify negative and non-negative sentiments, which

means it is easier for our models to predict negative sentiments. The SVM model has a higher performance score than both logistic and LASSO models. In addition, SVM and logistic models used the first 183 principle components, meaning that the models' features include 80% of the cumulative variance. The best one-stage model is the SVM.

3.9.2 Two-Stage Models

We introduced two-stage regression models for two cases: three headline sentiments and four headline sentiments. In the first case, we have positive, neutral, and negative two-stage models. In the second case, we classify four headline sentiments: positive, neutral, negative, and one strong sentiment (strong positive or strong negative). Therefore, we have strong positive and strong negative two-stage models. Table 3.9.2 includes the best models chosen from the two-stage models from the first case. From Table 3.9.2, we can conclude

Two-Stage Models	Sentiment	Thresholds	PC's	Performance Score
LASSO	Neutral	5%, 95%	183	106.4003
Logistic	Neutral	10%, 95%	302	105.7155
SVM	Neutral	5%, 95%	183	111.6266

Table 3.9.2: Performance of the best two-stage LASSO, logistic, SVM models, *neutral* represents classifying neutral vs non-neutral headline sentiments at the first stage and positive and negative in the second stage. *PC's* is the number of principal components. 183 PC's correspond to 80% of the cumulative variance and 302 PC's correspond to 90% of the cumulative variance

that the neutral two-stage regression models outperform the other models. Also, neutral two-stage models work better with the 5% and 95% quantile numbers as the thresholds, which is the opposite of one-stage negative models. In addition, the two-stage SVM model outperforms the two-stage LASSO model and has the highest performance score.

Now we start analyzing strong positive and strong negative two-stage models. First, we list the table for strong two-stage models. Table 3.9.3 shows that strong negative two-stage models outperform strong positive two-stage models, except for LASSO models. Still, the strong negative logistic models have the highest performance score among all the models.

Strong Two-Stage Models	Strong	PC's	Performance Score
LASSO	Positive	418	50.90337
Logistic	Positive	302	47.83819
SVM	Positive	418	51.27849
LASSO	Negative	183	50.13287
Logistic	Negative	232	53.66705
SVM	Negative	183	52.76148

Table 3.9.3: Performance of the best strong two-stage LASSO, logistic, SVM models, the stage combinations are fixed for strong two-stage models. *Strong* represents a strong positive or strong negative model, *PC's* is the number of principal components. 183 PC's correspond to 80% of the cumulative variance, 232 PC's correspond to 85% of the cumulative variance, and 418 PC's correspond to 95% of the cumulative variance

3.9.3 Three Stage Models

We also fixed the stage combinations and threshold of impact for three-stage models. We fixed the impact threshold for three-stage regression models to be 5%, 30%,70%, and 95% for strong negative, negative, positive, and strong positive, respectively. Table 3.9.4 illustrates

Best Three-Stage Models	Principal Components	Performance Score
LASSO	232	38.96683
Logistic	302	27.021055
SVM	232	38.75015

Table 3.9.4: Performance of the best three-stage LASSO, logistic, SVM models, 232 PC's correspond to 85% of the cumulative variance and 302 PC's correspond to 90% of the cumulative variance

both LASSO and SVM outperform logistic models. The best LASSO and SVM models both used the first 232 principal components in the three-stage models and have close performance scores. The first 418 principal components are not shown in the table, which means we do not need to keep 95% of the total variance when applying the best three-stage models.

3.9.4 Best Models in Each Scenario of Regression Models

In summary, this section provides us with two common findings. First, the models perform better in classifying negative headline sentiments. Second, the combination of PCA and SVM is stable in each scenario and usually gives high performance scores. Next, we calculate

the models' normalized performance scores to compare the best models with one-stage, two-stage, and three-stage approaches. The normalized performance scores are calculated as follows.

$$\text{Normalized performance score} = \frac{\text{Performance score}}{\text{Maximum performance score}} \tag{3.9.1}$$

Therefore, we need to find the maximum possible performance score for each scenario of regression models. For one-stage models, we can find that performance scores equal to a by (3.3.6), where a is a percentage number in Table 3.3.1. Thus the maximum performance score for one-stage models is 100. By (3.3.7), we know the performance score for two-stage models that classify three-states of sentiment is $a + e + i - c - g$, where a, e, i, c, g are percentage numbers in Table 3.3.3. To maximize it, we need to have c, g equal to 0 and a, e, i equal to 100. Thus the maximum performance score for normal two-stage models is 300. Accordingly, we calculate the maximum performance score for strong two-stage and three-stage models, which are 400 and 500, respectively. Finally, the normalized scores are calculated by dividing them by the maximum possible values for each scenario of regression models.

Now we are summarizing all the information in this section to find the best model in each scenario. Table 3.9.5 illustrates the best models in each scenario. The SVM is the most

Best Models	Regression	PC's	Thresholds	PS	NS
Positive one-stage	SVM	183	10%	19.79	0.2
Negative one-stage	SVM	183	10%	22.74	0.23
Positive two-stage	Logistic	302	5%, 95%	104.16	0.35
Neutral two-stage	SVM	183	5%, 95%	111.63	0.37
Negative two-stage	Logistic	302	10%, 95%	105.72	0.35
Strong Positive two-stage	SVM	418	Fixed	51.28	0.13
Strong Negative two-stage	Logistic	232	Fixed	53.67	0.13
Three-stage	LASSO	232	Fixed	38.97	0.08

Table 3.9.5: Performance of the best models in each scenario; *PC's* is the number of principal components; 183 correspond to 80% of the cumulative variance; 232 correspond to 85% of the cumulative variance, and 418 correspond to 95% of the cumulative variance; *NS* is the normalized performance scores; *PS* denotes the performance scores.

powerful model, and the logistic model is also frequently selected. Also, the first 183 or

232 PC's are selected frequently, indicating that our regression models require a relatively small number of principal components with a high cumulative variance. In addition, the neutral two-stage model has the highest normalized score, and two-stage models have higher normalized scores than other scenarios of regression models. The models that include strong sentiment have lower normalized performance scores than others. Thus it is difficult for models to classify strong sentiment from headlines accurately. In Chapter 4, the models in this table are applied to validation data. Then we make trading strategies based on the predictions from those models.

Chapter 4

Trading Strategies

In this thesis, we connect headline sentiments with stock returns. A positive or negative headline sentiment means the posted headline has a positive or negative impact on the stock price. Particularly, if a headline has positive sentiment, the stock price increases relative to yesterday's stock price. We construct trading strategies using predictions from regression models. Investors can make excess profits from the trading strategies are made based on the known information that is well collected and aggregated [11].

However, after the headlines are posted, we predict the stock return relative to the close price on one trading day before it. Generally, we use headlines posted on day i to predict the stock return relative to the close price on day $i-1$. Therefore, on the day $i-1$, we assume we already know the headlines posted on day i so that we can make a prediction of the stock return on day i and buy one-day call or put options. This situation is unrealistic in the real world because we can't know the financial news headlines before they are posted. Such trading strategies are unrealistic. However, they allow for testing the prediction power of our results. We also create realistic trading strategies. We believe that positive and negative headlines can affect the stock market for more than one day. Therefore, we assume the headlines posted on day i will impact the close price on day $i+1$ in the same way, so we can buy one-day call or put options on day i. In summary, the only difference between realistic and unrealistic trading strategies is the date of buying options.

Realistic trading strategies consist of options purchased on the same day when an impactful headline is posted.

Unrealistic trading strategies assume we know the headline's sentiment one trading day before the headline is posted and purchase the corresponding put or call option one trading day before the headline is posted.

This thesis focuses on unrealistic strategies because our models predict the stock return relative to the close price one trading day before. Thus, we can use this approach to test the prediction power of our models.

We classify the headlines' impact as strong negative, negative, neutral, positive, or strong positive and create ten trading strategies that use predictions for the sentiment and three trading strategies without predictions. Our trading strategies consist of standard European call and put options that are introduced in Section 4.1. The ten trading strategies presented below are made for each type of regression model. Strategy 1 to Strategy 7 below can both be realistic or unrealistic depending on the date we buy options. If we purchase an option on the same day when an impactful headline is posted and execute it one trading day later, we construct a realistic strategy. If we purchase an option one trading day before an impactful headline is posted and execute it one trading day later, we construct an unrealistic strategy.

Strategy 1 consists of only call options. It is built on the positive one-stage model that returns positive sentiment predictions. Strategy 1 buys one call option for each predicted positively impactful headline.

Strategy 2 consists of only put options. It is built on the negative one-stage model that returns negative sentiment predictions. Strategy 2 buys one put option for each predicted negatively impactful headline.

Strategy 3 consists of call and put options. It uses predictions from the positive and negative one-stage models and buys one call or one put option when positive or negative sentiments are predicted, respectively. If the predictions are overlapped, buy one call and one put option on the same day.

Strategy 4 consists of call and put options. It uses predictions from two-stage models that classify three sentiments. Since the two-stage model predicts positive, negative, and neutral sentiments without overlap, Strategy 4 buys one call or one put option when a positive or negative sentiment is predicted, respectively.

Strategy 5 consists of call and put options. Strategy 5 is built on the strong positive two-stage model that predicts strong positive, positive, neutral, and negative sentiments. Strategy 5 buys ten call options for each predicted strong positive sentiment and buys one call or one put when a positive or negative sentiment is predicted, respectively.

Strategy 6 consists of call and put options. Strategy 6 is built on the strong negative two-stage model that predicts strong negative, negative, neutral, and positive sentiments. Strategy 6 buys ten put options for each predicted strong negative sentiment and buys one call or one put option when a positive or negative sentiment is predicted, respectively.

Strategy 7 consists of call and put options. Strategy 7 is built on three-stage models that predict strong negative, negative, neutral, positive, and strong positive sentiments. Strategy 7 buys ten put or ten call options for each predicted strong negative or positive sentiment, respectively, and buys one call or one put on when a positive or negative sentiment is predicted, respectively.

Arbitrary 1 is a trading strategy that ignores predictions and buys call options daily during the validation period. Since there are 200 trading days with relevant headlines from 2018-05-31 to 2019-05-31. Arbitrary 1 consists of 200 call options.

Arbitrary 2 is a trading strategy that ignores predictions and buys put options daily during the validation period. Arbitrary 2 consists of 200 put options.

Arbitrary 3 is a trading strategy that ignores predictions, buys calls and puts options daily during the validation period. Arbitrary 3 consists of 200 call options and 200 put options.

4.1 European Call and Put Options

European options are contracts that allow their holders to buy or sell an underlying asset at a fixed price on a given day all specified in advance. European options can be either calls or puts. The call option allows its holders to buy the underlying asset for a strike price of K on the expiration date. The put option allows its holders to sell the underlying asset for a strike price of K on the expiration date. Thus the amount we can receive from exercising European call and put options is given by the payoff defined as follows.

$$\text{Payoff of call option} = \max(S - K, 0) \tag{4.1.1}$$

and

$$\text{Payoff of put option} = \max(K - S, 0) \tag{4.1.2}$$

where S is the underlying asset's price on the expiration date, and K is the strike price fixed when the option is written. We use at-the-money one-day-to-expiry options to construct our trading strategies. That is, we use the current stock price as the strike price. Suppose the option is purchased on the day t. The strike price is

$$K = S_t \tag{4.1.3}$$

The option expires on day $t + 1$ when the option's holder can exercise it. Therefore, if we buy a call option on day t for C_t dollars, then the profit from it is $\max(S_{t+1} - S_t, 0) - C_t$. Conversely, if we buy a put option for P_t dollars, then the profit is $\max(S_t - S_{t+1}, 0) - P_t$.

4.2 Black–Scholes Formula

The trading strategies consist of European call or put options. We select the Black–Scholes formula to calculate the price of options for multiple reasons. First, we only have a few alternatives because of our data set. Recall that our data set only includes the stock's prices, trading volume, and stock return, so we don't have historical option prices that can be used to calibrate an asset price model. Therefore, the parameters of any asset model we use

for pricing options can only be estimated under the real-world measure. However, we need risk-neutral parameters to price derivatives (options). Suppose we are using an incomplete asset price model (such as pure jump, jump-diffusion or stochastic volatility models). In that case, there is no unique risk-neutral probability measure; hence, there is no one-to-one mapping between real-world and risk-neutral model parameters. The Black–Scholes model is a complete market model, so we can estimate the volatility under the real-world measure (using the historical asset returns) and then use the same value as a risk-neutral parameter. Second, we can use nonlinear diffusion models such as the CEV model, which is a complete market model, but estimating its parameters from asset returns is more complex. Also, for one-day options, we would not notice a significant difference in prices produced by the CEV and Black–Scholes models. In conclusion, we select the Black–Scholes model to calculate the options' prices because we are using historical stock returns to estimate the volatility of options, and the Black–Scholes model is more simplified and efficient than others.

Specifically, we use the Black–Scholes model to calculate the no arbitrage values of European call and put options for the trading strategies. We calculate the initial price and the next-day profit of each option. Next, we sum up all the costs and profits to calculate the total cost and return at the end of the validation period. The Black–Scholes model gives us the pricing formula of call and put options as follows.

$$C_t = N(d_1)S_t - N(d_2)Ke^{-rh} \tag{4.2.1}$$

and

$$P_t = N(-d_2)Ke^{-rh} - N(-d_1)S_t \tag{4.2.2}$$

where h is $\frac{1}{252}$ (252 trading days in a year), t is the day we write the options, and N is the standard normal cumulative distribution function. Also, $d_1 = \frac{\ln \frac{S_t}{K} + (r + \frac{\sigma_t^2}{2})h}{\sigma_t \sqrt{h}}$ and $d_2 = d_1 - \sigma_t \sqrt{h}$. The C_t and P_t in (4.2.1) and (4.2.2) represent the no arbitrage prices of the call and put options, respectively. Since we are making trading strategies based on Apple's stock price, the S_t represents the close price of Apple stock on the day we buy call and put options. We are setting at-the-money options, so K equals the S_t (the close price

of Apple stock). Here, r is the risk-free interest rate, and we use the average annual yield of 10 years US treasure which is 2.91%. We set the σ_t to be the annual standard deviation of logreturn in the past years which is $\sigma_t^2 = \frac{1}{nh}\sum_{i=t-n}^{t-1}(x_i - \bar{x_t})^2$, where $n = 252$ is the total number of trading days in one year before we purchase the options, x_i denotes the logreturn on day i, and $\bar{x_t} = \frac{1}{n}\sum_{i=t-n}^{t-1} x_i$.

4.3 One-Stage Trading Strategies

With a one-stage approach, we construct positive and negative one-stage models. The trading strategies consist of at-the-money options with one day to expiry. Recall the discussion of realistic and unrealistic trading strategies at the beginning of this chapter. In this section, we compare realistic and unrealistic trading strategies with a one-stage approach. Specifically, with realistic trading strategies, we purchase one-day call options on days when predicted positive headlines are posted and one-day put options on days when predicted negative headlines are posted. In contrast, unrealistic trading strategies purchase one-day options one trading day before the headlines are posted and execute them on the same day when the headlines are posted.

4.3.1 Positive One-Stage Models

The positive one-stage model predicts whether a headline has positive or non-positive impacts on the stock. The best positive one-stage model is the SVM model with 90% as the upper thresholds and the first 183 principal components as the features. We use the SVM model on the test data from 2018-06-01 to 2019-05-31 and buy call options on the trading days with predicted positive headlines. The summary of our prediction on the test data is shown in Table 4.3.1. There are 200 trading days in the test data. Thus the sum of

	Predict Positive	Predict Non-Positive
Actual Positive	9	15
Actual Non-Positive	29	147

Table 4.3.1: Prediction table for positive one-stage SVM model

predicted positive and non-positive is 200. First, we create an unrealistic strategy based on

Table 4.3.1. The result of the unrealistic strategy is given by Table 4.3.2. We can conclude that there are 38 days that are predicted to have positive headlines. Thus 38 call opinions are bought for the trading strategy. With the test data, we can verify the quality of the trading strategy based on the prediction. In Table 4.3.2, *Purchased* is the total number of

Unrealistic Strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 1	38	22	48.0451	78.02912	29.98402	62.408 %
Arbitrary 1	200	106	252.7396	285.4352	32.69566	12.937%

Table 4.3.2: Results of unrealistic trading strategy for positive one-stage SVM model

options in this trading strategy. *Exercised* represents the total number of options exercised in the trading strategy, which means 22 options can be profitable. *Price* is the total cost of all 38 options, and *Payoff* is the total payoff of the exercise options. *Profit* is the total payoff minus the total cost, representing how much money we can earn from this strategy. *Return* is equal to the total profit divided by the total cost. From Table 4.3.2, we can conduct that the trading strategy works well and has a much higher return than Arbitrary 1. Although the total profit for our trading strategy is small, it can be amplified by increasing the number of purchased options. In conclusion, the unrealistic trading strategy made by the positive one-stage SVM model is powerful and successful.

We also create realistic trading strategies based on Table 4.3.1. Table 4.3.3 illustrates the result from the realistic trading strategy with positive one-stage regression model. The

Realistic Strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 1	38	20	48.32029	67.7855	19.46521	40.284 %
Arbitrary 1	200	106	246.9694	278.9186	31.9492	12.937%

Table 4.3.3: Results of realistic trading strategy for positive one-stage SVM model

realistic strategy also buys 38 call options, the same as the unrealistic one, because they use the same predictions but buy options on different days. The realistic strategy exercises 20 options that are close to the unrealistic strategy. Thus our assumption that the positive impact can last for more than one day is confirmed by the results. Although the realistic trading strategy exercises 20 options, it has a lower annual return than the unrealistic strategy. Because the market may be more sensitive to the newest headlines, the options

in the realistic strategy are exercised one trading day after the headlines are posted.

In conclusion, with the positive one-stage model, the realistic and unrealistic strategy buys the same number of options and exercises a close number of options, but the unrealistic one has a higher payoff and annual return. Both realistic and unrealistic strategies are more profitable than Arbitrary 1.

4.3.2 Negative One-Stage Models

The best negative one-stage model is the SVM negative one-stage model with 10% as the threshold of impact, and the first 183 principal components as predictors. Table 4.3.4 shows the predictions from negative one-stage SVM models on test data. Based on Table 4.3.4,

	Predict Negative	Predict Non-Negative
Actual Negative	11	14
Actual Non-Negative	31	144

Table 4.3.4: Prediction table for the negative one-stage SVM model

the unrealistic trading strategy is supposed to buy 42 put options but only 11 headlines are accurately predicted. Also, we created a trading strategy without using predictions. The trading strategy based on the predictions is more profitable than Arbitrary 2. Also,

Unrealistic strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 2	42	27	52.27045	124.4015	72.13103	137.996%
Arbitrary 2	200	94	248.2788	293.9115	45.63267	18.380%

Table 4.3.5: Results of unrealistic trading strategy for the negative one-stage SVM model

Arbitrary 2 has a higher annual return than Arbitrary 1, which means the market is more volatile when negative headlines are posted. In conclusion, with the one-stage approach, both positive and negative models are making profits, and the negative model is more profitable because the market seems to be more sensitive to negative headlines.

Realistic strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 2	42	22	52.143	67.95042	15.80742	30.31551%
Arbitrary 2	200	94	242.6105	287.2013	44.591	18.380%

Table 4.3.6: Results of realistic trading strategy for the negative one-stage SVM model

At the same time, we also create a realistic strategy based on Table 4.3.4. Table 4.3.6 shows the results from a realistic strategy with the negative one-stage model. The realistic strategy also buys 42 put options but only exercises 22 options and has a smaller annual return than the unrealistic strategy.

In summary, with the negative one-stage model, although the realistic and unrealistic strategy buys the same number of put options, the unrealistic strategy has a much higher annual return than the realistic strategy. Because the market may be very sensitive to negative headlines that the stock price significantly decreases on the day the negative headlines are posted. Thus on the next trading day, the negative impact still lasts but it is less profound than on the day before. Therefore, the realistic strategy is less profitable than the unrealistic strategy.

4.3.3 Combination of Positive and Negative One-Stage Models

The one-stage models can either predict positive or negative headline sentiments, but we can combine the predictions and make a trading strategy based on both predictions. For example, based on predictions from the positive one-stage SVM model, we buy a call option one trading day before the headlines are predicted to be positive. Conversely, using predictions from the negative one-stage LASSO models, we buy a put option one trading day before the predicted negative headline is posted. Since the predictions come from two different models, some overlapping predictions may exist. Thus for the overlapping predictions, we buy both call and put options. In summary, the trading strategy made from both positive and negative one-stage models consists of both call and put options. We buy a call option for predicted positive sentiment and a put option for predicted negative sentiment, both call and put options for overlapping predictions. And we add the results from Sections 4.3.1 and 4.3.2 to calculate the profit of the combined trading strategy. Table 4.3.7 shows the results from combined unrealistic strategies. Although we can make trading strategies based on positive and negative one-stage models, the return of the combined trading strategies is lower than negative one-stage trading strategies. However, the combined trading strategy still has a high return and exercises more options than any other one-stage trading

Unrealistic Strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 1	38	22	48.0451	78.02912	29.98402	62.408 %
Strategy 2	42	27	52.27045	124.4015	72.13103	137.996%
Strategy 3	80	49	100.3156	202.4306	102.115	101.794%
Arbitrary 3	400	200	501.0184	566.1199	78.32833	15.665 %

Table 4.3.7: Summarized results for one-stage unrealistic trading strategies

strategy.

Also, we create the combined realistic strategy for one-stage models. The annual return of

Realistic Strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 1	38	20	48.32029	67.7855	19.46521	40.284 %
Strategy 2	42	22	52.143	67.95042	15.80742	30.316%
Strategy 3	80	42	100.4633	135.7359	35.27263	35.120%
Arbitrary 3	400	200	501.0184	566.1199	78.32833	15.665 %

Table 4.3.8: Summarized results for one-stage realistic trading strategies

the combined realistic strategy is 35.10997% which is higher than Arbitrary 3. We find that the annual return of the combined unrealistic strategy is higher than the realistic strategy. In conclusion, with the one-stage approach, all unrealistic strategies work better than realistic ones. The annual return of the unrealistic strategy made with the negative one-stage model is the highest among all the strategies.

4.4 Two-Stage Trading Strategies

In Section 4.3, we compared the realistic and unrealistic strategies. Next, we focus on the analysis of unrealistic strategies because they correspond to our models. The two-stage logistic regression is proposed to classify three sentiments at once to generate trading strategies that consist of both call and put options. We can use the prediction of both positive, neutral, and negative headline sentiments to construct trading strategies. In this section, we only analyze unrealistic trading strategies.

4.4.1 Positive Two-Stage Models

Positive two-stage models classify positive and non-positive headlines in the first stage and neutral and negative in the second stage. The best positive two-stage model selected is the logistic model with 5% and 95% as the threshold and the first 302 principal components as the features. Table 4.4.1 illustrates the predictions from the best positive two-stage model.

	Predict Positive	Predict Neutral	Predict Negative
Actual Positive	4	7	2
Actual Neutral	16	142	11
Actual Neutral	1	16	1

Table 4.4.1: Prediction table for positive two-stage logistic model

We can see that the total number of headlines predicted to be positive is 21, and the total number of headlines predicted to be negative is 14. Thus, the trading strategy based on predictions consists of 21 call options and 14 put options. From Table 4.4.2, we can find that

Unrealistic strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 4	35	22	43.78625	86.32138	42.53513	97.143%
Arbitrary 3	400	200	495.2482	572.8301	77.58187	15.665 %

Table 4.4.2: Results of unrealistic trading strategy for the positive two-stage logistic model

the positive two-stage model has a higher return than the positive one-stage models and Arbitrary 3. The Arbitrary 3 strategy ignores the predictions and buys both call and put options every trading day. Thus the number of exercises is 200 because we exercise either the call or put options every day but not the both options. Using the positive two-stage logistic model we build a profitable trading strategy.

4.4.2 Neutral Two-Stage Models

Neutral two-stage models classify the neutral and non-neutral headlines in the first stage and positive and negative headlines in the second stage. The best neutral two-stage model is the SVM two-stage model with 5% and 95% as the threshold of impact, and the first 183 principal components as predictors. In Table 4.4.3, we can see that there are 26 headlines predicted to be positive and 22 headlines predicted to be negative. Thus our trading strategy

	Predict Positive	Predict Neutral	Predict Negative
Actual Positive	3	6	4
Actual Neutral	19	136	14
Actual Neutral	4	10	4

Table 4.4.3: Prediction table for neutral two-stage SVM

consists of 26 call and 22 put options. We make an unrealistic strategy based on Table 4.4.3. From Table 4.4.4, we can see that our trading strategy made from a neutral two-stage model

Unrealistic strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 4	48	28	60.51668	116.3231	55.80646	92.217 %
Arbitrary 3	400	200	495.2482	572.8301	77.58187	15.665 %

Table 4.4.4: Results of unrealistic trading strategy for the neutral two-stage SVM model

buys 48 options but only exercises 28 of them, which means the predictions are not accurate enough. The neutral trading strategy performs slightly better than the strategy based on the positive two-stage model.

4.4.3 Negative Two-Stage Models

Negative two-stage models classify the negative and non-negative headlines in the first stage and positive and normal headlines in the second stage. The best negative two-stage model selected is the logistic model, which uses 10% and 95% as the threshold of impact, and the first 302 principal components as the features. In Table 4.4.5, there are 9 headlines predicted

	Predict Positive	Predict Neutral	Predict Negative
Actual Positive	0	13	0
Actual Neutral	9	136	17
Actual Neutral	0	17	8

Table 4.4.5: Prediction table for negative two-stage logistic model

to be positive and 25 headlines predicted to be negative. Thus the trading strategy consists of 9 call and 25 put options. We can see that no positive headline is predicted correctly. However, we still can make profits from the predictions because the most neutral headlines predicted to be high still have a positive impact on the market but are defined to be neutral because they did not reach the threshold. Table 4.4.6 is the result of an unrealistic strategy

made with Table 4.4.5. The trading strategy made with the negative two-stage model is

Unrealistic Strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 4	34	25	41.37602	111.4111	70.03511	169.265%
Arbitrary 3	400	200	495.2482	572.8301	77.58187	15.665 %

Table 4.4.6: Results of unrealistic trading strategy for the negative two-stage logistic model

the best we have ever created. We can see that most of the options in the strategy are exercised, and the return is 169.265%, which is much higher than the return for Arbitrary 3. In addition, the negative two-stage model focuses more on classifying negative headlines. Thus the prediction for negative headlines is more accurate. Recall that we found the market was more sensitive to negative headlines, so the negative two-stage model could make more powerful trading strategies.

4.4.4 Strong Positive or Negative Two-Stage Models

Recall we also add strong sentiments to the two-stage logistic regressions. We classify the headlines into four states: negative, neutral, positive, strong positive, or strong negative. Due to the limit of two-stage models, we can only add one of the strong sentiments at once. In this case, we define the thresholds as 5%, 30%, 70%, and 95%, representing strong negative, negative, positive, and strong positive, respectively. First, we generate the trading strategy with strong positive two-stage models. The SVM model with the first 418 principal components is the best strong positive two-stage model. Table 4.4.7 is the prediction of the strong positive two-stage SVM model. Based on the prediction, the trading strategy

Actual\Predict	Strong Positive	Positive	Neutral	Negative
Strong Positive	5	1	3	4
Positive	8	12	12	18
Neutral	13	14	23	19
Negative	7	21	15	25

Table 4.4.7: Prediction table for strong positive two-stage SVM

consists of 378 call and 66 put options. Table 4.4.8 gives the result of this unrealistic strategy. The annual return of the aggressive trading strategy is higher than an arbitrary one. However, the return is not as good as that for the trading strategies made with two-stage

Unrealistic strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 5	444	285	556.1896	881.004	324.8144	58.400 %
Arbitrary 3	400	200	495.2482	572.8301	77.58187	15.665 %

Table 4.4.8: Results of unrealistic trading strategy for the strong positive two-stage SVM model

models, which do not include any strong sentiment.

Also, we generate the trading strategy based on strong negative two-stage models. In such models, we classify the negative and non-negative headlines at first. The negative headlines are classified as strong negative and negative ones, and the non-negative headlines are classified as positive and normal ones. The logistic model with the first 232 principal components is the best strong negative two-stage model. From Table 4.4.9, we can generate

Actual\Predict	Positive	Neutral	Negative	Strong Negative
Positive	21	20	15	7
Neutral	21	27	16	5
Negative	9	19	17	5
Strong Negative	1	7	6	4

Table 4.4.9: Prediction table for strong negative two-stage logistic model

a trading strategy that consists of 52 call and 264 put options. Thus we can create a table for the result of this trading strategy with test data. With Table 4.4.10, we can see that

Unrealistic strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 6	326	175	396.6357	708.3781	311.7424	78.597 %
Arbitrary 3	400	200	495.2482	572.8301	77.58187	15.665 %

Table 4.4.10: Results of unrealistic trading strategy for the strong negative two-stage logistic model

this trading strategy performs well and makes more profit than the trading strategy made from strong positive two-stage models. This also proves that the market is more sensitive to negative headlines, so trading strategies that focus on negative headlines are always more profitable.

In conclusion, although aggressive trading strategies buy more options, the overall annual return of aggressive trading strategies is not as good as those made with two-stage models that don't include any strong sentiment. In the next section, we will generate more

aggressive trading strategies based on the prediction made from three-stage models.

4.5 Three-Stage Trading Strategies

Three-stage logistic regression classifies five sentiments at once. Thus we can make a more aggressive trading strategy based on the predictions from the three-stage model. The threshold of impact for the three-stage model is fixed, and we used 5%, 30%, 70%, and 95% to represent strong negative, negative, positive, and strong positive, respectively. The best three-stage model selected is the LASSO model with the first 232 principal components.

Actual\Prediction	Strong Positive	Positive	Neutral	Negative	Strong Negative
Strong Positive	8	0	1	2	2
Positive	15	11	12	6	6
Neutral	7	17	28	13	4
Negative	8	13	13	9	7
Strong Negative	1	1	5	2	9

Table 4.5.1: Prediction table for three-stage LASSO

The rows represent the actual result, and the columns represent the predicted result. We buy 10 call and put options for predicted strong positive and strong negative headlines, respectively, and 1 option for each positive or negative one. With Table 4.5.1, we create a trading strategy consisting of 432 call options and 312 put options. Table 4.5.2 illustrates the results of this trading strategy. In Table 4.5.2, we can see that the aggressive trading

Unrealistic strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 7	744	478	927.753	2080.406	1152.653	124.241 %
Arbitrary 3	400	200	495.2482	572.8301	77.58187	15.665 %

Table 4.5.2: Results of unrealistic trading strategy for the three-stage LASSO model

strategy based on the three-stage logistic regression works better than the aggressive trading strategies constructed from two-stage models. Also, its annual return is much higher than Arbitrary 3. In conclusion, the trading strategy constructed from three-stage models is profitable. However, the return of all the aggressive trading strategies is not as high as that of the normal trading strategies. Still, aggressive trading strategies successfully extract more profitable headlines and provide us with additional choices when designing trading

strategies.

4.6 Summary of Unrealistic Trading Strategies

In this Chapter, we constructed unrealistic trading strategies for each kind of regression model, including one-stage, two-stage, and three-stage models. Table 4.6.1 summarizes the best unrealistic strategies. In Table 4.6.1, we find that the negative one-stage, negative

Unrealistic strategy	Model	Return
Strategy 2	Negative one-stage SVM	137.9958%
Strategy 4	Negative two-stage logistic	169.265%
Strategy 6	Strong Negative Two-stage logistic	78.59666 %
Strategy 7	Three-stage LASSO	124.2414 %

Table 4.6.1: Summary of unrealistic trading strategies.

two-stage, and strong negative two-stage models are the best among all one-stage and two-stage regression models. We can conclude that the market is more sensitive to negative headlines, and that is why the strategies made with negative models have higher returns than others. In addition, an unrealistic strategy with a negative two-stage logistic model has the highest annual return, and Strategy 2 has the second-highest return. The aggressive trading Strategy 6 and 7 do not demonstrate the same performance as the normal Strategy 2 and 4. In conclusion, the one-stage and two-stage regression models are sufficient to design trading strategies with strong performance.

Chapter 5

Conclusion and Future Work

5.1 Main Results

The NLP techniques were proposed to process financial news headlines and transform them into a binary data set with a size of 1502×1178. The PCA method was implemented to reduce the dimension of the data set. We introduced one-stage, two-stage, and three-stage models to analyze the data set and proposed logistic, LASSO, and SVM regression models to classify the headlines based on the stock returns. Table 5.1.1 illustrates the best models selected from training data. The neutral and negative in Table 5.1.1 represent the

Best models	Type	Thresholds	PC's	PS	NS
Negative one-stage	SVM	90%	183	22.74	0.23
Neutral two-stage	SVM	5%, 95%	183	111.63	0.37
Negative strong two-stage	Logistic	5%, 30%, 70% ,95%	232	53.67	0.13
Three-stage	LASSO	5%, 30%, 70%, 95%	232	38.97	0.08

Table 5.1.1: Results for best regression models on training data, and *NS* represents the normalized performance scores. The value of *PS* is the performance scores

sentiments analyzed in the first stage. We can investigate that 183 and 232 are the only number of principal components which means the projected data set with 80% and 85% of the cumulative variance, respectively, is suitable for building regression models. The neutral two-stage model has the highest normalized score, and the three-stage model has the lowest normalized score, which means it is more difficult to classify five sentiment categories. In addition, the trading strategies made based on the prediction from regression models had

outstanding performances. In Table 5.1.2, the first four trading strategies correspond to the

Unrealistic strategy	Purchased	Exercised	Price	Payoff	Profit	Return
Strategy 2	42	27	52.27045	124.4015	72.13103	137.996%
Strategy 4	48	28	60.51668	116.3231	55.80646	92.217 %
Strategy 6	326	175	396.6357	708.3781	311.7424	78.597 %
Strategy 7	744	478	927.753	2080.406	1152.653	124.241 %
Strategy 4	34	25	41.37602	111.4111	70.03511	169.265%

Table 5.1.2: Results of trading strategies corresponding to models with the highest performance score and the best trading strategy.

best models in Table 5.1.1. The last trading strategy in Table 5.1.2 is the best among all strategies, and it is created based on the negative two-stage logistic model. The negative two-stage logistic is not the best in two-stage models but creates the best trading strategy. The only explanation is that the market is more sensitive to negative headlines. Thus massive profits are made from the trading strategies made from the negative two-stage model. Strategies 6 and 7 are aggressive trading strategies. Such strategies work okay but don't work as well as Strategy 2 made with a negative two-stage model. Although aggressive trading strategies have lower return rates, they extract more profitable headlines and provide more choices for us when making trading strategies.

5.2 Future Work

Principal component analysis worked well in reducing the size of the data set, but it could be better if we had decreased the size of the data set at the start to solve this problem. We can dig into the NLP techniques to filter the features so that the NLP techniques select only impactful features. We can define a new algorithm to filter the features or completely clean the data. In addition, the NLTK package was used to calculate the sentiment scores of the headlines. However, the dictionary used in this package is so general that it could not classify the sentiment of financial news headlines well. Therefore, we can explore the sentiment analysis that analyzes the sentiment of financial texts by creating a new dictionary that includes the meaning of financial words.

In addition, we implemented the principal component analysis to reduce the dimension of

the data set. Some other statistical methods can also be used to reduce the dimension of the data set. We constructed logistic, LASSO, and SVM (linear kernel) regression models to analyze the data, and they are all linear classifiers. We can extend to introduce non-linear regression models to see how they work with our data set.

We created many unrealistic trading strategies consisting of call and put options based on the predictions from the models, but we assumed we already knew the financial news headlines one trading day before, which was impossible in the real world. The realistic strategies created with one-stage models had a much lower return than the unrealistic ones. Thus we can keep working on realistic trading strategies to make them more profitable.

5.3 Conclusion

The primary objective of this thesis is to create statistical learning models and construct trading strategies by analyzing financial news headlines. We collected financial news headlines and proposed NLP techniques to detect the sentiment of the headlines. Also, the NLP techniques were applied to clean and process the data so that we could extract important information from the headlines and use the important information to generate a data set.

Once the data set was generated, the principal component analysis was implemented to project the data set. We only kept the principal components that include 80% to 95% of the cumulative variance of the original data so that we could shrink the dimensionality of the data set. Next, we applied cross-validation to fairly assess the performance of regression models. Meanwhile, we introduced the one-stage, two-stage, and three-stage logistic regression models to classify the sentiments of headlines. The logistic, LASSO, and SVM models were proposed to generate regression models to help classify the headlines based on stock return at each stage.

Lastly, the trained models were applied to the validation data from 2018-05-31 to 2019-5-31. Based on the prediction made from the regression models, we could generate realistic and unrealistic trading strategies. The trading strategies that only consist of one kind of options were constructed based on predictions from one-stage trading strategies. They could be formed from either call or put options. With the predictions made from two-stage trad-

ing strategies, we could generate more diverse trading strategies that consist of both call and put options. Strong two-stage models were also applied to include one more strong sentiment in the two-stage models. We also introduced three-stage trading strategies to extract more profitable headlines and developed aggressive trading strategies to buy additional options for the headlines that are predicted to contain strong positive or strong negative sentiments. Also, all the trading strategies based on the predictions had much higher returns than the strategies made without predictions, proving the advantage of sentiment analysis in financial markets.

Bibliography

[1] W John Wilbur and Karl Sirotkin. "The automatic identification of stop words". In: *Journal of information science* 18.1 (1992), pp. 45–55.

[2] Michael I Jordan et al. *Why the logistic function? A tutorial discussion on probabilities and neural networks*. 1995.

[3] Rob J Hyndman and Yanan Fan. "Sample quantiles in statistical packages". In: *The American Statistician* 50.4 (1996), pp. 361–365.

[4] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[5] Joël Plisson, Nada Lavrac, Dunja Mladenic, et al. "A rule based approach to word lemmatization". In: *Proceedings of IS*. Vol. 3. 2004, pp. 83–86.

[6] Abhisek Ukil. "Support vector machine". In: *Intelligent Systems and Signal Processing in Power Engineering*. Springer, 2007, pp. 161–226.

[7] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[8] Payam Refaeilzadeh, Lei Tang, and Huan Liu. "Cross-validation." In: *Encyclopedia of database systems* 5 (2009), pp. 532–538.

[9] Hervé Abdi and Lynne J Williams. "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.

[10] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. "Regularization paths for generalized linear models via coordinate descent". In: *Journal of statistical software* 33.1 (2010), p. 1.

[11] Wenbin Zhang and Steven Skiena. "Trading strategies to exploit blog and news sentiment". In: *Fourth international aAAI conference on weblogs and social media*. 2010.

[12] Xavier Robin et al. "pROC: an open-source package for R and S+ to analyze and compare ROC curves". In: *BMC bioinformatics* 12.1 (2011), pp. 1–8.

[13] Jill C Stoltzfus. "Logistic regression: a brief primer". In: *Academic emergency medicine* 18.10 (2011), pp. 1099–1104.

[14] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[15] R Core Team. "R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria". In: *http://www. R-project. org/* (2013).

[16] G Vinodhini and RM Chandrasekaran. "Effect of feature reduction in sentiment analysis of online reviews". In: *Int J Adv Res Comput Eng Technol (IJARCET)* 2.6 (2013), pp. 2165–2172.

[17] Chuan-Ju Wang et al. "Financial sentiment analysis for risk prediction". In: *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. 2013, pp. 802–808.

[18] Clayton Hutto and Eric Gilbert. "Vader: A parsimonious rule-based model for sentiment analysis of social media text". In: *Proceedings of the international AAAI conference on web and social media*. Vol. 8. 1. 2014, pp. 216–225.

[19] Xiaodong Li et al. "News impact on stock price return via sentiment analysis". In: *Knowledge-Based Systems* 69 (2014), pp. 14–23.

[20] Duyu Tang et al. "Effective LSTMs for target-dependent sentiment classification". In: *arXiv preprint arXiv:1512.01100* (2015).

[21] Siavash Kazemian, Shunan Zhao, and Gerald Penn. "Evaluating sentiment analysis in the context of securities trading". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 2094–2103.

[22] Zhe Hui Hoo, Jane Candlish, and Dawn Teare. *What is an ROC curve?* 2017.

[23] Bradley Meyer, Marwan Bikdash, and Xiangfeng Dai. "Fine-grained financial news sentiment analysis". In: *SoutheastCon 2017*. IEEE. 2017, pp. 1–8.

[24] Anjuman Prabhat and Vikas Khullar. "Sentiment classification on big data using Naıve Bayes and logistic regression". In: *2017 International Conference on Computer Communication and Informatics (ICCCI)*. IEEE. 2017, pp. 1–5.

[25] Nurulhuda Zainuddin, Ali Selamat, and Roliana Ibrahim. "Hybrid sentiment classification on twitter aspect-based sentiment analysis". In: *Applied Intelligence* 48.5 (2018), pp. 1218–1232.

[26] Sadaf Abdul-Rauf et al. "Exploring transfer learning and domain data selection for the biomedical translation". In: *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*. 2019, pp. 156–163.

[27] VD Chaithra. "Hybrid approach: naive bayes and sentiment VADER for analyzing sentiment of mobile unboxing video comments". In: *International Journal of Electrical and Computer Engineering (IJECE)* 9.5 (2019), pp. 4452–4459.

[28] Huseyn Hasanli and Samir Rustamov. "Sentiment analysis of Azerbaijani twits using logistic regression, Naive Bayes and SVM". In: *2019 IEEE 13th International Conference on Application of Information and Communication Technologies (AICT)*. IEEE. 2019, pp. 1–7.

[29] KR1442 Chowdhary. "Natural language processing". In: *Fundamentals of artificial intelligence* (2020), pp. 603–649.

[30] Anita Yadav et al. "Sentiment analysis of financial news using unsupervised approach". In: *Procedia Computer Science* 167 (2020), pp. 589–598.

[31] Jacques Wainer and Gavin Cawley. "Nested cross-validation when selecting classifiers is overzealous for most practical applications". In: *Expert Systems with Applications* 182 (2021), p. 115222.