Lexicon-Based Sentiment Analysis and Regression Models for Predicting Tesla Share Prices and Sentiment

Introduction:

The accurate prediction of stock prices has always posed a significant challenge for investors and financial analysts due to the complex and dynamic nature of financial markets. Among the various factors influencing share prices, market sentiment plays a crucial role as it reflects market participants' speculation and expectations. Negative market sentiment is often associated with share price declines and market downturns, while positive sentiment tends to drive share prices higher. Understanding and predicting market sentiment can provide valuable insights for investors in making informed decisions.

In the context of this project, the hypothesis was formulated that there exists a linear relationship between sentiment and share price data. This hypothesis assumes that changes in sentiment levels would directly impact share prices in a linear manner. To investigate this relationship, linear regression was employed to predict share prices based on sentiment scores. Additionally, logistic regression was used to predict sentiment labels based on share price data. By analyzing the relationship between sentiment and share prices, it becomes possible to gain insights into the potential predictive power of sentiment analysis in forecasting share price movements.

To conduct the analysis, Tesla, a prominent growth stock, was chosen as the focus of this study. Unlike value stocks that are primarily influenced by financial fundamentals, growth stocks like Tesla are heavily affected by market speculation. Market participants often have high expectations for the growth potential of such stocks, leading to increased volatility and sensitivity to sentiment fluctuations. Examining Tesla's share price movements in relation to market sentiment can shed light on the impact of speculation and sentiment on the value of growth stocks.

By investigating the linear relationship between sentiment and share price data and employing logistic regression to predict sentiment labels, this study aims to enhance our understanding of the interplay between market sentiment and share prices, particularly in the context of a growth stock like Tesla. The findings from this analysis have the potential to provide valuable insights for investors and financial analysts, aiding in more informed decision-making and potentially improving the accuracy of stock price predictions.

Overview:

The objective of this project was twofold: first, to employ Lexicon-Based Sentiment Analysis on marketwatch articles related to Tesla, and second, to utilize linear regression and logistic regression models for predicting sentiment and share prices, respectively. The sentiment analysis approach employed the widely used Loughran and McDonald Sentiment Word Lists, augmented with a custom vocabulary of positive and negative financial terms. These terms, derived from the field of corporate finance and my

knowledge of its concepts, captured sentiment nuances by considering the modifiers associated with subject terms. Particularly, for example, the words in the negative financial term vocabulary indicated a negative sentiment when modified with a positive word and the reverse when modified with a negative word. An example of this is the word “debt”, where when modified with Loughran and McDonald positive words like “grew” or “tremendous”, indicates a negative sentiment whereas when modified with Loughran and McDonald a negative word like “low” indicates positive sentiment.

To compute sentiment scores, dependency parsing was performed on the marketwatch articles, allowing us to identify the subject-modifier relationships within the sentences. By leveraging the Loughran and McDonald Sentiment Word Lists and the custom vocabulary, sentiment scores were assigned based on the presence of subject-modifier pairs. Word pairs that have this subject-modifier relationship were given a weight of 2 to their respective sentiment score. Moreover, words that did not have this subject-modifier relationship but were in the Loughran and McDonald Sentiment Word Lists were assigned a score of 1 if in the positive word list and -1 if in the negative word list. Additionally, document frequency (df) was employed as a weight when calculating the weighted average of sentiment scores from tokens within the article that matched the vocabulary terms.

The predictive models were evaluated using different pairs of sentiment scores and price lists. One pair, referred to as "arrays," contained the original sentiment scores associated with the same share prices. Conversely, the other pair, known as "arrays1," featured averaged sentiment scores for articles with identical share prices, resulting in each sentiment score having a unique share price.

Evaluation metrics were applied to assess the performance of the models. For the linear regression model, mean squared error (MSE) and R-squared (R2) were used to measure the accuracy of the predicted share prices. The logistic regression model's performance was evaluated using precision, recall, F1 score, and accuracy to determine the accuracy of sentiment prediction.

Methods:

Data Scraping and Data Merging

The process of scraping articles from MarketWatch involved utilizing selenium webdriver and action chains to extract relevant information from the website. The Selenium WebDriver was set up using the Chrome browser, enabling navigation to the MarketWatch page dedicated to Tesla stock. Once on the page, the WebDriver located the element containing the news articles using an XPath expression. ActionChains were then employed to perform a scroll-down action on the element, allowing access to more articles. The scroll action was repeated multiple times to ensure a sufficient number of articles were retrieved.

To scrape the articles, a loop iterated through the elements containing the articles. Within each iteration, the loop attempted to retrieve pertinent information such as the article's link, timestamp, and text. Several steps were taken for each article, including checking for the presence of a link subclass,

retrieving the link itself, obtaining the timestamp, parsing the timestamp string into a datetime object, and finally downloading and parsing the article's text using the Article class from the newspaper library. The extracted information, including the date, time, and text, was then stored in a dictionary named "articles," categorized by date.

The scraped articles were saved in a CSV file named "articles.csv" in the format of date, time, and text. Using the csv module, the CSV file was created and opened in write mode, and the data was written row by row using a csv.writer.

To ensure robustness and handle any potential errors during the scraping process, proper exception handling was implemented. Exceptions such as NoSuchElementException, HTTPError, and newspaper.article.ArticleException were caught, allowing the process to continue with the next article. The historical data files were manually downloaded from NASDAQ. They were then loaded and merged into a single dataframe. The function merge_historical was created to handle this process. This dataframe was then merged with a dataframe created from the "articles.csv" file based on date.

Data Preprocessing

The first step in the analysis involved converting the articles into a list of words that contain positive and negative vocabulary words. This preprocessing step was performed to facilitate quicker IDF (Inverse Document Frequency) calculation. The text_to_vocab() function was used for this purpose. The function iterated over the articles and tokenized them using the SpaCy library. It then extracted the positive and negative vocabulary words present in the tokens and stored them in separate dictionaries, pos_text and neg_text. These dictionaries were saved as JSON files, namely "pos_text.json" and "neg_text.json", respectively.

IDF Calculation

The next step was to calculate the IDF scores of the positive and negative words. The idf_scores() function was implemented for this task. It iterated over the positive and negative vocabulary words and calculated their document frequency (doc_freq) by checking their presence in the pos_text and neg_text dictionaries, respectively. The IDF score was then computed using the formula (1 + N) / (1 + doc_freq), where N represents the total number of documents in the dataset. To avoid zero probabilities, 1 was added to the score before taking the logarithm. The IDF scores for positive and negative words were stored in separate dictionaries, pos_words and neg_words, respectively. These dictionaries were saved as JSON files, namely "pos_words.json" and "neg_words.json".

Sentiment Score Calculation

The calculate_sentiment_score() function was developed to compute the sentiment score for each article. It takes the preprocessed text and the index of the article as input parameters. The function initializes variables weights_total and weighted_sum to keep track of the total weights and the weighted sum, respectively. It retrieves the positive and negative words present in the article from the pos_text and neg_text dictionaries based on the provided index. The function then iterates over the tokens in the preprocessed text and applies filtering conditions to include only non-stop words and alphabetic tokens that belong to the positive or negative vocabulary. The sentiment score is calculated based on the presence of positive or negative financial terms, as well as the dependencies between the words. If a word is in the

positive vocabulary, a score of 1 is given. If a word is in the negative vocabulary, a score of -1 is given. If the subject of a sentence is a word in the positive or negative financial term vocabulary and its modifier is in either the negative or positive vocabularies, its sentiment score is given a weight of 2. Otherwise, the sentiment scores are weighted by their IDF scores and added to the weighted_sum and weights_total variables. Finally, the overall sentiment score is computed as the ratio of the weighted sum to the weights total. If the weighted sum is zero, the sentiment score is considered neutral. The function returns the calculated sentiment score.

Sentiment Analysis

The sentiment_analysis() function performs sentiment analysis by calculating sentiment scores for each article and associating them with the corresponding share prices. Unlike sentiment_analysis_average(), it does not average anything and contains all sentiment scores including those that are associated with identical share prices. It iterates over the articles and their respective share prices, preprocesses the text using SpaCy, and calls the calculate_sentiment_score() function to obtain the sentiment score. The sentiment scores and share prices are stored in separate lists. The dataset is then split into a training set and a test set, with 80% of the data used for training. The function returns numpy arrays containing the sentiment scores for the training set (X_train), the share prices for the training set (Y_train), the actual share prices for the test set (Y_actual), and the sentiment scores for the test set (X_test). These arrays are saved in a file named "arrays.npz" for future use.

Average Sentiment Analysis

The average() function is utilized to compute the average sentiment score for texts that have the same share price. It takes a dictionary, sentiment_price_dict, as input, which maps sentiment scores to share prices. The average sentiment score for each unique share price was calculated by iterating through the sentiment scores. The updated share prices and their corresponding average sentiment scores were stored in separate lists (prices_updated and average_scores, respectively). The prices_updated and average_scores are then used as a dataset to form the training set and test set. This is done by splitting the dataset into a training set and a test set, with 80% of the data used for training. The function returns numpy arrays containing the sentiment scores for the training set (X_train), the share prices for the training set (Y_train), the actual share prices for the test set (Y_actual), and the sentiment scores for the test set (X_test). These arrays are saved in a file named "arrays1.npz" for future use.

Training Linear Regression Model

A linear regression model was trained to predict share prices based on sentiment scores. The training process involved loading the preprocessed training and test data from a file (arrays1.npz or arrays.npz), reshaping the input data to match the expected shape for scikit-learn models, initializing a linear regression model, fitting the model to the training data, predicting share prices for the test data using the trained model, evaluating the model's performance using mean squared error (MSE) and R2 score, and plotting the actual and predicted values on a scatter plot.

Training Logistic Regression Model

A logistic regression model was trained using scikit-learn Logistic Regression to predict the general sentiment (-1, 0, +1) based on share prices. The training process involved loading the preprocessed training and test data from a file (arrays1.npz or arrays.npz), reshaping the share prices array to match the shape expected by scikit-learn models, converting sentiment scores to binary values (0 or 1) based on negative or positive sentiment, initializing a logistic regression model, fitting the model to the training data, predicting sentiment labels and probabilities for the test data using the trained model, and plotting the actual sentiment labels, predicted sentiment labels, and predicted probabilities on a scatter plot. Additionally, precision, recall, F1 score, and accuracy were computed for both label-based and probability-based predictions.

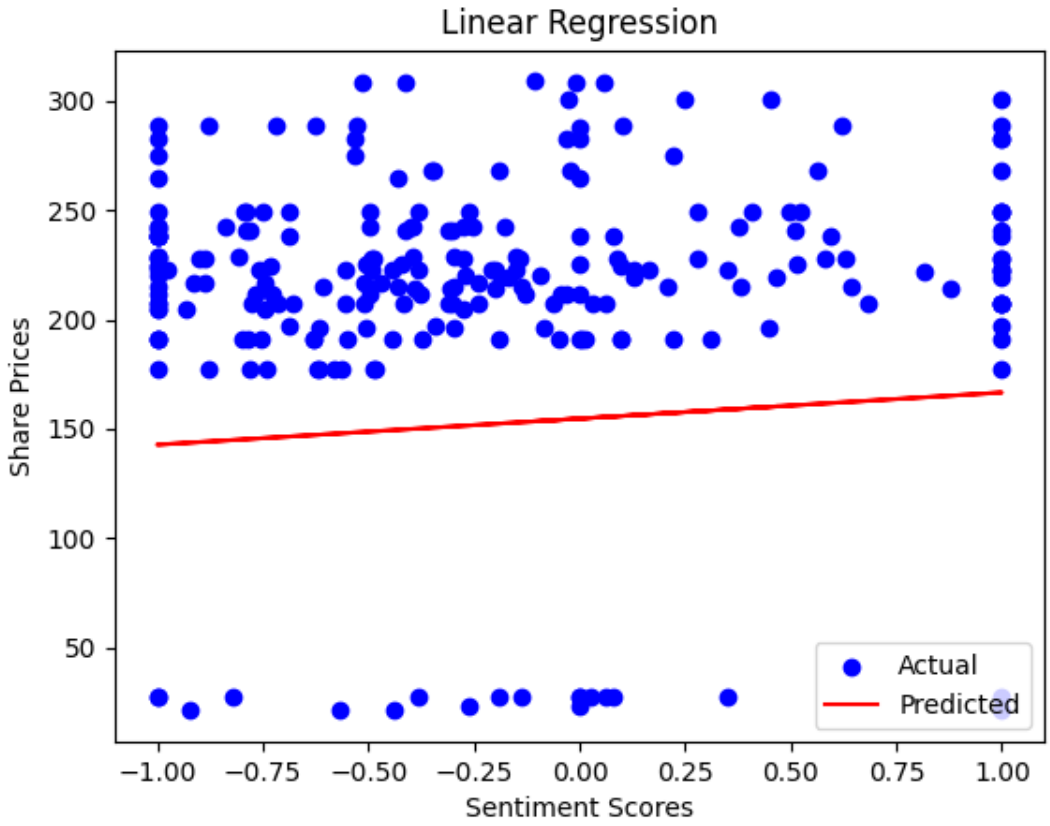
Results

The purpose of this performance analysis was to compare the performance of the linear regression and logistic regression models using two sets of sentiment scores: arrays.npz and arrays1.npz. These datasets differed in the way sentiment scores were associated with share prices. While arrays.npz included sentiment scores that shared share prices, arrays1.npz consisted of sentiment scores with unique shared share prices. As a result, arrays1.npz had fewer data points compared to arrays.npz, raising concerns about the potential impact on the models' performance.

For arrays.npz, the performance of the models was as follows:

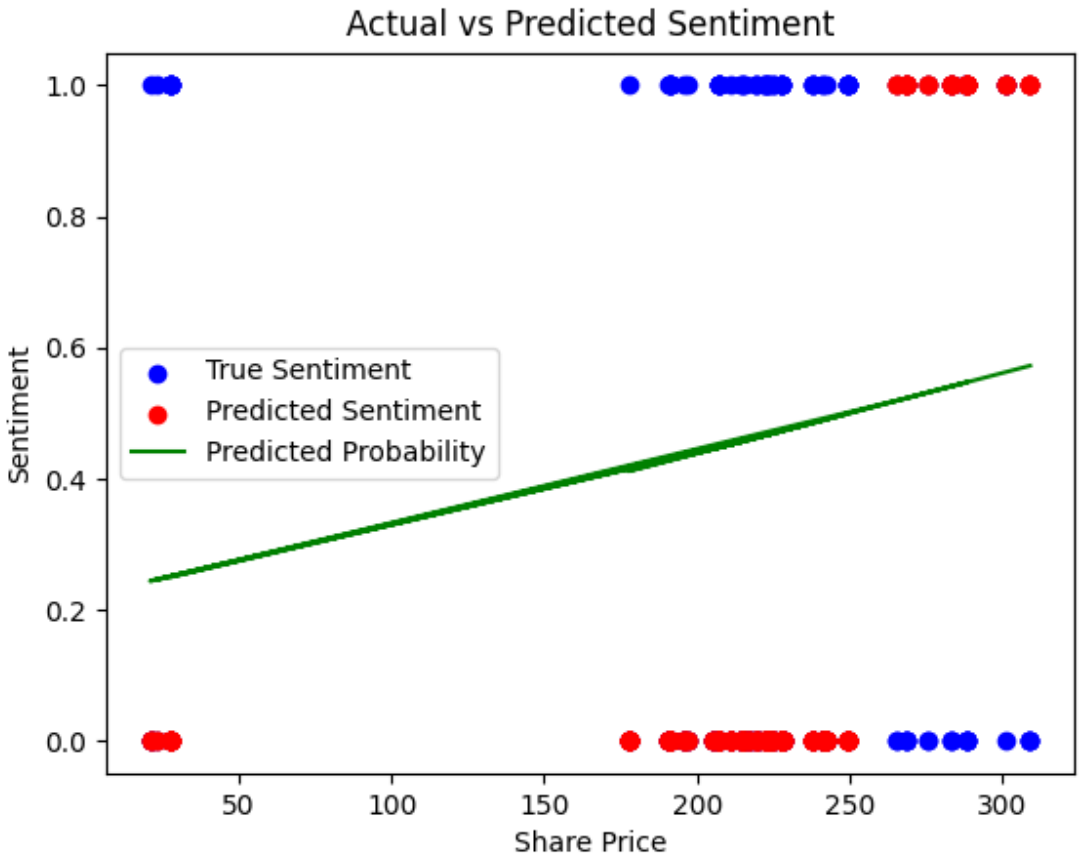
Linear Regression Model:

Mean Squared Error (MSE): 7549.112683641544
R2 Score: -0.9409795059093049
Logistic Regression:



Logistic Regression Model:

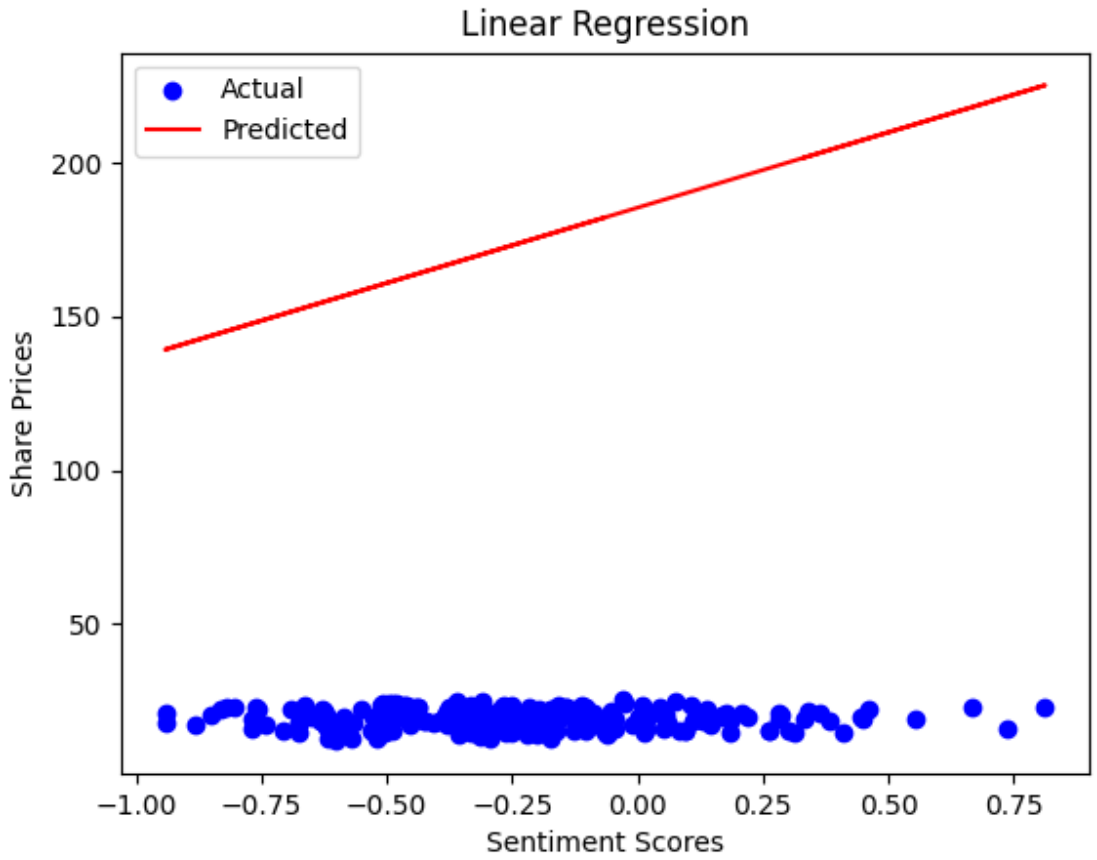
Precision (Prediction): 0.4166666666666667
Recall (Prediction): 0.189873417721519
F1 Score (Prediction): 0.26086956521739135
Accuracy (Prediction): 0.6473029045643154
Precision (Probability): 0.4166666666666667
Recall (Probability): 0.189873417721519
F1 Score (Probability): 0.26086956521739135
Accuracy (Probability): 0.6473029045643154



In contrast, when using arrays1.npz, the models' performance was as follows:

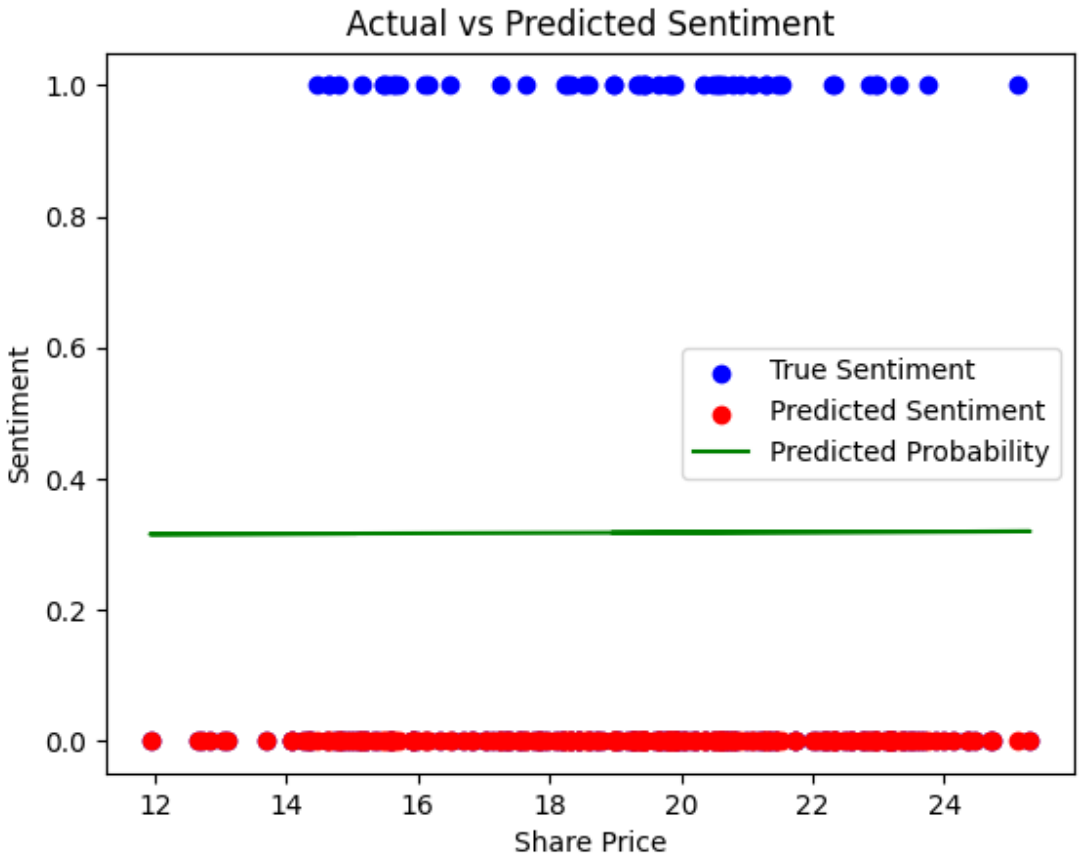
Linear Regression Model:

Mean Squared Error (MSE): 24234.835462987994
R2 Score: -2651.77183258162



Logistic Regression:

Precision (Prediction): 0.0
Recall (Prediction): 0.0
F1 Score (Prediction): 0.0
Accuracy (Prediction): 0.771551724137931
Precision (Probability): 0.0
Recall (Probability): 0.0



Discussion:

The discussion section aims to interpret and analyze the results obtained from the performance evaluation of the linear regression and logistic regression models using arrays.npz and arrays1.npz datasets. These results provide insights into the models' performance in sentiment analysis and prediction tasks.

The performance evaluation of the linear regression model using arrays.npz revealed a relatively high Mean Squared Error (MSE) value of 7549.112683641544. This indicates that, on average, the squared difference between the predicted and actual share prices was large. Additionally, the negative R2 score of -0.9409795059093049 suggests that the model's predictions did not fit the data well. These findings indicate that the linear regression model's performance in predicting share prices based on sentiment scores was poor when using arrays.npz.

Regarding logistic regression with arrays.npz, the precision values for both prediction and probability-based evaluations were 0.4166666666666667. This precision value represents the fraction of correctly predicted positive instances, indicating a moderate level of accuracy in identifying positive sentiment labels. The recall (true positive rate) was 0.189873417721519, indicating that the model correctly identified a relatively small proportion of actual positive instances. The F1 score, which combines precision and recall, had a value of 0.26086956521739135, suggesting a moderate overall

accuracy of the model in sentiment classification. The accuracy, measuring the overall correctness of predictions, was 0.6473029045643154, indicating a moderate level of predictive accuracy.

In contrast, when utilizing arrays1.npz, the models' performance showed a notable decline. The linear regression model yielded a considerably higher MSE value of 24234.835462987994 compared to arrays.npz. This implies that the predictions made by the model had a larger average squared difference from the actual share prices, indicating reduced accuracy. Furthermore, the negative R2 score of -2651.77183258162 indicates an extremely poor fit of the linear regression model to the data, suggesting its inability to capture the underlying patterns and relationships.

For logistic regression using arrays1.npz, the precision, recall, and F1 score for both prediction and probability-based evaluations were all zero, except for the accuracy, which remained at 0.771551724137931. These findings indicate that the logistic regression model struggled to effectively classify sentiment labels based on share prices when utilizing arrays1.npz.

The observed decline in model performance when using arrays1.npz supports the suspicion that the reduced number of data points had a detrimental effect on the models' performance. The higher MSE, negative R2 score, and absence of meaningful precision, recall, and F1 score in logistic regression suggest that the models' ability to accurately predict share prices and classify sentiment labels was compromised due to the reduced dataset size.

These results underscore the importance of having an adequate and representative dataset for achieving reliable results in sentiment analysis and prediction tasks. The reduced data points in arrays1.npz limited the models' ability to capture the complex relationship between sentiment scores and share prices accurately. Therefore the results and analysis support the suspicion that the reduced data points in arrays1.npz had a detrimental effect on the models' performance. It is crucial to consider the dataset size and ensure a sufficient amount of data to obtain more accurate and reliable predictions and classifications in sentiment analysis.
Furthermore, these poor performance of these models that can be concluded from the results indicate that our hypothesis that there is a linear relationship between sentiment and share price data is most likely false.

However, it is important to acknowledge the limitations of this study. The analysis was conducted using specific datasets, and the observed trends may not be universally applicable. Further research with larger and more diverse datasets is necessary to validate and generalize these findings. Moreover, It is important to consider that sentiment analysis alone may not be sufficient for accurate share price prediction. Other fundamental and technical factors, such as company financials, industry trends, and external events, should be integrated into a comprehensive prediction model to enhance accuracy and robustness.

In conclusion, the discussion highlights the impact of dataset size on the performance of the linear regression and logistic regression models in sentiment analysis and prediction tasks. The results indicate that a reduced dataset size resulted in poorer model performance, as evidenced by higher MSE, negative R2 score, and limited precision, recall, and F1 score values.

References

Loughran and McDonald Sentiment Word Lists: (https://sraf.nd.edu/textual-analysis/resources/)
Marketwatch: "https://www.marketwatch.com/investing/stock/tsla?mod=search_symbol"
Nasdaq: https://www.nasdaq.com/market-activity/stocks/tsla/historical