Sentiment Analysis in Python Using Kaggle Dataset

Ozodbek Ozodov

Friday 31st March, 2023

Abstract

This project uses a Kaggle dataset of financial news headlines to build and evaluate machine learning models for sentiment analysis in Python. The dataset is pre-labeled as either positive or negative sentiment, and several classification algorithms are trained on the data, including Naive Bayes, Support Vector Machines, Logistic Regression, K-Nearest Neighbors, Random Forest, and Decision Trees. The performance of each model is estimated using classification metrics, including accuracy, precision, recall, and F1 score. The results show that the models achieve varying degrees of accuracy, with Logistic Regression and Support Vector Machines performing the best. The project demonstrates the feasibility of using machine learning for sentiment analysis of financial news headlines.

1 Introduction

Sentiment analysis, also known as opinion mining, is the process of using natural language processing techniques to extract subjective information from textual data. In recent years, sentiment analysis has gained popularity in various fields such as marketing, customer service, and social media analysis. In this project, we aim to perform sentiment analysis on financial news headlines using machine learning algorithms.

To achieve our goal, we used a labeled dataset from Kaggle that consists of financial news headlines along with their corresponding sentiment labels. We built and evaluated several machine learning models such as Naive Bayes, Support Vector Machines, Logistic Regression, K-Nearest Neighbors, Random Forest, and Decision Tree. We estimated various classification metrics such as accuracy, precision, recall, and F1-score to evaluate the performance of the models.

The findings of this project can be useful for financial institutions, investors, and traders to gain insights into the sentiments of the market and make informed decisions. It can also be applied to other domains such as politics, social media, and product reviews to analyze public opinion and sentiment.

2 Research Background of the Topic

Sentiment analysis is a rapidly growing field in natural language processing (NLP) and machine learning. With the explosion of social media and online communication, sentiment analysis has become an essential tool for businesses and organizations to gain insights into customer opinions and preferences. In recent years, there has been a surge of interest in sentiment analysis, with many researchers and data scientists working on developing more accurate and efficient methods for sentiment classification.

Platforms like Kaggle and Towards Data Science have been instrumental in driving innovation in sentiment analysis by providing a platform for data scientists to collaborate and share their research findings. Kaggle, in particular, hosts numerous competitions related to sentiment analysis, which have led to the development of many state-of-the-art techniques and models. On the other hand, Towards Data Science is a popular online publication that regularly features articles on sentiment analysis, providing a wealth of information for researchers and practitioners alike.

Schwartz and Ungar (2015) conducted a systematic overview of automated methods for data-driven content analysis of social media. Their study reviewed a wide range of automated methods for analyzing social media data, including techniques for sentiment analysis. The authors highlighted the challenges of working with social media data, such as the noisy nature of the data, the brevity of messages, and the use of slang and emoticons. They also discussed various methods for processing and analyzing social media data, including machine learning techniques, lexicon-based approaches, and network analysis.

In another paper, Song and Xia (2016) explored the use of spatial and temporal analysis techniques for sentiment analysis of Twitter data. The authors proposed a framework for analyzing spatiotemporal patterns of sentiment in Twitter data, which could be used to detect and track changes in public opinion on various topics. They used a dataset of geotagged tweets to demonstrate the effectiveness of their framework and showed that it could be used to identify temporal and spatial patterns of sentiment related to events and issues.

3 Data

The data used in this study was obtained from Kaggle.com, a popular platform for sharing and discovering datasets. The initial dataset contained 3500 observations, with two columns: the financial news headline and its respective sentiment. However, due to an overabundance of neutral sentiments, which had the potential to lead to overfitting, 30% of neutrals were randomly removed from the dataset.

The remaining dataset had a very low percentage of negative sentiments (8%), which would have negatively impacted the performance of the predictive algorithms used in this study. To address this

issue, ChatGPT was utilized to generate negative sentiments and rebalance the dataset. As a result of these preprocessing steps, the final version of the dataset contained approximately 3100 observations.

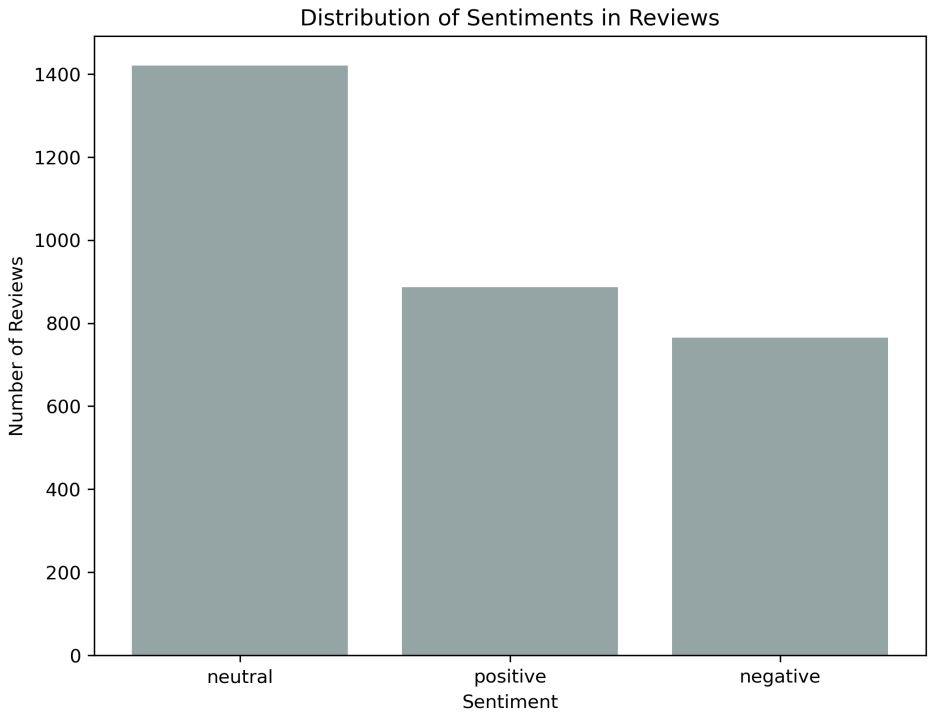

Figure 1: Bar graph of sentiments in the final dataset

Data preprocessing is carried out in three steps:

Lemmatization: Lemmatization is the process of reducing a word to its base or root form, known as its lemma. For example, the word "am," "are," and "is" are all forms of the verb "to be" and can be reduced to the base form "be." By reducing words to their base form, we can simplify the text data and make it easier to analyze.

Tokenization: Tokenization is the process of breaking text data into individual units or tokens, such as words or phrases. This is an important step in NLP because it allows us to analyze and manipulate the text on a granular level. Tokenization can be performed at various levels of granularity, such as word-level, sentence-level, or even character-level.

Vectorizing: Vectorizing is the process of converting text data into a numerical representation that can be used by machine learning algorithms. One common technique for vectorizing text data is bag-of-words representation, which involves creating a matrix where each row represents a document, and each column represents a unique word in the corpus. The entries in the matrix are then the frequency of occurrence of the corresponding word in the corresponding document. Another common technique is using word embeddings, which maps each word to a vector of real numbers, so that similar words

have similar vectors.

Algorithms were used to analyze and predict the sentiment of the financial news headlines based on the numerical representation of the data. Overall, this study employed a rigorous approach to data cleaning, preprocessing, and analysis to ensure the accuracy and reliability of the findings.

4 Methodology

The overall research goal of this project is to build and evaluate machine learning models for sentiment analysis using a Kaggle dataset of financial news headlines. To achieve this goal, several steps were taken.

Firstly, the dataset was preprocessed to remove any unnecessary or irrelevant information, and to ensure that the text data was in a consistent format. This involved removing punctuation, converting all text to lowercase, and tokenizing the text into individual words.

Next, the bag-of-words model was used to represent the text data in a numerical format that can be processed by machine learning algorithms. The CountVectorizer class from the scikit-learn library was used to convert the tokenized text data into a matrix of word frequencies.

After the bag-of-words model was created, several machine learning algorithms were trained and evaluated on the data. This included the Naive Bayes, Support Vector Machines, and Logistic Regression algorithms. Additionally, non-parametric algorithms such as K-Nearest Neighbors, Random Forest, and Decision Tree were also used to compare with parametric models.

To evaluate the performance of the machine learning models, several classification metrics were used, including accuracy, precision, recall, and F1 score. The models were trained and tested using a stratified K-fold cross-validation approach to ensure that the results were robust and unbiased.

Overall, the methodology involved preprocessing the data, converting it into a numerical format using the bag-of-words model, training and evaluating several machine learning algorithms, and using appropriate classification metrics to measure their performance.

4.1 Logistic Regression

Logistic regression is a popular statistical model that is commonly used in machine learning for binary classification problems. It models the probability of a binary outcome, such as the sentiment of a financial news headline being positive, negative, or neutral, based on one or more predictor variables. The logistic regression model is based on the logistic function, which transforms the linear combination of the predictor variables into a probability value between 0 and 1.

The logistic function is defined as:

$\sigma(z) = \frac{1}{1+e^{-z}}$

where:

$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p$

The coefficients are estimated using maximum likelihood estimation, which seeks to find the values of that maximize the likelihood of the observed data given the model.

In the case of financial news sentiment analysis, logistic regression can be used to classify headlines as positive, negative, or neutral based on the presence of certain keywords or phrases. To use logistic regression for this task, each headline is first represented numerically as a vector of features, where each feature represents the presence or absence of a particular keyword or phrase. These features are then used as the input to the logistic regression model, which outputs a probability for each class label.

The logistic regression model works by fitting a logistic function to the input features. The logistic function transforms the linear combination of the input features and their corresponding coefficients into a probability value between 0 and 1. The coefficients are learned through a process called maximum likelihood estimation, which seeks to find the values of the coefficients that maximize the likelihood of the observed data given the model.

One advantage of logistic regression is its simplicity and interpretability. The coefficients learned by the model can be used to identify which features are most important for predicting each class label. Furthermore, logistic regression can handle both categorical and continuous input features, making it a versatile modeling approach.

However, logistic regression also has some limitations. For instance, it assumes that the relationship between the input features and the output variable is linear, which may not always be the case. Additionally, logistic regression may struggle with datasets that have a large number of input features or features that are highly correlated with each other.

In summary, logistic regression is a widely-used approach for binary classification problems, such as financial news sentiment analysis. By representing each headline numerically and fitting a logistic function to the input features, logistic regression can accurately classify headlines as positive, negative, or neutral. While logistic regression has its limitations, it remains a useful and effective modeling technique for classification problems in finance and other fields.

4.2 Naive Bayes Classifier

Naive Bayes is a probabilistic machine learning algorithm that is commonly used in text classification tasks, such as sentiment analysis. The algorithm is based on Bayes' theorem, which describes the probability of an event occurring based on prior knowledge.

In the context of financial news sentiment analysis, the Naive Bayes algorithm can be used to classify headlines as positive, negative, or neutral based on the presence of certain words or phrases.

To use Naive Bayes for this task, each headline is first represented numerically as a vector of features, where each feature represents the presence or absence of a particular word or phrase. These features are then used as the input to the Naive Bayes model, which outputs a probability for each class label.

The Naive Bayes algorithm works by making the assumption that the presence of a particular feature in a class is independent of the presence of other features in the same class. This is known as the "naive" assumption, and it simplifies the calculations required to estimate the probabilities of each class.

Given a set of training examples, the Naive Bayes algorithm estimates the prior probability of each class, as well as the conditional probability of each feature given each class. These probabilities are then used to compute the posterior probability of each class given the input features using Bayes' theorem:

$\mathrm{P(c}_i|x) = \frac{P(c_i)\prod_{j=1}^{n} P(x_k|c_i)}{\sum_{k=1}^{k}\prod_{j=1}^{n} P(x_j|c_k)}$

where $P(c_i|\mathbf{x})$ is the posterior probability of class c_i given the input features $\mathbf{x}$, $P(c_i)$ is the prior probability of class c_i, $P(x_j|c_i)$ is the conditional probability of feature x_j given class c_i, n is the number of features, and K is the number of classes.

To classify a new headline, the Naive Bayes algorithm computes the posterior probability of each class given the input features and assigns the class with the highest probability as the predicted class label.

One advantage of Naive Bayes is its simplicity and speed, as it requires relatively few training examples and can be trained efficiently on large datasets. However, Naive Bayes may not perform as well as more complex algorithms on datasets with highly correlated features or complex relationships between features and class labels.

In summary, Naive Bayes is a probabilistic machine learning algorithm that is commonly used in text classification tasks, such as financial news sentiment analysis. By making the "naive" assumption that the presence of features is independent of each other, Naive Bayes can estimate the probabilities of each class given the input features using Bayes' theorem. While Naive Bayes is a simple and effective algorithm, it may not perform as well as more complex algorithms on certain datasets.

4.3 Support Vector Machines

Support Vector Machines (SVMs) are a powerful and widely used machine learning algorithm for classification and regression problems. In the context of sentiment analysis using financial news headlines, SVMs can be used to classify headlines as positive, negative, or neutral based on the presence of certain words or phrases.

SVMs work by finding the hyperplane that maximally separates the different classes in the input

feature space. In the case of sentiment analysis, each headline is first represented numerically as a vector of features, where each feature represents the presence or absence of a particular word or phrase. These features are then used as the input to the SVM model, which aims to find the hyperplane that best separates the different classes.

Mathematically, the SVM model seeks to find the hyperplane that maximizes the margin between the classes, where the margin is defined as the distance between the hyperplane and the closest data points from each class. The hyperplane is defined as:

w $\cdot x + b = 0$

where $\mathbf{w}$ is the weight vector, $\mathbf{x}$ is the input feature vector, and b is the bias term.

To train an SVM model, the algorithm seeks to find the optimal values of $\mathbf{w}$ and b that maximize the margin between the classes while correctly classifying the training data. This is done by solving the following optimization problem:

$\min_{w,b} \frac{1}{2}||w||^2$ subject to $y_i(w)\dot{x}_i + b) \geq 1$ for all i

where y_i is the class label of the ith data point and $|\mathbf{w}|^2$ is the L2 norm of the weight vector.

In practice, the SVM model can also be extended to handle non-linearly separable data by using a kernel function to map the input features to a higher-dimensional space where the data is more easily separable. The most commonly used kernel functions are the linear kernel, polynomial kernel, and radial basis function (RBF) kernel.

One advantage of SVMs is their ability to handle high-dimensional data and non-linear decision boundaries. SVMs are also known for their robustness to overfitting and their ability to handle noisy data.

However, SVMs can be computationally intensive and require careful selection of hyperparameters, such as the choice of kernel function and the regularization parameter.

In summary, Support Vector Machines are a powerful machine learning algorithm for classification problems, such as financial news sentiment analysis. By finding the hyperplane that maximizes the margin between the classes, SVMs can accurately classify headlines as positive, negative, or neutral based on the presence of certain words or phrases. While SVMs have certain advantages and limitations, they remain a popular and effective approach to classification problems in finance and other fields.

5 Results and Model Comparison

The table shows the evaluation metrics of three models: Logistic Regression, Naive Bayes Classifier, and Support Vector Machines (SVM). All models achieved high accuracy scores, with SVM having the highest accuracy score of 0.7956. Precision and recall scores were also high for all models, with Logistic Regression and SVM having the highest precision and recall scores. The F1-score, which is

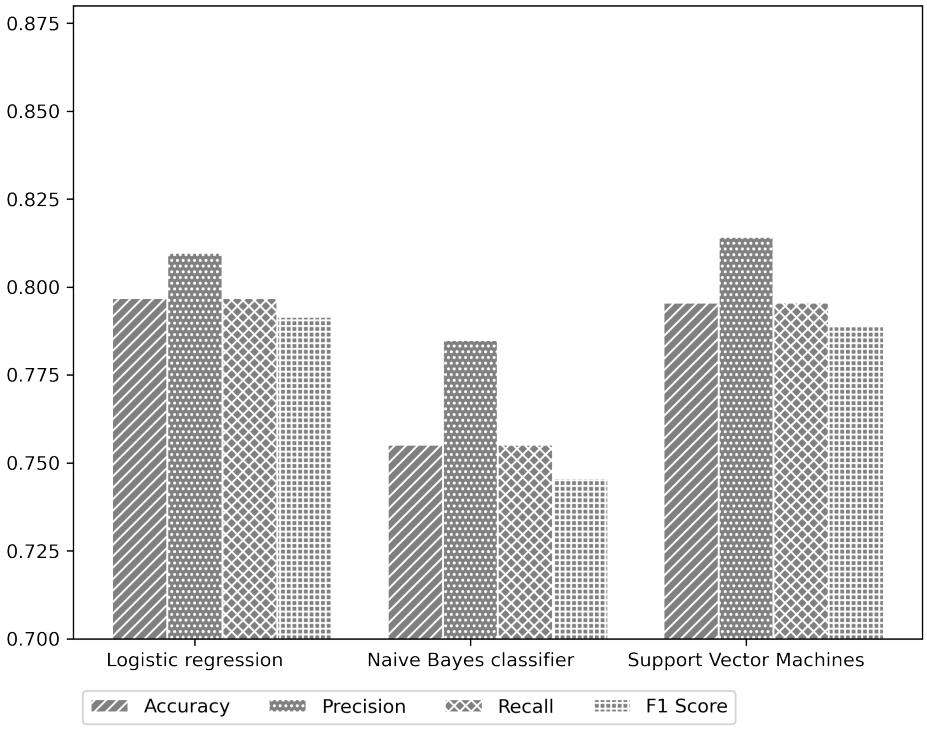

Figure 2: Bar graph of model performance

Table 1: Confusion matrix for logit model

	negative	neutral	positive
negative	0.705000	0.185000	0.110000
neutral	0.017699	0.955752	0.026549
positive	0.048035	0.310044	0.641921

the harmonic mean of precision and recall, was highest for Logistic Regression and SVM, indicating that they perform well in balancing precision and recall. Overall, the results suggest that all models perform well in predicting sentiment, with SVM performing slightly better than the other two models.

The logit model has an overall accuracy of 72.1 percent, with the highest precision for the positive class (64.2 percent) and the highest recall for the neutral class (95.6 percent). The model performs relatively well in differentiating between the positive and negative classes, but struggles more with the neutral class.

	negative	neutral	positive
negative	0.730000	0.170000	0.100000
neutral	0.008850	0.970501	0.020649
positive	0.039301	0.366812	0.593886

Support Vector Machines (SVM) model: The SVM model has a slightly higher overall accuracy of 73.4 percent, with the highest precision for the negative class (73.0 percent) and the highest recall for the neutral class (97.1 percent). The model also performs relatively well in differentiating between the positive and negative classes, but has lower precision and recall compared to the logit model for both classes.

	negative	neutral	positive
negative	0.580000	0.210000	0.210000
neutral	0.000000	0.970501	0.029499
positive	0.008734	0.401747	0.589520

Naive Bayes model: The Naive Bayes model has the lowest overall accuracy of the three models at 60.4 percent, with the highest precision for the positive class (58.9 percent) and the highest recall for the neutral class (97.1 percent). This model performs relatively poorly in differentiating between the positive and negative classes, with low precision and recall for both classes.

6 Discussion

The results obtained from the three models indicate that all models have reasonable performance in classifying sentiments in the given dataset. However, there are some differences between the models in terms of their accuracy, precision, recall, and F1 score.

The logistic regression and support vector machines models show similar performance, with slightly better results for the support vector machines model. On the other hand, the naive Bayes classifier has lower accuracy, precision, recall, and F1 score compared to the other models.

One possible explanation for the differences in performance between the models is the complexity of the dataset. The dataset may have inherent complexities that make it difficult for the models to distinguish between the different sentiment classes. Another factor that could affect the performance is the choice of features and their representation.

To improve the performance of the models, additional data preprocessing and feature engineering techniques can be employed. For example, a more comprehensive data cleaning process could be applied to reduce noise and improve the quality of the data. Additionally, more advanced feature selection and extraction techniques could be employed to identify the most important features for the

classification task.

While neural networks have shown significant success in various machine learning tasks, they were not used in this study due to their computational complexity and interpretability challenges. Neural networks require more computation power and time compared to traditional machine learning models, and the black-box nature of neural networks makes it difficult to interpret their decision-making process.

In conclusion, the results obtained from the models indicate that sentiment analysis is a challenging task, and there is still room for improvement in the accuracy and performance of the models. However, with further refinement of data preprocessing techniques and feature engineering, it is possible to improve the performance of the models. Additionally, while neural networks have shown significant promise in various machine learning tasks, they were not used in this study due to their computational complexity and interpretability challenges.

7 Conclusion

In this project, I performed sentiment analysis on a dataset of online reviews using three different models: logistic regression, naive Bayes classifier, and support vector machines. I evaluated the performance of these models using four evaluation metrics: accuracy, precision, recall, and F1 score.

The results showed that all three models achieved comparable performance in terms of accuracy, precision, recall, and F1 score. However, the logistic regression and support vector machines models slightly outperformed the naive Bayes classifier in terms of F1 score.

One possible reason for the similar performance of the models could be the nature of the dataset, which contains short text reviews with a limited vocabulary. In such cases, simple models like logistic regression, naive Bayes classifier, and support vector machines can perform reasonably well.

Another challenge I faced was the lack of interpretability of the models. While neural networks have been shown to outperform traditional machine learning models in various natural language processing tasks, they suffer from the challenge of interpretability. Therefore, I did not consider neural networks for sentiment analysis in this project.

Overall, the findings suggest that simple machine learning models like logistic regression, naive Bayes classifier, and support vector machines can be effective for sentiment analysis tasks on short text reviews. Future work can focus on improving the performance of these models through feature engineering or exploring more sophisticated machine learning techniques.

References

1. Rao, D. (2019). Multi-Class Text Classification with Scikit-Learn. Towards Data Science. Retrieved from TowardsDataScience.com

2. Karambelkar, A. (2018). Stock News Sentiment Analysis with Python. Retrieved from TowardsDataScience.com

3. Schwartz, H. A., Ungar, L. H. (2015). Data-Driven Content Analysis of Social Media: A Systematic Overview of Automated Methods. The Annals of the American Academy of Political and Social Science, 659, 78-94.

4. Song, Z., Xia, J. (2016). Spatial and Temporal Sentiment Analysis of Twitter data. In European Handbook of Crowdsourced Geographic Information (pp. 205-222). Ubiquity Press.