SENTIMENT ANALYSIS OF FINANCIAL NEWSTO DEVELOP A MODEL THAT CAN ACCURATELY CLASSIFY NEWS ARTICLES AS POSITIVE OR NEGATIVE FOR THE MARKET.

KUSH HEMANI

Problem Statement:

The aim of this study is to use NLP techniques to carry out sentiment analysis and analyse Indian stock market data and develop a model that can accurately classify news articles as positive or negative for the market. The study's findings will have significant implications for investors and financial analysts, who can leverage the insights to make informed investment decisions.

Objectives:

The objectives of this study are:

1. To evaluate the effectiveness of NLP techniques in carrying out sentiment analysis on Indian stock market data.
2. To identify the challenges in carrying out sentiment analysis on Indian stock market data and propose solutions to overcome them.

Code:

The code uses deep learning, logistic regression, random forests, Naïve-Bayes, and XGBoost to carry out the sentiment analysis.

Deep Learning Methodology

The code for deep learning is a machine learning model for sentiment analysis of news headlines using deep learning with the Keras library. Below is a detailed explanation of the code and methodology:

Libraries Used:

- pandas: for data manipulation and analysis
- numpy: for numerical operations and array handling
- matplotlib: for data visualization
- seaborn: for advanced data visualization
- nltk: for natural language processing tasks such as text preprocessing, tokenization, and lemmatization
- tensorflow: for building and training deep learning models
- sklearn: for machine learning tasks such as splitting data into train and test sets, and evaluating the model's performance
- keras: for building and training deep learning models
- xgboost: for building and training gradient boosting models

Data Loading and Exploration:
The code reads a CSV file containing news headlines and their corresponding sentiment labels. It checks for any missing data and visualizes the class distribution to ensure there is no data imbalance.

Text Pre-processing:

The code uses the NLTK library to pre-process the text data. It first tokenizes the headlines using a regular expression tokenizer. Then, it converts all the text to lowercase and lemmatizes each token. Finally, it removes any stop words from the text.

Data Splitting:
The pre-processed text data is split into training and testing sets using the train_test_split function from the sklearn library.

Tokenization and Padding:
The text data is tokenized using the Keras Tokenizer function. This function converts the text into sequences of integers. The maximum sequence length is determined by the longest headline in the training set. The sequences are then padded with zeros to ensure they are all the same length.

Model Architecture:
The deep learning model is built using the Keras library. The first layer is an embedding layer that converts the integer sequences into dense vectors of fixed size. The second layer is a bidirectional LSTM layer that takes advantage of the sequential nature of the data by processing it in both forward and backward directions. The third layer is a global max pooling layer that extracts the most important features from the LSTM output. The fourth layer is a dense layer with a ReLU activation function that reduces the dimensionality of the data. The fifth layer is a dropout layer that helps prevent overfitting. Finally, the sixth layer is a dense layer with a softmax activation function that outputs the probability distribution over the three sentiment classes.

Model Training and Evaluation:
The model is trained using the fit function from Keras. The loss function used is sparse categorical cross-entropy, and the optimizer used is Adam. Early stopping is used to prevent overfitting. Once the model is trained, it is evaluated on both the training and testing sets using various metrics such as accuracy, precision, recall, and F1 score. The confusion matrix is also plotted using the seaborn library.

ML Algorithm Code:

The given code performs sentiment analysis on a dataset of news headlines using various machine learning algorithms. The code first imports required libraries such as pandas, matplotlib, seaborn, nltk, sklearn.

The code then loads the news headlines dataset from a CSV file using pandas. It checks for missing data in the dataset by creating a boolean mask of missing values using the isnull() function and printing the resulting mask. It also checks the class distribution of the target variable 'sentiment' by creating a bar chart using the value_counts() and plot() functions from pandas and matplotlib, respectively.

The code then preprocesses the text data by removing stopwords, lemmatizing words using the WordNetLemmatizer from nltk, and tokenizing the text using a regular expression tokenizer. The preprocessed text is stored in a new column 'text' in the dataframe.

The code then splits the preprocessed data into training and test sets using the train_test_split() function from sklearn. It defines two types of vectorizers, CountVectorizer and TfidfVectorizer, and fits them on the training data and transforms the test data using the fit_transform() and transform() functions, respectively.

These vectorizers are used to convert the text data into numerical features that can be used for machine learning models.

Next, the code defines three machine learning models: Multinomial Naive Bayes, Logistic Regression, and Random Forest Classifier, and defines hyperparameters for each of them using dictionaries. It then performs hyperparameter tuning using the GridSearchCV function from sklearn, which trains the models on the training data with different hyperparameter values using cross-validation, and returns the best hyperparameters for each model.

The code then trains each model on the training data using the best hyperparameters obtained from hyperparameter tuning, and predicts the sentiment on the training and test data using each model. It also calculates evaluation metrics such as accuracy, precision, recall, and F1 score for each model on both the training and test data using functions from sklearn such as confusion_matrix(), accuracy_score(), precision_score(), recall_score(), and f1_score().

Finally, the code creates a subplot of six confusion matrices (three for the training set and three for the test set) using the heatmap() function from seaborn, and displays them using matplotlib. The confusion matrices show the true and predicted labels for each sentiment class and allow for visualization of the performance of each model.
In summary, the code performs sentiment analysis on news headlines using various machine learning algorithms, and evaluates their performance using evaluation metrics and visualization of confusion matrices.

XGBoost

The code for the XGBoost does the following:

1. The code begins by importing the necessary libraries including pandas, matplotlib, seaborn, nltk, and sklearn.
2. The dataset is read into a pandas DataFrame using the read_csv function.
3. The code checks for missing data in the DataFrame using the isnull function and prints out the results.
4. The code checks for data imbalance by counting the number of headlines in each sentiment category and plotting a bar graph to visualize the distribution.
5. The NLTK library is used to remove stop words and lemmatize the remaining words in each headline using the preprocess_text function.
6. The data is split into training and testing sets using the train_test_split function from sklearn.
7. The CountVectorizer and TfidfVectorizer functions from sklearn are used to transform the text data into numerical vectors.

8. An XGBoost classifier is defined with random_state=42.
9. A grid search is performed to find the best hyperparameters for the XGBoost classifier using GridSearchCV from sklearn.
10. The XGBoost classifier is trained on the training data with the best hyperparameters.
11. The classifier is used to predict the sentiment of the training data and the accuracy, precision, recall, and F1 score are calculated and printed.
12. The same metrics are calculated and printed for the test data.
13. A confusion matrix is plotted using seaborn to visualize the performance of the classifier on the test data.