Project 1(One)

FAANG Stock News Sentiment Analysis

Introduction

This project focuses on analyzing the sentiment of news articles related to FAANG (Facebook, Apple, Amazon, Netflix, Google) stocks. The goal is to understand the overall market sentiment surrounding these companies and its potential impact on stock prices. By leveraging sentiment analysis techniques, we aim to extract insights from textual data and identify trends and patterns in investor sentiment.

Analysis Components

- **Data Preprocessing**: Cleaning and preprocessing textual data to remove noise, such as stopwords, punctuation, and special characters. Tokenization and lemmatization techniques may also be applied to normalize the text.
- **Sentiment Analysis**: Utilizing sentiment analysis algorithms to classify news articles into positive, negative, or neutral sentiment categories. Techniques such as bag-of-words, TF-IDF, or pre-trained language models like BERT may be used for sentiment classification.
- **Statistical Analysis**: Conducting statistical tests to analyze the accuracy of the model.

Technologies Used

- Python , Pandas, Scikit-learn , NLTK , Matplotlib , Seaborn.

Model

Random Forest Classifier: Random Forest is utilized as a classification model to classify sentiment categories based on the textual features extracted from news articles.

Repository Structure

- **Dataset**: 'FAANG_STOCK_NEWS_Dataset.csv' contains raw data files downloaded from Kaggle.
- **Notebooks**: 'FAANG_Stock_News_Analysis.ipynb' contains python notebook with code for data preprocessing , wrangling and sentiment analysis.
- **README.md**: This document providing an overview of the project, instructions, and guidelines for contributors.

Usage

1. Clone the repository to your local machine.
2. Install the required dependencies and libraries.
3. Explore the notebooks and scripts to understand the sentiment analysis process and run the code as needed.
4. Contribute to the project by adding new sentiment analysis techniques, improving existing models, or enhancing visualizations.

Project 2(Two)

Samsung Vs Apple Comparison Module

Overview:

The Apple vs Samsung Comparison Module provides a comprehensive analysis of the performance and financial metrics of Apple Inc. and Samsung Electronics Co., Ltd. Users can compare various aspects such as revenue, net income, market capitalization, stock prices, and other key metrics to gain insights into the competitive landscape between these two technology giants.

Features:

1. **Revenue Analysis:** Compare the annual revenue trends of Apple and Samsung over the years.
2. **Net Income Comparison:** Analyze the net income of Apple and Samsung to understand their profitability.
3. **Market Capitalization Trends:** Explore the market capitalization trends of both companies and how they have evolved over time.
4. **Stock Price Comparison:** Visualize the stock price movements of Apple and Samsung and identify any significant trends.
5. **Dashboard Visualization:** Present all the comparative analyses in an intuitive and user-friendly dashboard format for easy interpretation.

Usage:

1. **Module Execution:** Run the module script or application to perform data processing, analysis, and visualization.
2. **Dashboard Interaction:** Explore the dashboard interface to interact with different visualizations and gain insights into the comparison between Apple and Samsung.
3. **Customization:** Modify the module parameters or dashboard layout as needed to tailor the analysis to specific requirements or preferences.

Dependencies:

- Python (for data processing and analysis)
- Pandas, Matplotlib (for data manipulation and visualization)
- Tableau (for dashboard creation and visualization)

Tableau Dashboard:

To access the interactive dashboard for visualizing the comparison between Apple and Samsung, please visit Tableau Public

Project 3(Three)

Apple Stock Price Prediction using LSTM

This project focuses on predicting the stock prices of Apple Inc. (AAPL) using Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN). LSTM models are particularly effective for sequential data like stock prices due to their ability to capture long-term dependencies.

Dataset

The dataset used for this project consists of historical stock price data for Apple Inc.

Methodology

Data Preprocessing: The raw stock price data is preprocessed to handle missing values, normalize the features, and create sequential input-output pairs suitable for training LSTM models.

Model Training: LSTM models are trained using the preprocessed data. The models are configured with appropriate hyperparameters and trained on a subset of the data. The training process involves optimizing the model parameters to minimize the prediction error.

Model Evaluation: The trained LSTM models are evaluated using a separate test dataset to assess their performance in predicting future stock prices. Evaluation metrics such as mean squared error (MSE) or root mean squared error (RMSE) are calculated to quantify the model's accuracy.

Prediction: Once trained and evaluated, the LSTM models are used to make predictions for future stock prices. These predictions provide insights into potential price movements and trends, aiding investors in making informed decisions.

Repository Structure

- Data: 'AAPL.csv' contains the raw and preprocessed datasets used for training and testing.
- Notebooks: 'Apple_Stock_Price_Analysis.ipynb' contains the Jupyter notebooks with code for data preprocessing, model training, evaluation, and visualization.
- Models: Saved model checkpoints or files for the trained LSTM models.
- README.md: This document providing an overview of the project and instructions for replicating the analysis.

Usage

- Clone the repository to your local machine.
- Install the required dependencies and libraries.
- Explore the Jupyter notebooks and Python scripts to understand the data preprocessing, model training, and prediction process.
- Execute the notebooks or scripts to preprocess the data, train the LSTM models, and make predictions for future stock prices.
- Analyze the model performance using evaluation metrics and visualize the predicted stock prices.
- Experiment with different hyperparameters, architectures, or features to improve the model's accuracy.

Project 4(Four)

Simple Search Engine

- The Simple Search Engine operates on a straightforward keyword-based search methodology. It utilizes the PyPDF2 library in Python to extract text content from uploaded PDF files. Once a PDF is uploaded, the search engine reads through each page, preprocesses the text, and indexes it based on stemmed tokens.
- When a user submits a search query, the search engine identifies relevant documents by finding the intersection of all pages containing the queried keywords. This process involves preprocessing the query, tokenizing it into individual terms, and identifying the stemmed forms of these terms.
- The search engine then looks up each stemmed term in its index to retrieve a list of document IDs where the term appears. By taking the intersection of these sets of document IDs for all query terms, the engine determines the pages that contain all the keywords provided by the user.
- Finally, the search engine presents the text content of these identified pages to the user, allowing them to access the relevant information they are looking for. This simple yet effective approach provides users with a quick and intuitive way to search for specific information within PDF documents.