Intelligent System
Assignment 1 - Option B
Applying Macroeconomic Indicators to Predict Stock Prices

Xuan Tuan Minh Nguyen - 103819212

October 25, 2024

Contents

Abstract

The major purpose of this research is to showcase **the application of machine learning** in predicting stock prices with the combination of **historical** data of **Macroeconomic Indicators**. In this research, we will focusing on researching the affect of the **Macroeconomic Indicators** to the stock price prediction. We will be using two most popular models for time series prediction, which are the **XGBoost** and **RandomForest** models. In addition, the stock price data will be taken from **Yahoo Finance**, and the Macroeconomic Indicator data will be taken from **FRED**. The research will be conducted on **Tesla Inc.** (TSLA) stock from the date of 2015-01-01 to 2023-08-25. This research will deep dive into **differentiating** each **potential solutions** to improve the performance of the **v0.5** codebase, **pick and implement** the solution that fits the most for the context. And **visualize** the **prediction results** with the actual stock prices to evaluate the performance of the models.

1 Introduction

With the significant development of the stock market, more and more investors are interested in pouring their money into the stock market to gain profits. However, with the unpredictable nature of the stock market, it is very difficult for investors to see when is the perfect time to buy or sell their stocks. Therefore, with the phenomenon growth of the **machine learning** field, investors could empower the **machine learning** to predict the future direction of the stock market and could make their own decisions based on the predicted results.

In this research, we will be exploring the application of **machine learning** in the context of predicting the stock prices. Stock price prediction could be a very complicated task as there are multiple factors to consider, such as the **fundamental analysis** and **technical analysis**. In this research, we will be analyzing the potential solutions to improve the performance of the **v0.5** model, which itself strongly relies on the **historical stock prices** and have a bad performance in the prediction accuracy. After that, we will be discussing and picking the best solution to give our **v0.5** an update to **v0.6**, which has a better and more reliable prediction accuracy.

2 Review of the Potential Solutions

Nowadays, there are multiple different approaches that have been proposed to improve the performance of the traditional stock price prediction models. These approaches aimed for increase the accuracy and broaden the scope of the prediction features by adding more data sources to the model, extending the modeling and data processing techniques and inputting external factors that could affect the stock prices. These solutions could be categorized into:

2.1 Improving the current Ensemble Prediction Model

To improve the current ensemble prediction model, using an ensemble of multiple models could be a good solution. Two key approaches that could be used to improve the performance of the ensemble prediction model are:

- **Stacking Ensemble**: With the powerful non-linear nature of the stacking ensemble, which allows it to combine the predictions from multiple models to create a new model that has a better performance than the individual models, we could use the stacking ensemble to ensemble the predictions from the state of the are models, such as **Random Forest** and **XGBoost** models, to achieve a higher preidction accuracy and reducing the generalization error. [6]

- **Bagging / Boosting Ensemble**: In addtion to the stacking ensemble, the Bagging and Boosting are the two most popular ensemble methods to improve the performance of the prediction model. The bagging ensemble method is used to reduce the variance of the prediction model, while the boosting ensemble method is used to reduce the bias of the prediction model. Thus could help to improve the prediction accuracy and reducing the generalization error. [6]

2.2 News and Social Media Content

The sentiment of a market could strongly affect the stock prices, as the sentiment of the market could affect the behavior of the investors, which in turn could affect the stock prices. Therefore, with the development of the **natural language processing** field, more and more research aimed to use the sentiment analysis to predict the future direction of the stock prices. The following methods were explored:

- **News and Social Media Content**: By collecting the news and social media content, we could use the sentiment analysis to predict the future direction of the stock prices, like what [2] did.
- **Sentiment Analysis**: Natural Language Processing (NLP) is a field that focuses on the interaction between computers and humans using natural language. With the development of the NLP field, we could use the sentiment analysis to predict the future direction of the stock prices, which could result in a higher accuracy. [2]

2.3 Macroeconomic Indicators

It is a well-known fact that some macroeconomic indicators could have a strong correlation with the stock prices, such as the **unemployment rate** and **inflation rate** [4]. The following methods were found to be effective in predicting the stock prices:

- **Inclusion of Macroeconomic Indicators**: Beside from the stock-specific data, such as the open, close, high and low prices, the volume of the stocks traded and the adjusted close price, we could include the macroeconomic indicators that have a strong correlation with the stock prices to improve the prediction accuracy, such as **GDP**, **inflation rate** and **unemployment rate**. [4]
- **Technical Indicators**: By adding technical indicators, such as **RSI**, **MACD** and **William Percent Range**, we could see the overbought and oversold levels on a chart and can help predict where a price is likely to go next, based on past performance. [3]

2.4 Advanced Model Optimization Techniques

Different optimization techniques are researched and have been widely used for improving the performance of the prediction model, such as:

- **Hyperparameter Optimization**: By finding the optimal hyperparameters for the model, we could improve the prediction accuracy and reducing the generalization error. [1]
- **Random Search**: Random search is a technique that randomly samples a set of hyperparameters from a given distribution and then evaluates the performance of the model with those hyperparameters. This technique can help to find the optimal hyperparameters for the model, which could improve the prediction accuracy and reducing the generalization error. [1]

- **Early Stopping and Learning Rate Scheduling**: Early stopping is a technique that stops the training process when the performance of the model starts to decrease, which could help to prevent overfitting. Learning rate scheduling is a technique that adjusts the learning rate during the training process, which could help to increase the prediction accuracy and reducing the error.

2.5 Advanced Model Applications

Moreover, there are some advanced applications that could be used to improve the performance of the prediction model, such as:

- **Deep Reinforcement Learning**: Deep Reinforcement Learning (DRL) allows agents to learn directly from raw sensory input, such as images or sounds, without relying on handcrafted features or domain-specific knowledge. Thus making DRL highly flexible and capable of automating tasks, including stock trading. [7]
- **Time series Analysis**: Using time-series based models, such as **ARIMA**, we could effectively predict the short-term stock price and stock returns movements, thanks to the ability of time-series models to capture the underlying patterns and trends in the data. [5]

3 Implementation

After carefully reviewing and considering the researched potential approaches, I have decided to go with the **Macroeconomic Indicators** approach, as it has a strong correlation with the stock prices, has a high compatibility with the current v0.5 codebase, and has a good prediction accuracy. In addition, the **Macroeconomic Indicators** approach could give me a good point of view to how the macroeconomic indicators could affect the stock prices and how we could use it for the prediction.

3.1 Environment Setup

3.1.1 Create a new environment using Conda CLI

There are several different procedures to create an environment, but the given procedure below will encapsulate all of the required steps to create a safe and clean environment using Conda [8].

- Navigate to the Github repository that contains the source code for v0.5.
- Once navigated, download the source code by clicking on Code → Download ZIP or use the following command in the CLI (Terminal):

```
git clone https://github.com/cobeo2004/cos30018.git
```

- Once the source code is successfully cloned (downloaded), navigate to the Week 6/v0.5 folder and execute the file conda-config.sh using the following command:

```
bash conda-config.sh
```

- The given file config.sh will execute the following procedure:

– Generate an environment with a pre-defined name (you can change the name if you want to) in Python 3.10.9 by using the command:

```
conda create -n cos30018_env_w7_v0.5 python=3.10.9
```

– Activate the created environment using:

```
conda activate cos30018_env_w7_v0.5
```

– Check and validate if the conda environment is successfully initialized by running `conda info --envs` for listing conda environments and see which environment that we are in and current Python version using `python --version`.

3.1.2 Installing required dependencies

Once the environment is successfully initialized, we can start installing the dependencies (libraries) that are required by the program. There are multiple pathways to install dependencies in Python, but the most popular steps are:

- Scan through the code to find out the required dependencies; for example, consider the file stock_prediction.py. We could see that there are quite a few required dependencies, such as: numpy matplotlib pandas tensorflow scikit-learn pandas-datareader yfinance TA-lib statsmodels plotly pmdarima fredapi xgboost. Especially, **statsmodels** will be used for finding parameters for ARIMA model, **fredapi** for getting macroeconomic indicators data and **xgboost** for constructing XGBoost model.
- Once dependencies are scanned, use the following command to install the dependencies:

```
pip install numpy matplotlib pandas tensorflow scikit-learn
pandas-datareader yfinance TA-lib statsmodels plotly pmdarima fredapi
xgboost
```

- Another step is to list all required libraries into a requirements.txt file, and using the following command to install the required dependencies:

```
pip install -U -r requirements.txt
```

4 Understanding the Machine Learning 3

4.1 Exploring the required parameters for the ARIMA model

4.1.1 Codebase

4.1.2 Parameters

- **data: pandas.DataFrame**: The processed data that is used for finding the autoregressive order, difference order and moving average order.

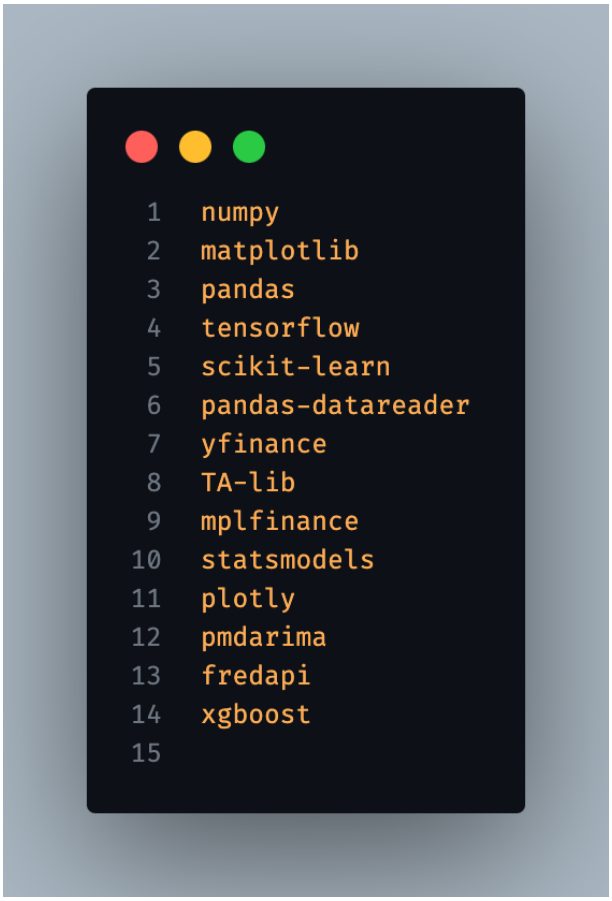

Figure 1: Example of requirements.txt

Figure 2: Codebase for function to display the plots related to ARIMA parameters

4.1.3 Functionalities

In general, the given code will be used for plotting three graphs that correlate to three different essential parameters that the ARIMA model used, which are the number of lag observations in the model (p), the differencing degree to make the time series stationary (d), and the moving average order (q).

- To observe the number of lags in the model (p), we will use the partial autocorrelation plot that is taken from **statsmodels** library and look for any crosses for the upper confidence interval, which is illustrated in the code and the plot below:

Figure 3: Codebase used for plotting PACF

After a short investigation, we could see that first two lags have a significant correlation, which stays at 1 while the others stay around 0. Thus, a suggested p value that we obtained is 2.

To observe the moving average order (q), we will use the autocorrelation function plot that is also acquired from **statsmodels** library and look for any crosses for the upper confidence interval, which is illustrated in the code and the plot below:

The ACF plot points out that there is a significant autocorrelation at the first lag of the plot, thus suggests that the moving average order (q) is 1.

Figure 4: PACF Results of TSLA stock from 2015-01-01 and 2023-08-25

Figure 5: Codebase used for plotting ACF

To observe the differencing degree (d), we will inspect the original time series of the stock price and its first and second time series differences.

The given code above creates three subplots that sequently correlates to:

- The original time series of the stock price.
- The first difference of the stock price's time series.
- The second difference of the stock price's time series.

Based on the given three plots, we could see that:

- The original time series observed a clear upward trend, especially from 2020 onwards. This trend indicates that the series is non-stationary, which suggests that differencing is necessary to achieve stationarity.
- The first-order differencing of the original data appears much more stable compared to the original series plot. The mean seems to be roughly constant around zero, and the variance looks more consistent throughout the time period. This is a significant improvement in terms of stationarity.
- The second-order differencing doesn't seem to be a substantial improvement over the first-order differencing. Thus, might be introducing unnecessary complexity or noise into the data.

Therefore, first-order differencing (d=1) is sufficient to achieve stationarity in this time series.

4.2 Function to create ARIMA model

4.2.1 Codebase

4.2.2 Parameters

- **train_data: pd.DataFrame**: The training dataset that contains historical stock price data.
- **test_data: pd.DataFrame**: The testing dataset that contains historical stock price data for evaluation.
- **p_range: Tuple[int, int]**: The range of p (number of lag) values to consider in the ARIMA model. Default is (0, 5).
- **d_range: Tuple[int, int]**: The range of d (differencing degree) values to consider in the ARIMA model. Default is (0, 2).
- **q_range: Tuple[int, int]**: The range of q (moving average) values to consider in the ARIMA model. Default is (0, 5).
- **m: int**: The number of periods in each season for seasonal ARIMA. Default is 7.
- **seasonal: bool**: Whether to include seasonal components in the ARIMA model. Default is True.

Figure 6: ACF Results of TSLA stock from 2015-01-01 and 2023-08-25

Figure 7: Codebase used for plotting time series

4.2.3 Functionalities

In general, the major purpose of this function is to use the power of Auto Regressive Integrated Moving Average (ARIMA) model to predict the price of the stock based on the close price. The function performs the following steps:

- **Data Extraction**: The function starts by extracting the value of the 'Close' price from both pre-processed train and test data, taken from the implemented `create_dataset()` function.
- **Modelling and training the ARIMA**: Once the data has been processed, the function will call to the `auto_arima()` function from the module `pmdarima` that helps creating the arima model with all of the required parameters and help selecting which model is the best parameters for the context. Specifically, the `auto_arima()` function will consider the specified ranges for the required p, d, and q parameters, and seasonal components if required. Then, the function will perform trainings with different configurations to find out which model fits the best.
- **Model Summary**: Displays general informations of the selected ARIMA model, including the parameters and fit statistics.
- **Prediction**: Then, the function will perform some prediction based on the length of the test data to cover all of the periods on the test data.
- **Calculate Root Mean Square Error (RMSE)**: Once the prediction is finished, the function will calculate the Root Mean Square Error (RMSE) to indicate the error between the predicted values and the actual test data.
- **Return**: Finally, the function wil return an array that contains the predicted stock prices for the test period and the Root Mean Square Error of the predictions compared to the actual test data.

4.3 Function to calculate the ensemble predictions

4.3.1 Codebase

4.3.2 Parameters

- **dl_pred: np.ndarray**: The prediction results array from the deep learning model, could be from LSTM, GRU or RNN.
- **arima_pred: np.ndarray**: The prediction results array from the ARIMA model.
- **dl_rmse: float**: Root Mean Square Error (RMSE) value from the deep learning model predictions.
- **arima_rmse: float**: Root Mean Square Error (RMSE) value from the ARIMA model predictions.
- **test_data: pd.DataFrame — np.ndarray**: The actual test data for comparison and error calculation.

Figure 8: Time series differences of TSLA stock from 2015-01-01 and 2023-08-25

Figure 9: Codebase for function to create ARIMA model

4.3.3 Functionalities

The purpose of this function is to create an ensemble prediction by combining the predictions from a deep learning model with an ARIMA model. The function performs the following procedures:

- **Data Preparation**: Firstly, the function will flatten the deep learning predictions array if needed and starts modifying the length of predictions and test data to ensure they have the same size as different size of data could result in conflictions when working with charts and further calculations.
- **Performing ensemble prediction**: It then converts the deep learning predictions and the ARIMA predictions into a `numpy.array`, then sum all values from both arrays and divide it by 2 to get a unified ensemble prediction array.
- **Data cleaning**: Then, to ensure that the data is cleaned to perform a smooth plotting experience and for scalability, we search for NaN values in the test data and replace it with 0. Once the NaN values are replaced, the function prints the data after replacing NaN and if there are any NaN values in the deep learning and ARIMA prediction arrays.
- **Error Calculation**: Once the data is cleaned, the function will calculate the Root Mean Square Error (RMSE) for the ensemble predictions array and the average RMSE from both deep learning model predictions and the ARIMA predictions.
- **Return data**: Finally, the function will return ensemble predictions result as an `numpy.ndarray`, the Root Mean Square Error (RMSE) value for the ensemble predictions and the average RMSE value for both deep learning predictions and the ARIMA predictions.

5 Deploying and Testing the Codebase

5.1 Codebase used for testing

Please note that, since the given image below that shows the codebase for testing is too long, please refer to the file `main.py` for the full testing code.

References

[1] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. `https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf`, 2012. Accessed: 2024-10-25.

[2] Shri Bharathi and A Geetha. Sentiment analysis for effective stock market prediction. *Int. J. Intelligent Engineering and Systems*, 2018. Accessed: 2024-10-25.

[3] Milan Cutkovic. Best trading indicators: Most popular technical indicators / axi AU. `https://www.axi.com/au/blog/education/trading-indicators`. Accessed: 2024-10-25.

[4] Md Sabbirul Haque, Md Shahedul Amin, Jonayet Miah, Duc Minh Cao, and Ashiqul Haque Ahmed. Boosting stock price prediction with anticipated macro policy changes. *arXiv*, October 2023.

Figure 10: Codebase for calculate the ensemble predictions

Figure 11: Codebase for testing

[5] Hardikkumar. Stock market forecasting using time series analysis with ARIMA model. `https://www.analyticsvidhya.com/blog/2021/07/stock-market-forecasting-using-time-series-analysis-with-arima-model/`, July 2021. Accessed: 2024-10-25.

[6] Minqi Jiang, Jiapeng Liu, Lu Zhang, and Chunyu Liu. An improved stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms. *Physica A*, 541(122272):122272, March 2020.

[7] Razib Hayat Khan, Jonayet Miah, Md Minhazur Rahman, Md Maruf Hasan, and Muntasir Mamun. A study of forecasting stocks price by using deep reinforcement learning. In *2023 IEEE World AI IoT Congress (AIIoT)*, pages 0250–0255. IEEE, June 2023.

[8] Kofi Opoku Nti, Adebayo Adekoya, and Benjamin Weyori. Random forest based feature selection of macroeconomic variables for stock market prediction. *SSRN Electron. J.*, July 2019.